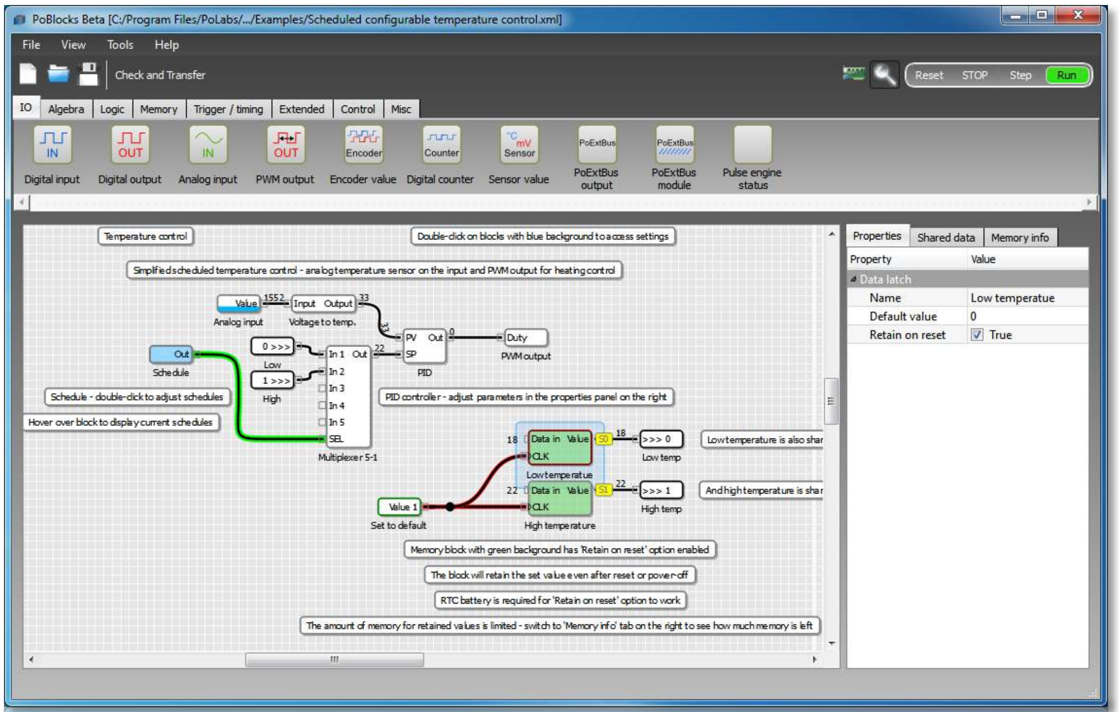
PoBlocks

User manual

Version: March 31, 2014



Temperature control

Double-click on blocks with blue background to access settings

Simplified scheduled temperature control - analog temperature sensor on the input and PWM output for heating control

Schedule - double-click to adjust schedules

Hover over block to display current schedules

PID controller - adjust parameters in the properties panel on the right

Low temperature is also shar

And high temperature is shar

Memory block with green background has 'Retain on reset' option enabled

The block will retain the set value even after reset or power-off

RTC battery is required for 'Retain on reset' option to work

The amount of memory for retained values is limited - switch to 'Memory info' tab on the right to see how much memory is left

Property	Value
Data latch	
Name	Low temperatue
Default value	0
Retain on reset	☑ True

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.

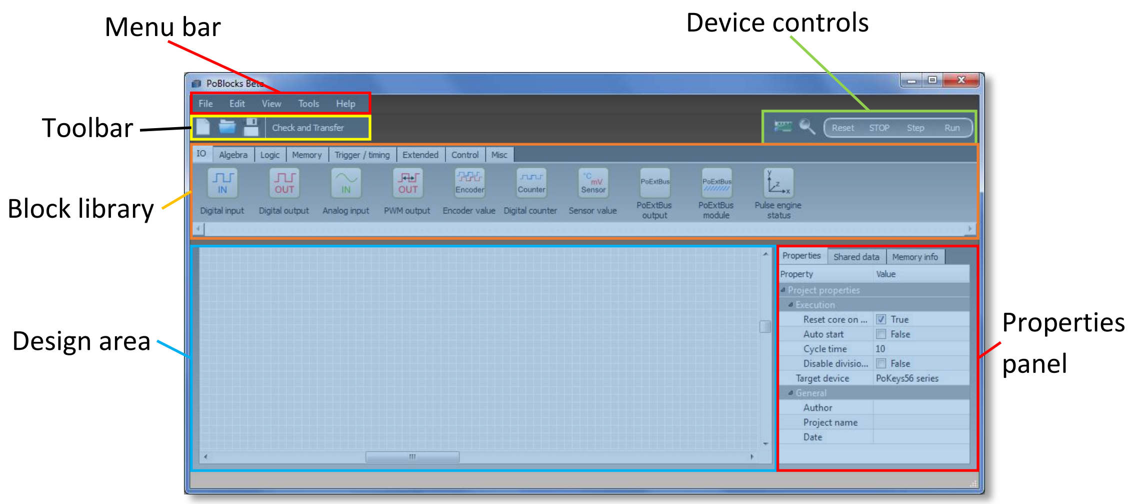
PoBlocks features an integrated Task manager that displays the target and actual cycle times of the tasks being executed (Task manager is found in PoBlocks application menu under 'Tools > Task manager'). For more details, see 'Task manager' below.

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.

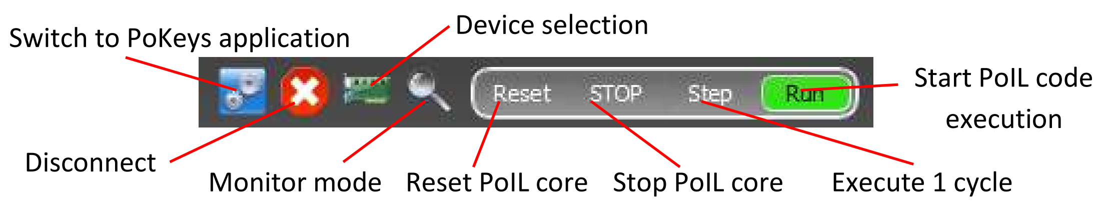


Property	Value
Project properties	
Execution	
Reset core on ...	True
Auto start	False
Cycle time	10
Disable divisio...	False
Target device	PoKeys56 series
General	
Author	
Project name	
Date	

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:

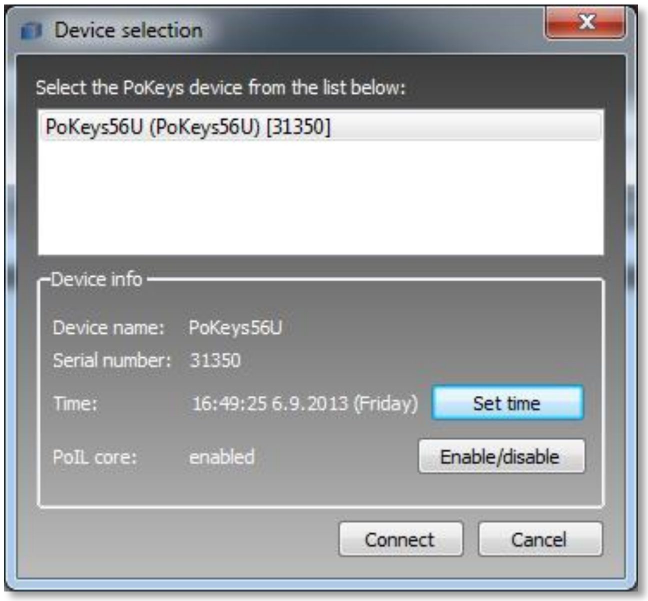
- **Switch to PoKeys application**: disconnect and open PoKeys application for advanced peripheral setup and testing. This option is only available if PoKeys application is installed to the default installation path.
- **Disconnect**: click this to disconnect from current device

- **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'.


Device selection

Select the PoKeys device from the list below:

PoKeys56U (PoKeys56U) [31350]

Device info

Device name: PoKeys56U

Serial number: 31350

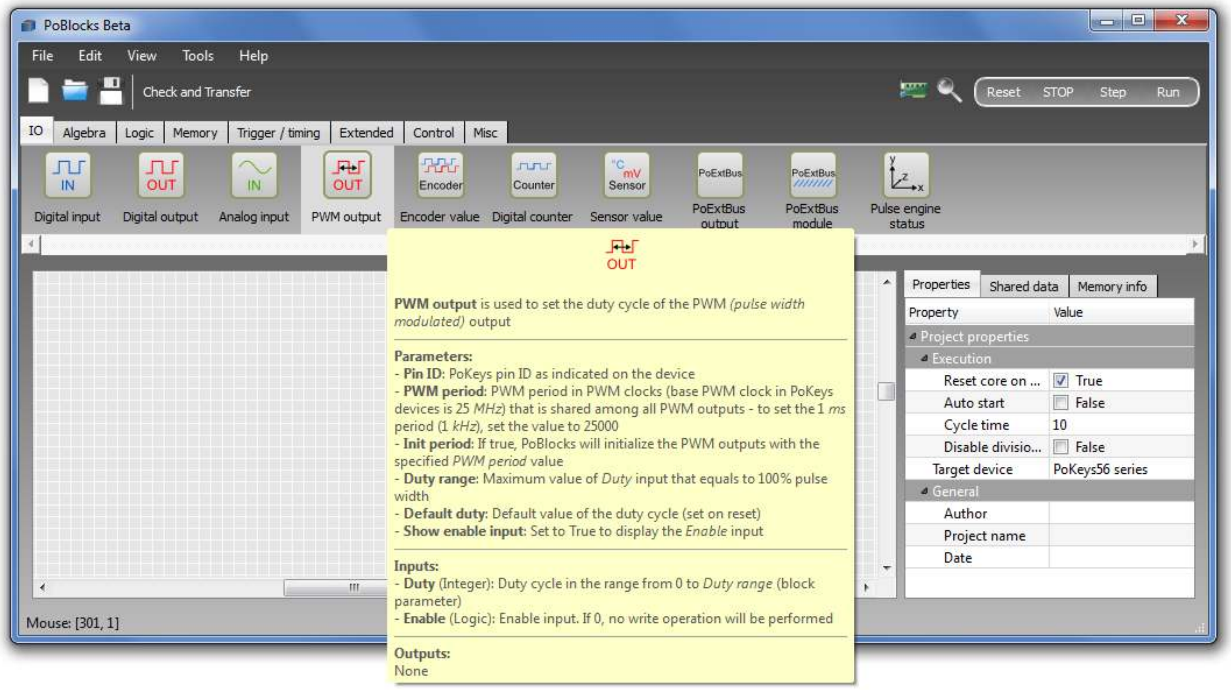
Time: 16:49:25 6.9.2013 (Friday)

PoIL core: enabled

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.



OUT

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

Parameters:
- **Pin ID**: PoKeys pin ID as indicated on the device
- **PWM period**: 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**: If true, PoBlocks will initialize the PWM outputs with the specified *PWM period* value
- **Duty range**: Maximum value of *Duty* input that equals to 100% pulse width
- **Default duty**: Default value of the duty cycle (set on reset)
- **Show enable input**: Set to True to display the *Enable* input

Inputs:
- **Duty** (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 | Shared data | Memory info

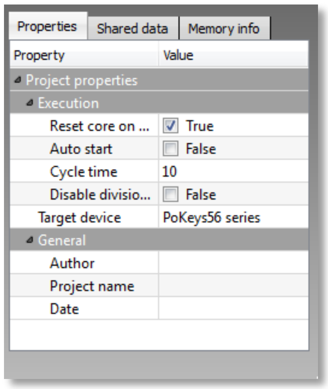
Property	Value
Project properties	
Execution	
Reset core on ...	✓ True
Auto start	False
Cycle time	10
Disable divisio...	False
Target device	PoKeys56 series
General	
Author	
Project name	
Date	

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



Property	Value
Project properties	
Execution	
Reset core on ...	✓ True
Auto start	False
Cycle time	10
Disable divisio...	False
Target device	PoKeys56 series
General	
Author	
Project name	
Date	

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

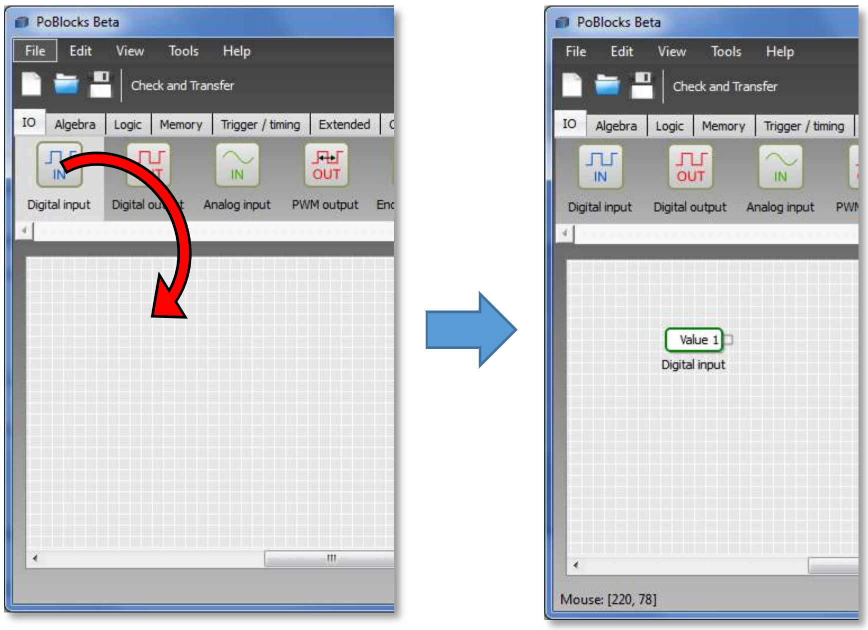
Target device: a drop-box menu used to select the device type that will be used to execute the diagram's code

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.

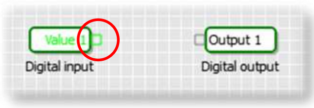
8.4. Copying blocks

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.

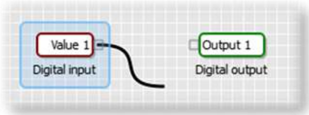
8.5. Connecting blocks

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.

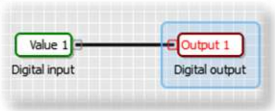
Start connecting the blocks by clicking on one of the empty ports (a port that can accept a connection will be highlighted in green). You can then either drag the connection to another port and release the mouse button or release the button immediately and click again on the destination block's port.



1. Start at source port



2. Drag the connection

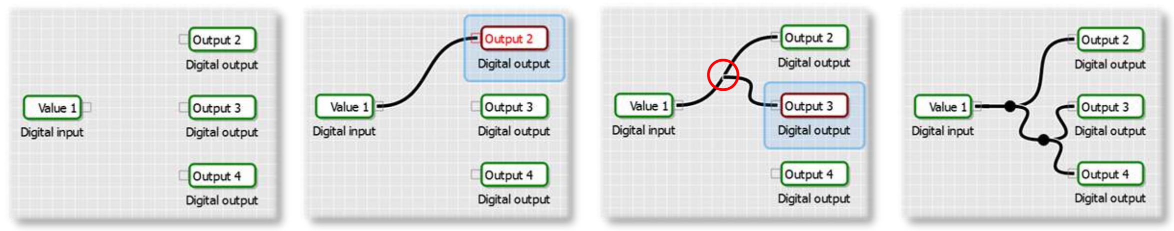


3. End at destination block

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

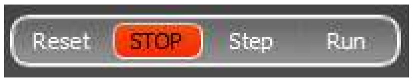
After PoBlocks diagram is finished, a built-in compiler is used to translate it into PoIL code that is then transferred and executed on a PoKeys device. During compiling, the diagram is checked for errors and if any are detected, a pop-up error list window will appear and compiler process will be stopped.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

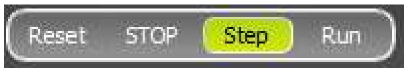
- The blocks are first put into a proper execution sequence based on their function and how they are connected to other blocks. PoIL core executes code of one block after the other and the proper sequence is required to obtain correct result. If the diagram consists of algebraic loops (closed loops containing 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)

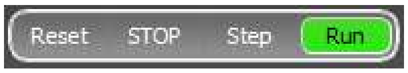
When a PoKeys device is selected in the 'Device selection' dialog, an operating mode of the PoIL core is automatically displayed in device controls as a highlighted item.



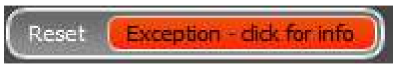
STOP: PoIL core is in STOPPED state, no PoIL code is being executed



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



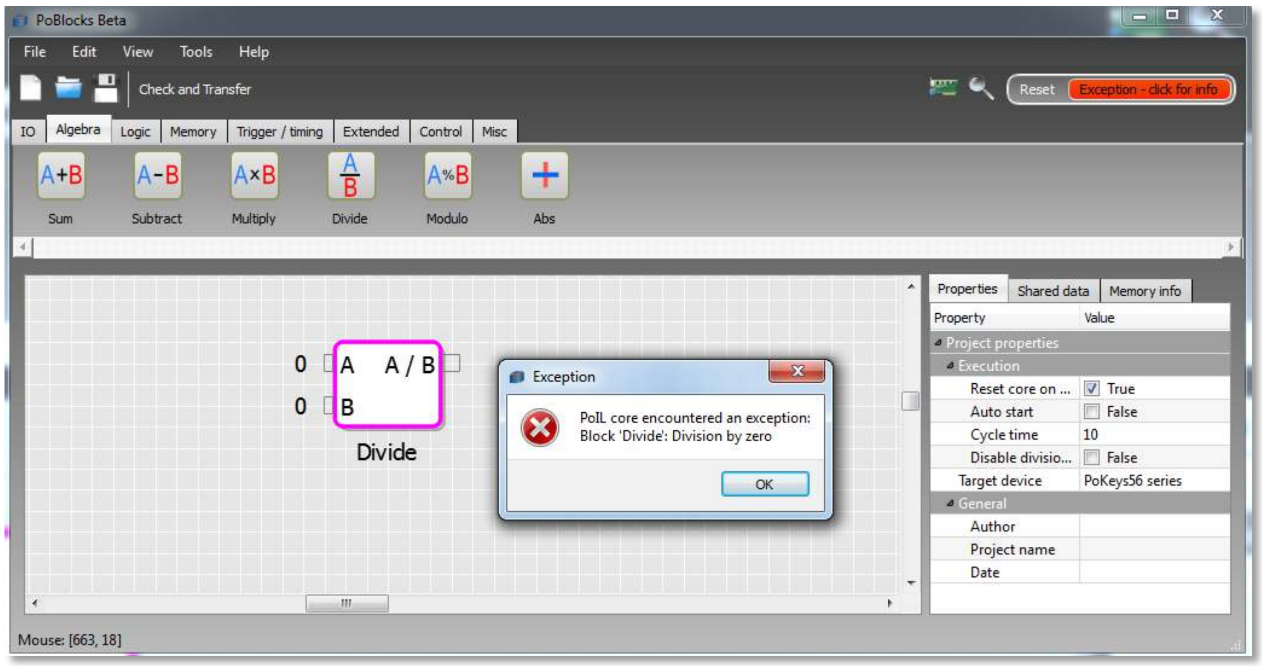
Run: PoIL core is running normally



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.



Exception

PoIL core encountered an exception:
Block 'Divide': Division by zero

OK

Property	Value
Project properties	
Execution	
Reset core on ...	☑ True
Auto start	☐ False
Cycle time	10
Disable divisio...	☐ False
Target device	PoKeys56 series
General	
Author	
Project name	
Date	

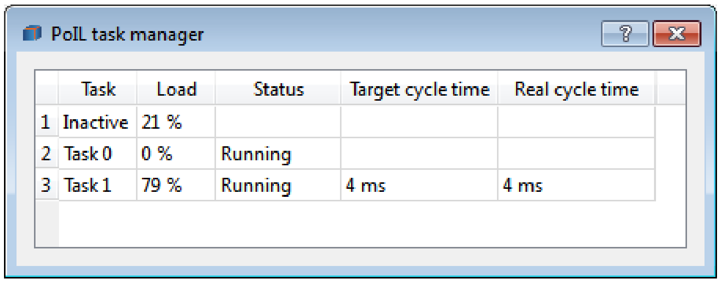
12. Monitor mode

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.

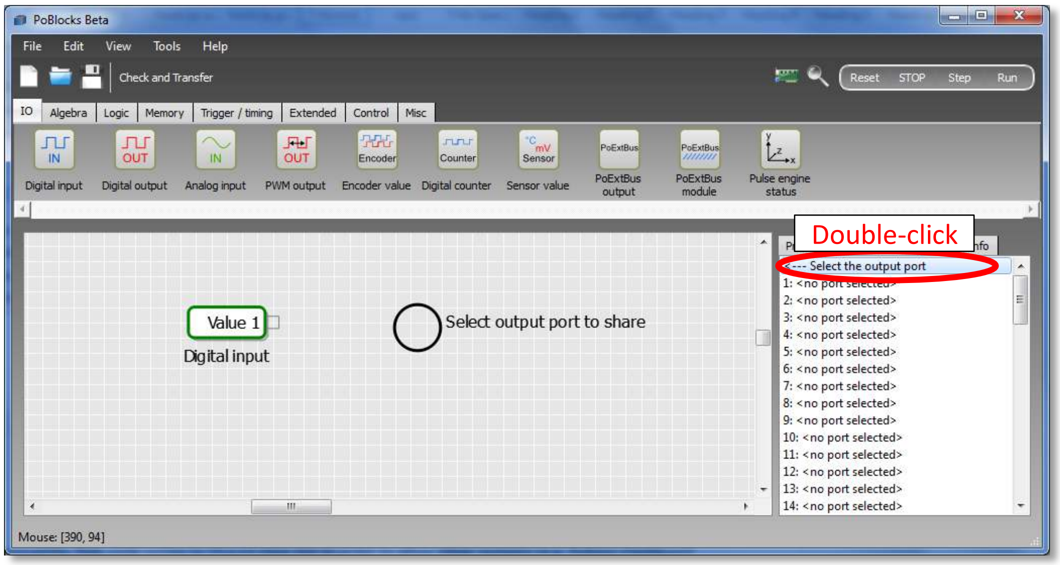
Last two columns show the 'Target cycle time' (set in the project settings) and 'Real cycle time' (true/real cycle time managed by the device).

14. Shared data

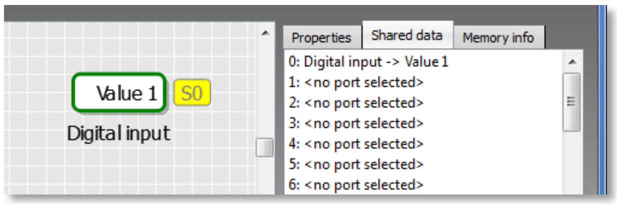
In order to share data between PoIL code and other PoKeys systems (Web dashboard and Modbus interface on Ethernet PoKeys devices, third-party applications on other devices or computers, connected to PoKeys, etc.) a shared data slots were introduced in PoKeys devices with PoIL core support. Each Shared 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).

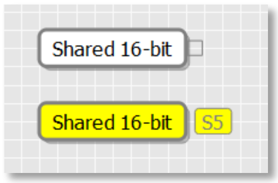


Shared slot ID (displayed as a shared slot index in the list and on every port with shared slot assigned to) is used to refer to this slot from other systems.

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.

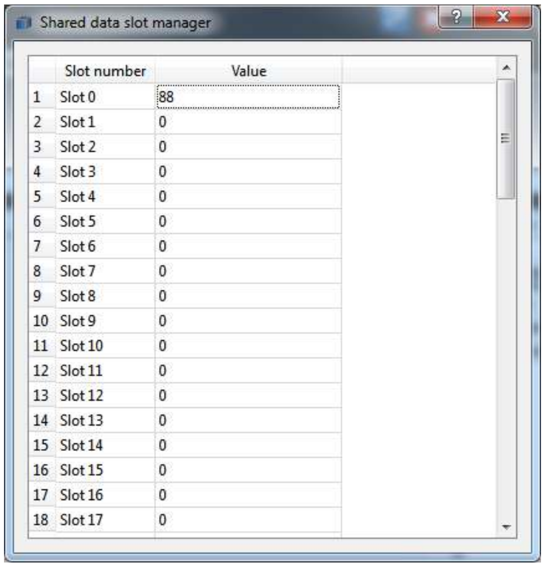
14.2. Reading data from shared data slots

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).



	Slot number	Value
1	Slot 0	88
2	Slot 1	0
3	Slot 2	0
4	Slot 3	0
5	Slot 4	0
6	Slot 5	0
7	Slot 6	0
8	Slot 7	0
9	Slot 8	0
10	Slot 9	0
11	Slot 10	0
12	Slot 11	0
13	Slot 12	0
14	Slot 13	0
15	Slot 14	0
16	Slot 15	0
17	Slot 16	0
18	Slot 17	0

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

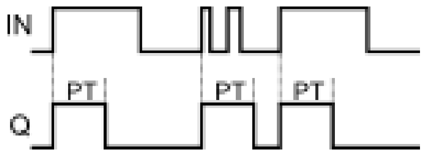
Description	Down counter decrements the counter on the *Clock* input positive (rising) changes from value, specified by *PV* input, to 0
Inputs	***Clock*** (Logic): Counter pulse signal
	Load PV (Logic): When 1, counter value *Value* is loaded with value of *PV*
	PV (32-bit integer): Initial counter 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

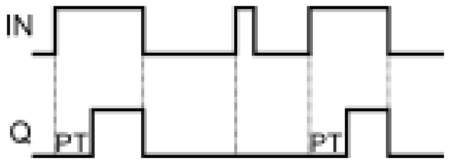
Description	Up/down counter increments the counter vaue on the *Clock up* input positive (rising) changes and decrements the counter on the *Clock down* input positive (rising) changes. The counter value *Value* is limited to the range between (including) 0 to *PV*
Inputs	***Clock up*** (Logic): Counter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

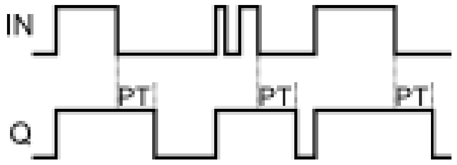
Description	Pulse timer triggers on *IN* high input signal and turns off after the period defined by *PT*, uneffected by the *IN* signal state during this period as illustrated below
Inputs	***IN*** (Logic): Timer activation signal input

	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

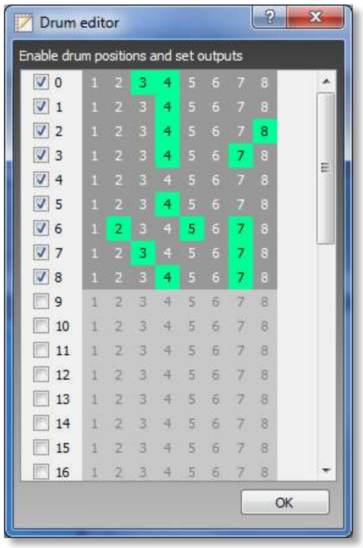
Description	Off timer or (off delay timer) activates on *IN* high input signal state. The timer starts counting on the *IN* input signal falling edge and turns off after the period defined by *PT*. The counter is reset if the input signal *IN* goes to high state and the timer stays activated as illustrated 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.



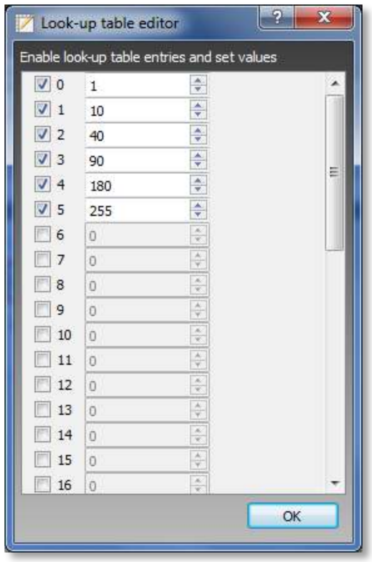
Drum editor

Enable drum positions and set outputs

☑ 0	1	2	🟩 3	🟩 4	5	6	7	8
☑ 1	1	2	3	🟩 4	5	6	7	8
☑ 2	1	2	3	🟩 4	5	6	7	🟩 8
☑ 3	1	2	3	🟩 4	5	6	🟩 7	8
☑ 4	1	2	3	4	5	6	7	8
☑ 5	1	2	3	🟩 4	5	6	7	8
☑ 6	1	🟩 2	3	4	🟩 5	6	🟩 7	8
☑ 7	1	2	🟩 3	4	5	6	🟩 7	8
☑ 8	1	2	3	🟩 4	5	6	🟩 7	8
☐ 9	1	2	3	4	5	6	7	8
☐ 10	1	2	3	4	5	6	7	8
☐ 11	1	2	3	4	5	6	7	8
☐ 12	1	2	3	4	5	6	7	8
☐ 13	1	2	3	4	5	6	7	8
☐ 14	1	2	3	4	5	6	7	8
☐ 15	1	2	3	4	5	6	7	8
☐ 16	1	2	3	4	5	6	7	8

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.



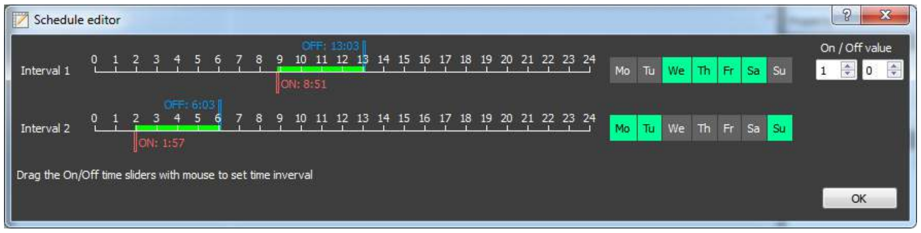
✓	0	1
✓	1	10
✓	2	40
✓	3	90
✓	4	180
✓	5	255
☐	6	0
☐	7	0
☐	8	0
☐	9	0
☐	10	0
☐	11	0
☐	12	0
☐	13	0
☐	14	0
☐	15	0
☐	16	0

16.56. Extended - Schedule

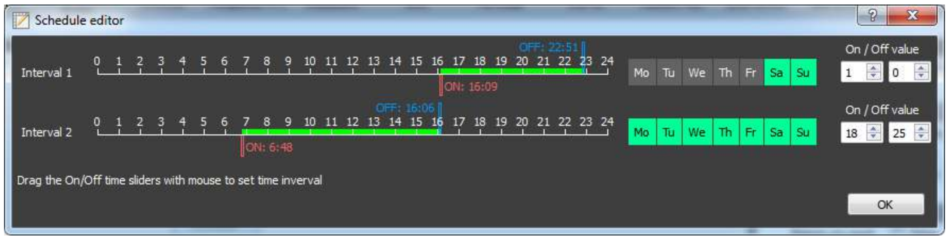
Description	Schedule is used to setup activation schedule by configuring multiple intervals. To setup the intervals, double click on the block in the scheme.
Inputs	None
Outputs	***Out*** (8-bit integer): Activation signal ***Out 1 - 10*** (8-bit integer): Interval-specific optional ouputs. ***Rule 1 - 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

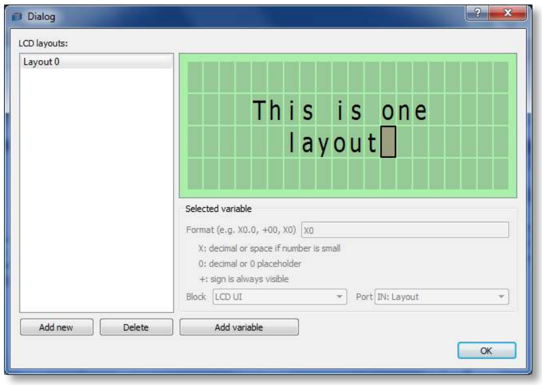
output values) content.
LCD UI block must be stimulated on the refresh input in order to refresh the contents of the LCD. LCD refresh is a time-consuming operation and is not done by PoBlocks or PoIL code by itself – user has to define a proper refresh time using clock signal block or generate a signal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

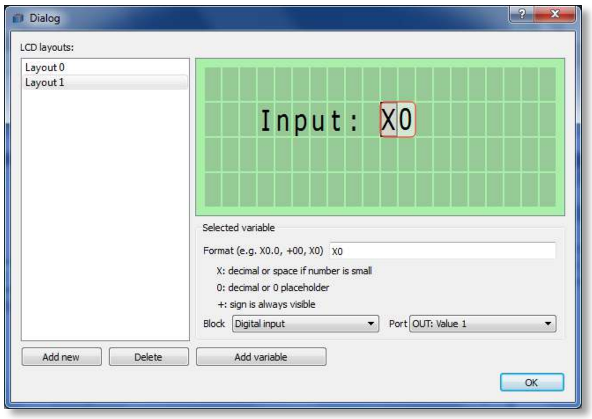
Properties ***Use secondary pins*** (Logic): If True, LCD will be initialized on secondary pins as described in PoKeys manual
LCD rows (Integer, 1 to 4): Number of rows in the LCD display
LCD columns (Integer, 1 to 20): Number of columns in the LCD display

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.



Format (e.g. X0.0, +00, X0) X0
X: decimal or space if number is small
0: decimal or 0 placeholder
+: sign is always visible
Block LCD UI
Port IN: Layout

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.



Selected variable

Format (e.g. X0.0, +00, X0) X0

X: decimal or space if number is small

0: decimal or 0 placeholder

+: sign is always visible

Block Digital input

Port OUT: Value 1

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^{2}C read operation block
Inputs	***!CLK*** (Logic/clock): Inverted clock input signal
	Address (8-bit integer): I^{2}C 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^{2}C 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^{2}C write operation block
Inputs	***!CLK*** (Logic/clock): Inverted clock input signal
	Address (8-bit integer): I^{2}C 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^{2}C 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

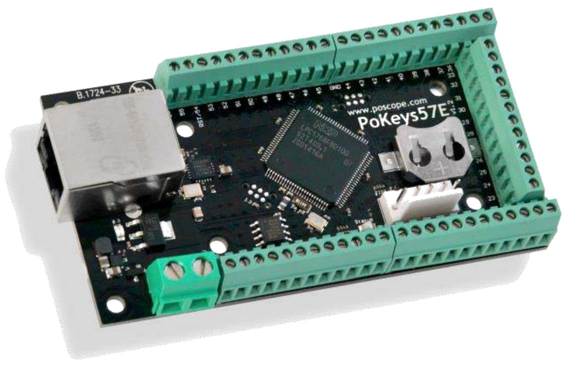
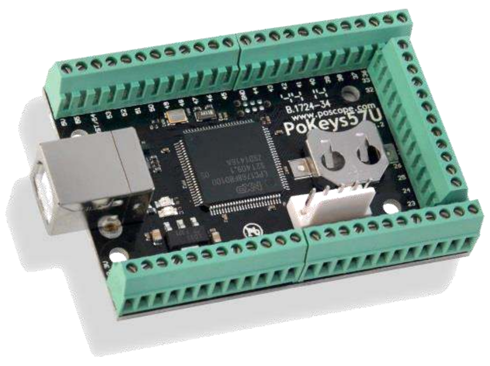
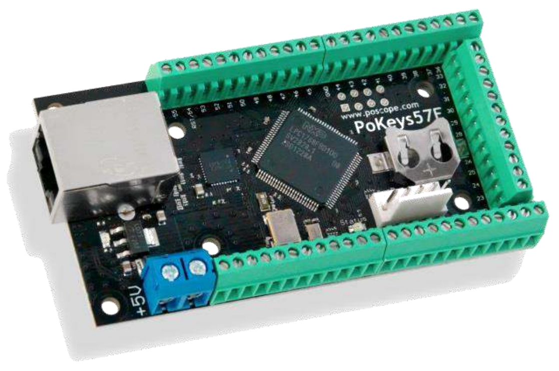
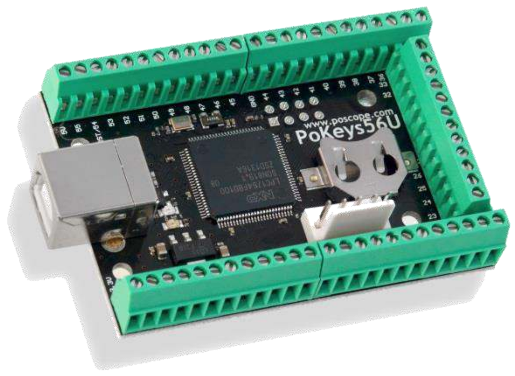
We provide upgrades, free of charge, from our web site at www.poscope.com. We reserve the right to charge for updates or replacements sent out on physical media.

Trademarks

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

User's manual

Version: 21/12/2016

Please read the following notes

Contents

1. Description

PoKeys products line consists of simple, easy-to-use USB and network devices with the extended list of features making them powerful input/output devices. PoKeys55, PoKeys56U and PoKeys57U devices also incorporate a virtual USB keyboard and joystick with a simple setup. PoKeys devices enable user to design specially built robust computer interfaces. The devices are highly adjustable and as such require no complex knowledge on device programming. Virtual USB keyboard supports emulation of single key presses or various programmable series of keys (keyboard macro sequences) while virtual USB joystick emulation supports mapping of analog inputs to joystick axes and mapping of digital inputs to joystick buttons. All peripherals can be tested via included software with highly intuitive graphical user interface. Chosen settings can be stored on device, so no special software is needed on target system.

If additional input and output capabilities are needed, the devices provide 55 digital 5V tolerant inputs or outputs, 5 10-bit analog inputs (on PoKeys55) / 7 12-bit analog inputs (on PoKeys56 and PoKeys57 series devices) with adjustable software low-pass filter and one 10-bit analog output (on PoKeys55). These are complemented with 6 high-speed fully configurable PWM (pulse width modulation) outputs. User can freely set PWM period and PWM duty cycles. PWM module runs at 12 MHz (on PoKeys55) / 25 MHz (on PoKeys56 and PoKeys57 series devices) and allows high-speed output switching.

PoKeys devices also support up to 25 pairs of quadrature encoder signal inputs for which can be freely connected to any of the 55 inputs on PoKeys devices. These inputs increment or decrement the counters that can be read via provided software of other third-party applications. Changes in those signals can trigger virtual keyboard presses on USB versions of PoKeys devices. PoKeys devices also support three additional fast encoders, when higher frequencies of the quadrature encoder signals are in use. PoKeys56 and PoKeys57 series devices add additional support for one ultra-fast quadrature encoder signal that can handle even greater frequencies of these signals with ease.

In the cases when the application requires more than 55 inputs or outputs, PoKeys devices have the inbuilt support for matrix keyboards sized up to 16x8 with freely assignable key codes for virtual USB keyboard, while the PoExtBus bus feature adds the support for additional 80 digital outputs.

PoKeys devices also support hd44780-compatible character LCD displays and two serially driven 8x8 matrix LED displays.

Third-party application developers that are adding the support for PoKeys devices, are encouraged to use the supplied communication DLL that can be simply used in the different .NET framework based applications and various other programming languages that provide support for ActiveX interface. There is even an open-source cross-platform C library available at https://bitbucket.org/mbosnak/pokeyslib.

To aid developers that are communicating with PoKeys devices on the low-level, the extensive documentation on device communication protocol can be downloaded free of charge from the product webpage.

2. Features

2.1. PoKeys55

- Compatible with USB 1.1/2.0 HID standard
- Standard English USB keyboard simulation (with triggering support for up/down keys)
- Standard USB joystick simulation (6 axis, 32 buttons with triggering support)
- 55 digital inputs with pull-up resistors, freely mappable to virtual USB keyboard's keys
- 55 software controlled digital outputs
- 5 analog inputs (10-bit), freely mappable to any of virtual USB joystick axes (with adjustable low-pass filtering support)
- 1 software controlled 10-bit analog output, controlled via included software
- Up to 25 encoder pair inputs
- Up to 64 256-character long keyboard macro sequences
- Up to 16x8 matrix keyboard with triggered keys/alternate function support
- Two 8x8 matrix LED display support
- Up to 6 high-speed fully configurable PWM outputs support (12 MHz PWM timer)
- HD44780-based character LCD support (up to 4x20 characters)
- PoExtBus support for adding up to 10 external shift registers
- Support for Connection signal output
- Intuitive and user-friendly software
- Third-party support via communication DLL library and extensive protocol specification document that allows porting to other systems

2.2. PoKeys56U / PoKeys57U

- Compatible with USB 1.1/2.0 HID standard
- Standard English USB keyboard simulation (with triggering support for up/down keys)
- Standard USB joystick simulation (6 axis, 32 buttons with triggering support)
- 55 digital inputs with pull-up resistors, freely mappable to virtual USB keyboard's keys
- 55 software controlled digital outputs
- 7 analog inputs (12-bit) with adjustable low-pass filtering support
- Up to 26 encoder pair inputs (3 high-speed encoder inputs, 1 ultra high speed encoder input)
- Digital counters on specific digital input pins
- 3-axis 25 kHz or 8-axis 125 kHz highly customizable Pulse engine v2 with safety 5 kHz charge-pump output
- Two 8x8 matrix LED display support
- Up to 64 256-character long keyboard macro sequences
- Up to 16x8 matrix keyboard with triggered keys/alternate function support
- Two 8x8 matrix LED display support
- Up to 6 high-speed fully configurable PWM outputs support (25MHz PWM timer)
- HD44780-based character LCD support (up to 4x20 characters)
- PoExtBus support for adding up to 10 external shift registers
- PoNET devices support (48-key CNC keyboard mapped to matrix keyboard)
- Support for Connection signal output
- Fail-safe support in case of communication interruption

- Support for up to 10 sensors on I^{2}C bus, up to 10 sensors on 1-wire bus and up to 7 analog sensors (PoKeys56) or up to 100 sensors on I^{2}C, 1-wire buses (PoKeys57)
- Communication compatible with PoKeys55 on application layer (data packet structure)
- Intuitive and user-friendly software
- Third-party support via communication DLL library and extensive protocol specification document that allows porting to other systems

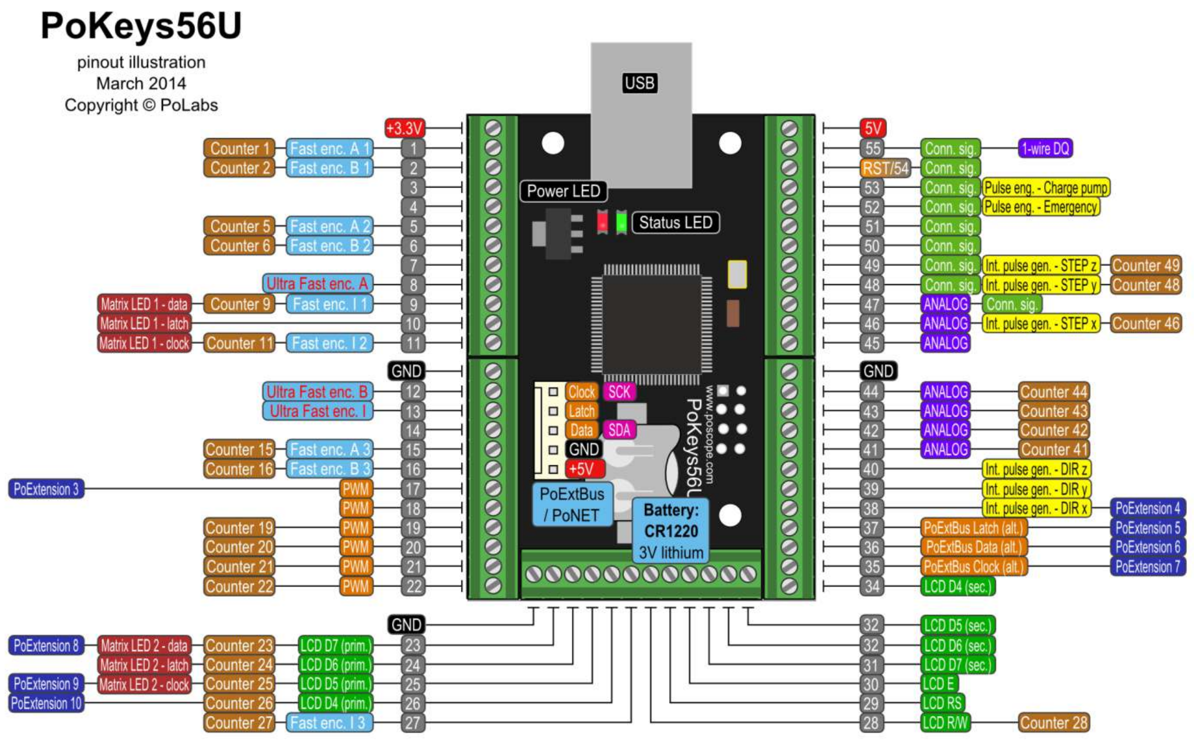
2.3. PoKeys56E / PoKeys57E

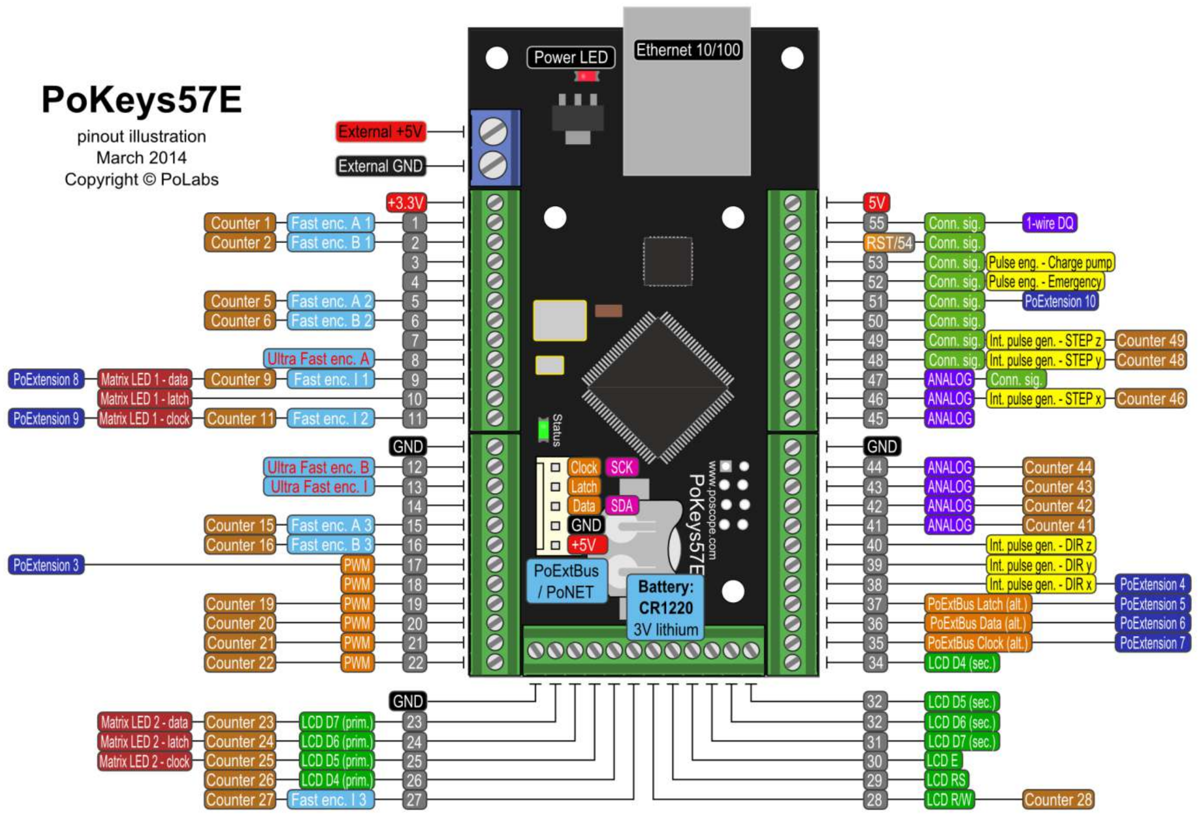
- Ethernet 10/100 with DHCP client or fixed IP support
- TCP connection with device
- 55 digital inputs with pull-up resistors
- 55 software controlled digital outputs
- 7 analog inputs (12-bit) with adjustable low-pass filtering support
- Up to 26 encoder pair inputs (3 high-speed encoder inputs, 1 ultra high speed encoder input)
- 3-axis 25 kHz or 8-axis 125 kHz highly customizable Pulse engine v2 with safety 5 kHz charge-pump output
- Digital counters on specific digital input pins
- Up to 16x8 matrix keyboard
- Two 8x8 matrix LED display support
- Up to 6 high-speed fully configurable PWM outputs support (25 MHz PWM timer)
- HD44780-based character LCD support (up to 4x20 characters)
- PoExtBus support for adding up to 10 external shift registers
- PoNET devices support (48-key CNC keyboard)
- Modbus TCP support (access to digital IO, analog inputs, encoders' counters, digital counters values, PWM outputs, LCD display, LED matrix display, PoExtBus devices, matrix keyboard status)
- Support for up to 10 sensors on I^{2}C bus, up to 10 sensors on 1-wire bus and up to 7 analog sensors (PoKeys56) or up to 100 sensors on I^{2}C, 1-wire buses (PoKeys57)
- Web interface with newly designed dashboard and I/O status display with multiple user accounts
- Support for communication with devices on I^{2}C and 1-wire buses
- Support for Connection signal output
- Fail-safe support in case of communication interruption
- Communication compatible with PoKeys55 on application layer (data packet structure)
- Intuitive and user-friendly software
- Third-party support via communication DLL library and extensive protocol specification document that allows porting to other systems

3. Device comparison

	PoKeys55	PoKeys56U/57U	PoKeys56E/57E
Number of pins on the board	55	55	55
Digital inputs	55	55	55
Digital outputs	55	55	55
Analog inputs	5x 10-bit	7x 12-bit	7x 12-bit
Analog ouputs	1x 10-bit	0	0
Number of encoders (normal)	25	25	25
Number of fast encoders	3	3	3
Number of ultra fast encoders	0	1	1
Number of PWM outputs	6 (12 MHz clock)	6 (25 MHz clock)	6 (25 MHz clock)
Number of digital counters	0	24	24
Pulse engine	not suported	int. 3-axis 25 kHz ext. 8-axis 125 kHz	int. 3-axis 25 kHz ext. 8-axis 125 kHz
LCD support	Alphanumeric up to 4x20	Alphanumeric up to 4x20	Alphanumeric up to 4x20
Matrix keyboard support	Up to 16x8	Up to 16x8	Up to 16x8
Matrix LED	Two 8x8	Two 8x8	Two 8x8
Keyboard emulation	Yes	Yes	No
Joystick emulation	Yes	Yes	No
Connectivity	USB	USB	Ethernet (IPv4 + DHCP)
Modbus TCP support	No	No	Yes
I2C sensors	No	Yes	Yes
1-wire sensors	No	Yes	Yes
PoExtBus support	up to 80 outputs	up to 80 outputs	up to 80 outputs
PoNET bus support	No	up to 16 devices	up to 16 devices
Power supply	powered from USB	powered from USB	external power supply
Number of devices per computer	127 per USB root hub	127 per USB root hub	practically unlimited
Automatic device discovery	Yes	Yes	Yes (in local network)
Free configuration application	Yes	Yes	Yes
Free firmware upgrades	Yes	Yes	Yes
Communication DLL support	Yes	Yes	Yes
Availability	Out of production	Available	Available

4. Device hardware description


PoKeys56U
pinout illustration
March 2014
Copyright © PoLabs
USB
Power LED
Status LED
+3.3V
Counter 1 Fast enc. A 1 1
Counter 2 Fast enc. B 1 2
3
4
Counter 5 Fast enc. A 2 5
Counter 6 Fast enc. B 2 6
7
Ultra Fast enc. A 8
Matrix LED 1 - data Counter 9 Fast enc. I 1 9
Matrix LED 1 - latch 10
Matrix LED 1 - clock Counter 11 Fast enc. I 2 11
GND
Ultra Fast enc. B 12
Ultra Fast enc. I 13
14
Counter 15 Fast enc. A 3 15
Counter 16 Fast enc. B 3 16
PoExtension 3 PWM 17
PWM 18
Counter 19 PWM 19
Counter 20 PWM 20
Counter 21 PWM 21
Counter 22 PWM 22
GND
PoExtension 8 Matrix LED 2 - data Counter 23 LCD D7 (prim.) 23
Matrix LED 2 - latch Counter 24 LCD D6 (prim.) 24
PoExtension 9 Matrix LED 2 - clock Counter 25 LCD D5 (prim.) 25
PoExtension 10 Counter 26 LCD D4 (prim.) 26
Counter 27 Fast enc. I 3 27
5V
55 Conn. sig. 1-wire DQ
RST/54 Conn. sig.
53 Conn. sig. Pulse eng. - Charge pump
52 Conn. sig. Pulse eng. - Emergency
51 Conn. sig.
50 Conn. sig.
49 Conn. sig. Int. pulse gen. - STEP z Counter 49
48 Conn. sig. Int. pulse gen. - STEP y Counter 48
47 ANALOG Conn. sig.
46 ANALOG Int. pulse gen. - STEP x Counter 46
45 ANALOG
GND
44 ANALOG Counter 44
43 ANALOG Counter 43
42 ANALOG Counter 42
41 ANALOG Counter 41
40 Int. pulse gen. - DIR z
39 Int. pulse gen. - DIR y
38 Int. pulse gen. - DIR x PoExtension 4
37 PoExtBus Latch (alt.) PoExtension 5
36 PoExtBus Data (alt.) PoExtension 6
35 PoExtBus Clock (alt.) PoExtension 7
34 LCD D4 (sec.)
32 LCD D5 (sec.)
32 LCD D6 (sec.)
31 LCD D7 (sec.)
30 LCD E
29 LCD RS
28 LCD R/W Counter 28
Clock SCK
Latch
Data SDA
GND
+5V
PoExtBus / PoNET
Battery: CR1220 3V lithium
www.poscope.com
PoKeys56U


PoKeys57E
pinout illustration
March 2014
Copyright © PoLabs
Power LED
Ethernet 10/100
External +5V
External GND
+3.3V
Counter 1 Fast enc. A 1 1
Counter 2 Fast enc. B 1 2
3
4
Counter 5 Fast enc. A 2 5
Counter 6 Fast enc. B 2 6
7
Ultra Fast enc. A 8
PoExtension 8 Matrix LED 1 - data Counter 9 Fast enc. I 1 9
Matrix LED 1 - latch 10
PoExtension 9 Matrix LED 1 - clock Counter 11 Fast enc. I 2 11
GND
Ultra Fast enc. B 12
Ultra Fast enc. I 13
14
Counter 15 Fast enc. A 3 15
Counter 16 Fast enc. B 3 16
PoExtension 3 PWM 17
PWM 18
Counter 19 PWM 19
Counter 20 PWM 20
Counter 21 PWM 21
Counter 22 PWM 22
GND
Matrix LED 2 - data Counter 23 LCD D7 (prim.) 23
Matrix LED 2 - latch Counter 24 LCD D6 (prim.) 24
Matrix LED 2 - clock Counter 25 LCD D5 (prim.) 25
Counter 26 LCD D4 (prim.) 26
Counter 27 Fast enc. I 3 27
5V
55 Conn. sig. 1-wire DQ
RST/54 Conn. sig.
53 Conn. sig. Pulse eng. - Charge pump
52 Conn. sig. Pulse eng. - Emergency
51 Conn. sig. PoExtension 10
50 Conn. sig.
49 Conn. sig. Int. pulse gen. - STEP z Counter 49
48 Conn. sig. Int. pulse gen. - STEP y Counter 48
47 ANALOG Conn. sig.
46 ANALOG Int. pulse gen. - STEP x Counter 46
45 ANALOG
GND
44 ANALOG Counter 44
43 ANALOG Counter 43
42 ANALOG Counter 42
41 ANALOG Counter 41
40 Int. pulse gen. - DIR z
39 Int. pulse gen. - DIR y
38 Int. pulse gen. - DIR x PoExtension 4
37 PoExtBus Latch (alt.) PoExtension 5
36 PoExtBus Data (alt.) PoExtension 6
35 PoExtBus Clock (alt.) PoExtension 7
34 LCD D4 (sec.)
32 LCD D5 (sec.)
32 LCD D6 (sec.)
31 LCD D7 (sec.)
30 LCD E
29 LCD RS
28 LCD R/W Counter 28
Status
Clock SCK
Latch
Data SDA
GND
+5V
PoExtBus / PoNET
Battery: CR1220 3V lithium
www.poscope.com
PoKeys57E

I = input, O = output, I/O input or output, SF = special function

Pin	Type	Description
1	I/O	General purpose digital input/output pin (limited current source capability on PoKeys56E/57E)
	I	**Fast encoder 1A** – A channel input for the fast encoder 1
	I	**Counter 1** – Counter input signal for counter 1 (not available on PoKeys55)
2	I/O	General purpose digital input/output pin (limited current source capability on PoKeys56E/57E)
	I	**Fast encoder 1B** – B channel input for the fast encoder 1
	I	**Counter 2** – Counter input signal for counter 2 (not available on PoKeys55)
3	I/O	General purpose digital input/output pin
4	I/O	General purpose digital input/output pin
5	I/O	General purpose digital input/output pin (when used as output on PoKeys56E/57E, pin 6 must also be specified as output)
	I	**Fast encoder 2A** – A channel input for the fast encoder 2
	I	**Counter 5** – Counter input signal for counter 5 (not available on PoKeys55)
6	I/O	General purpose digital input/output pin (when used as output on PoKeys56E/57E, pin 5 must also be specified as output)
	I	**Fast encoder 2B** – B channel input for the fast encoder 2
	I	**Counter 6** – Counter input signal for counter 6 (not available on PoKeys55)
7	I/O	General purpose digital input/output pin
8	I/O	General purpose digital input/output pin
	I	**Ultra fast encoder input A** (not available on PoKeys55)
9	I/O	General purpose digital input/output pin
	SF	PoKeys56E/57E: external pulse generator - see Pulse engine manual
	O	**Matrix LED 1** serial data output
	I	**Counter 9** – Counter input signal for counter 9 (not available on PoKeys55)
10	I/O	General purpose digital input/output pin
	O	**Matrix LED 1** output latch clock output
11	I/O	General purpose digital input/output pin
	SF	PoKeys56E/57E: external pulse generator - see Pulse engine manual
	O	**Matrix LED 1** serial clock output
	I	**Counter 11** – Counter input signal for counter 11 (not available on PoKeys55)
12	I/O	General purpose digital input/output pin
	I	**Ultra fast encoder input B** (not available on PoKeys55)
13	I/O	General purpose digital input/output pin
	I	**Ultra fast encoder index** signal input (not available on PoKeys55)
14	I/O	General purpose digital input/output pin
15	I/O	General purpose digital input/output pin
	I	**Fast encoder 3A** – A channel input for the fast encoder 3
	I	**Counter 15** – Counter input signal for counter 15 (not available on PoKeys55)
16	I/O	General purpose digital input/output pin
	I	**Fast encoder 3B** – B channel input for the fast encoder 3
	I	**Counter 16** – Counter input signal for counter 16 (not available on PoKeys55)
17	I/O	General purpose digital input/output pin
	O	**PWM channel 6** output

18	I/O	General purpose digital input/output pin
	O	**PWM channel 5** output
19	I/O	General purpose digital input/output pin
	O	**PWM channel 4** output
	I	**Counter 19** – Counter input signal for counter 19 (not available on PoKeys55)
20	I/O	General purpose digital input/output pin
	O	**PWM channel 3** output
	I	**Counter 20** – Counter input signal for counter 20 (not available on PoKeys55)
21	I/O	General purpose digital input/output pin
	O	**PWM channel 2** output
	I	**Counter 21** – Counter input signal for counter 21 (not available on PoKeys55)
22	I/O	General purpose digital input/output pin
	O	**PWM channel 1** output
	I	**Counter 22** – Counter input signal for counter 22 (not available on PoKeys55)
23	I/O	General purpose digital input/output pin
	I/O	**LCD data 7** (primary configuration)
	O	**Matrix LED 2** serial data output
	SF	PoKeys56U: external pulse generator - see Pulse engine manual
	I	**Counter 23** – Counter input signal for counter 23 (not available on PoKeys55)
24	I/O	General purpose digital input/output pin
	I/O	**LCD data 6** (primary configuration)
	O	**Matrix LED 2** output latch clock output
	I	**Counter 24** – Counter input signal for counter 24 (not available on PoKeys55)
25	I/O	General purpose digital input/output pin
	I/O	**LCD data 5** (primary configuration)
	O	**Matrix LED 2** serial clock output
	SF	PoKeys56U: external pulse generator - see Pulse engine manual
	I	**Counter 25** – Counter input signal for counter 25 (not available on PoKeys55)
26	I/O	General purpose digital input/output pin
	I/O	**LCD data 4** (primary configuration)
	SF	PoKeys56U: external pulse generator - see Pulse engine manual
	I	**Counter 26** – Counter input signal for counter 26 (not available on PoKeys55)
27	I/O	General purpose digital input/output pin
	I	**Counter 27** – Counter input signal for counter 27 (not available on PoKeys55)
28	I/O	General purpose digital input/output pin
	O	**LCD R/W** (primary and secondary configuration)
	I	**Counter 28** – Counter input signal for counter 28 (not available on PoKeys55)
29	I/O	General purpose digital input/output pin
	O	**LCD RS** (primary and secondary configuration)
30	I/O	General purpose digital input/output pin
	O	**LCD E** (primary and secondary configuration)
31	I/O	General purpose digital input/output pin
	I/O	**LCD data 7** (secondary configuration)
32	I/O	General purpose digital input/output pin
	I/O	**LCD data 6** (secondary configuration)
33	I/O	General purpose digital input/output pin

	I/O	**LCD data 5** (secondary configuration)
34	I/O	General purpose digital input/output pin
	I/O	**LCD data 4** (secondary configuration)
35	I/O	General purpose digital input/output pin
	SF	External pulse generator dedicated IO connection - see Pulse engine manual
	O	**PoExtBus** serial clock if PoExtBus on dedicated connector is not used
36	I/O	General purpose digital input/output pin
	SF	External pulse generator dedicated IO connection - see Pulse engine manual
	O	**PoExtBus** serial data if PoExtBus on dedicated connector is not used
37	I/O	General purpose digital input/output pin
	SF	External pulse generator dedicated IO connection - see Pulse engine manual
	O	**PoExtBus** output latch clock if PoExtBus on dedicated connector is not used
38	I/O	General purpose digital input/output pin
	SF	External pulse generator dedicated IO connection - see Pulse engine manual
	O	Pulse engine **DIR x** - x-axis direction output (not available on PoKeys55)
39	I/O	General purpose digital input/output pin
	O	Pulse engine **DIR y** - y-axis direction output (not available on PoKeys55)
40	I/O	General purpose digital input/output pin
	O	Pulse engine **DIR z** - z-axis direction output (not available on PoKeys55)
41	I/O	General purpose digital input/output pin
	I	**Analog input 41** (not available on PoKeys55)
	I	**Counter 41** – Counter input signal for counter 41 (not available on PoKeys55)
42	I/O	General purpose digital input/output pin
	I	**Analog input 42** (not available on PoKeys55)
	I	**Counter 42** – Counter input signal for counter 42 (not available on PoKeys55)
43	I/O	General purpose digital input/output pin
	I	**Analog input 43**
	I	**Counter 43** – Counter input signal for counter 43 (not available on PoKeys55)
44	I/O	General purpose digital input/output pin
	I	**Analog input 44**
	I	**Counter 44** – Counter input signal for counter 44 (not available on PoKeys55)
45	I/O	General purpose digital input/output pin
	I	**Analog input 45**
46	I/O	General purpose digital input/output pin
	I	**Analog input 46**
	I	**Counter 46** – Counter input signal for counter 46 (not available on PoKeys55)
	O	Integrated Pulse engine **STEP x** - x-axis step output (not available on PoKeys55)
47	I/O	General purpose digital input/output pin
	I	**Analog input 47**
	O	Connection signal output
48	I/O	General purpose digital input/output pin (limited current source capability on PoKeys56U)
	I	**Counter 48** – Counter input signal for counter 48 (not available on PoKeys55)
	O	Integrated Pulse engine **STEP y** - y-axis step output (requires additional 470 Ohm pull-up resistor on PoKeys56U) (not available on PoKeys55)
	O	Connection signal output

49	I/O	General purpose digital input/output pin (limited current source capability on PoKeys56U)
	I	**Counter 49** – Counter input signal for counter 49 (not available on PoKeys55)
	O	Integrated Pulse engine **STEP z** - z-axis step output (requires additional 470 Ohm pull-up resistor on PoKeys56U) (not available on PoKeys55)
	O	Connection signal output
50	I/O	General purpose digital input/output pin
	O	Connection signal output
51	I/O	General purpose digital input/output pin
	SF	PoKeys56E/57E: external pulse generator - see Pulse engine manual
	O	Connection signal output
52	I/O	General purpose digital input/output pin
	I	Pulse engine **EMGERGENCY** - emergency switch input (not available on PoKeys55)
	O	Connection signal output
53	I/O	General purpose digital input/output pin
	O	Pulse engine **CHARGEPUMP** - 5 kHz charge pump signal output (not available on PoKeys55)
	O	Connection signal output
54	I/O	General purpose digital input/output pin
	I	Recovery mode (**RST**) signal - PoKeys enters recovery mode if this signal is tied to GND at startup
	O	Connection signal output
55	I/O	General purpose digital input/output pin
	I/O	**1-wire** communication bus (not available on PoKeys55)
	O	Connection signal output

4.1. PoKeys56U/E pin functions graphical overview

Pin	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
DI	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
DO	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
AI	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
AO	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Fast encoders	●	●	○	○	●	●	○	○	○	○	○	○	○	○	●	●	○	○	○	○	○	○	○	○	○	○	○	○
Ultra fast encoders	○	○	○	○	○	○	○	●	○	○	○	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
LCD	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	●	●	●	●	○	●
LED	○	○	○	○	○	○	○	○	●	●	●	○	○	○	○	○	○	○	○	○	○	○	●	●	●	○	○	○
Auxilary output	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
PWM	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	●	●	●	●	●	●	○	○	○	○	○	○
Connection signal	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Configuration reset	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Counters	●	●	○	○	●	●	○	○	●	○	●	○	○	○	●	●	○	○	●	●	●	●	●	●	●	●	●	●
Special functions	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
1-wire	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

Pin	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55
DI	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
DO	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
AI	○	○	○	○	○	○	○	○	○	○	○	○	●	●	●	●	●	●	●	○	○	○	○	○	○	○	○
AO	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Fast encoders	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Ultra fast encoders	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
LCD	●	●	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
LED	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Auxilary output	○	○	○	○	○	○	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
PWM	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Connection signal	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	●	●	●	●	●	●	●	●	●
Configuration reset	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	●	○
Counters	○	○	○	○	○	○	○	○	○	○	○	○	●	●	●	●	○	●	○	●	●	○	○	○	○	○	○
Special functions	○	○	○	●	●	●	●	●	●	●	●	●	○	○	○	○	○	●	●	●	●	●	●	●	●	●	○
1-wire	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	●

○ not supported
● supported

4.2. PoKeys55 pin functions graphical overview **(serial numbers above 11500)**

Pin	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
DI	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
DO	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
AI	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
AO	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Fast encoders	●	●	○	○	●	●	○	○	○	○	○	○	○	○	●	●	○	○
LCD	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
LED	○	○	○	○	○	○	○	○	●	●	●	○	○	○	○	○	○	○
Auxilary output	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
PWM	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	●	●
Connection signal	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Configuration reset	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Special functions	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

Pin	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
DI	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
DO	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
AI	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
AO	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Fast encoders	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
LCD	○	○	○	○	●	●	●	●	○	●	●	●	●	●	●	●	○	○	○
LED	○	○	○	○	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○
Auxilary output	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	●	●	●
PWM	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Connection signal	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Configuration reset	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Special functions	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

Pin	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55
DI	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
DO	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
AI	○	○	○	○	○	●	●	●	●	●	○	○	○	○	○	○	○	○
AO	○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	○
Fast encoders	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
LCD	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
LED	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Auxilary output	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
PWM	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Connection signal	○	○	○	○	○	○	○	○	○	●	●	●	●	●	●	●	●	●
Configuration reset	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	●	○
Special functions	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	●	○

○	not supported
●	supported

4.3. Status LEDs

Each PoKeys devices has the following status LEDs:

- Red LED – power status: when the power is applied to PoKeys device, this LED is lit
- Green LED – connection/communication status: this LED reflects various operating modes of the PoKeys device
    - Fast blinking on startup: device has started in the recovery mode. This enables to update the firmware and clear the settings in the device.
    - Slow blinking on startup (PoKeys56E/57E): PoKeys device is waiting for DHCP settings
    - Solid: device is in normal operation mode

5. Requirements

5.1. USB devices

- One available USB 1.1 or USB 2.0 port
- USB HID device driver enabled operating system (Windows 98 SE/ME/2000/XP/Vista/Windows 7/8/10, Linux, Mac OS)
- Included software requires Windows XP/Vista/Windows 7/Windows 8/Windows 10 with .NET frameworks 2.0 and 3.5 installed (ONLY FOR SYSTEMS WHERE THE DEVICES WILL BE CONFIGURED, TARGET SYSTEM NEEDS NO SOFTWARE INSTALLATION FOR THE DEVICE TO OPERATE AS A STANDARD USB KEYBOARD AND JOYSTICK).

5.2. Network devices

PoKeys57E v1.2

- Ethernet connection between host computer and PoKeys57E device
- 5-12 V DC power supply with 400 mA current capability. If any additional peripheral is connected to PoKeys57E, use appropriately more powerful power supply.
- Included software requires Windows XP/Vista/Windows 7/Windows 8/Windows 10 with .NET frameworks 2.0 and 3.5 installed (ONLY FOR SYSTEMS WHERE THE DEVICES WILL BE CONFIGURED, TARGET SYSTEM NEEDS NO SOFTWARE INSTALLATION FOR THE DEVICE TO OPERATE AS A STANDARD NETWORK DEVICE).

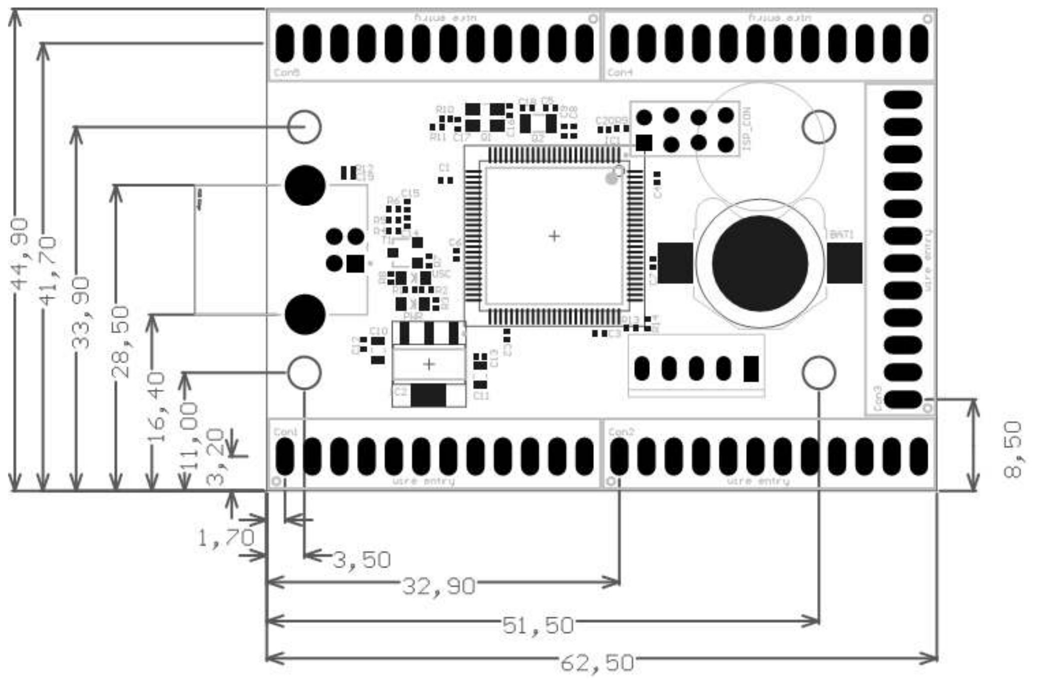
PoKeys56E/PoKeys57E v1.1

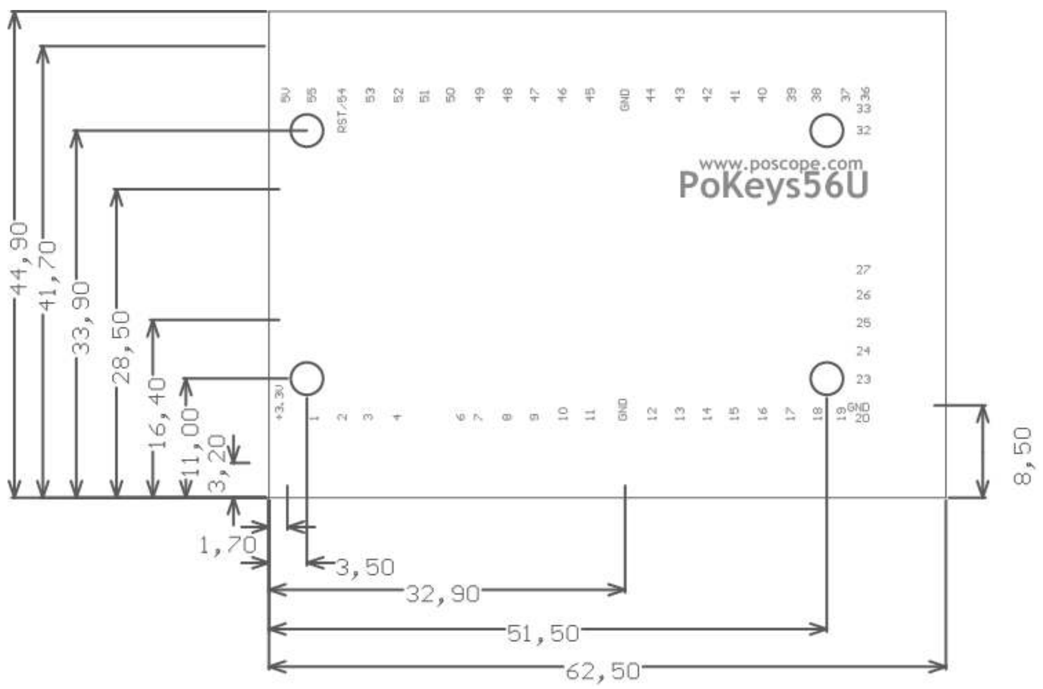
- Ethernet connection between host computer and PoKeys56E device
- 5V DC power supply with 400 mA. If any additional peripheral is connected to PoKeys56E, use appropriately more powerful power supply.
- Included software requires Windows XP/Vista/Windows 7/Windows 8/Windows 10 with .NET frameworks 2.0 and 3.5 installed (ONLY FOR SYSTEMS WHERE THE DEVICES WILL BE CONFIGURED, TARGET SYSTEM NEEDS NO SOFTWARE INSTALLATION FOR THE DEVICE TO OPERATE AS A STANDARD NETWORK DEVICE).

6. Technical specifications

6.1. PoKeys55, PoKeys56U and PoKeys57U dimensions

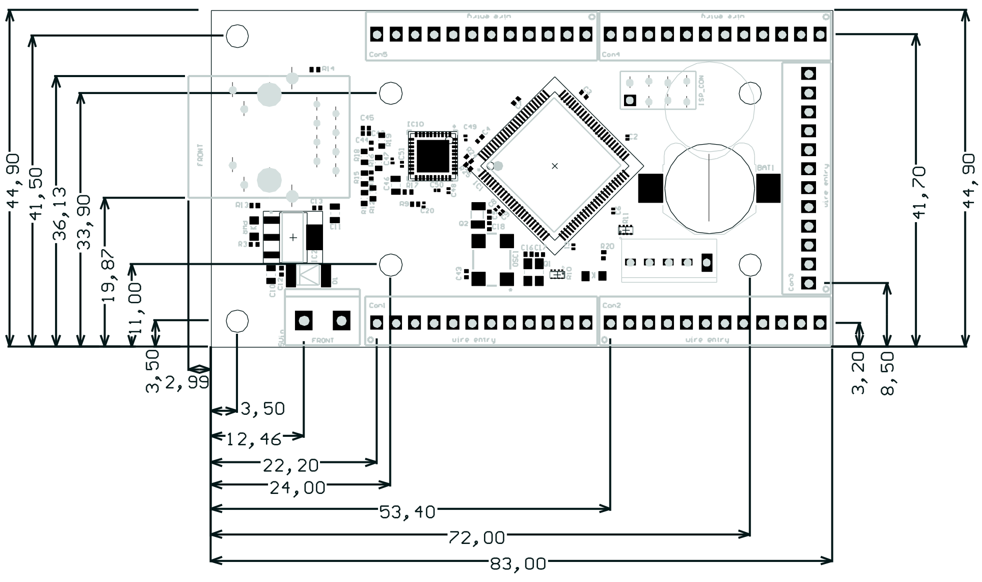
Measurements are in mm, all holes have 3mm diameter.

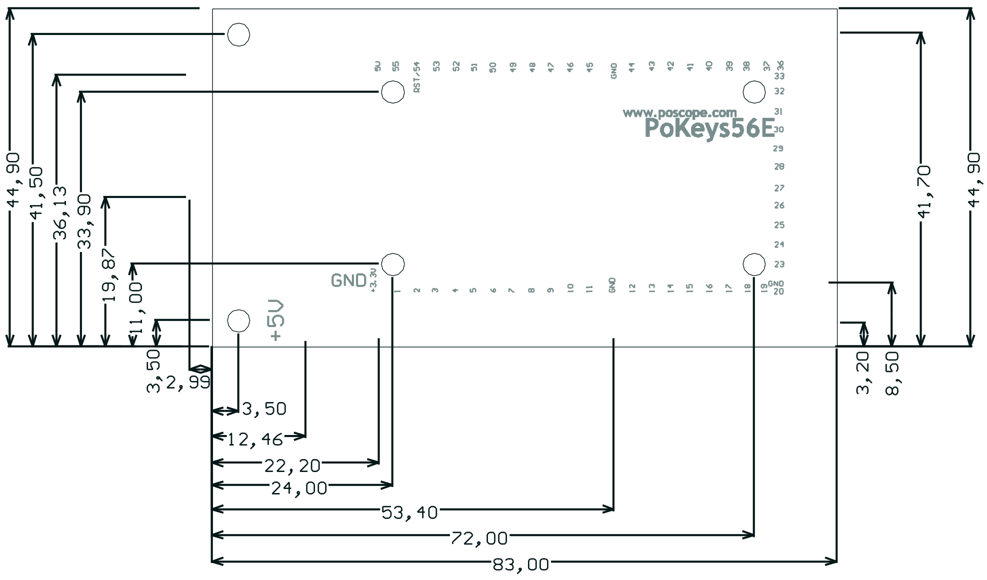




6.2. PoKeys56E/57E dimensions

Measurements are in mm, all holes have 3mm diameter.





6.3. Electrical specification – limiting values

Symbol	Parameter	Min	Max	Unit
V_{IA}	analog input voltage on ADC related pins	-0.5	3.6	V
V_I	Input voltage on other pins	-0.5	5.5	V
V_{esd}	electrostatic discharge	-4000	4000	V

6.4. Electrical specification – static characteristic

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_I	input voltage	pin configured as digital input	0	-	5.5	V
V_O	output voltage	pin configured as digital output	0	-	3.3	V
V_{IH}	HIGH-level input voltage		2.0	-	-	V
V_{I_analog}	analog input voltage	pin configured as analog input	0	-	3.3	V
V_{O_analog}	analog output voltage	pin configured as analog output	0	-	3.3	V
V_{IL}	LOW-level input voltage		-	-	0.8	V
V_{hys}	hysteresis voltage		0.4	-	-	V
V_{OH}	HIGH-level output voltage	I_{OH} = -4 mA	2.9	-	-	V
V_{OL}	LOW-level output voltage	I_{OH} = 4 mA	-	-	0.4	V
I_{OH}	HIGH-level output current	V_{OH} = 2.9 V	-4	-	-	mA
I_{OL}	LOW-level output current	V_{OL} = 0.4 V	4	-	-	mA
I_{OHS}	HIGH-level short-circuit output current	V_{OH} = 0 V	-	-	-45	mA
I_{OLS}	LOW-level short-circuit output current	V_{OL} = 3.3V	-	-	50	mA
I_{pu}	pull-up current		0 (V_I >= 3.3V)	-50	-100	µA
I_{5V}	5V pin source current				400	mA
I_{5V}	Power supply current for PoKeys57E v1.2 **without peripherals**	U_{in} = 5 V U_{in} = 12 V	100 50	120 60	150 75	mA mA
I_{5V}	5V power supply current for PoKeys56E and PoKeys57E v1.1 **without peripherals**		250	300	400	mA
$I_{3.3V}$	Maximum current of 3.3V power supply for external devices		-	-	120	mA
V_{BATT}	Lithium battery for RTC functionality (CR1220)		-	3	-	V

6.5. Environment specifications

Parameter	Min	Typ	Max	Unit
Power supply range				
- PoKeys56U, *56E, *57U	4.5	-	5.5	V
- PoKeys57E v1.2	4.5	-	15	V
Operating temperature	0	-	60	°C
Storage temperature	-40	-	85	°C
Humidity	5	-	95 (non-condensing)	% RH

6.6. Power supply for PoKeys57E v1.2

PoKeys57E requires external 5-12 V power supply to be connected to the board. PoKeys57E uses switching power supply to power the 5 V power supply bus (also available on pin +5V for external peripherals that require +5V). If any additional peripheral is connected to PoKeys57E board, combined current of peripheral and PoKeys57E board must be taken into consideration when selecting the power supply.

Take care when selecting proper power supply for PoKeys57E. The power supply may never exceed the voltage of +15 V or the PoKeys57E can get damaged.

6.7. Power supply for PoKeys56E (and PoKeys57E v1.1)

PoKeys56E requires external 5V power supply to be connected to the board. If any additional peripheral is connected to PoKeys56E board, combined current of peripheral and PoKeys56E board must be taken into consideration when selecting the power supply.

Due to the protection diode in the circuit, a 0.8 V voltage drop can be observed between input voltage and 5V pin. When supplying the PoKeys56E board with 5V power supply, take care when connecting peripherals that require 5V supply voltage (e.g., LCD displays).

7. Installation

PoKeys55 and PoKeys56U are USB 1.1/2.0 compliant devices and as such require no additional drivers for operation as a standard USB keyboard and joystick.

PoKeys56E/57E is an Ethernet device that requires an Ethernet connection between host computer and the device and the external power supply. No additional drivers are required. Network firewalls must allow all traffic on TCP/UDP port 20055.

To operate the device after the device has been configured there is no software installation necessary on a target system.

To configure the device the supplied software must be installed and the requirements listed in previous section of this manual must be met.

8. PoKeys configuration options

8.1. Digital inputs and outputs

Any of the 55 pins, available on the terminals on the PoKeys PCB, can be configured as digital input or output with selectable polarity. All pins have a weak pull-up resistor enabled and are 5V tollerant.

At every startup, all digital input or output configured pins are preset as digital input pins (with the weak pull-up resistor pulling the state of the pin to logical 1). To enable output function on a selected pin, SetPinFunction command must be executed. This behaviour can be disabled using the SetAutoSetOutputs command or via PoKeys configuration software (Settings > Initialize outputs on startup). Each output pin can sink or source up to 4 mA of current, with the limitation that the pins combined source or sink current does not exceed 100 mA.

All digital input pins on PoKeys55 or PoKeys56U can be configured to simulate a configurable USB keyboard key. When there is a high state on pin (on low state when using inverted polarity option) PoKeys device sends a USB message with the key code and modifier associated with this pin. Moreover, PoKeys device can simulate a series of key presses, what is called a macro sequence. Up to 64 different macro sequences can be setup with the combined total length of 3584 characters with each macro sequence shorter than 128 keys. All macros can be labelled with a 7-character name. Character codes (most frequently used are listed in the Appendix to this document) are USB HID standard keyboard codes.

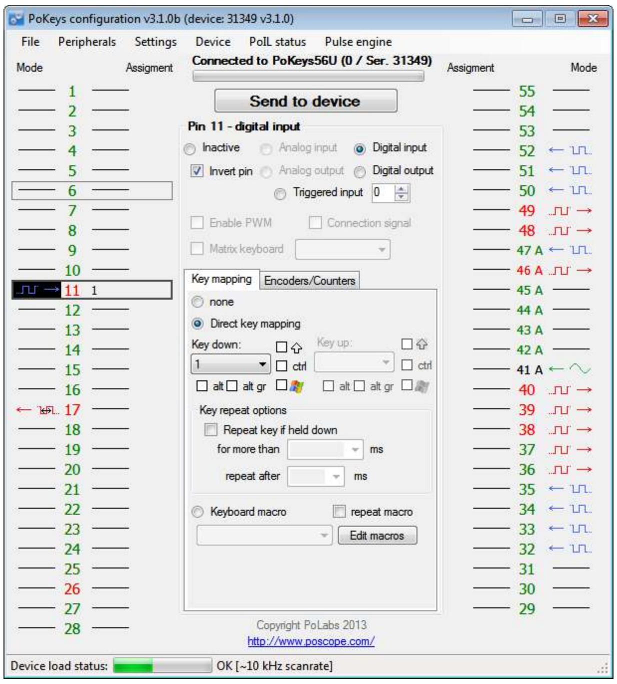
An extension to the USB keyboard mapping described above, PoKeys supports also triggered mapping of inputs to USB keyboard keys. In triggered mapping mode, only pin state transitions (low-to-high or high-to-low trigger a USB key press) with different key combinations for each transition.

Type-matic like repeat and delay is an additional extension to the triggered key mapping. Instead of relying on the user's system to trigger key repeat events, PoKeys can be configured to simulate repeated key presses at the predefined rate (period between two key presses is adjustable in 5 ms cycles – 0.78 to 200 repeats possible[1]) after a predefined delay (adjustable in steps of 5 ms – 0 to 1275 ms possible).

PoKeys configuration software usage

There is graphical representation for configuration of each PoKeys device's pin on left and right side of main window. To change pin function, click on pin name and change its function in central 'Pin settings' frame.

[1] The maximum repeat rate depends on the user's system


PoKeys configuration v3.1.0b (device: 31349 v3.1.0)

File Peripherals Settings Device PoIL status Pulse engine

Connected to PoKeys56U (0 / Ser. 31349)

Send to device

Pin 11 - digital input

Inactive | Analog input | [✓ Digital input]

[✓ Invert pin] | Analog output | Digital output

Triggered input 0

Enable PWM | Connection signal

Matrix keyboard

Key mapping | Encoders/Counters

none | [✓ Direct key mapping]

Key down: 1 | ctrl | alt | alt gr — Key up: ctrl | alt | alt gr

Key repeat options: Repeat key if held down; for more than ms; repeat after ms

Keyboard macro | repeat macro | Edit macros

Copyright PoLabs 2013 http://www.poscope.com/

Device load status: OK [~10 kHz scanrate]

Figure 1: PoKeys configuration window

There are 6 main pin functions possible: inactive, digital input, triggered digital input, digital output, analog input and analog output.

Inactive

Any pin (except those fixly mapped to an activated peripheral) can be set as inactive. Inactive pin is put in high-Z state with internal pull-up resistors enabled.

Digital input

Any one of the 55 pins can be configured as digital input by selecting ‘Digital input’ option box. If the pin polarity is wished to be inverted, check the 'Invert pin' box.

There are several additional possibilities for digital input pin functions.

Direct key mapping – only for USB devices

Digital input set up for direct key mapping acts like a keyboard key. When there is a high state on pin (on low state when using inverted option), PoKeys55/PoKeys56U sends a key associated with this pin. Select a keyboard key from drop-down box and check appropriate key modifiers (Shift, Ctrl, …).

Example: Send Alt-F4

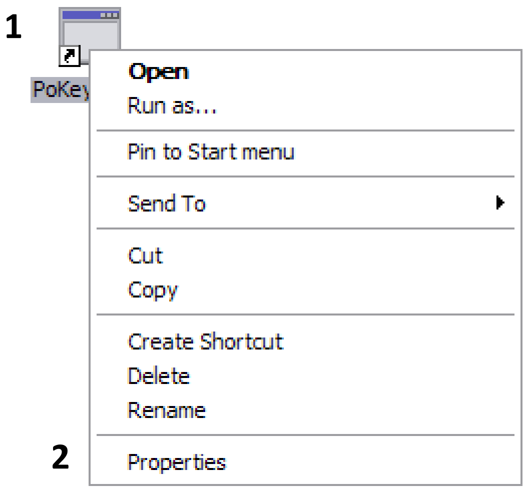
Select F4 from drop-down box and check Alt checkbox.

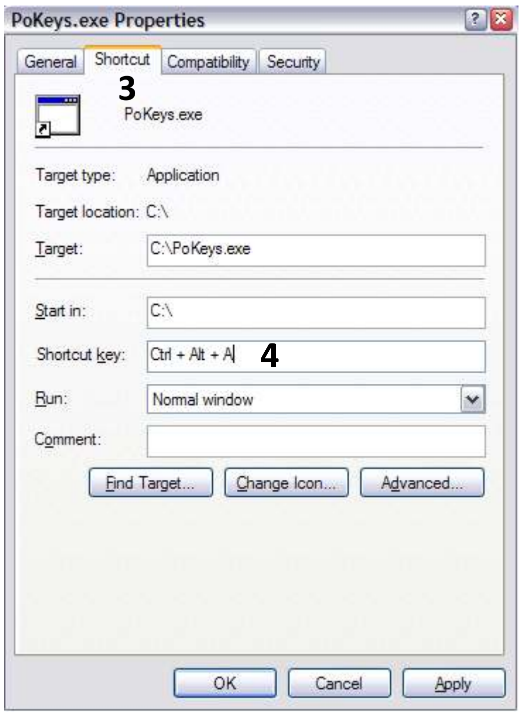
Example: Send ((opening bracket)

This key kombination differs from your system regional settings. As the PoKeys55/PoKeys56U devices emulate a system keyboard, key associations depend on current sytem keyboard regional setting. To send an opening bracket symbol, one possible solution is to press Shift-8 (in most non-English countries) or to press Shift-9. Out of this reason there are no such secondary keys listed in drop-down box and must be entered by user as described above.

Example: starting a program on Windows using PoKeys55/PoKeys56U device

On a Windows operating system, users can assign a custom shortcut key to any program shortcut. Find the shortcut and then right click on it to show the context menu (Step 1). Select Properties (Step 2), and under the Shortcut tab (Step 3), click on the 'Shortcut key' text box. Proceed by typing in a combination that you wish to assign to a particular program (Step 4). Next, open the PoKeys application and connect to the desired PoKeys device. Click on the pin that will function as a launch trigger for your application (Step 5). Under Key mapping, select the same keyboard combination that you assigned to the program shortcut (Step 6). Click on the 'Send to device' button (Step 7) to transfer settings to the device. This will activate the new shortcut.





PoKeys.exe

Target type: Application

Target location: C:\

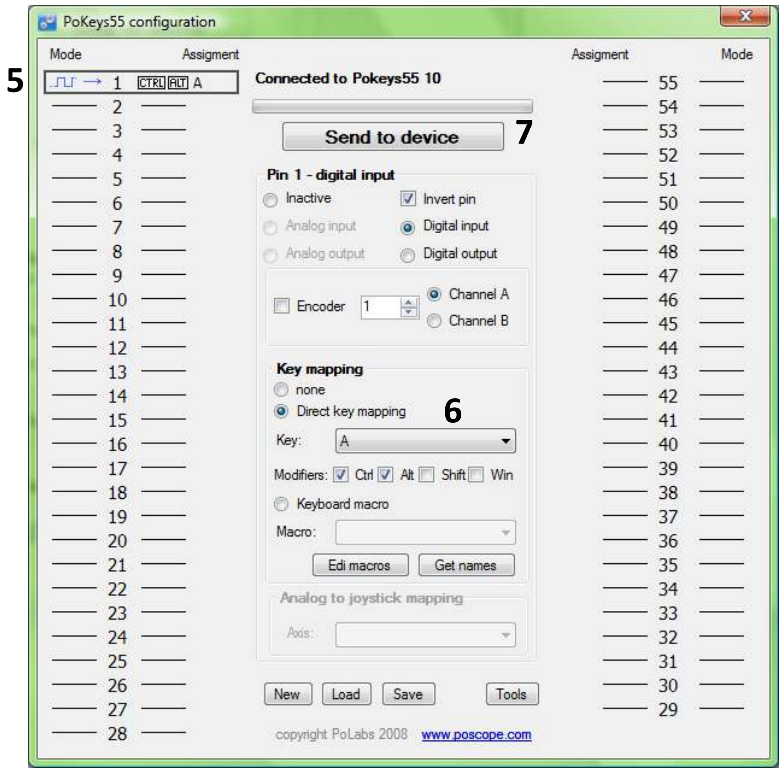
Target: C:\PoKeys.exe

Start in: C:\

Shortcut key: Ctrl + Alt + A

Run: Normal window

Comment:



Connected to Pokeys55 10

Pin 1 - digital input

[✓ Invert pin]

[✓ Digital input]

Encoder: 1

[✓ Direct key mapping]

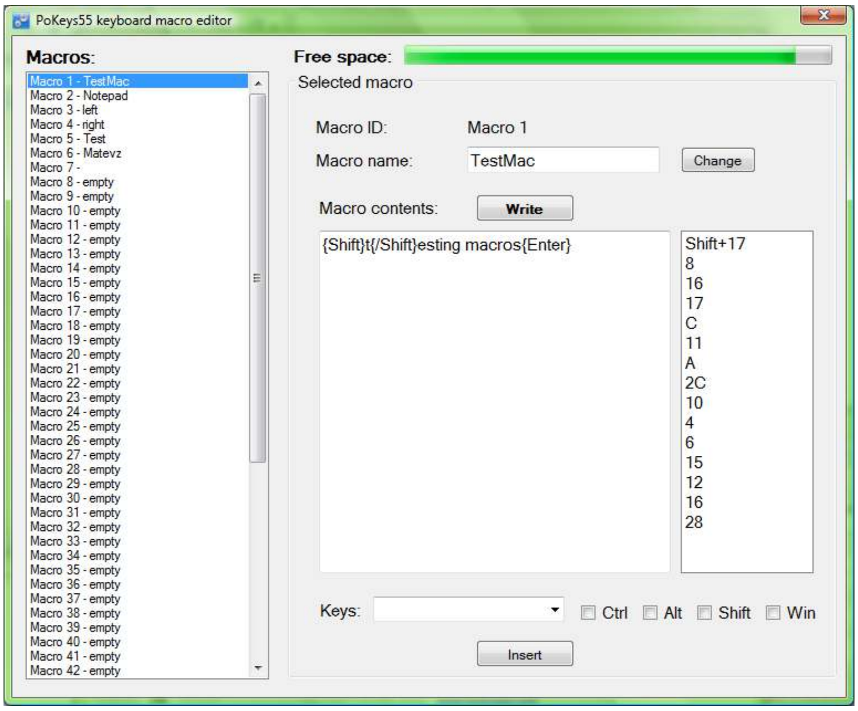
Key: A

Modifiers: [✓ Ctrl] [✓ Alt] Shift Win

Figure 2: Setting up PoKeys device

Keyboard macro – only for USB devices

PoKeys55 and PoKeys56U devices support keyboard macros – the key press combinations that can be up to 256 keys long. To define a keyboard macro, first select Keyboard macro mapping option for one of the pins. 'Edit macros' and 'Get names' command buttons become enabled. To add, change or delete macros click the 'Edit macros' button. The following dialog appears



Macro 1 - TestMac
Macro 2 - Notepad
Macro 3 - left
Macro 4 - right
Macro 5 - Test
Macro 6 - Matevz
Macro 7 -
Macro 8 - empty
Macro 9 - empty
Macro 10 - empty
Macro 11 - empty
Macro 12 - empty
Macro 13 - empty
Macro 14 - empty
Macro 15 - empty
Macro 16 - empty
Macro 17 - empty
Macro 18 - empty
Macro 19 - empty
Macro 20 - empty
Macro 21 - empty
Macro 22 - empty
Macro 23 - empty
Macro 24 - empty
Macro 25 - empty
Macro 26 - empty
Macro 27 - empty
Macro 28 - empty
Macro 29 - empty
Macro 30 - empty
Macro 31 - empty
Macro 32 - empty
Macro 33 - empty
Macro 34 - empty
Macro 35 - empty
Macro 36 - empty
Macro 37 - empty
Macro 38 - empty
Macro 39 - empty
Macro 40 - empty
Macro 41 - empty
Macro 42 - empty

Macro ID: Macro 1
Macro name: TestMac
Macro contents:

```
{Shift}t{/Shift}esting macros{Enter}
```

```
Shift+17
8
16
17
C
11
A
2C
10
4
6
15
12
16
28
```

Figure 3: Macro editing dialog

First select the macro you want to edit. To change macro name, enter desired macro name (up to 7 characters long) in 'Macro name' text box and click 'Change' button. This name is used only to help user differentiate between multiple macros.

To set macro contents, simply enter text into 'Macro contents' text box. If there is an invalid character found, the text appears red. When finished, click Write to write macro to device.

List box at the right displays final macro sequence that is sent to PC when the macro gets activated.

Triggered digital input – only for USB devices

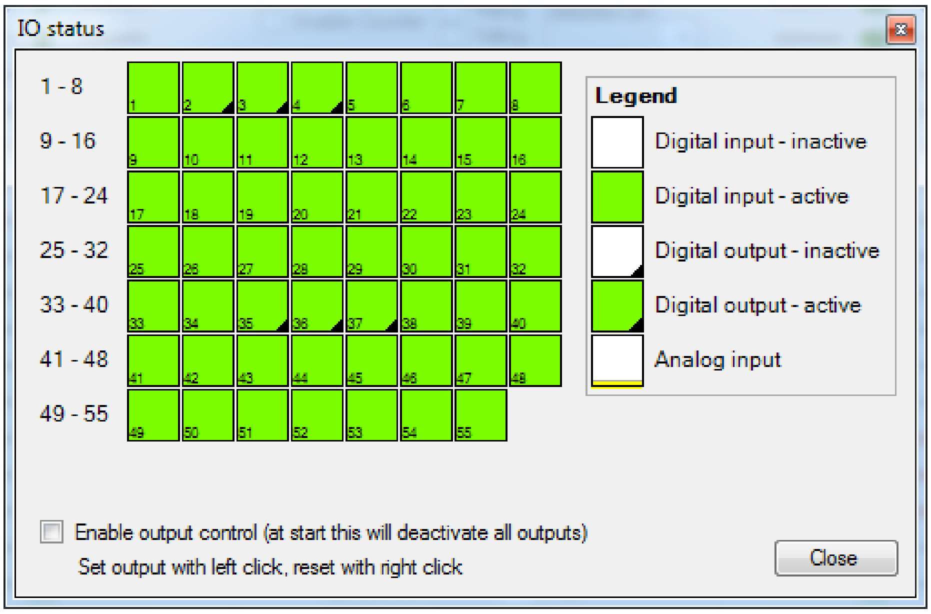
Any one of the 55 pins can be configured as triggered digital input by selecting ‘Triggered input’ option box. This pin mode enables user to select a key that is pressed only when a transition in a signal occurs. Different keys can be selected for ‘LOW-to-HIGH’ or ‘Key up’ event and for ‘HIGH-to-LOW’ or ‘Key down’ event. Selecting the keys is similar to Direct key mapping described above.

Digital output

Any one of the 55 pins can be configured as digital output by selecting ‘Digital output’ option box. If the polarity of the pin is wished to be inverted, check the 'Invert pin' box. On startup, all pins (although optionally configured as digital output) are by default initialized in high-Z state (behaving like inputs). To use the outputs, SetPinData should be called before attempting to set the output state. However, there is an option to disable this behaviour - use the ‘Settings > Initialize outputs on startup’ option to either enable or disable output activation on PoKeys startup.

View status of digital inputs and outputs

Go to ‘Peripherals > Digital inputs and outputs…’ to display the status dialog as shown below.



Enable output control (at start this will deactivate all outputs)

Set output with left click, reset with right click

Figure 4: Input and output status dialog

There are 55 pins represented as colored squares in the dialog, organized in the rows of 8 pins. Each square contains a pin index number in the lower left corner, while the lower right corner is used to indicate a digital output (small black triangle is displayed on pins, configured as digital outputs). The color of the square resembles the current state of the pin – green for the activated (HIGH state) and white for the unactivated (LOW state).

To change the digital output state, first enable 'Enable output control' option, then either left or right click with mouse on the square representing the digital output to activate or deactivate this output.

Applicable PoKeys library commands

Read or set single pin state	
GetInput SetOutput	Reads current state of digital input Sets digital output state
Read or set single pin function	
GetPinData SetPinData	Reads pin function Set pin function
Read or set single pin key or macro mapping	
GetPinKeyMapping SetPinKeyMapping	Reads pin key mapping Sets pin key mapping options
GetPinMacroMapping SetPinMacroMapping	Reads pin macro mapping Set pin macro mappping options
Read or set triggered input mapping	
GetTriggeredInputMapping SetTriggeredInputMapping	Reads triggered input mapping options Sets triggered input mapping options
Read or set type-matic like delays and rates	
GetTypeMaticDelaysAndRates SetTypeMaticDelaysAndRates	Gets the typematic delay and repeat rate Sets the typematic delay and repeat rate

Read or write multiple inputs and outputs	
BlockGetInput1 / BlockGetInput2 / BlockGetInputAll55 BlockSetOutput1/BlockSetOutput2/ BlockSetOutputAll55	Read block of inputs (1 to 32) / (33 to 55) / (1 to 55) Set block of outputs (1 to 32) / (33 to 55) / (1 to 55)
Read or set multiple pin configurations	
GetAllPinConfiguration GetAllPinKeyCodes GetAllPinKeyMapping GetAllPinKeyModifierss SetAllPinConfiguration SetAllPinKeyCodes SetAllPinKeyMapping SetAllPinKeyModifierss	Read pin configuration for all pins Read pin key codes for all pins Read pin key mapping for all pins Read pin key modifiers for all pins Set pin configuration for all pins Set pin key codes for all pins Set pin key mapping for all pins Set pin key modifiers for all pins
Read or set automatic output initalization option	
GetAutoSetOutputs SetAutoSetOutputs	Get automatic outputs initialization option Set automatic outputs initialization option
Macro-related commands	
GetMacroKeys GetMacroName MacroCreate MacroDelete MacroGetActiveMacros MacroGetFreeSpace MacroGetLength MacroModifyLength MacroSaveConfiguration SetMacroKeys SetMacroName	Read key mapping for a macro Read macro name Reserve the space for a new macro Delete a specific macro Read the macro activation status Get free space for macros Read macro length Modify the length of a macro Save configuration of macros Set key mapping for a macro Set macro name
COM_Execute command	
COM_ExecuteInner with the following parameters: - GetDigitalPinValues - SetDigitalPinValues - GetPinConfiguration - SetPinConfiguration	 Read digital inputs Set digital outputs Read pins configuration Set pins configuration
Raw basic device report	
GetFullDeviceReport	Set and read full device report

8.2. Connection signal output

Connection signal pin status can be set for pins 48 to 55. When USB connection with PC is established (or ethernet link is established on PoKeys56E/57E), pin for which 'Connection signal pin status' is enabled, will be put into high state (or low state if pin polarity is setup as inverted). If the connection with PC is lost and power through USB is still available (or ethernet link is lost on PoKeys56E/57E), pin will go into low state (or high state if pin polarity is setup as inverted).

This command must not be confused with Failsafe settings, which are based on the communication timeout and not only link status.

PoKeys configuration software usage

To enable connection signal output, select one of the pins 48 to 55, then first select 'Digital output' option for this pin and check the 'Connection signal' option.

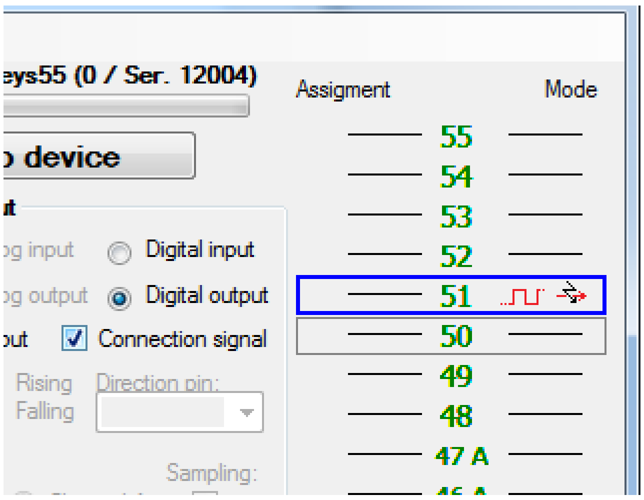


Figure 5: Connection signal setup

Applicable PoKeys library commands

Read or set connection signal setup	
GetConnectionSignalData SetConnectionSignalData	Retrieves connection signal pins data Sets connection signal pins data

8.3. Digital counters (only on PoKeys56 devices)

Selected pins of the PoKeys56U and PoKeys56E/57E can be setup to count the number of signal transitions on those pins. Pin digital counter can be setup to be incremented/decremented on rising, falling or rising and falling edges of the input signal. If needed, additional pin can be selected to toggle between incrementing and decrementing mode.

For the list of pins that support digital counter option go to *Device pin functions* chapter of this manual.

If switches are used in the combination with digital counters, external debouncing circuit must be installed.

PoKeys configuration software usage

To setup digital counter inputs, first set the selected pin as 'Digital input' and if digital counter is available on the selected pin, the 'Enable counter' option will be enabled. Check this option and check 'rising' and/or 'falling' edge counting option.

To enable selection between incrementing or decrementing counter modes, a direction pin can be selected in the 'Direction pin' drop-down menu. If no pin is selected, the counter mode defaults to incrementing mode.

Digital counters values status page (menu 'Peripherals > Digital counters values…') can be used to check the proper working of the configured digital counters.

Applicable PoKeys library commands

Read or set single pin digital counter configuration	
GetPinData SetPinData GetDigitalCounterDirectionPin SetDigitalCounterDirectionPin	Reads pin data with counter options Set pin data with counter options Reads direction pins for the digital counters Sets direction pins for the digital counters
IsCounterAvailable	Checks if specified pin supports digital counter
Read or set multiple pin digital counter configuration	
GetAllPinConfiguration SetAllPinConfiguration	Read pin configuration for all pins with digital counter options Set pin configuration for all pins with digital counter options
Read or set digital counters values	
GetDigitalCountersValues	Reads current values of digital counters
ResetDigitalCounters	Resets values of digital counters
COM_Execute command	
COM_ExecuteInner with the following parameters: - GetDigitalCounterValues - GetPinConfiguration - SetPinConfiguration	

8.4. Encoders

PoKeys devices can handle decoding of up to 26 (25 on PoKeys55) pairs of quadrature encoder signals. A and B signals of 25 'normal' encoders can be connected to any digital input and are intended for hand-driven rotational encoder switches with the quadrature signal frequencies up to 1 kHz.

Three fast encoders input pairs are available only on selected input pins (pins 1-2 as encoder 1, pins 3-4 as encoder 2, pins 15-16 as encoder 3 on PoKeys55 devices and pins 1-2 as encoder 1, pins 5-6 as encoder 2, pins 15-16 as encoder 3 on PoKeys56U and PoKeys56E/57E devices) and can handle quadrature signal frequencies to about 100 kHz. When activated, fast encoders logically replace the 'normal' encoders 1, 2 and 3.

On PoKeys56U and PoKeys56E/57E devices, ultra-fast encoder support is available on pins 8, 12 with the optional index signal input on pin 13. This can handle even higher frequencies (up to 5 MHz with digital filtering disabled). As ultra-fast encoders use hardware-specific functions, only x2 and x4 step multiplication factors are available.

Samilarly as simple digital inputs, encoders can be assigned to direct key mapping or keyboard macro (only with USB devices). This is possible for both directions (CW and CCW) separately.

If needed, encoder inputs can be incremented or decremented on every detected signal edge, increasing the resolution of the encoder for a factor of 4.

PoKeys configuration software usage

To enable encoder input on the selected pin, define the pin as digital input, select encoder index with numerical up-down selector and select appropriate encoder channel. The last step is to check the box ‘Encoder’.

If needed, encoder inputs can be incremented or decremented 4x faster, therefore each complete step will produce increment or decrement of 4 sub-steps. Using this setting, higher precision can be obtained.

To assign a key combination (only on USB devices) associated with the encoder, use the same procedure as described in the ‘Digital inputs and ouputs’ section. Each encoder is connected to PoKeys device with two signals and each of the pins that the encoder is connected to changes the keyboard mapping operation when encoder option is activated. Instead of being activated on each A and B signal front, keys are triggered on encoder value increment or decrement event. The key mapping settings however are setup on the pins that the encoder is connected to.

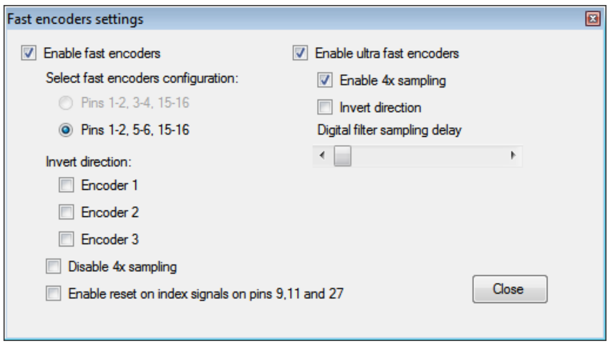
Enabling fast encoders

To enable fast encoders, go to menu 'Peripherals > Fast encoders settings’, then check ‘Enable fast encoders’ option. There are additional options for inverting the encoders’ directions, disabling 4x step multiplication and enabling the index signal on pins 9, 11 and 27. If index signal input is enabled, encoder value is automatically reset on low to high index signal transition.

Fast encoders inputs are fixed to pins 1-2 for fast encoder input 1, pins 3-4 (PoKeys55) or pins 5-6 (PoKeys56U/E) for fast encoder input 2 and pins 15-16 for fast encoder input 3.

Enabling ultra-fast encoders

To enable ultra-fast encoders, go to menu 'Peripherals > Fast encoders settings', then check 'Enable ultra fast encoders' option. There are additional options to enable 4x step multiplication (sampling) and inverting the encoder direction. Digital filter sampling delay slider enables setting the digital filter delay parameter - leftmost position equates to no digital filtering, rightmost position equates to digital filtering with filter delay constant set to 1000 (sampling frequency reduced to less than 25 kHz).



Fast encoders settings

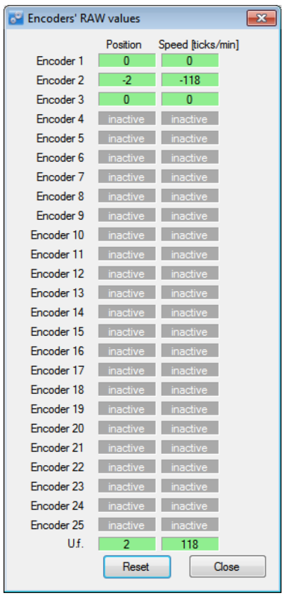
- [✓] Enable fast encoders
- Select fast encoders configuration:
  - ○ Pins 1-2, 3-4, 15-16
  - ● Pins 1-2, 5-6, 15-16
- Invert direction:
  - ☐ Encoder 1
  - ☐ Encoder 2
  - ☐ Encoder 3
- ☐ Disable 4x sampling
- ☐ Enable reset on index signals on pins 9,11 and 27
- [✓] Enable ultra fast encoders
  - [✓] Enable 4x sampling
  - ☐ Invert direction
  - Digital filter sampling delay

Displaying encoder raw values

To open encoder raw values dialog, go to Peripherals menu and select 'Encoder RAW values'. The following dialog below appears. It simply shows the list of all encoders and their current values. In additional column, current encoder speed is displayed.

Status of the fast encoders is displayed in green, while the status of an ordinary encoder is displayed in light grey. Inactive encoders are displayed as dark grey boxes.

At the bottom of the window, there is a command button that can be used to reset the encoders' values.



Encoders' RAW values

	Position	Speed [ticks/min]
Encoder 1	0	0
Encoder 2	-2	-118
Encoder 3	0	0
Encoder 4	inactive	inactive
Encoder 5	inactive	inactive
Encoder 6	inactive	inactive
Encoder 7	inactive	inactive
Encoder 8	inactive	inactive
Encoder 9	inactive	inactive
Encoder 10	inactive	inactive
Encoder 11	inactive	inactive
Encoder 12	inactive	inactive
Encoder 13	inactive	inactive
Encoder 14	inactive	inactive
Encoder 15	inactive	inactive
Encoder 16	inactive	inactive
Encoder 17	inactive	inactive
Encoder 18	inactive	inactive
Encoder 19	inactive	inactive
Encoder 20	inactive	inactive
Encoder 21	inactive	inactive
Encoder 22	inactive	inactive
Encoder 23	inactive	inactive
Encoder 24	inactive	inactive
Encoder 25	inactive	inactive
U.f.	2	118

Figure 6: Encoders' RAW values

Applicable PoKeys library commands

Read single encoder value	
GetEncoderValue GetEncoderRAWValue	Read current encoder value Read encoder RAW value
Read or set single encoder configuration	
GetEncoderSettings SetEncoderSettings	Read encoder settings Set encoder settings
Read or set single encoder keyboard mapping configuration	
GetEncoderKeyMappingDirA GetEncoderKeyMappingDirB SetEncoderKeyMappingDirA SetEncoderKeyMappingDirB	Reads encoder mapping configuration for direction A Reads encoder mapping configuration for direction B Set encoder key mapping for direction A Set encoder key mapping for direction B
Read or set multiple encoders' values	
GetEncoders1_13_RAWValue GetEncoders14_25_RAWValue GetEncoders14_26_RAWValue	Read RAW value for encoders 1 to 13 Read RAW value for encoders 14 to 25 Read RAW value for encoders 14 to 26 (with ultra fast encoder as 26)
SetEncoders1_13_RAWValue SetEncoders14_25_RAWValue	Write RAW value for encoders 1 to 13 Write RAW value for encoders 14 to 25
ResetEncoderValue	Reset encoder value to 0
Read or set multiple encoders' configurations	
GetAllEncoderChannelPinMappings GetAllEncoderKeyMappingsDirA GetAllEncoderKeyMappingsDirB GetAllEncoderOptions	Get encoder pin mapping for all encoders Get encoder pin codes and modifiers for all encoders for direction A Get encoder pin codes and modifiers for all encoders for direction B Read encoder options for all encoders
SetAllEncoderOptions SetAllEncoderChannelPinMappings SetAllEncoderKeyMappingsDirA SetAllEncoderKeyMappingsDirB	Set encoder options for all encoders Set encoder pin mapping for all encoders Set encoder pin codes and modifiers for all encoders for direction A Get encoder pin codes and modifiers for all encoders for direction B
Read or set fast encoders configuration	
GetFastEncodersStatus GetFastEncodersStatusWOptions SetFastEncodersStatus SetFastEncodersStatusWOptions	Read fast encoder settings Read fast encoder settings and options Set fast encoder settings Set fast encoder settings
Read or set ultra-fast encoders configuration	
GetUltraFastEncodersSettings SetUltraFastEncodersSettings	Get ultra fast encoder settings Set ultra fast encoder settings
COM_Execute command	
COM_ExecuteInner with the following parameters: - GetEncoderValues - SetEncoderValues - GetPeripheralConfiguration - SetPeripheralConfiguration	

8.5. Matrix keyboard

Matrix keyboard is a set of buttons, connected into a mesh. All buttons in a row share one contact, same goes for each of the buttons in the column. If a button is pressed, a key press is detected with a periodic scanning of each of the rows and columns. PoKeys devices use digital outputs for setting the voltage levels on rows and read column voltage levels using digital inputs that already have internal pull-up resistors, so no external circuitry is needed.

PoKeys devices support matrix keyboards of up to 16x8 in size, simpler 3x3, 4x3, 4x4 and others are of course fully supported.

Similarly to simple digital inputs, keys of the matrix keyboard connected to the PoKeys55/PoKeys56U device can be configured as USB keyboard keys. Direct mapping, mapping to macro sequence and triggered mapping are all supported. Additional alternate function can be used to assign two different keyboard keys to each of the matrix keyboard buttons. If additional (and freely selectable from the list of digital inputs) Fn+ input pin is inactive, the default function key is used. If the Fn+ key input pin is activated, an alternate function key is used instead of the default.

On all devices, the status of key presses of the matrix keyboard can be read using the PoKeys library commands without the need to setup the mapping described above.

Figure 7: Standard 4x3 matrix keyboard

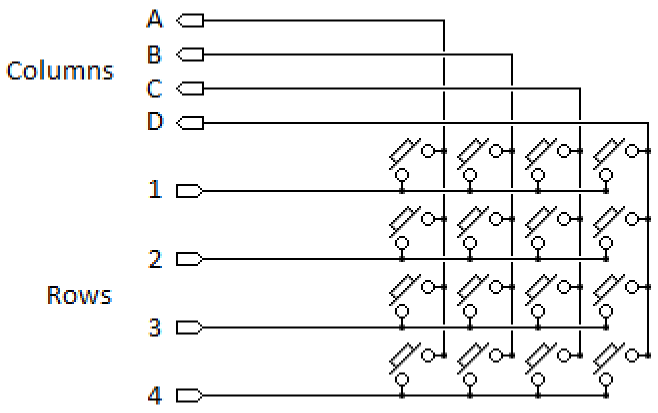
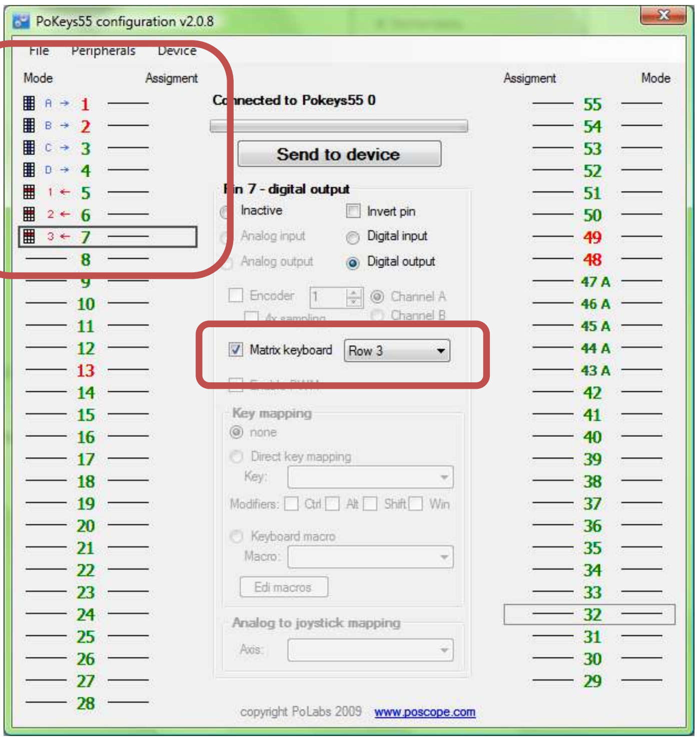


Figure 8: 4x4 matrix keyboard internal structure

PoKeys configuration software usage

Before any matrix keyboard configuration can be done, go to ‘Peripherals > Matrix keyboard…’, check the ‘Enable matrix keyboard’ option and select the number of rows and columns. Close the dialog and continue by selecting column and row pins.


PoKeys55 configuration v2.0.8
File
Peripherals
Device
Mode
Assigment
Connected to Pokeys55 0
Send to device
Key mapping
Analog to joystick mapping
copyright PoLabs 2009 www.poscope.com

Pin 7 - digital output

Inactive | Invert pin
Analog input | Digital input
Analog output | [✓ Digital output]

Encoder 1 | Channel A
Channel B

[✓ Matrix keyboard] Row 3

Figure 9: Assigning row and column pins

Matrix keyboard column selection

Each free digital input pin can be assigned as matrix keyboard column input. Make sure the selected pin is configured as digital input, then check the ‘Matrix keyboard’ option for the pin and select the appropriate column letter from the list.

Matrix keyboard row selection

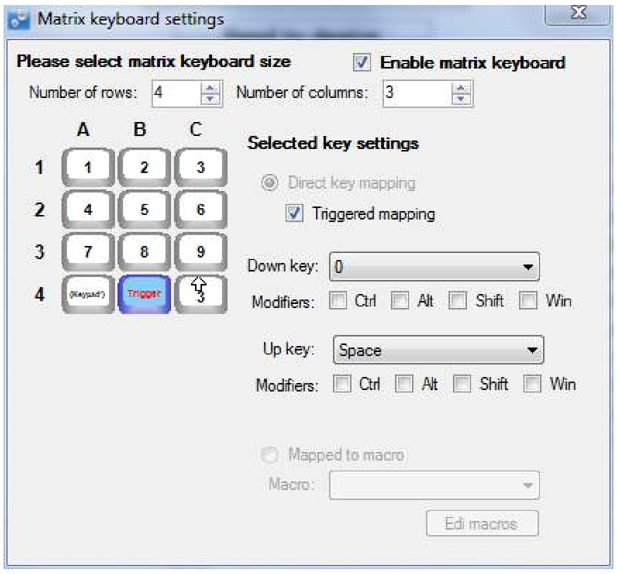
Each free digital output pin can be assigned as matrix keyboard row output. Make sure the selected pin is configured as digital output, then check the ‘Matrix keyboard’ option for the pin and select the appropriate row number from the list.

Keyboard mapping configuration

For configuring the keyboard mapping, open the ‘Peripherals > Matrix keyboard…’ menu. Matrix keyboard is schematically drawn in the dialog below. On the right, key mapping settings can be selected (on PoKeys55/PoKeys56U devices). To setup mapping, click on one of the keys in the matrix keyboard drawing and select appropriate key mapping options on the right.

To test the matrix keyboard, first make sure that the settings have been saved to device (close the Matrix keyboard dialog and click on ‘Send to device’ button). The matrix keyboard dialog can then be used to test the matrix keyboard – just press any key on your matrix keyboard and the appropriate button in the matrix keyboard drawing will be highlighted.

To setup different key presses for ‘key press’ and ‘key release’ events, check ‘Triggered mapping’ option and select different settings for ‘Down key’ (‘key press’ event) and ‘Up key’ (‘key release’ event).



Matrix keyboard settings

Please select matrix keyboard size ☑ Enable matrix keyboard

Number of rows: 4 Number of columns: 3

Selected key settings

◉ Direct key mapping

☑ Triggered mapping

Down key: 0

Modifiers: ☐ Ctrl ☐ Alt ☐ Shift ☐ Win

Up key: Space

Modifiers: ☐ Ctrl ☐ Alt ☐ Shift ☐ Win

○ Mapped to macro

Macro:

Edi macros

Figure 10: Matrix keyboard configuration for a 4x3 matrix keyboard

Applicable PoKeys library commands

Read matrix keyboard status	
GetMatrixKeyboardKeyStatus	Read matrix keyboard status
Read or set basic matrix keyboard configuration	
GetMatrixKeyboardConfiguration SetMatrixKeyboardConfiguration	Read matrix keyboard configuration Set matrix keyboard configuration
Read or set matrix configuration with alternate function support	
GetMatrixKeyboardConfiguration_AlternateFunctionSupport SetMatrixKeyboardConfiguration_AlternateFunctionSupport	Read matrix keyboard configuration (for alternate functions mapping) Set matrix keyboard configuration (for alternate functions mapping)
Read or set matrix configuration with triggered mapping support	
GetMatrixKeyboardConfiguration_TriggerSupport SetMatrixKeyboardConfiguration_TriggerSupport	Read matrix keyboard configuration (for triggered mapping) Set matrix keyboard configuration (for triggered mapping)
COM_Execute command	
COM_ExecuteInner with the following parameters: - GetMatrixKeyboardConfiguration - SetMatrixKeyboardConfiguration - GetMatrixKeyboardKeysStatus	

8.6. Analog inputs

Analog input function is only available for pins 43 to 47 (on PoKeys55 devices) or for pins 41 to 47 (on PoKeys56 devices). On PoKeys55 and PoKeys56U devices, these analog inputs can also be freely mapped to any of the 6 joystick axis; X, Y, Z, rotation X, rotation Y and throttle.

Analog inputs (10-bit on PoKeys55 and 12-bit pn PoKeys56U/PoKeys56E/PoKeys57E) are sampled at a fixed rate of 10 kHz and fed through adjustable discrete low-pass filter with the following equation

$$y(k) = y(k-1) * \frac{filter}{filter + 1} + u(k) * \frac{1}{filter + 1}$$

where y(k) is the output analog value, u(k) is a new A/D sample and *filter* is a user-adjustable constant. Sample u(k) is produced according to the following equation

PoKeys56U/PoKeys56E/PoKeys57E	PoKeys55
$\boldsymbol{u(k) = \frac{U(k)[V]}{3.3\,V} * 4095}$	$u(k) = \frac{U(k)[V]}{3.3\,V} * 1023$

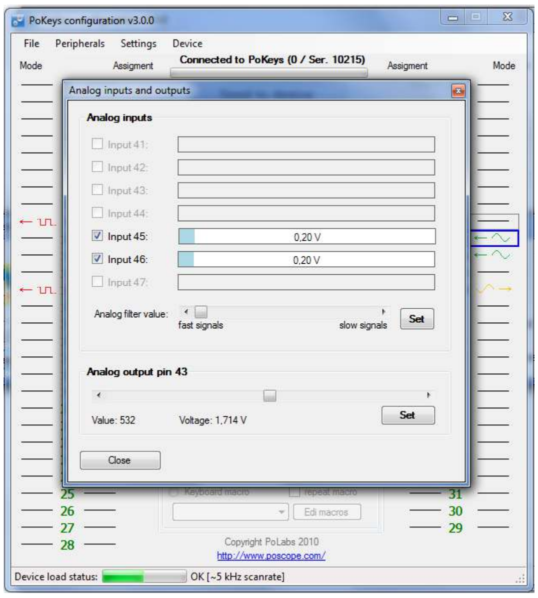
where U(k) (in Volts) is a voltage present on the selected analog input pin.

PoKeys configuration software usage

To open analog inputs dialog, go to Peripherals menu and select 'Analog inputs and outputs'. Dialog below appears. To enable display of analog input channel, check the appropriate check box. It is enabled only when the input is set up as analog input.

The progress bar displays the current voltage at the pin with the maximum at 3.3V. Below the input selection boxes user can set low-pass filtering for analog inputs. When analog input signal appears to be flickering or jumping due to analog signal noise, move the value for the filter to the right towards label 'slow signals' and then press Set button.

If pin 43 is set as analog output (only on PoKeys55 devices), analog value can be set for this pin.



Connected to PoKeys (0 / Ser. 10215)

[ ] Input 41:
[ ] Input 42:
[ ] Input 43:
[ ] Input 44:
[✓] Input 45: 0,20 V
[✓] Input 46: 0,20 V
[ ] Input 47:

Value: 532 Voltage: 1,714 V

Device load status: OK [~5 kHz scanrate]

Figure 11: Analog inputs and outputs dialog

Applicable PoKeys library commands

Read single analog input value	
GetAnalogInput	Read analog input value
Read or set single pin function	
GetPinData SetPinData	Reads pin function Set pin function
Read all analog inputs	
GetAllAnalogInputs	Read all analog inputs in one command
Read analog input maximum value	
GetAnalogValueMax	Returns the maximum value of analog input
Read or set analog low-pass filter parameter	
GetAnalogRCFilterValue SetAnalogRCFilterValue	Read low-pass filter parameter Set low-pass filter parameter
COM_Execute command	
COM_ExecuteInner with the following parameters: - GetAnalogPinValues	

8.7. Joystick mapping

Each axis of the PoKeys virtual joystick can be assigned an analog input source. In addition, analog to digital mapping option can be enabled, which allows user to connect an analog joystick to a PoKeys devices and simulate key presses for each direction of the joystick. User can freely select dead band and saturation ranges.

Besides mapping the analog inputs to virtual joystick axes, digital inputs (or encoder switching events) can be mapped to any of the 32 virtual joystick buttons (either directly or 'triggered') or 4-way POV hat selector. The triggered mapping to joystick buttons enables used to select different pins that triggers selected joystick button on off-to-on transition (Down Event) and on on-to-off transition (Up Event).

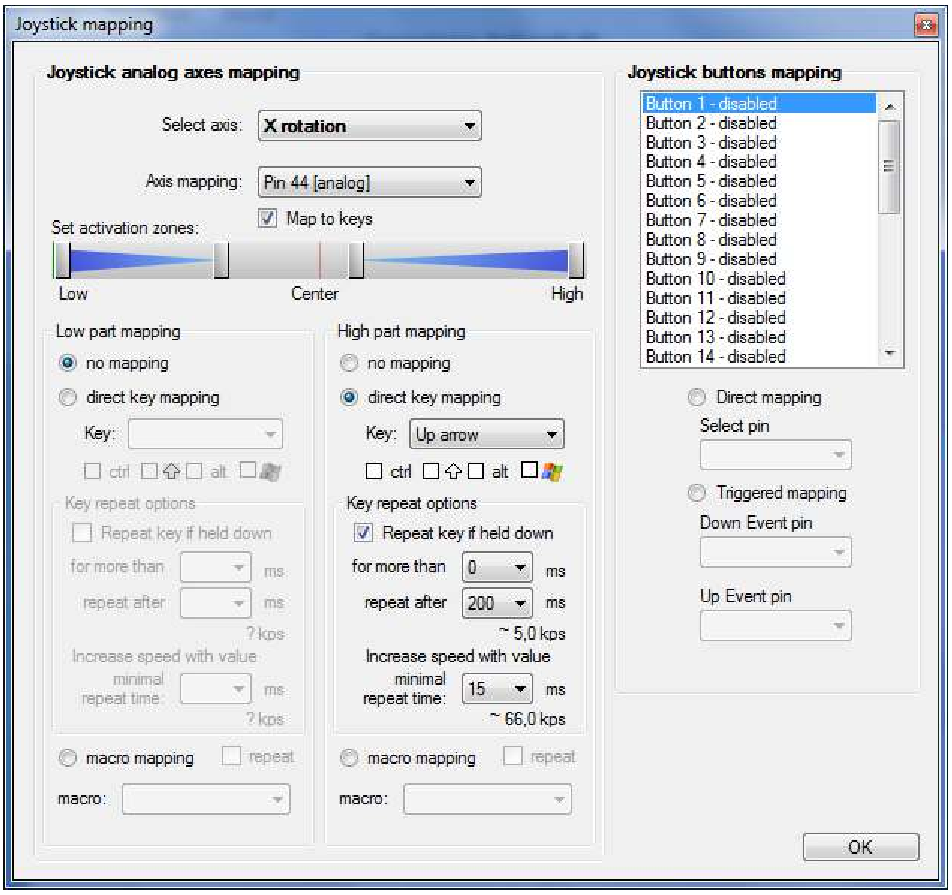
PoKeys configuration software usage

Joystick axis and buttons mapping can be setup via Joystick mapping dialog. Go to 'Peripherals' and select 'Joystick settings…'. The dialog on Figure 12 appears.

Each axis can be assigned an analog input. In addition, analog to digital mapping option can be enabled. This allows user to connect an analog joystick to a PoKeys devices and simulate key presses for each direction of the joystick. To do so, first check 'Map to key' option. Then set the dead band (when input value will be between lower and upper dead band margins, no keys will be activated) using sliders. In the lower part of the window, select the mapping options.

For simple direct mapping (pin input status is directly reflected in joystick button status) use the 'Direct mapping' option and select pin number to be associated with selected joystick button. If more advanced behavior is needed (joystick button is pressed for a short time only on transitions of pin status), user should select 'Triggered mapping' option to select one pin that triggers selected joystick button on off-to-on transition (Down Event) and one pin that triggers this joystick button on on-to-off transition (Up Event).

If joystick button mapping is to be used in connection with encoder inputs, use 'Triggered mapping' option (encoder's values cannot be directly translated into direct mapping) and select a pin with the appropriate encoder channel. For example: pins 5 and 6 are set up as digital inputs with encoder (channel A on pin 5 and channel B on pin 6). When pin 5 is selected as Down event pin for joystick Button 2 and pin 6 is selected as Down event pin for joystick Button 3, rotating the encoder in positive direction will trigger joystick Button 2 on each detent. Similarly, rotating the encoder in negative direction, joystick Button 3 will be triggered on each detent. Up Event pin option cannot be used in connection with encoders.


Joystick mapping

Joystick analog axes mapping

Select axis: X rotation

Axis mapping: Pin 44 [analog]

☑ Map to keys

Set activation zones:

Low Center High

Low part mapping
- [✓ no mapping]
- direct key mapping

High part mapping
- no mapping
- [✓ direct key mapping]

Key: Up arrow

Key repeat options

☑ Repeat key if held down

for more than 0 ms

repeat after 200 ms

~ 5,0 kps

Increase speed with value

minimal repeat time: 15 ms

~ 66,0 kps

Joystick buttons mapping

Button 1 - disabled
Button 2 - disabled
Button 3 - disabled
Button 4 - disabled
Button 5 - disabled
Button 6 - disabled
Button 7 - disabled
Button 8 - disabled
Button 9 - disabled
Button 10 - disabled
Button 11 - disabled
Button 12 - disabled
Button 13 - disabled
Button 14 - disabled

Figure 12: Joystick mapping settings

Applicable PoKeys library commands

Read or set joystick axes and buttons mapping	
GetJoystickMapping SetJoystickMapping	Read joystick mapping options Set joystick mapping options
Read or set joystick triggered mapping	
GetJoystickTriggeredMapping SetJoystickTriggeredMapping	Read joystick triggered mapping options Set joystick triggered mapping options
Read or set analog to digital mapping	
GetJoystickDigitalMapping SetJoystickDigitalMapping	Read joystick analog to digital mapping configuration Set joystick analog to digital mapping configuration

8.8. PWM outputs

PoKeys55, PoKeys56U and PoKeys56E/57E devices support PWM output function on 6 pins (pins 17 to 22). Different duty cycles can be assigned to each PWM output however, all outputs share the same PWM period. PWM outputs can be easily amplified using an external transistor and used for control of loads with increased current demand. PoKeys PWM outputs can also be used to drive various R/C servo motors that accept PWM signal with 50 Hz frequency (20 ms PWM period) and duty cycles between 5 and 10 % (1 to 2 ms).

PoKeys devices have an in-built PWM module that operates at a fixed clock frequency (12 MHz in PoKeys55 devices and 25 MHz in PoKeys56U and PoKeys56E/57E devices). Both the PWM period and the PWM duty cycles must be expressed as number of module clock cycles (i.e. 20 ms PWM period equates to 0.020 x 25 000 000 = 500 000 on PoKeys56U/E devices).

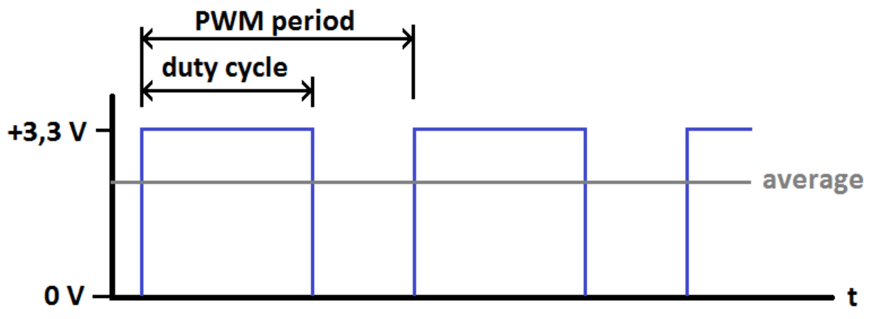
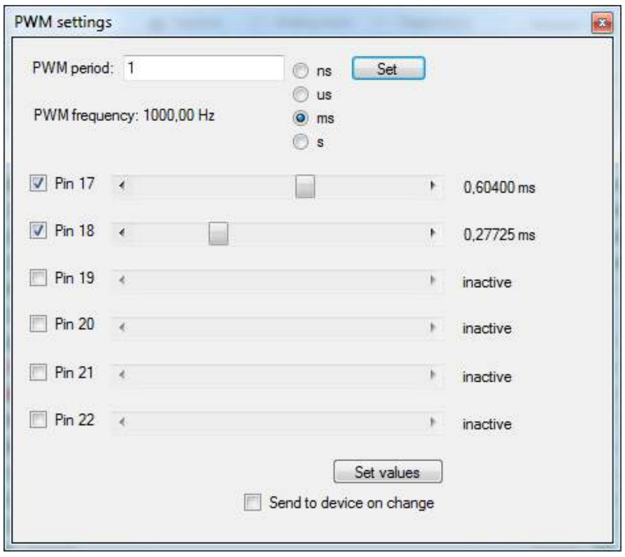


Figure 13: PWM output

PoKeys configuration software usage

PoKeys device's PWM (pulse width modulation) module can be setup via Peripherals > PWM outputs….



PWM settings

PWM period: 1 ( ) ns ( ) us (●) ms ( ) s — Set

PWM frequency: 1000,00 Hz

☑ Pin 17	0,60400 ms
☑ Pin 18	0,27725 ms
☐ Pin 19	inactive
☐ Pin 20	inactive
☐ Pin 21	inactive
☐ Pin 22	inactive

Set values

☐ Send to device on change

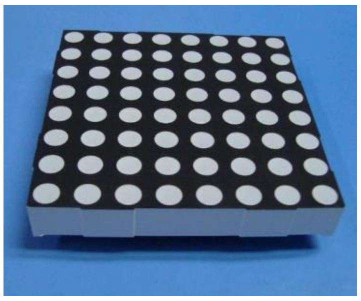
Figure 14: PWM outputs settings

In this window, user can enter PWM period and set PWM duties for each channel. Channels can be independently enabled or disabled. After a change is made, user must click 'Set values' button or check 'Send to device on change' checkbox. Left position of a slider means 0% and right position 100% respectively.

Applicable PoKeys library commands

Read or set PWM outputs configuration	
SetPWMOutputs GetPWMOutputs	Set PWM period and duty cycles Get PWM period and duty cycles
Read PWM module frequency	
GetPWMFrequency	Read the PWM module base frequency
COM_Execute command	
COM_ExecuteInner with the following parameters: - SetPWMOutputs	

8.9. Matrix LED output



PoKeys devices support 2 additional external 8x8 LED matrix displays. Each display is based on 2 8-bit shift registers, one controlling the row and the other controlling the column signals. PoKeys refreshes the display at the rate of 2800 rows per second (which gives around 350 Hz refresh rate of the whole 8x8 display).

Once enabled, PoKeys device assigns pins 9, 10 and 11 to be used with display 1, and pins 23, 24, 25 to be used with display 2.

Function	Matrix LED display 1	Matrix LED display 2
data	9	23
latch	10	24
clock	11	25

Table 1: Matrix LED displays pin assignments

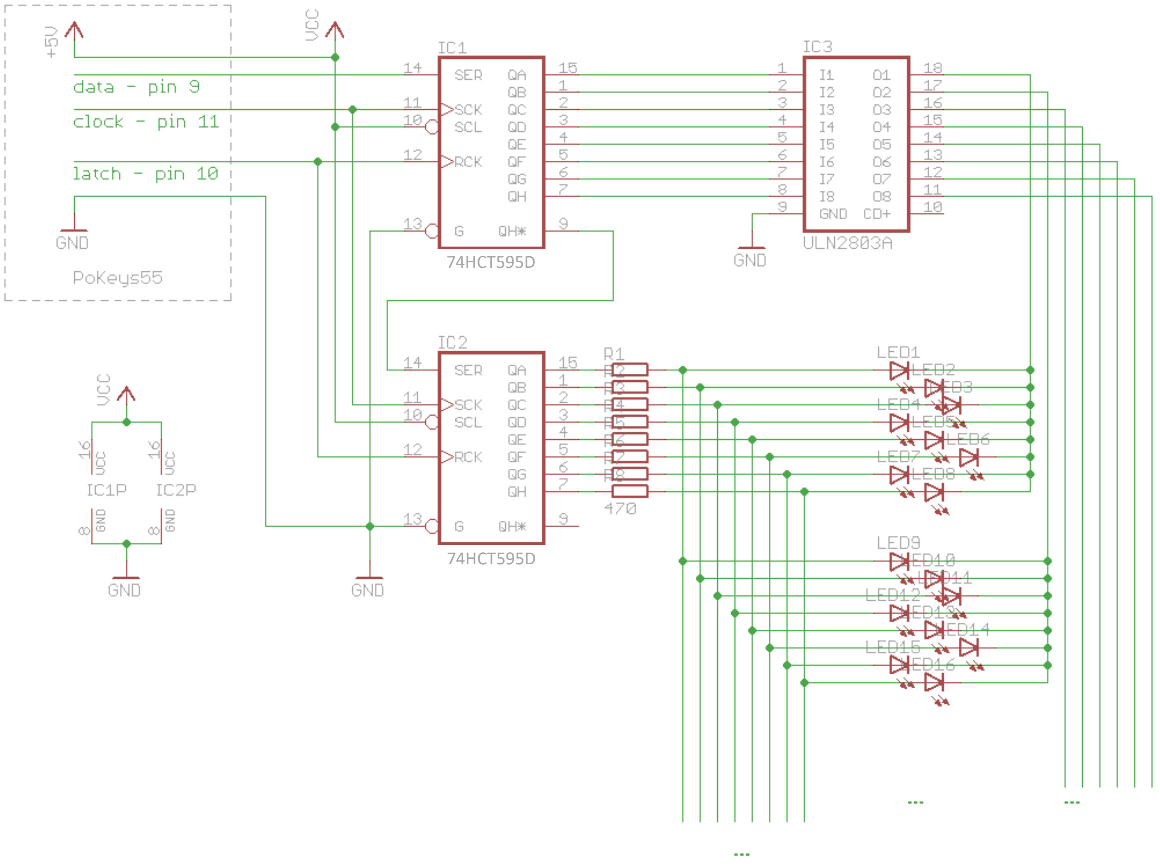


Figure 15: Matrix LED display deserialization circuit

PoKeys configuration software usage

To set-up matrix LED displays, open 'Peripherals > LED displays…'. The following dialog appears

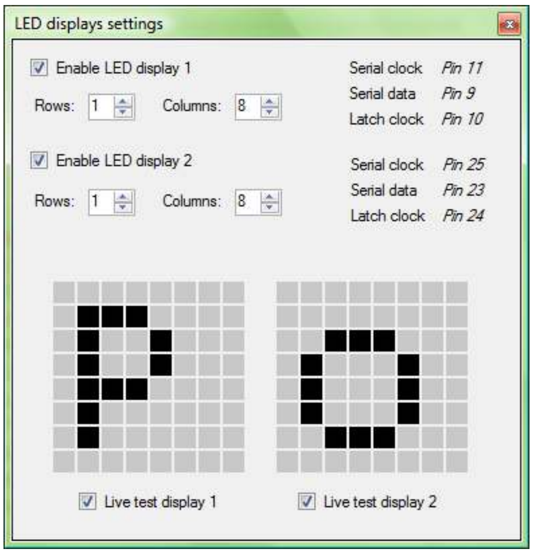


Figure 16: Matrix LED setup dialog

For each of the displays, number of rows and columns can be selected. On the bottom, after enabling 'Live test display x', user can test the displays in real-time by clicking on gray rectangles. Left-click turn-s selected pixel on, while right-click resets selected pixel.

Applicable PoKeys library commands

Read or set matrix LED settings	
MatrixLEDGetSettings MatrixLEDSetSettings	Get matrix LED settings Set matrix LED settings
Matrix LED commands	
MatrixLED1ClearAll MatrixLED1SetPixel MatrixLED1Update MatrixLED2ClearAll MatrixLED2SetPixel MatrixLED2Update	Clear matrix LED 1 Set pixel on matrix LED 1 Matrix LED 1 update Clear matrix LED 2 Set pixel on matrix LED 2 Matrix LED 2 update
COM_Execute command	
COM_ExecuteInner with the following parameters: - SetMatrixLEDconf - SetMatrixLEDstat	

8.10. LCD

PoKeys devices support connecting one alphanumeric LCD module up to a size of 4x20 (4 rows, 20 columns). The selection of the module is limited by support for HD44780 or compatible chipset. Usually these displays come in various sizes - 1/2/4 line with 8/16/20 characters and colors (black letters on green background, white letters on blue background ...).

Figure 17: Typical 2x16 character LCD

These displays share standard pin-out that is listed in the table below:

Pin	Symbol	Function	PoKeys pin
1	Vss	Ground	GND
2	Vdd	Positive supply (usually 5V)[2]	5V (usually) or 3.3V
3	Vo	Contrast adjustment	Variable resistor between GND and supply or PWM output
4	RS	Instruction/data input	Pin 29
5	R/W	Read/write	Pin 28
6	E	Enable signal	Pin 30
7	DB0	Data bus – bit 0	Not connected
8	DB1	Data bus – bit 1	Not connected
9	DB2	Data bus – bit 2	Not connected
10	DB3	Data bus – bit 3	Not connected
11	DB4	Data bus – bit 4	Pin 26 (or secondary 34)
12	DB5	Data bus – bit 5	Pin 25 (or secondary 33)
13	DB6	Data bus – bit 6	Pin 24 (or secondary 32)
14	DB7	Data bus – bit 7	Pin 23 (or secondary 31)
15	Backlight (optional)		
16	Backlight (optional)		

Table 2: LCD pin assignments

LCD display can be used to display various data. A third-party application or a script can execute all supported operations, including LCD initializing, clearing, moving cursor, setting display shifting mode, custom character defining and displaying text.

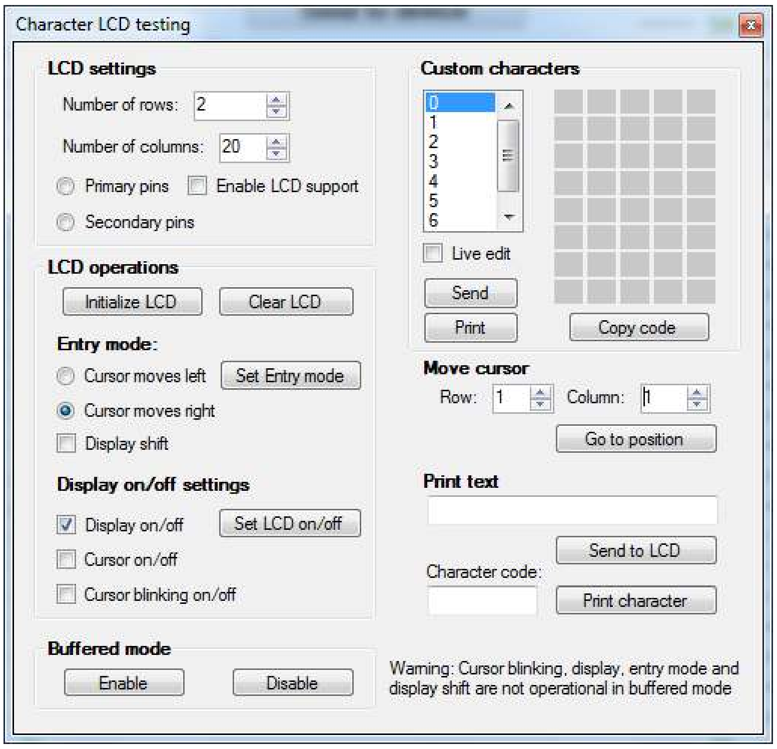
[2] Positive supply voltage depends on LCD used. User should find this information in datasheet of the LCD in use. **On PoKeys56E take special care that the power supply voltage for the LCD is adequate (there is 0.8 V of voltage drop between power supply input and 5V pin).**

PoKeys56U and PoKeys56E/57E devices by default function in buffered LCD mode. In this mode, any LCD-related command that is sent to PoKeys device is first buffered and when possible, PoKeys device executes the LCD refresh on its own. In this mode, some LCD operations are not operational – cursor movement is controlled by the PoKeys device and cursor move or display commands may not work as expected. If this low-level control is desirable, buffered mode must be deactivated first. In other cases, it is advisable to use buffered mode in order to allow better load balancing in PoKeys devices and in the end attain greater communication speed.

Before LCD initialization, the LCD module size (number of rows and columns) must be specified and appropriate pin assignment (primary or secondary, see the table above) must be selected. Secondary pins must be selected in case of matrix LED display 2 in use.

PoKeys configuration software usage

Functions of this interface can be tested through PoKeys settings application. Just open Peripherals > Test LCD… and dialog below will appear.



Warning: Cursor blinking, display, entry mode and display shift are not operational in buffered mode

Figure 18: Character LCD testing dialog

LCD settings

In this part, user can set number of rows and columns in the LCD used. Support for LCD can be enabled or disabled also. Data pins for LCD can be selected on primary (23 to 26) or secondary (31 to 34) pins. ***Secondary pins must be selected in case of matrix LED display 2 is in use.***

LCD operations

Before user can start using the LCD, LCD module must be initialized. This is done via 'Initialize LCD' button. Button 'Clear LCD' clears LCD display and moves cursor to home position.

User can also set entry mode settings of LCD module. Cursor can be set-up to move either right (normally) or left after each character displayed. If 'Display' shift is enabled, whole display shifts with every new character displayed.

Settings are processed after user clicks button 'Set Entry mode' and work only in ‘unbuffered’ mode described above.

Display on/off settings

User can set on/off switches for whole display, cursor and cursor blinking.

Settings are processed after user clicks button 'Set LCD on/off' work only in ‘unbuffered’ mode described above.

Custom characters

Simple interface enables to draw up to 8 custom characters. These characters can then be used on display. Selecting 'Live edit' mode will transfer the character each time a change is made to any of the pixels. Character can be previewed via button 'Print', which puts current custom character on the LCD display.

Move cursor

This section enabled user to move cursor to any position on the screen works only in ‘unbuffered’ mode described above.

Print text

Sends entered text to display module. If advanced characters are needed, enter character code in lower text box and press 'Print character'.

Applicable PoKeys library commands

Read or set LCD module configuration	
LCDGetSettings **LCDSetSettings**	Read LCD module settings Set LCD module settings
LCD operations	
LCDInit **LCDClear** **LCDSetEntryMode** **LCDDisplayOnOffControl** **LCDSetMode** **LCDGotoXY**	Initialize LCD display Clear LCD display Set entry mode for LCD Set LCD display on/off status Set LCD display mode Move cursor to a specified position in the display
Display text on LCD	
LCDPrint **LCDPutc**	Print a string on the LCD display Put a single character on LCD display
LCDDefineCustomCharacter	Define a custom character in the LCD display memory
COM_Execute command	
COM_ExecuteInner with the following parameters: - LCD	

8.11. PoExtBus

PoExtBus bus support enables user to add additional 10 8-bit shift registers to the project based on PoKeys device. This gives additional 80 digital outputs that can be easily controlled with included dll interface or other 3rd party interface for PoKeys device.

On PoKeys55, device assigns pins 35, 36 and 37 to be used with PoExtBus. On PoKeys56 series, there is a dedicated connector on the board, which serves for the PoExtBus functions. Marking the pin closer to the bottom of the board (the oposite side of either Ethernet or USB connector) as pin 1, the PoExtBus devices should be connected as follows:

Pin 1	Power supply 5V
Pin 2	Ground
Pin 3	Data
Pin 4	Latch
Pin 5	Clock

If PoExtBus functionality is moved to PoKeys pins 35, 36, 37, the following table shows the proper connections.

Function	ExtBus – PoKeys pin
Clock	35
Data	36
Latch	37

Table 3: PoExtBus pin assignments

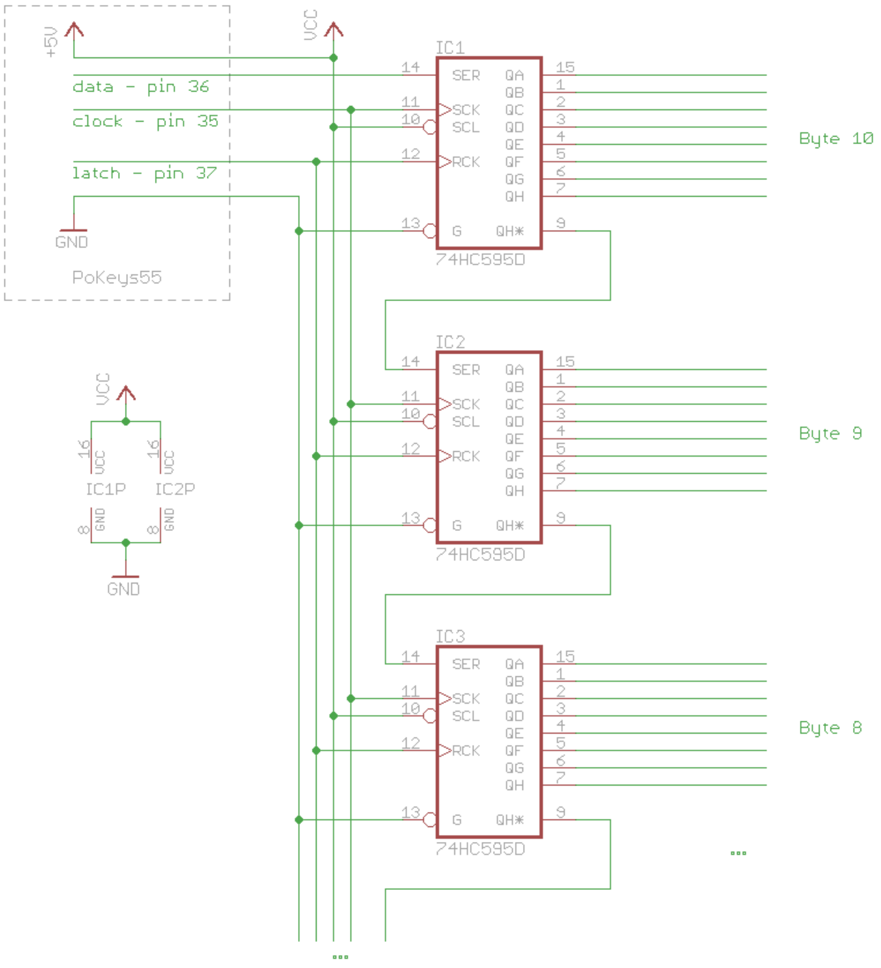
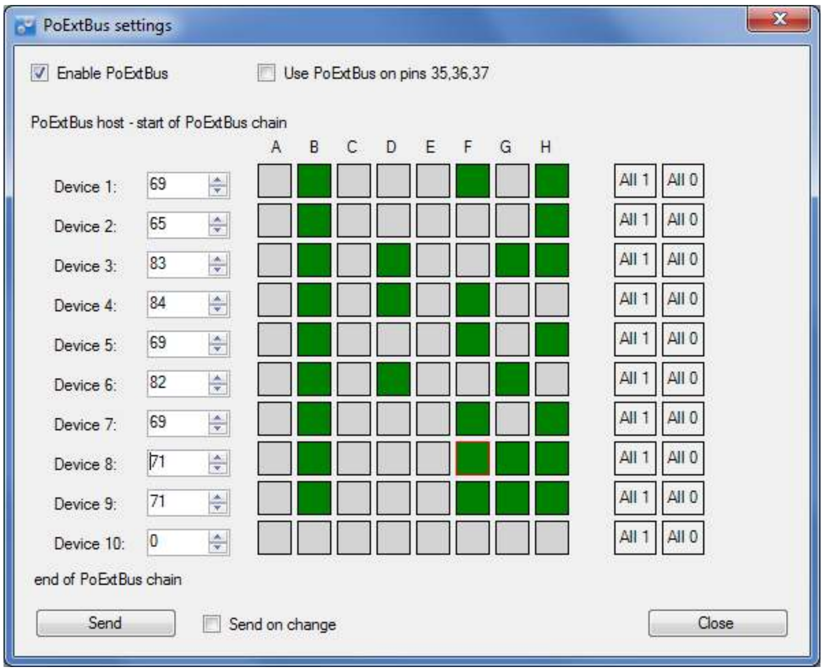


Figure 19: PoExtBus deserialization circuit

PoKeys configuration software usage

To set-up and test PoExtBus, open 'Peripherals' > PoExtBus...'. The dialog below (Figure 20) appears. By using mouse left and right clicks, user can turn on or off each of the outputs.



☑ Enable PoExtBus ☐ Use PoExtBus on pins 35,36,37

PoExtBus host - start of PoExtBus chain

		A	B	C	D	E	F	G	H
Device 1:	69	⬜	🟩	⬜	⬜	⬜	🟩	⬜	🟩
Device 2:	65	⬜	🟩	⬜	⬜	⬜	⬜	⬜	🟩
Device 3:	83	⬜	🟩	⬜	🟩	⬜	⬜	🟩	🟩
Device 4:	84	⬜	🟩	⬜	🟩	⬜	🟩	⬜	⬜
Device 5:	69	⬜	🟩	⬜	⬜	⬜	🟩	⬜	🟩
Device 6:	82	⬜	🟩	⬜	🟩	⬜	⬜	🟩	⬜
Device 7:	69	⬜	🟩	⬜	⬜	⬜	🟩	⬜	🟩
Device 8:	71	⬜	🟩	⬜	⬜	⬜	🟩	🟩	🟩
Device 9:	71	⬜	🟩	⬜	⬜	⬜	🟩	🟩	🟩
Device 10:	0	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜

end of PoExtBus chain

☐ Send on change

Figure 20: PoExtBus setup dialog

Applicable PoKeys library commands

Read or set PoExtBus configuration and status	
AuxilaryBusGetData AuxilaryBusSetData	Read PoExtBus configuration Set PoExtBus configuration and data

PoExtBus connector type

- Female wire-side connector: Molex 22-01-2055
- Cable contacts: 08-50-0032 (5 pcs needed)
- Prepared cables: 88941-0700 (5 pcs needed)

8.12. PoNET

PoKeys56 devices support PoNET devices that extend PoKeys56 functionality. Multiple PoNET and PoExtBus devices can be linked together. While the PoNET devices and PoExtBus devices share the same ExtBus connector, found on PoKeys56 boards, user should pay attention in connecting devices of both types together (see the schematics below).

All PoNET devices must be connected in parallel to each other and directly to the PoKeys56 board, while the PoExtBus devices should be connected in series after the PoNET devices.

Adding new devices

After connecting new PoNET device, go to 'Peripherals > PoNET...'. The following dialog will appear

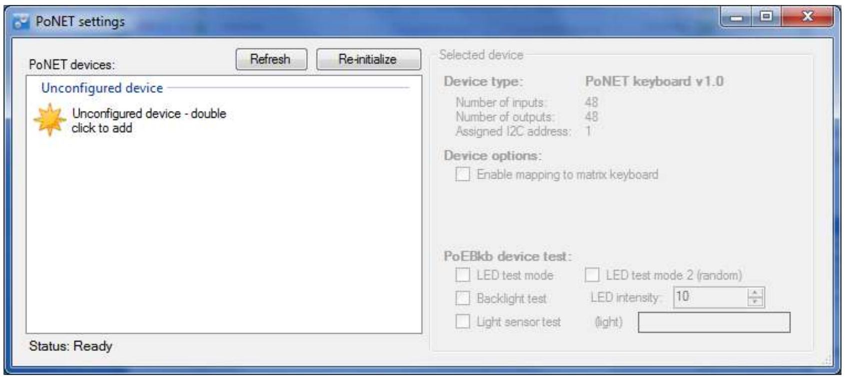


Figure 21: PoNET settings dialog with an unconfigured device

In order to register new device, double click on the 'Unconfigured device' icon. In the next 10 seconds press any key on the device that is about to be added. If the process is successful, status LED on the device will stop blinking and will be constantly lit. The device will also be listed as in the dialog below:

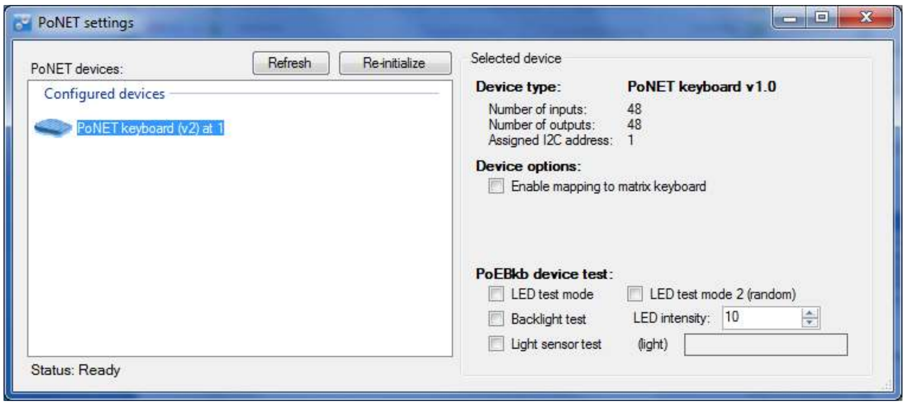
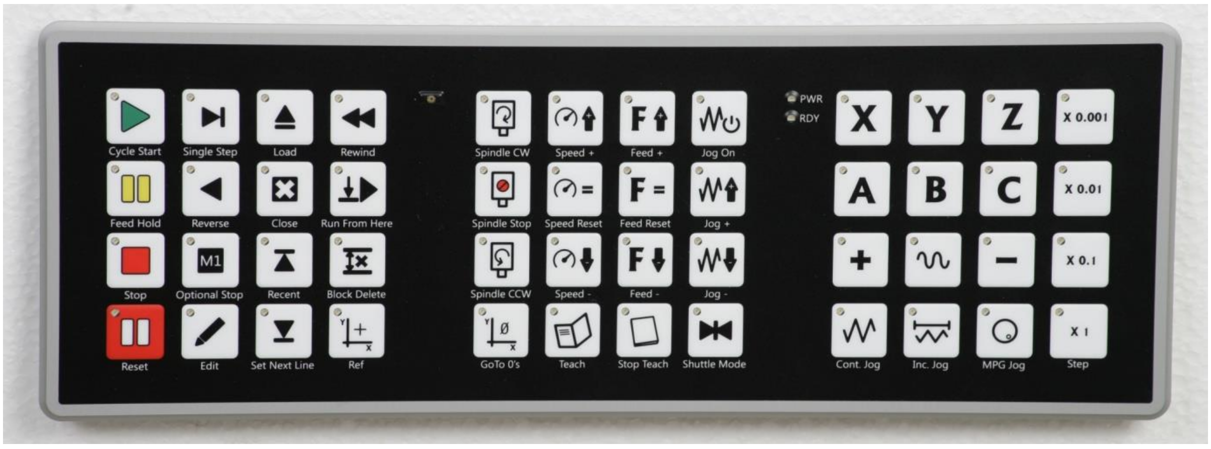


Figure 22: PoNET settings dialog with a successfully configured device

PoNET kb48CNC keyboard



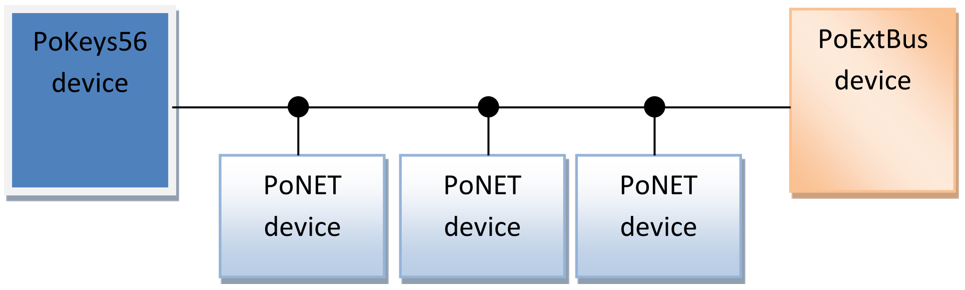
The device can be virually mapped to PoKeys matrix keyboard. Third party software can set the status of LEDs under the keys, read the light sensor that measure the amount of light in the environment and set the intensity of the LEDs.

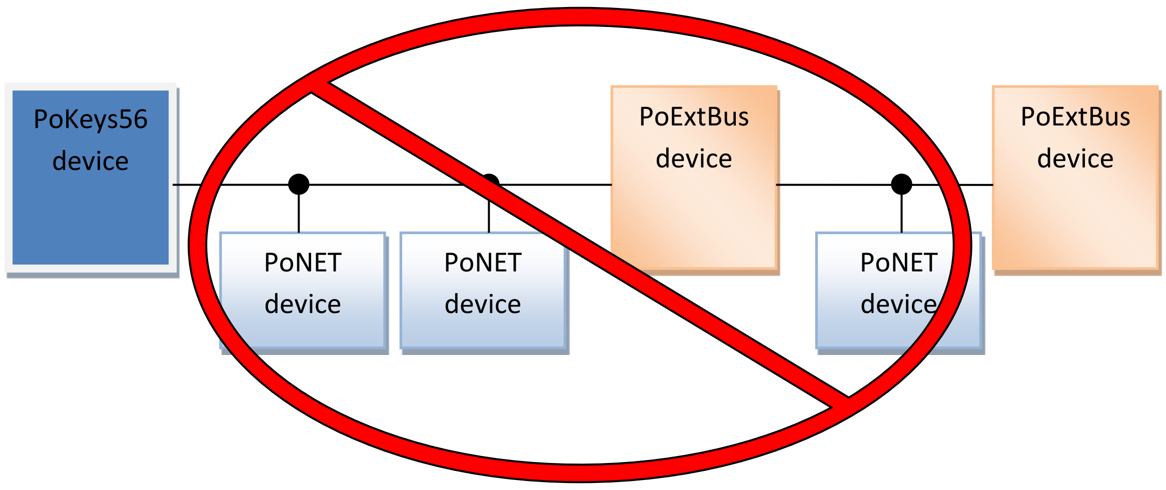
To map the PoNET keyboard to PoKeys matrix keyboard, initialize the PoNET bus as described above in 'Adding new devices', select the keyboard in the device list and check the checkbox 'Enable mapping to matrix keyboard'.

PoNET/PoExtBus connection example

Pay attention to the following:

- **PoNET device can be connected only to PoKeys56 board or other PoNET device**
- **PoExtBus device can be connected to either PoKeys board, PoNET or PoExtBus device.**
- **There should be not PoExtBus device connected between any PoNET devices and PoKeys board**
- **There could be only one PoExtBus device connected besides other PoNET devices**





Applicable PoKeys library commands

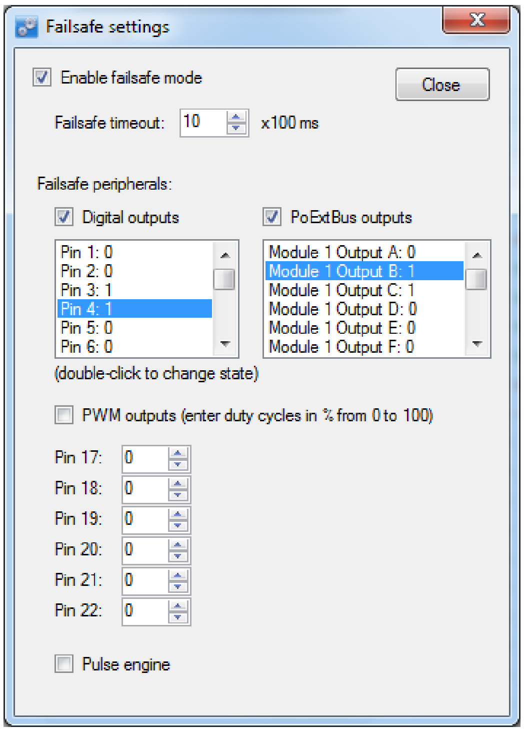
Device discovery and initialization commands	
PoNET_AddDeviceStart	Starts 'Register device' procedure
PoNET_AddDeviceStatus	Retrieves the status of 'Register device' procedure
PoNET_AddDeviceStop	Stops the 'Register device' procedure
PoNET_CheckForNewDevices	Checks for new devices
PoNET_ReinitializePoNET_noClear	Unregisters all PoExtBus Pro devices (but keeps the configuration)
PoNET_ReinitializePoNET_wClear	Unregisters all PoExtBus Pro devices and clears the configuration
Read internal state of the PoI2C engine in PoKeys	
PoNET_GetState	Function returns the internal PoI2C state of the PoKeys device. Look in the protocol specification document under PoI2C settings and communication
Read or write to selected PoNET device	
PoNET_GetFWversion	Reads firmware version
PoNET_GetLightIntensity	Reads light intensity using light sensor
PoNET_GetModuleSettings	Retrieves module settings
PoNET_GetModuleStatus	Reads data from the module
PoNET_SetModuleSettings	Sets the module options
PoNET_SetModuleStatus	Writes data to module
PoNET_SetPWM	Sets PWM duty cycle for LED intensity

8.13. Failsafe settings

PoKeys56U and PoKeys56E/57E devices support the configuration of the failsafe state for the digital outputs, PWM outputs, PoExtBus devices and PoKeys Pulse engine.

When the communication with the device is interrupted for longer than a period defined in the failsafe configuration, peripherals listed above enter the failsafe mode, which can be setup in 'Failsafe settings' dialog (Peripherals > Failsafe settings…)

Peripheral	Failsafe setting
Digital outputs	Active Off / Active On
PoExtBus outputs	Active Off / Active On
PWM outputs	Fixed duty cycle in %
Pulse engine	No setting – Pulse engine enters emergency mode on failsafe activation



☑ Enable failsafe mode

Failsafe timeout: 10 x100 ms

☑ Digital outputs

Pin 1: 0
Pin 2: 0
Pin 3: 1
Pin 4: 1
Pin 5: 0
Pin 6: 0

☑ PoExtBus outputs

Module 1 Output A: 0
Module 1 Output B: 1
Module 1 Output C: 1
Module 1 Output D: 0
Module 1 Output E: 0
Module 1 Output F: 0

(double-click to change state)

☐ PWM outputs (enter duty cycles in % from 0 to 100)

Pin 17: 0
Pin 18: 0
Pin 19: 0
Pin 20: 0
Pin 21: 0
Pin 22: 0

☐ Pulse engine

Figure 23: Failsafe settings dialog

Applicable PoKeys library commands

Failsafe configuration	
N/A	N/A

8.14. I²C protocol

The I2C bus was designed by Philips in the early '80s to allow easy communication between components which reside on the same circuit board. Philips Semiconductors migrated to NXP in 2006. The name I2C translates into "Inter IC". Sometimes the bus is called IIC or I²C bus.

PoKeys56U and PoKeys56E/57E devices support communication with I^{2}C slave devices, connected to the PoExtBus connector. As I^{2}C, PoNET and PoExtBus use the same connector, PoExtBus, PoNET and I^{2}C functions are automatically switched by the PoKeys device.

However, in case of problems with noise caused by the I^{2}C or PoNET devices communication, PoExtBus functionality can be moved to pins 35, 36, 37 as with PoKeys55 devices).

Marking the pin closer to the bottom of the board (the oposite side of either Ethernet or USB connector) as pin 1, the I^{2}C devices should be connected as follows:

Pin 1	Power supply 5V
Pin 2	Ground
Pin 3	Serial data
Pin 4	
Pin 5	Serial clock

Protocol can be tested via PoKeys configuration software. Click on Peripherals > I2C bus test... The following dialog appears.

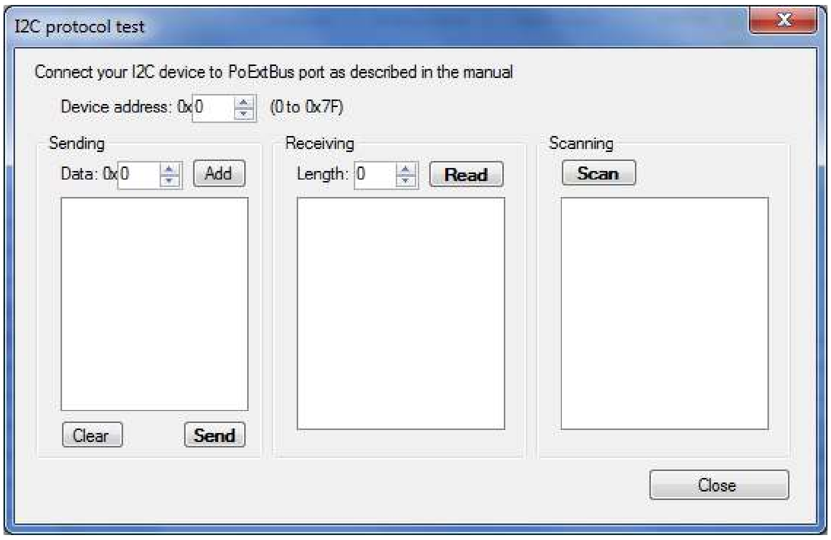


Figure 24: I^{2}C protocol test dialog

Applicable PoKeys library commands

I2C bus commands	
I2CStartBusScan I2CReadBusScanResults	Initiates I2C bus scan for devices Reads results of I2C bus scan
I2CStartRead I2CGetReadStatus	Initiates read from I2C device Read status of reading from I2C bus and retrieve data is

	successfull
I2CStartWrite I2CGetWriteStatus	Writes data to I2C bus Read status of writing to I2C bus

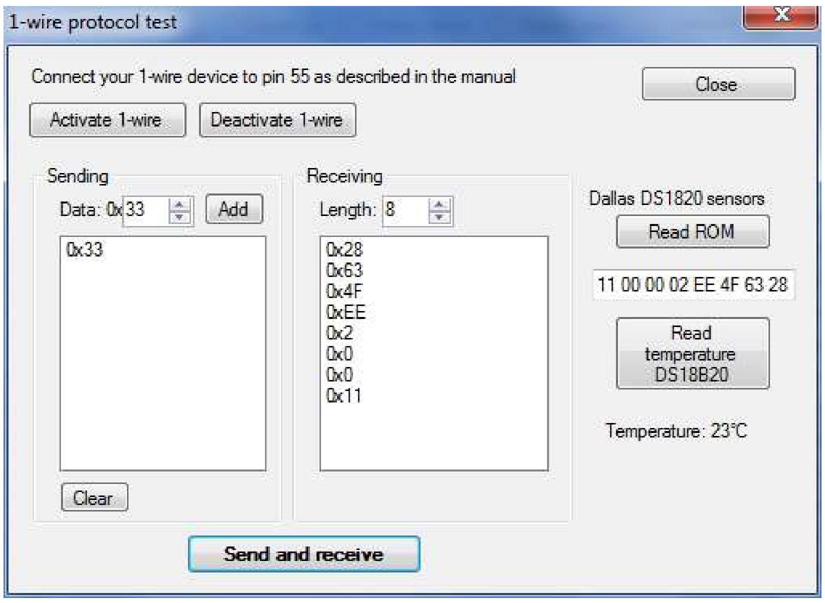
8.15. 1-wire

1-Wire is a device communications bus system designed by Dallas Semiconductor Corp. that provides low-speed data, signaling, and power over a single signal. 1-Wire is similar in concept to I²C, but with lower data rates and longer range. It is typically used to communicate with small inexpensive devices such as digital thermometers and weather instruments.

PoKeys56E/57E devices support communication with 1-Wire slave devices (without parasitic power supply), connected to the pin 55 with external pull-up resistor (of approximately 5 kΩ). In PoKeys57 series devices, pin for 1-wire communication can be changed.

Protocol can be tested via PoKeys configuration software. Click on Peripherals > 1-Wire bus test... The following dialog appears.

The DS18B20 test is deprecated in PoKeys57 series devices. Use EasySensors instead.



0x33

0x28
0x63
0x4F
0xEE
0x2
0x0
0x0
0x11

11 00 00 02 EE 4F 63 28

Temperature: 23°C

Figure 25: 1-Wire protocol test dialog

Applicable PoKeys library commands

1-wire device configuration and status commands	
prot1WireSetStatus prot1WireGetStatus	Activate or deactivate 1-wire bus support on pin 55 Retrieves 1-Wire bus activation status
prot1WireStartWriteAndRead prot1WireGetReadStatus	Initiates write and read process to/from 1-wire device Read status of write and read command and retrieve data if command was executed

8.16. Sensors

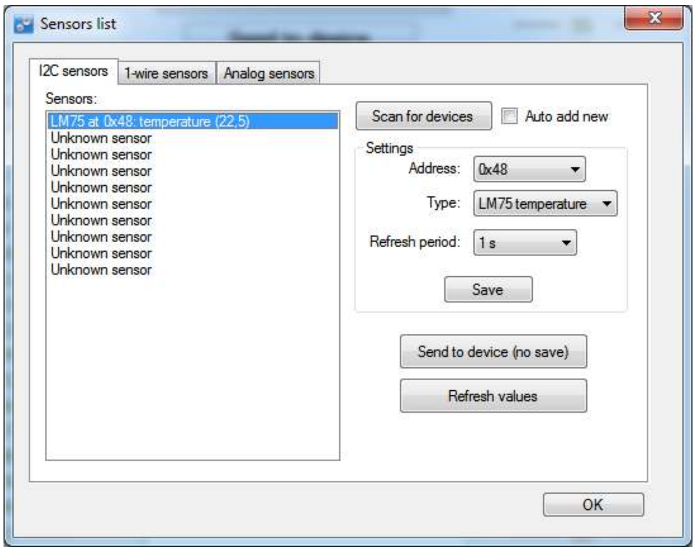
Note: This feature was superseded with EasySensors in PoKeys57 series devices. The content below is related to PoKeys56 series devices

In order to automate the procedure of starting the measurements and reading the results, PoKeys supports up to 10 sensors on I^{2}C bus, up to 10 1-wire sensors and up to 7 analog sensors. The configuration dialog is available through the menu Peripherals>Sensors.

I²C sensors

To setup a I^{2}C sensor, click on an empty ('Inactive') entry in the list, select the sensor's address (the address can be displayed by clicking the 'Scan for devices' command button. A list of addresses of all active I2C devices will be displayed), sensor type and the refresh period. Click Save. Sensor is activated by Sending the configuration to device from the main PoKeys configuration application window. The following list of sensors is supported:

- LM75 temperature sensor for the temperature range -55 °C to +125 °C with the resolution of 0.5 °C. The sensor has configurable address and up to 8 sensors can be connected.
- SHT21 temperature and humidity sensor for the temperature range -40 °C to +125 °C with the resolution of 0.01 °C and air relative humidity in range 0 to 100 %. Due to specifics of the PoKeys sensor list entries, this sensor is represented as temperature sensor and as relative humidity sensor separately. Two entries are needed to read both air temperature and humidity.
- MCP3425 A/D converter with selectable gain
- MMA7660 3-axis accelerometer
- BH1750 light sensor



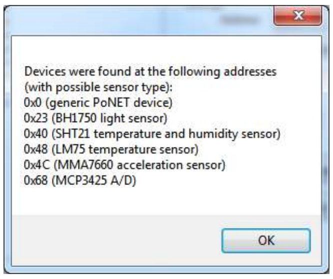
Sensors:
LM75 at 0x48: temperature (22,5)
Unknown sensor
Unknown sensor
Unknown sensor
Unknown sensor
Unknown sensor
Unknown sensor
Unknown sensor
Unknown sensor
Unknown sensor

Address: 0x48
Type: LM75 temperature
Refresh period: 1 s

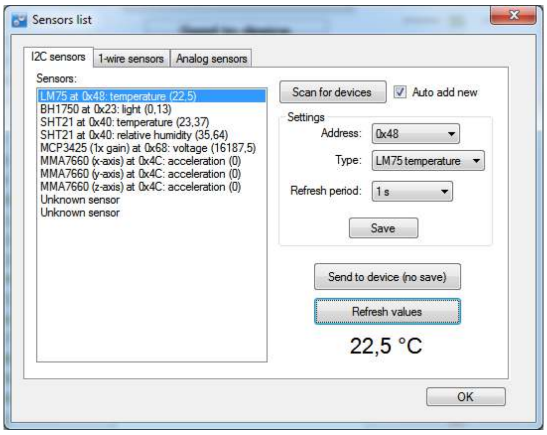
Figure 26: Configuration of I^{2}C sensors

Scanning for I2C sensors

I^{2}C sensors are accessed by a unique 7-bit address on the I^{2}C bus. PoKeys supports checking (scanning) all I^{2}C bus addresses for present devices. To start the scan, click on the 'Scan for devices' button. The operation may take a few seconds to execute and a list of detected I^{2}C device addresses is displayed. As some of the devices support I^{2}C address reconfiguration using dedicated pins and I^{2}C addresses are not unique for each device type, the device list displays only the most possible sensors at each detected address.


Devices were found at the following addresses
(with possible sensor type):
0x0 (generic PoNET device)
0x23 (BH1750 light sensor)
0x40 (SHT21 temperature and humidity sensor)
0x48 (LM75 temperature sensor)
0x4C (MMA7660 acceleration sensor)
0x68 (MCP3425 A/D)

When option ‘Auto add new’ is checked, detected devices that are recognized by the I^{2}C bus address, are automatically added to the sensors list.



Sensors:
LM75 at 0x48: temperature (22,5)
BH1750 at 0x23: light (0,13)
SHT21 at 0x40: temperature (23,37)
SHT21 at 0x40: relative humidity (35,64)
MCP3425 (1x gain) at 0x68: voltage (16187,5)
MMA7660 (x-axis) at 0x4C: acceleration (0)
MMA7660 (y-axis) at 0x4C: acceleration (0)
MMA7660 (z-axis) at 0x4C: acceleration (0)
Unknown sensor
Unknown sensor

[✓] Auto add new

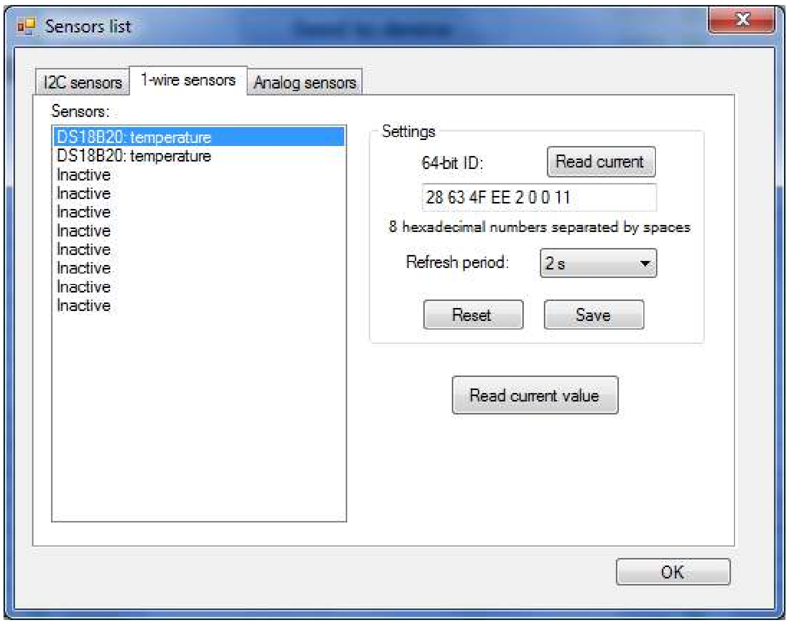
Settings
Address: 0x48
Type: LM75 temperature
Refresh period: 1 s

22,5 °C

1-wire sensors

To setup a 1-wire sensor, click on an empty ('Inactive') entry in the list, enter the sensor's 64-bit ID sequence and select the refresh period. If there is only one 1-wire sensor present on the 1-wire bus, the ID sequence can be read by clicking on the 'Read curent' command button. Multiple sensors can be setup this way – one sensor at a time. When all sensors are then connected to the bus, the PoKeys will identify each with the help of ID sequence. The following list of 1-wire sensors is supported:

- DS18B20 temperature sensor for the temperature range of -55 °C to +125 °C with the resolution of 0.0625 °C. As each sensor has its own unique sequence ID, up to 10 sensors can be connected to PoKeys.
- DS18S20 temperature sensor


Sensors:

DS18B20: temperature
DS18B20: temperature
Inactive
Inactive
Inactive
Inactive
Inactive
Inactive
Inactive
Inactive

Settings

64-bit ID: 28 63 4F EE 2 0 0 11

8 hexadecimal numbers separated by spaces

Refresh period: 2 s

Figure 27: Configuration of 1-wire sensors

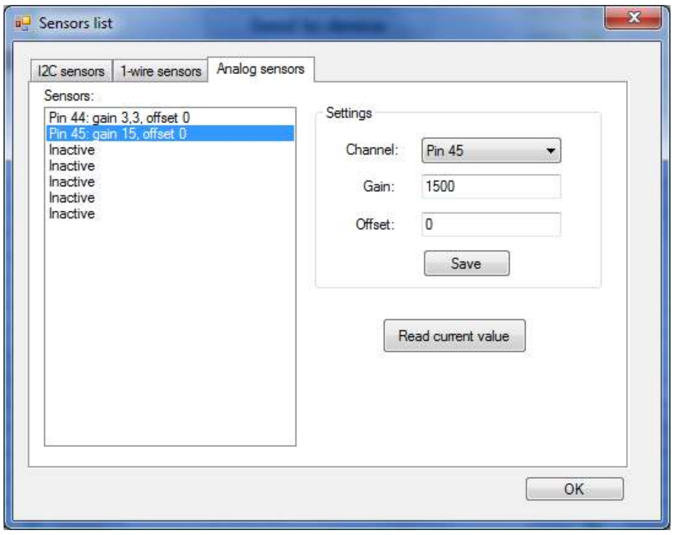
Analog sensors

In addition to I^2C and 1-wire sensors, PoKeys supports also analog sensors that are connected to analog voltage inputs on the PoKeys board. For each analog sensor, a 2-parameter linear transformation is supported. The following formula is used:

$$u = AD_{val} * \frac{A_{gain}}{4096} + A_{offset}$$

Where AD_val is a measurement of the analog-to-digital converter (a value between 0 and 4095), A_{gain} is gain (32-bit integer number) and A_{offset} is result offset (32-bit integer number). The value of u is a integer number that gets divised by 100 for the display (the temperature of 15.58 °C is represented by $u = 1558$. A gain of 330 therefore gives the true voltage on the analog input pin.

Example: the analog temperature sensor outputs the voltage with the following characteristics: 10 mV per °C and 500 mV offset at 0 °C. Gain A_{gain} of 330 will make the value u equal to the measured voltage. Since 1 °C step is equal to 10 mV step in voltage, this gain should be multiplied by a factor of 100, resulting in the gain of 33000. 500 mV offset equals to 50 °C offset, 50 should therefore be multiplied by 100 and get 5000. The final values for gain and offset are therefore 33000 and -5000.


Sensors list

Sensors:
Pin 44: gain 3,3, offset 0
Pin 45: gain 15, offset 0
Inactive
Inactive
Inactive
Inactive
Inactive

Settings
Channel: Pin 45
Gain: 1500
Offset: 0

Figure 28: Configuration of analog sensors

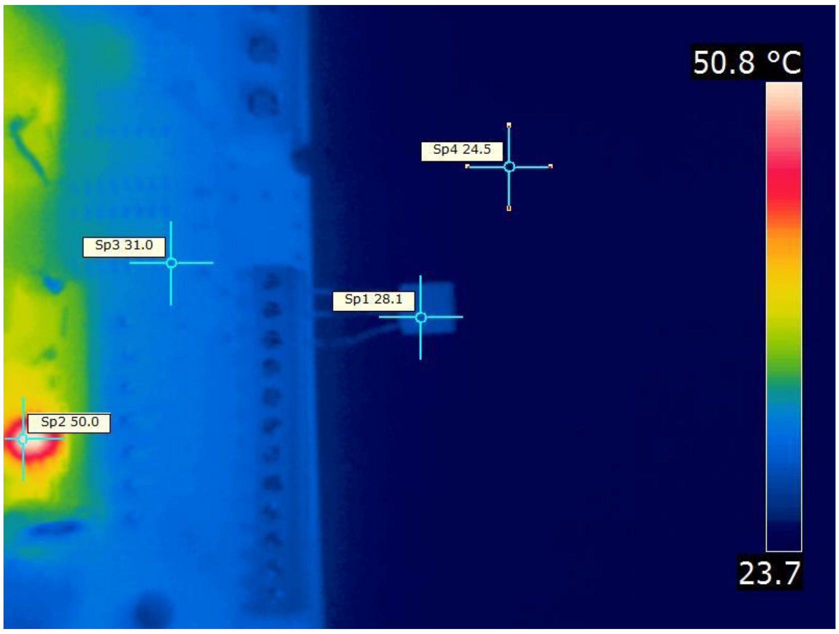


Figure 29: Thermal image of a temperature sensor being connected directly to a circuit board. The thermal conductivity of the sensor's leads causes the sensor to register higher temperature than the ambient real temperature. Use properly longer connections between sensor and sensor host board for the accurate readings

Applicable PoKeys library commands

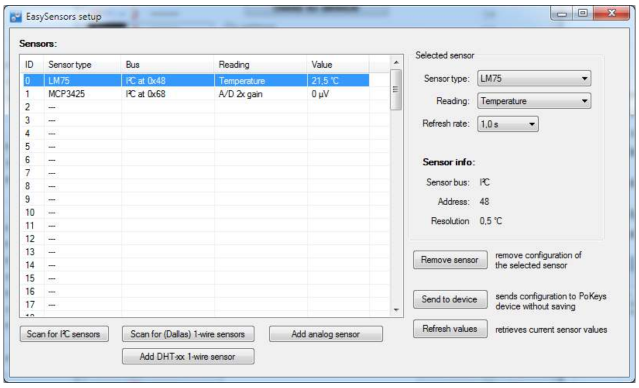
Sensor configuration	
GetSensorSetup SetSensorSetup	Retrieve the sensor setup and current sensor value Send the sensor setup to device
Read sensors (PoTLog27 specific commands)	
GetAllSensorIDs GetAllSensorValues GetAllSensorValuesString	PoTLog27 command to retrieve 64-bit IDs of all connected sensors PoTLog27 command to retrieve the values of all sensors PoTLog27 command to retrieve the values of all sensors in one string

8.17. EasySensors

PoKeys57 series devices implement a feature called EasySensors. It is an improvement of the original support for various sensors in PoKeys56 series devices. EasySensors feature allows the user to setup up to 100 sensors on various communication buses (including I^{2}C, 1-wire, DHTxx 1-wire and analog inputs). The feature is accessible in Peripherals > EasySensors menu.

EasySensors configuration dialog

The main EasySensors configuration dialog contains a list of all configured sensors with corresponding sensor information. Here, the sensory type, reading and refresh rate can be changed.



ID	Sensor type	Bus	Reading	Value
0	LM75	I²C at 0x48	Temperature	21,5 °C
1	MCP3425	I²C at 0x68	A/D 2x gain	0 µV
2	---			
3	---			
4	---			
5	---			
6	---			
7	---			
8	---			
9	---			
10	---			
11	---			
12	---			
13	---			
14	---			
15	---			
16	---			
17	---			

Sensor type: LM75
Reading: Temperature
Refresh rate: 1,0 s

Sensor info:
Sensor bus: I²C
Address: 48
Resolution 0,5 °C

Remove sensor – remove configuration of the selected sensor
Send to device – sends configuration to PoKeys device without saving
Refresh values – retrieves current sensor values

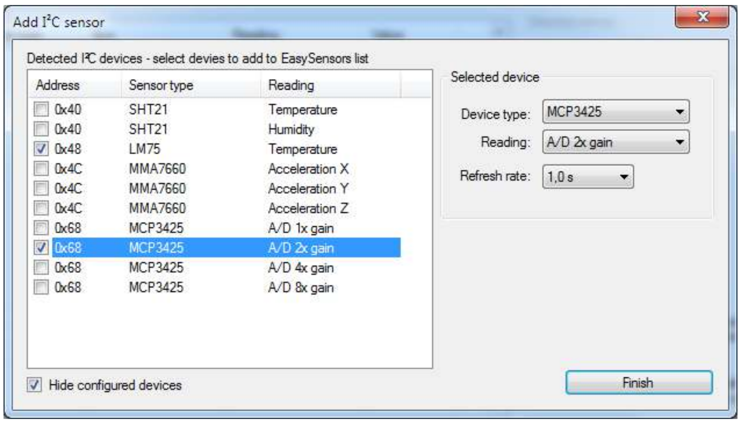
Figure 30: EasySensors configuration dialog

The changed sensor entries are marked with red color - in order for the changes to take effect, click on 'Send to device' button, which transfers the EasySensors configuration to device (note that this action does not save the settings to device's non-volatile memory and 'Send to device' on main PoKeys configuration window must be clicked to do that).

There are 4 types of sensors supported and each type of sensor can be added by clicking a corresponding button at the bottom of the dialog.

Scan for I^{2}C sensors

This command opens the 'Add I^{2}C sensor' dialog, which is automatically populated with detected I^{2}C devices and sensor type suggestions. Sensor type, reading and refresh rate can be changed using the selections on the right part of the diagram. In order to add the configuration of I2C sensor, check the corresponding item in the detected I^{2}C devices list and click on 'Finish.



Detected I²C devices - select devies to add to EasySensors list

Address	Sensor type	Reading
☐ 0x40	SHT21	Temperature
☐ 0x40	SHT21	Humidity
✓ 0x48	LM75	Temperature
☐ 0x4C	MMA7660	Acceleration X
☐ 0x4C	MMA7660	Acceleration Y
☐ 0x4C	MMA7660	Acceleration Z
☐ 0x68	MCP3425	A/D 1x gain
✓ 0x68	MCP3425	A/D 2x gain
☐ 0x68	MCP3425	A/D 4x gain
☐ 0x68	MCP3425	A/D 8x gain

✓ Hide configured devices

Device type: MCP3425
Reading: A/D 2x gain
Refresh rate: 1,0 s

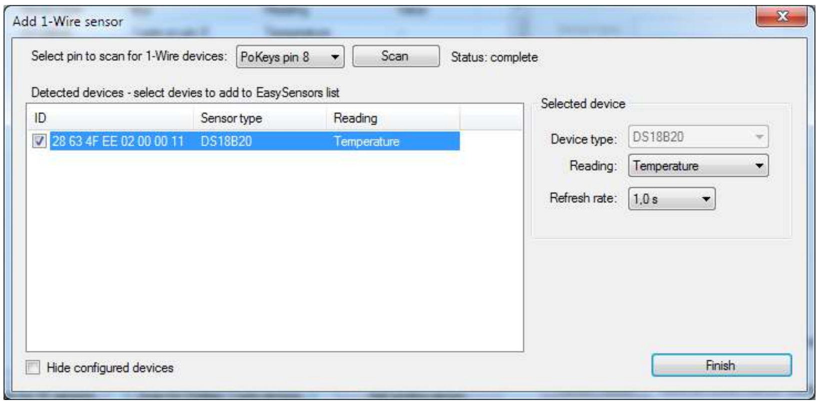
Figure 31: Scanning for I^{2}C sensors

Scan for 1-wire sensors

The command opens the ‘Add 1-Wire sensor’ dialog that allows the user to select PoKeys pin, where the 1-Wire bus is connected to. By clicking ‘Scan’, PoKeys device scans the 1-Wire bus for devices. The EasySensors 1-wire bus implementation can detect multiple 1-Wire devices on the bus at once and thus simplifying the configuration process.

The list of available devices is then shown in the ‘Detected devices’ list (if ‘Hide configured devices’ option is selected, already configured 1-Wire devices are not displayed in the list). On the right side, reading type and refresh rate of the selected sensor can be configured.

To add the sensor to the EasySensors list, check the checkbox of the sensor in the list and click ‘Finish’.



Add 1-Wire sensor

Select pin to scan for 1-Wire devices: PoKeys pin 8

Status: complete

Detected devices - select devies to add to EasySensors list

ID	Sensor type	Reading
✓ 28 63 4F EE 02 00 00 11	DS18B20	Temperature

Selected device

Device type: DS18B20

Reading: Temperature

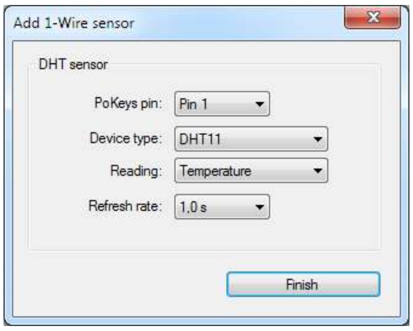
Refresh rate: 1,0 s

Hide configured devices

Figure 32: Scanning for I^2C sensors

Add DHTxx 1-Wire sensor

Three types of DHTxx 1-Wire sensors are supported by PoKeys device (sensors differ in temperature and humidity resolution and accuracy). Any of the available PoKeys pins can be selected as DHTxx 1-Wire bus, but only one sensor can be connected per pin.



Add 1-Wire sensor

DHT sensor

PoKeys pin: Pin 1

Device type: DHT11

Reading: Temperature

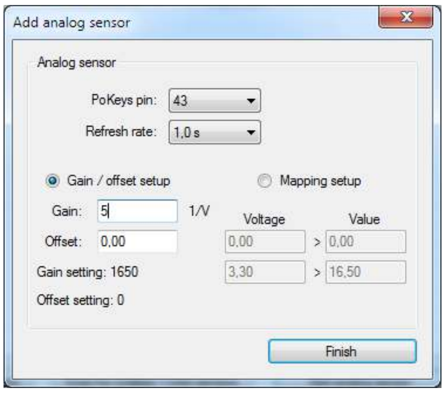
Refresh rate: 1,0 s

Figure 33: Adding the DHTxx 1-Wire sensor

Add analog sensor

EasySensors feature supports reading of simple analog sensors connected to PoKeys analog inputs. A linear transformation is applied to analog input reading to produce the sensor value. There are two possible ways to set up the sensor

a) Using gain and offset: specify the gain and offset characteristics of the sensor. The gain specifies the number of sensor units per 1 V of analog voltage, while the offset specifies the sensor value of analog voltage input of zero.


Add analog sensor

Analog sensor

PoKeys pin: 43

Refresh rate: 1,0 s

[✓ Gain / offset setup] Mapping setup

Gain: 5 1/V

Offset: 0,00

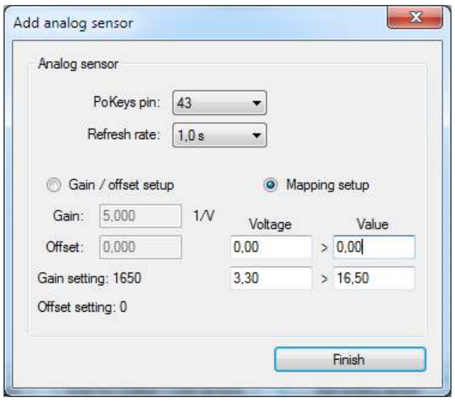
Gain setting: 1650

Offset setting: 0

Voltage		Value
0,00	>	0,00
3,30	>	16,50

Figure 34: Analog sensor setup with gain / offset option

b) Using two point mapping: in this mode, two sensor values with corresponding analog voltage must be entered into the fields provided. Gain and offset of the sensor are automatically calculated.


Add analog sensor

Analog sensor

PoKeys pin: 43

Refresh rate: 1,0 s

Gain / offset setup [✓ Mapping setup]

Gain: 5,000 1/V

Offset: 0,000

Gain setting: 1650

Offset setting: 0

Voltage		Value
0,00	>	0,00
3,30	>	16,50

Figure 35: Analog sensor setup with mapping

Analog sensors use the following formula to convert the analog input voltage into sensor reading

$$u = AD_{val} * \frac{A_{gain}}{4096} + A_{offset}$$

where AD_val is a measurement of the analog-to-digital converter (a value between 0 and 4095), A_{gain} is gain (32-bit integer number) and A_{offset} is result offset (32-bit integer number). The value of u is a integer number that gets divised by 100 for the display (the temperature of 15.58 °C is represented by $u = 1558$. A gain of 330 therefore gives the true voltage on the analog input pin. The analog sensor dialog shows the gain and offset settings in the bottom left corner of the dialog and should not be confused with Gain and Offset setup parameters.

List of supported sensors

- LM75 (Maxim, TI, NXP) temperature sensor for the temperature range -55 °C to +125 °C with the resolution of 0.5 °C. The sensor has configurable address and up to 8 sensors can be connected.
- SHT21 (Sensirion) temperature and humidity sensor for the temperature range -40 °C to +125 °C with the resolution of 0.01 °C and air relative humidity in range 0 to 100 %.
- Si7020 (SiLabs) air temperature and relative humidity sensor
- Si1141 (SiLabs) IR and visible light intensity and reflection
- MCP3425 (Microchip) A/D converter with selectable gain
- MCP9600 (Microchip) thermocouple to temperature converter
- iAQ-Core C (AMS) indoor air quality module, measures VOC (volatile organic compound) and CO_2 levels
- MMA7660 (NXP) 3-axis accelerometer
- LIS2DH12 (ST) 3-axis accelerometer
- BH1750 (Rohm) light sensor
- BME280 (Bosch) air temperature, relative humidity and absolute pressure
- ADS1115 (TI) 16-bit 4-channel A/D converter with selectable gain (up to 16x)
- DS18B20 (Maxim) temperature sensor for the temperature range of -55 °C to +125 °C with the resolution of 0.0625 °C. As each sensor has its own unique sequence ID, up to 10 sensors can be connected to PoKeys.
- DS18S20 (Maxim) temperature sensor
- DS2413 (Maxim) GPIO inputs
- DHT11, DHT21 and DHT22 (AM2302) temperature and humidity sensors

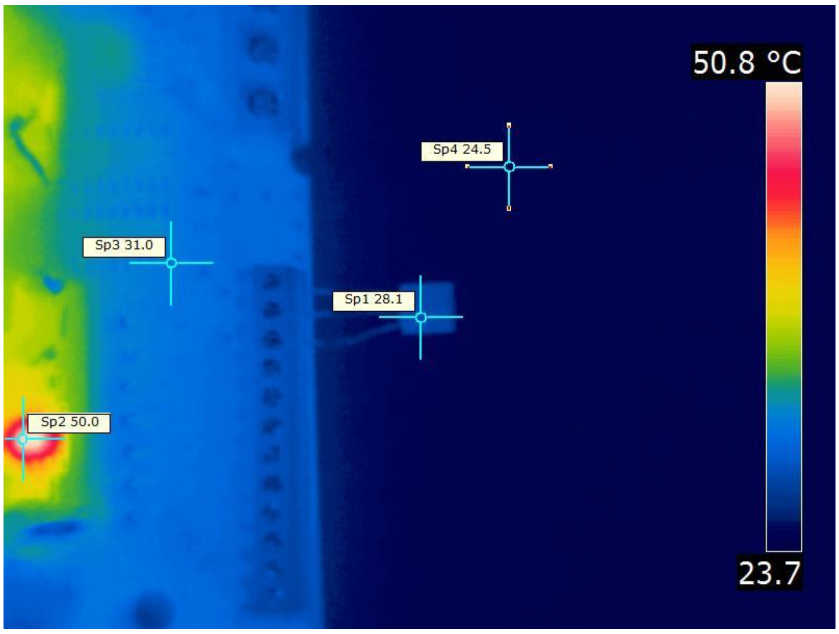


Figure 36: Thermal image of a temperature sensor being connected directly to a circuit board. The thermal conductivity of the sensor's leads causes the sensor to register higher temperature than the ambient real temperature. Use properly longer connections between sensor and sensor host board for the accurate readings

8.18. USB interface configuration

Note: some configuration options may be limited to PoKeys57 series

The operation of the USB PoKeys devices can be adjusted in the Device > USB interface', as shown on the figure below.

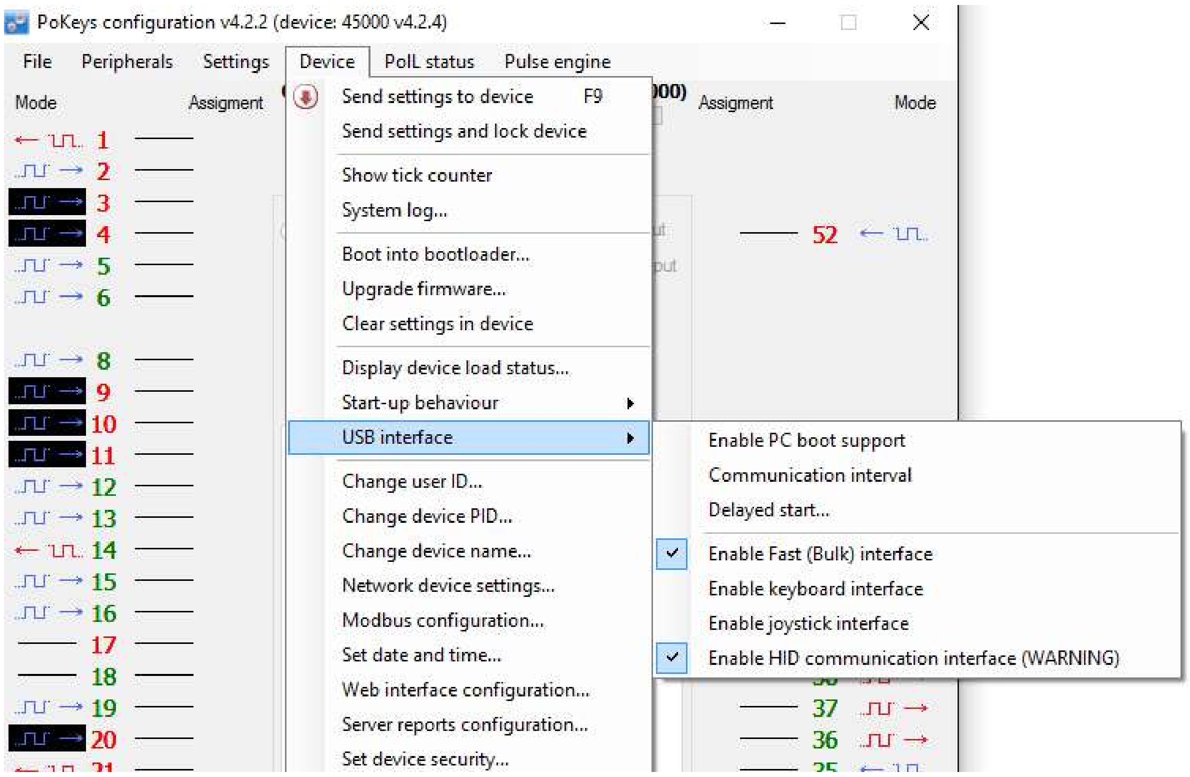


Figure 37: Accessing USB interface options

By adjusting these options, the user is given the possibility to configure how device reacts on the system start, change the communication interval and configure which interfaces are visible to the system. In some situations, omitting the unused interfaces can result in achieving better performance of the device due to lower load to the system drivers.

Start options

USB PoKeys devices can be configured to either report the support for boot operation or not. Some PC BIOS versions have problems configuring non-simple (plain keyboard and mouse) devices and may halt the boot sequence if such device is present. By omitting the PC boot support, this is usually overcome.

If this however does not fix the problems during the boot sequence, delayed start option give the possibility to delay the PoKeys device registration on USB for the predetermined delay, which will result in BIOS not detecting the device and continuing the boot sequence. Adjust the value according to your system in this case.

Communication interval

By default, PoKeys USB devices use 1 millisecond communication interval for the communication interface. If for any reason slower communication is required, it can be adjusted using the

Communication interval settings dialog. Communication intervals between 1 millisecond and 20 milliseconds can be configured.

Enabling/disabling the interfaces

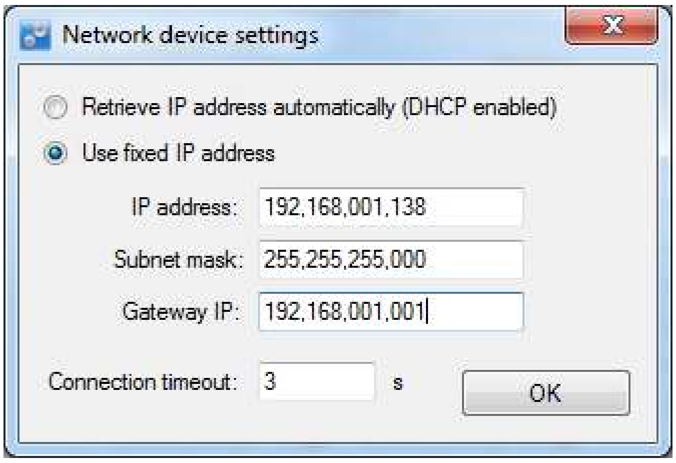
Each USB PoKeys device uses 4 USB interfaces (i.e. USB devices) and thus appears as 'USB Composite Device' in the Device manager. If not needed, some (or all) interfaces can be disabled.

Note: if all communication interfaces are disabled, configuration of the device will no longer be possible. In that case, follow the instructions in the section 'Restoring factory defaults' to restore the device's functionality.

8.19. PoKeys56E and PoKeys57E

PoKeys56E/57E is a network type of PoKeys device. The device can be connected to Ethernet 10/100 network with standard RJ-45 cable.

By default, the device is set to use the DHCP functionality of the network router. User can later turn on or off the DHCP support. If DHCP is not needed/wanted, fixed IP address can be defined for the device. To set the network settings of the device, go to Device menu and click Network device settings...


Network device settings

Retrieve IP address automatically (DHCP enabled) [✓ Use fixed IP address]

IP address: 192,168,001,138

Subnet mask: 255,255,255,000

Gateway IP: 192,168,001,001

Connection timeout: 3 s

Figure 38: Device network settings

The device communicates using TCP and UDP port of 20055. **Please ensure the firewall settings allow communication with this port. Also, please make sure that your network card (which you have connected PoKeys56E/57E to) has an IP address assigned with the subnet mask 255.255.255.0 (check it in IPv4 settings in your system).**

By default, PoKeys56E/57E device is configured to close the connection with the host after 3 seconds of inactivity. This connection timeout value can be set in the dialog, shown in Figure 38.

Connecting to PoKeys56E/57E device for the first time

1. Connect PoKeys56E/57E with your Ethernet network using the RJ-45 cable
2. Connect power supply for PoKeys56E/57E board
    a. If your network uses DHCP to assign IP addresses to each device, PoKeys56E/57E will be assigned a new IP address automatically
    b. If your network uses fixed IP addresses, PoKeys56E/57E device will wait for discovery packet from the PoKeys configuration software (during this time, LED will blink). In this process, temporary IP address from the same subnet will be assigned to PoKeys56E/57E board.
3. Open PoKeys configuration software and wait for devices to be detected
4. If different network configuration is needed, select device and click on Configure button to select new settings.
5. Click on Connect button to connect to PoKeys56E/57E device and start configuring

Device discovery

If device IP address is set as fixed, use can connect directly to PoKeys56E/57E device. Otherwise, UDP discovery packet should be send as UDP broadcast packet (for details, please see the protocol specification document). All PoKeys56E/57E devices that receive this packet, respond with their current IP address and serial number.

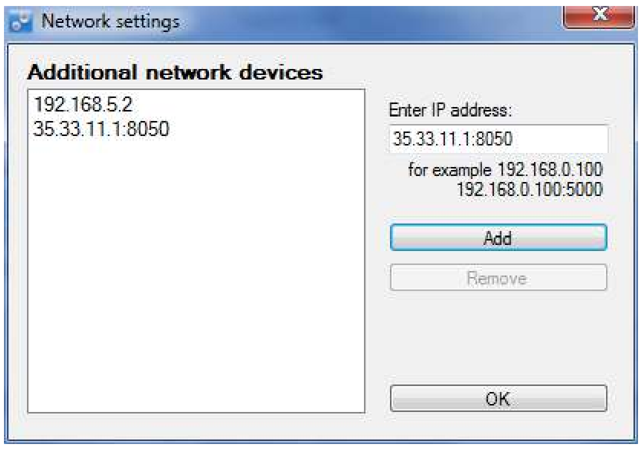
Default settings

DHCP:	**enabled**
Port:	20055
Security:	Full access

After receiveing the UDP discovery packet and if the DHCP server is not available, PoKeys56E/57E will use the temporary address of x.x.x.250, where x.x.x is the subnet address (with 255.255.255.0 subnet mask). This enables user to reconfigure the device with the proper IP address. To do this, click on the Configure button.

Connecting to device in other network

When the device is not detected automatically (either there is a firewall blocking the UDP broadcast messages or the device is not in the same network as a computer), custom IP address of the device can be entered by clicking on the 'Network settings... ' button. The following dialog appears.



192.168.5.2
35.33.11.1:8050

35.33.11.1:8050

for example 192.168.0.100
192.168.0.100:5000

Figure 39: Additional network settings

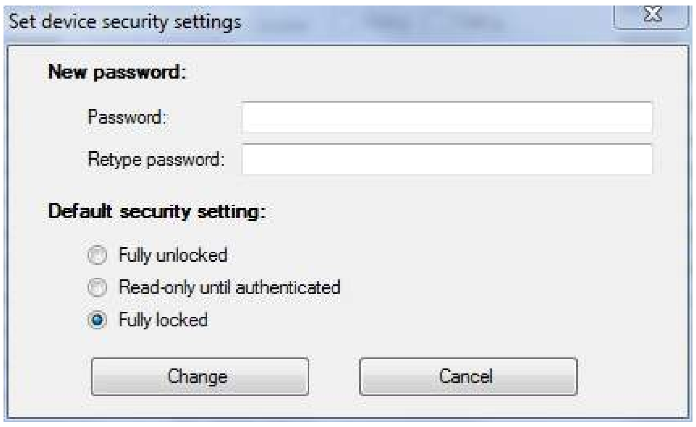
IP address of the device can be entered in the text box on the right and added to the list by clicking the button 'Add'. The list of additional devices is saved on application exit.

Security

Due to exposed nature of a network device, an authentication mechanism was implemented in PoKeys56E/57E that allows three levels of access rights:

- Full access (default): the device is fully accessible from the network
- Read-only access: unauthorized users are allowed only to fetch data from the device, while an authenticated users can acccess all functions of the device
- Full lock: unauthorised users can not neither read or write to the device. A user password is required to unlock access.

The security is set up in PoKeys configuration software – on the Device menu, click Set device security... The password can contain any character and can be up to 32 characters long.


Set device security settings

New password:

Password:

Retype password:

Default security setting:

- Fully unlocked
- Read-only until authenticated
- [✓ Fully locked]

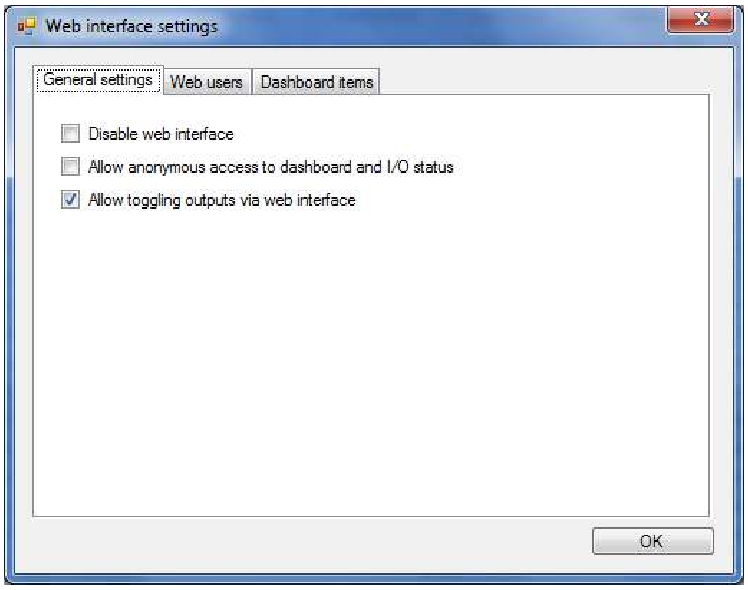
Figure 40: Device security settings window

Applicable PoKeys library commands

PoKeys56E/57E user authorization commands	
AuthorizeUser	Authorizes user on the device - changes the security level to the specified value
GetSecurityStatus	Reads the current security status of network device

Web interface

PoKeys56E/57E devices can be monitored through the simple web interface (that is already enabled by default). The interface can be disabled or configured in dialog accessible via menu Device->Web interface configuration. The following dialog appears

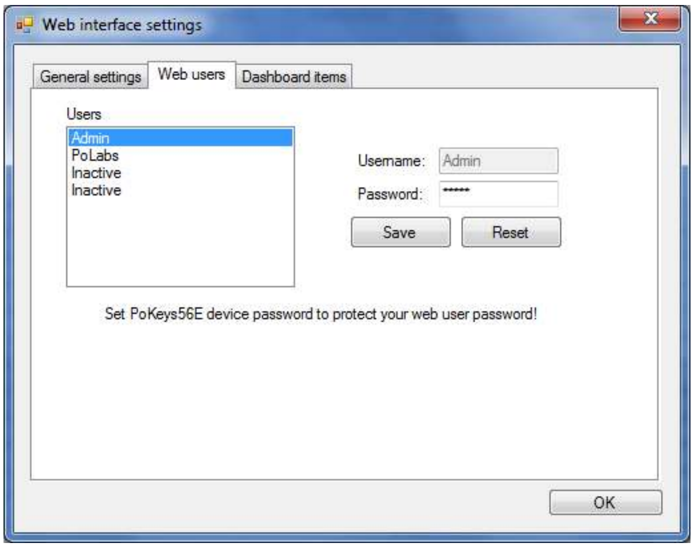


☐ Disable web interface
☐ Allow anonymous access to dashboard and I/O status
☑ Allow toggling outputs via web interface

Figure 41: Web interface settings – general settings

The dialog presents the following options on the 'General settings' tab:

- Disable web interface: check this field to disable web interface.
- Allow anonymous access to dashboard and I/O status: if this field is checked, users can access web interface directly without entering user name and password.
- Allow toggling outputs via web interface: if this field is checked, users can toggle the pins that are setup as outputs. If this field is unchecked, users are only presented with the status of each pin.



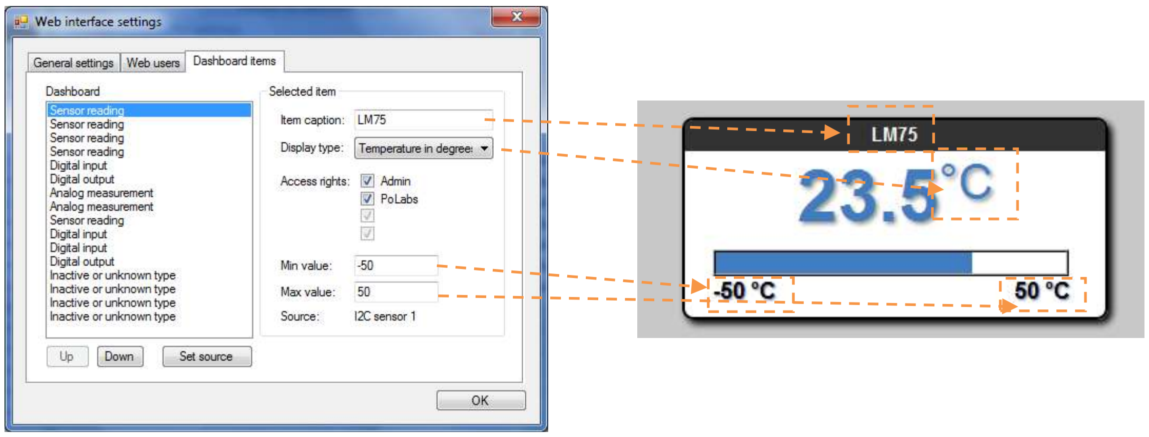
Users
Admin
PoLabs
Inactive
Inactive

Username: Admin
Password: ******

Set PoKeys56E device password to protect your web user password!

Figure 42: Web interface settings – web users setup

On the 'Web users' tab, usernames and passwords can be configured for up to 4 users (first user is fixed and named 'Admin' with the factory set password 'root0'). Usernames (except for Admin) and passwords can be up to 8 (ASCII) characters long.



Dashboard:
Sensor reading
Sensor reading
Sensor reading
Sensor reading
Digital input
Digital output
Analog measurement
Analog measurement
Sensor reading
Digital input
Digital input
Digital output
Inactive or unknown type
Inactive or unknown type
Inactive or unknown type
Inactive or unknown type

Selected item
Item caption: LM75
Display type: Temperature in degree:
Access rights: [✓ Admin] [✓ PoLabs]
Min value: -50
Max value: 50
Source: I2C sensor 1

Figure 43: Web interface settings – dashboard items

This page of Web interface settings is used to defined which items will be displayed on the Dashboard page of web interface (see below). PoKeys supports up to 16 items on dashboard screen. Each item displays the status of one sensor, one input or one output.

To set up a new dashboard item, select a 'Inactive or unknown type' entry in dashboard list and press 'Set source' command button. In a list that appears, select either 'Digital input', 'Digital output' or 'Sensor'. For the digital inputs and outputs, a dialog for entering pin number will appear, while for the sensors, a sensor selection dialog will appear. After selecting either digital input/output or sensor source, additional options will be available. For each item in the list, a 8 character item caption can be assigned. Available display types depend on the selected item data source (see Table 4 for details). For display types that include a progress bar, additional min and max value for the progress bar can be specified.

Item data source	Supported display types
Digital input	Digital input status
Digital output	Digital output status
	Digital output status with buttons
Sensor	Value in V
	Value in V and bar graph
	Value in mA
	Value in mA and bar graph
	Value in A
	Value in A and bar graph
	Temperature in degrees C
	Temperature in degrees C and bar graph
	Relative humidity in %
	Relative humidity in % and bar graph

Table 4: Display types for different item data sources

Access rights settings enables administrator to select which users will be able to display the sensor readings. This is intented mainly for display items that can change outputs.

To open the interface, use your internet browser and type in the IP address of the PoKeys56E/57E board. After entering the username and password, the following page will be displayed:

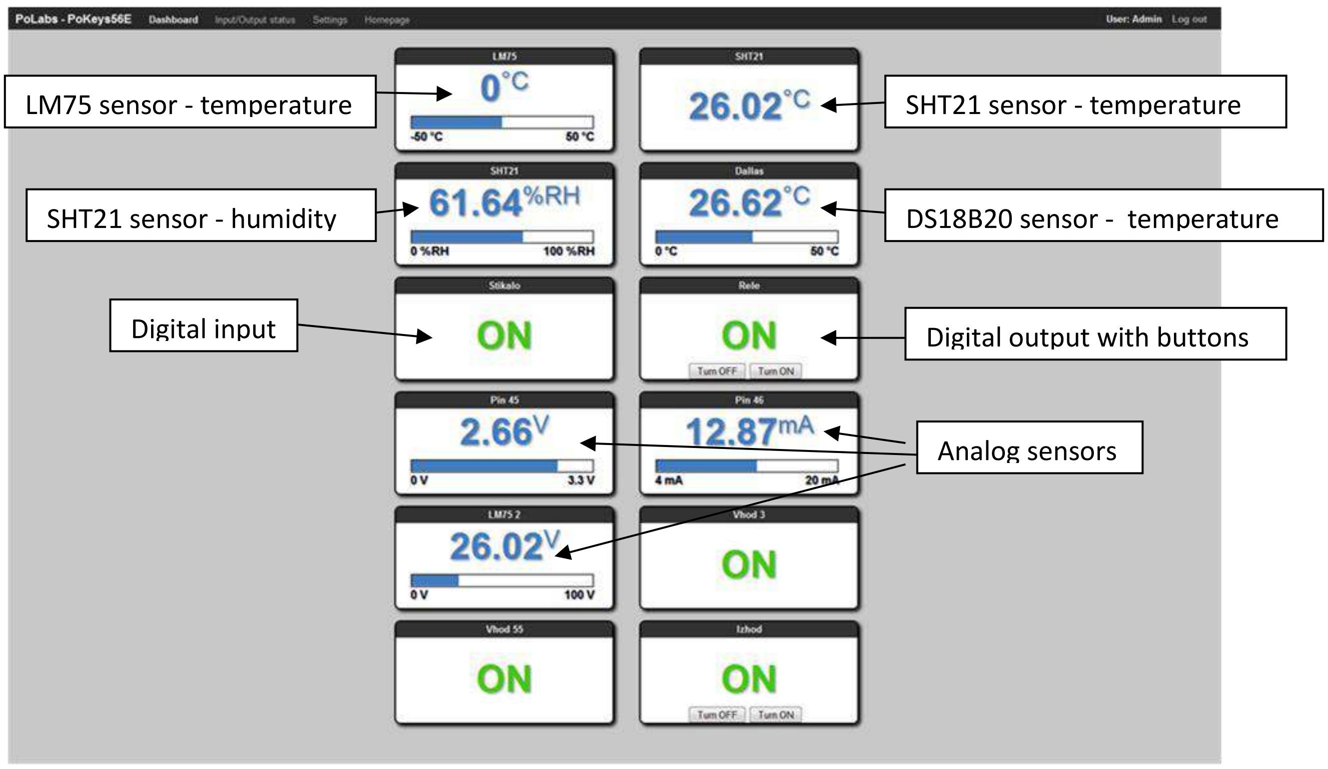


Figure 44: PoKeys56E dashboard

Developers can access status of the inputs by fetching the file /devStat.xml from PoKeys56E/57E board.

Applicable PoKeys library commands

Web interface configuration commands	
GetWebSettings SetWebSettings	Retrieve web interface settings (enable/disable interface, anonymous access, outputs enable) from the device Send web interface settings (enable/disable interface, anonymous access, outputs enable) to device
GetWebUserData SetWebUserData	Retrieve user name and password for the selected web user account Send user name and password for the selected web user account
GetDashboardItem SetDashboardItem	Retrieve the dashboard item setup Send the dashboard item setup

Modbus

PoKeys56E/57E supports slave (server) operation of Modbus TCP communication protocol. Modbus TCP compatible devices on the network can read the values from the device and set the outputs. To elevate the security, user can define which peripherals are accessible via Modbus TCP.

Modbus TCP uses TCP protocol on port 502 (default), which can be changed in Modbus settings (accessible from the menu Device – Modbus configuration...). The Modbus TCP connection is disconnected after 3 seconds of inactivity (this default value can be changed in the Modbus settings).

Discrete inputs/outputs

Supported operations:

0x01: Read coils

0x02: Read discrete input

0x05: Write single coil

0x0F: Write multiple coils

Address (0-based)	Access (R – Read, W – Write)	Description
0-54	R/W	55 pin inputs/outputs
100-126 (100-149 on PoKeys57 series)	R	Sensor OK statuses
200-263	R/W	PoIL shared data (binary data) - overlapped with 32-bit PoIL shared data at 1000-1127 (16-bit)
1000-1127	R	Matrix keyboard inputs
1400-1527	W	I2C Matrix keyboard LED
1600-1727	R/W	LED matrix
2000-2079	R/W	PoExtBus
2000		Device 10 – Output H
2001		Device 10 – Output G
…		
2079		Device 1 – Output A

Registers

Supported operations:

0x03: Read holding register

0x04: Read input register

0x06: Write single register

0x10: Write multiple registers

Address (0-based)	Access (R – Read, W – Write)	Description
0-1	R	Serial number of the device (PoKeys57 only)
10-16	R	Analog inputs
20-45	RW	Encoder counter values (lower 16-bit)

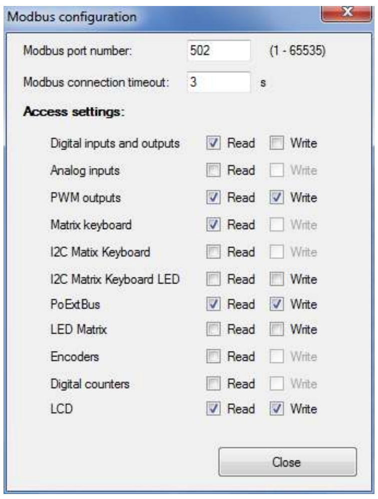
100-154	RW	Digital counter values
200-213	RW	PWM
200,201		PWM period (MSB first)
202,203		PWM duty1 (MSB first) – pin 22
…		
212,213		PWM duty6 (MSB first) – pin 17
300-304	RW	PoExtBus
400-453 (400-499 on PoKeys57 series)	R	Sensors (32-bit values, LSB first)
500-579	RW	LCD buffer
590	W	LCD configuration (0=disabled, 1=primary or 2=secondary) - writing to this register will re-init and clear the LCD
591	W	Number of rows (lower byte) and number of columns (upper byte) of the LCD module
592	W	Not used
593	W	Clear LCD (both bytes = 0xAA)
600	R	Tick counter (lower 16-bit)
610	RW	RTC seconds
611	RW	RTC minutes
612	RW	RTC hours
613	RW	RTC day
614	RW	RTC day of week
615	RW	RTC month
616	RW	RTC year
700-751	R[W]	Digital encoder values (32-bit values, LSB first) - any write to these registers causes the reset of the encoder value to 0
800-909	RW	Digital counter values (32-bit values, LSB first)
1000-1127	RW	PoIL shared data (32-bit, LSB first)

PoExtBus channel mapping:

Address (0-based)	**Register description**															
300	A	B	C	D	E	F	G	H	A	B	C	D	E	F	G	H
	Device 10								**Device 9**							
301	A	B	C	D	E	F	G	H	A	B	C	D	E	F	G	H
	Device 8								**Device 7**							
302	A	B	C	D	E	F	G	H	A	B	C	D	E	F	G	H
	Device 6								**Device 5**							
303	A	B	C	D	E	F	G	H	A	B	C	D	E	F	G	H
	Device 4								**Device 3**							
304	A	B	C	D	E	F	G	H	A	B	C	D	E	F	G	H
	Device 2								**Device 1**							

where

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Modbus word bit
A	B	C	D	E	F	G	H	A	B	C	D	E	F	G	H	PoExtBus device bit mapping
Device 10								Device 9								


Modbus configuration

Modbus port number:	502	(1 - 65535)
Modbus connection timeout:	3	s

Access settings:

Digital inputs and outputs	✓ Read	Write
Analog inputs	Read	Write
PWM outputs	✓ Read	✓ Write
Matrix keyboard	✓ Read	Write
I2C Matix Keyboard	Read	Write
I2C Matrix Keyboard LED	Read	Write
PoExtBus	✓ Read	✓ Write
LED Matrix	Read	Write
Encoders	Read	Write
Digital counters	Read	Write
LCD	✓ Read	✓ Write

Figure 45: Modbus configuration

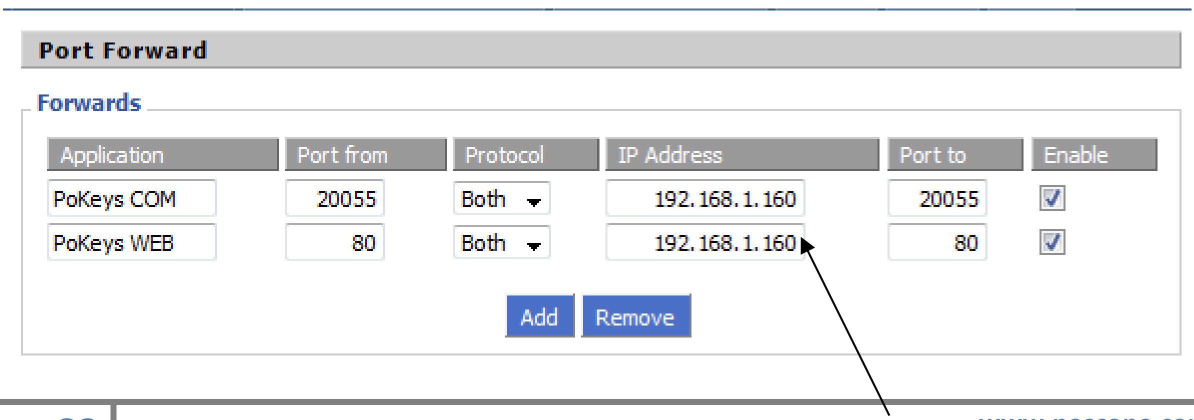
Applicable PoKeys library commands

Modbus configuration commands	
GetModbusSettings	Retrieves modbus settings
SetModbusSettings	Set modbus settings

Connecting to PoKeys56E/57E across the internet

To access the PoKeys56E/57E from the internet, connect the PoKeys56 board to your router and configure the following port forwarding (consult router manual for instructions on port forwarding):

- For the WEB interface, forward the port 80
- For the use of the PoKeys software, forward the port 20055


Port Forward

Forwards

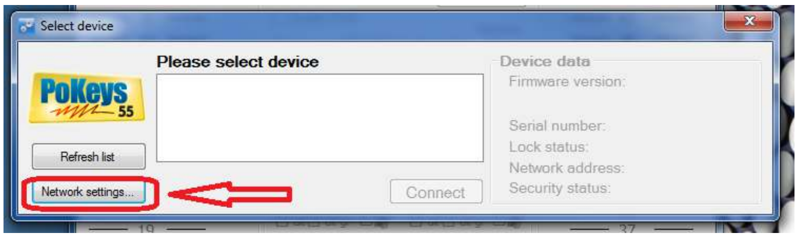
Application	Port from	Protocol	IP Address	Port to	Enable
PoKeys COM	20055	Both	192.168.1.160	20055	✓
PoKeys WEB	80	Both	192.168.1.160	80	✓

IP address of PoKeys56E board

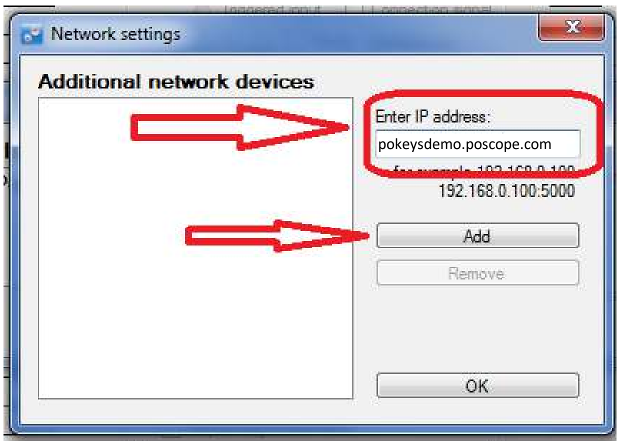
Using PoKeys software with PoKeys56E/57E device across the internet

Suppose that the PoKeys56E/57E is at the *pokeysdemo.poscope.com* (change this address to the address of your own PoKeys56E/57E device).

Open PoKeys software and click on 'Network settings'.



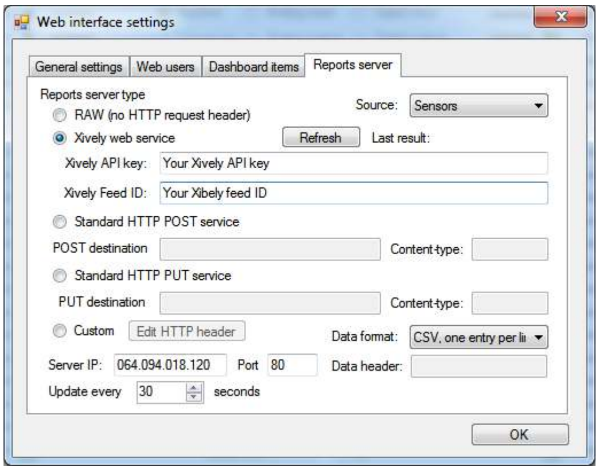
The following dialog will appear. Enter the address of the PoKeys56 board in the field on the right and click on 'Add'.



Then click on 'OK'. If the connection was established successfully, PoKeys56E/57E will be shown in the list of the devices.

8.20. Reporting data to network server with PoKeys56E/57E device

PoKeys56E/57E devices can automatically report sensor values to various network servers using the HTTP POST, HTTP PUT or text-only protocols.



Web interface settings

General settings | Web users | Dashboard items | Reports server

Reports server type
- RAW (no HTTP request header)
- [✓ Xively web service]
  - Xively API key: Your Xively API key
  - Xively Feed ID: Your Xibely feed ID
- Standard HTTP POST service
  - POST destination: Content-type:
- Standard HTTP PUT service
  - PUT destination: Content-type:
- Custom (Edit HTTP header)

Source: Sensors

Refresh | Last result:

Data format: CSV, one entry per li

Server IP: 064.094.018.120 Port 80

Data header:

Update every 30 seconds

OK

Figure 46: Reports server settings for Xively service

To use this reporting feature, user must specify request type (HTTP POST/PUT, Xively, custom HTTP header or raw), server IP and update rate.

The request header is constructed of two parts – HTML header and data header, divided by double new line character (\n). Example header (Xively.com web service):

`PUT /v2/feeds/62592.csv HTTP/1.1`
`Host: api.xively.com`
`X-ApiKey:`
`CAb3XX634daSAKxZc3M0M3I1NWVZVT0g`
`User-Agent: PoKeys56E`
`Content-Type: text/csv`
	The HTTP header without 'Connection' and 'Content-length' tags must be provided by the user

`Content-Length: 010`
`Connection: close`
	The 'Connection' and 'Content-length' tags are automatically inserted by PoKeys56E device

`Test1234,-15000.00`	Data is inserted at the end of the packet

The above example is specified as

```
PUT /v2/feeds/62592.csv HTTP/1.1
Host: api.xively.com
X-ApiKey: CAb3XX634daSAKxZc3M0M3I1NWVZVT0g
User-Agent: PoKeys56E
Content-Type: text/csv

```

The extra new line at the end is essential.

More customized header:

`POST /myScript.php HTTP/1.1`
`Host: api.xively.com`
`User-Agent: MyCustomDeviceAgent`
	The HTTP header without 'Connection' and 'Content-length' tags must be provided by the user

`Content-Type: text/csv`
`Content-Length: 010`
`Connection: close`
 	
`MyData: Test1234,-15000.00`
	The 'Connection' and 'Content-length' tags are automatically inserted by PoKeys56E device

The above example must be specified as

```
POST /myScript.php HTTP/1.1
Host: api.xively.com
User-Agent: MyCustomDeviceAgent
Content-Type: text/csv

MyData:
```

Again, an extra new line character in fifth line is essential.

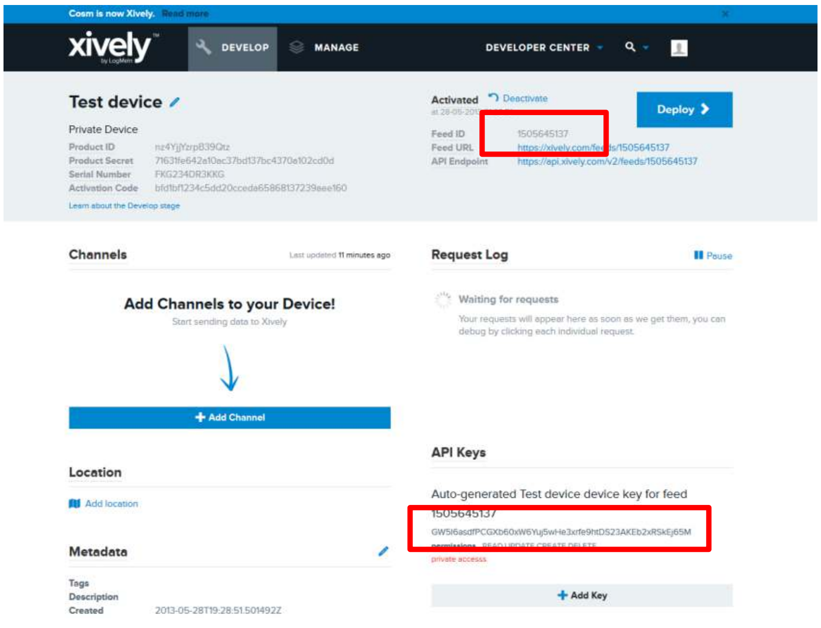
Total length of final header and data is limited to 350 bytes.

Setup for Xively web service

Xively web service is an on-line database service allowing users to connect sensor-derived data (eg. energy and environment data from objects) to the Web and to build their own applications based on that data.

PoKeys56E/57E devices feature a direct support for the Xively web service. To configure PoKeys device for Xively, follow these steps:

1. Sign-up for a free account at http://xively.com/
2. Navigate to Develop and create a new private device
3. Under the created device information, you can find 'Feed ID' and 'API key' for your newly created device, as shown in the figure below.



Test device

Private Device

Product ID	nz4YjjYzrpB39Qtz
Product Secret	71631fe642a10ac37bd137bc4370a102cd0d
Serial Number	FKG234DR3KKG
Activation Code	bfd1bf1234c5dd20cceda65868137239aee160

Activated

Feed ID	1505645137
Feed URL	https://xively.com/feeds/1505645137
API Endpoint	https://api.xively.com/v2/feeds/1505645137

Channels

Last updated 11 minutes ago

Add Channels to your Device!

Start sending data to Xively

Request Log

Waiting for requests

Your requests will appear here as soon as we get them, you can debug by clicking each individual request.

Location

Metadata

Tags	
Description	
Created	2013-05-28T19:28:51.501492Z

API Keys

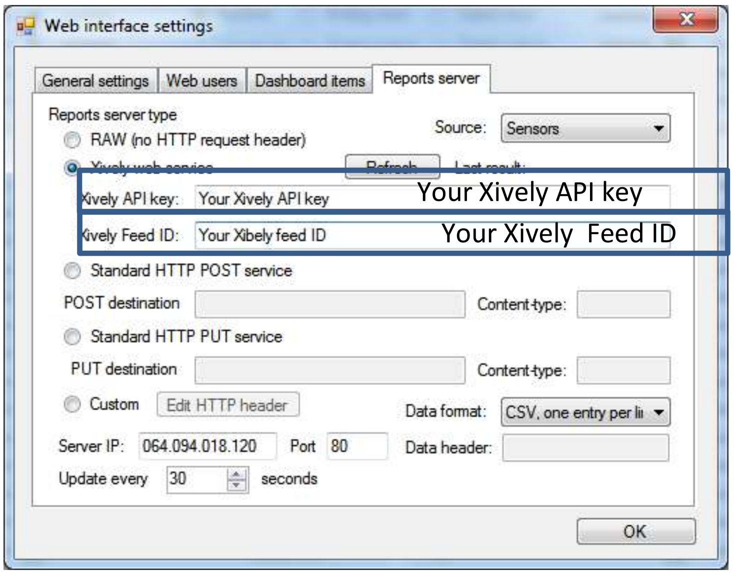
Auto-generated Test device device key for feed 1505645137

GW5I6asdfPCGXb60xW6Yuj5wHe3xrfe9htDS23AKEb2xRSkEj65M

permissions READ UPDATE CREATE DELETE

private accesss

4. Open Device > Web interface settings dialog and go to 'Reports server' tab. Check 'Xively web service' option.
5. Enter your Xively API key and Feed ID, created in step 3, leave other fields with default values


Web interface settings

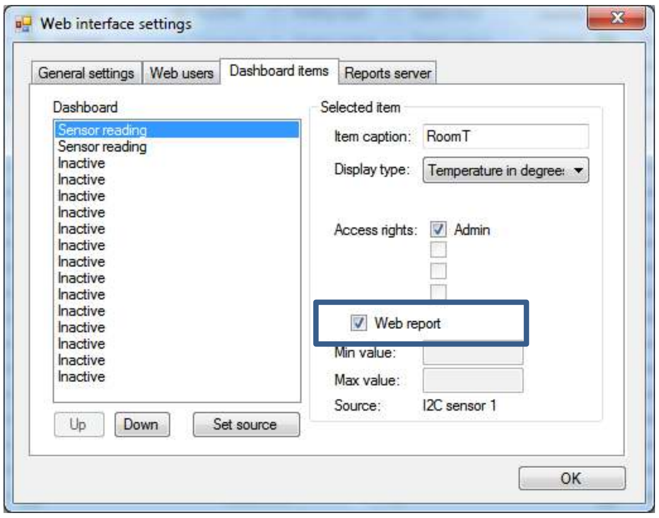
General settings | Web users | Dashboard items | Reports server

Reports server type
RAW (no HTTP request header)
[✓ Xively web service]
Xively API key: Your Xively API key
Xively Feed ID: Your Xibely feed ID
Standard HTTP POST service
POST destination
Standard HTTP PUT service
PUT destination
Custom
Source: Sensors
Content-type:
Data format: CSV, one entry per li
Data header:
Server IP: 064.094.018.120 Port 80
Update every 30 seconds

Figure 47: Xively settings page

6. Select the update rate at the bottom.
7. Click OK and click Send to device button

8. Go back to Device > Web interface settings to page 'Dashboard items'. Define entries as described in the 'Web interface' chapter of this manual. To enable uploading of the dashboard item to the Xively service, select 'Web report' as the user. The 'Item caption' field is used to identify the datastream in the selected Xively feed.
Make sure that item caption does not contain any invalid characters for Xively channel name (+, -, _, letters and numbers are allowed)


Web interface settings

General settings | Web users | Dashboard items | Reports server

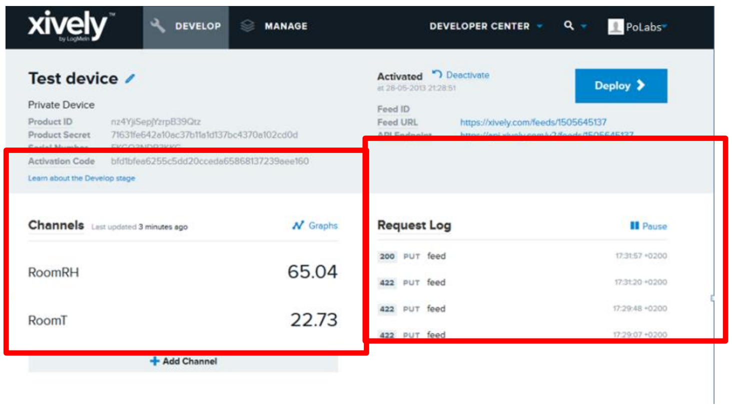
Dashboard
Sensor reading
Sensor reading
Inactive
Inactive
Inactive
Inactive
Inactive
Inactive
Inactive
Inactive
Inactive
Inactive
Inactive
Inactive
Inactive
Inactive

Selected item
Item caption: RoomT
Display type: Temperature in degree
Access rights: [✓ Admin]
[✓ Web report]
Min value:
Max value:
Source: I2C sensor 1

Figure 48: Item configuration for the Xively service

9. Save the settings again by clicking 'Send to device' button.

10. Make sure the PoKeys device has properly configured network settings and that it is connected to the internet
11. After a update interval, check the status of the Xively updates in the Xively 'Develop' page



Test device

Private Device

Product ID nz4YjiSepjYzrpB39Qtz
Product Secret 71631fe642a10ac37b11a1d137bc4370a102cd0d
Activation Code bfd1bfea6255c5dd20cceda65868137239aee160

Activated Deactivate
at 28-05-2013 21:28:51

Feed ID
Feed URL https://xively.com/feeds/1505645137

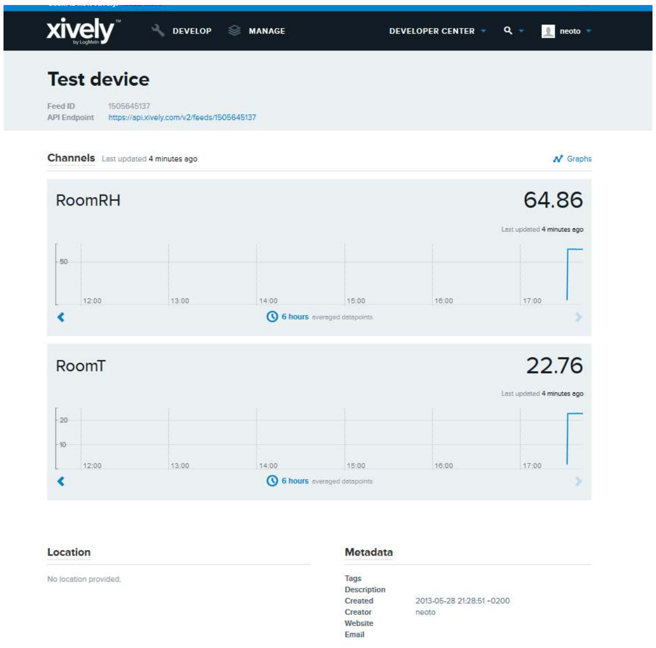
Channels Last updated 3 minutes ago

RoomRH	65.04
RoomT	22.73

Request Log

```
200  PUT  feed    17:31:57 +0200
422  PUT  feed    17:31:20 +0200
422  PUT  feed    17:29:48 +0200
422  PUT  feed    17:29:07 +0200
```

12. Open your Xively feed by clicking the Feed URL – you should see the recorded data



Test device

Feed ID 1505645137
API Endpoint https://api.xively.com/v2/feeds/1505645137

Channels Last updated 4 minutes ago

RoomRH 64.86
Last updated 4 minutes ago

RoomT 22.76
Last updated 4 minutes ago

Location

No location provided.

Metadata

Tags	
Description	
Created	2013-05-28 21:28:51 +0200
Creator	neoto
Website	
Email	

Figure 49: Xively feed overview page

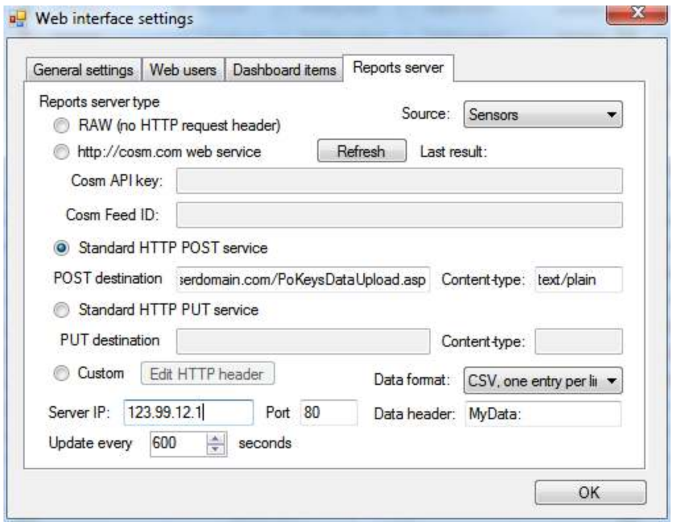
Setup for standard HTTP POST or PUT data upload

Let's assume that the user wants to send sensor data to his server script that accepts POST method for data upload. The user's server script is available at the address www.userdomain.com/PoKeysDataUpload.asp. The user also wants that the data in the POST stream to have the HTTP type of 'text/plain' and preceeded with the string 'MyData: '. The data should be transferred every 10 minutes. The settings are displayed in Figure 50.

To setup PoKeys to use this settings, the following steps must be taken (enter text without ' quotes):

1. Open Device > Web configuration menu and switch to 'Reports server' tab.

2. Select 'Standard HTTP POST service' option
3. Enter 'www.userdomain.com/PoKeysDataUpload.asp' in the POST destination field
4. Enter 'text/plain' in the 'Content-type' field.
5. Enter 'MyData: ' in the 'Data header' field
6. Enter the server's IP address and port number in the fields below.
7. Enter 600 in the update time field
8. Click OK and click Send to device button.
9. Go back to Device > Web interface settings to page 'Dashboard items'. Define entries as described in the 'Web interface' chapter of this manual. To enable uploading of the dashboard item to the web server, select 'Web report' as the user. The 'Item caption' field is used to identify the datastream.
10. Save the settings again by clicking 'Send to device' button.
11. Make sure the PoKeys device has properly configured network settings and that it is connected to the internet

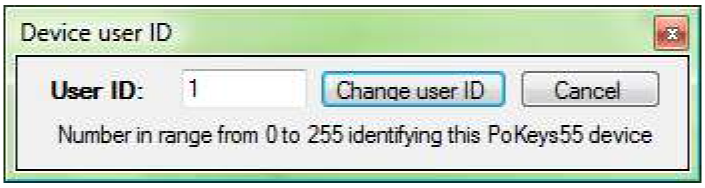

Web interface settings

Reports server type
( ) RAW (no HTTP request header)
( ) http://cosm.com web service
Cosm API key:
Cosm Feed ID:
(●) Standard HTTP POST service
POST destination: ›erdomain.com/PoKeysDataUpload.asp
Content-type: text/plain
( ) Standard HTTP PUT service
PUT destination:
Content-type:
( ) Custom
Source: Sensors
Last result:
Data format: CSV, one entry per li
Server IP: 123.99.12.1
Port: 80
Data header: MyData:
Update every 600 seconds

Figure 50: Example of HTTP POST service setup

8.21. Changing User ID number

Users can freely assign their own User ID number that represents a specific PoKeys device (enables distinguishing between different PoKeys devices in case there is more than one connected to a single host PC). To change the User ID number, go to 'Device' > 'Change user ID' menu. Simply enter any number between 0 and 255, and click the 'Change user ID' button.


Device user ID

User ID: 1

Number in range from 0 to 255 identifying this PoKeys55 device

Figure 51: Device user ID dialog

8.22. Saving current configuration to file

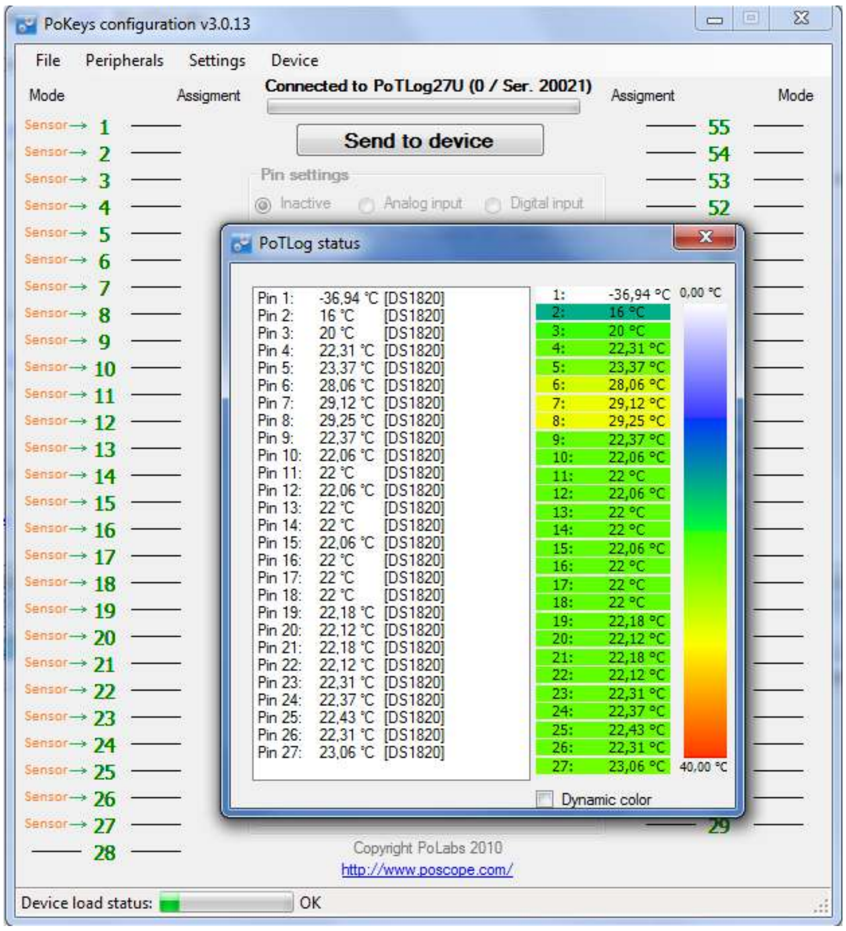
To save the current configuration to a file, go to ‘File’ > 'Save' menu and select a new filename. To reload a saved configuration from a file, go to ‘File’ > ‘Open’ menu and select the appropriate file. To transfer new settings to the device, click on the ‘Save to device’ button.

9. PoTLog27 firmware

PoKeys56U and PoKeys56E/57E devices support an alternate firmware that allows the connection of up to 27 DS1820 or DS18B20 temperature sensors (one sensor per pin, pins 1-27 only). Firmware automatically detects the sensors on each power-up and starts reading the temperatures. Temperatures of both types of temperature sensors are read at the rate of 1 measurement per 800 miliseconds (around 1,25 measurements per second) at the resolution of 12-bit (approx. 0,06 °C).

To activate PoTLog27 firmware, go to menu Device > Update firmware to PoTLog27... and wait for the update to finish. After the successfull update, the device will reboot into the PoTLog27 firmware.

To check the configuration and current temperatures, go to menu Peripherals > PoTLog status... The following dialog will appear, showing the list of all detected sensors and their temperatures.



Connected to PoTLog27U (0 / Ser. 20021)

```
Pin 1:   -36,94 °C [DS1820]
Pin 2:   16 °C     [DS1820]
Pin 3:   20 °C     [DS1820]
Pin 4:   22,31 °C  [DS1820]
Pin 5:   23,37 °C  [DS1820]
Pin 6:   28,06 °C  [DS1820]
Pin 7:   29,12 °C  [DS1820]
Pin 8:   29,25 °C  [DS1820]
Pin 9:   22,37 °C  [DS1820]
Pin 10:  22,06 °C  [DS1820]
Pin 11:  22 °C     [DS1820]
Pin 12:  22,06 °C  [DS1820]
Pin 13:  22 °C     [DS1820]
Pin 14:  22 °C     [DS1820]
Pin 15:  22,06 °C  [DS1820]
Pin 16:  22 °C     [DS1820]
Pin 17:  22 °C     [DS1820]
Pin 18:  22 °C     [DS1820]
Pin 19:  22,18 °C  [DS1820]
Pin 20:  22,12 °C  [DS1820]
Pin 21:  22,18 °C  [DS1820]
Pin 22:  22,12 °C  [DS1820]
Pin 23:  22,31 °C  [DS1820]
Pin 24:  22,37 °C  [DS1820]
Pin 25:  22,43 °C  [DS1820]
Pin 26:  22,31 °C  [DS1820]
Pin 27:  23,06 °C  [DS1820]
```

```
1:  -36,94 °C   0,00 °C
2:  16 °C
3:  20 °C
4:  22,31 °C
5:  23,37 °C
6:  28,06 °C
7:  29,12 °C
8:  29,25 °C
9:  22,37 °C
10: 22,06 °C
11: 22 °C
12: 22,06 °C
13: 22 °C
14: 22 °C
15: 22,06 °C
16: 22 °C
17: 22 °C
18: 22 °C
19: 22,18 °C
20: 22,12 °C
21: 22,18 °C
22: 22,12 °C
23: 22,31 °C
24: 22,37 °C
25: 22,43 °C
26: 22,31 °C
27: 23,06 °C    40,00 °C
```

Figure 52: PoTLog status dialog

Note: the following functions are disabled in PoTLog27 mode:

- Encoders
- Macros
- Triggered key mapping and delayed/repeated key actions
- PWM outputs
- Joystick analog to digital mapping
- Connection signal
- PoExtBus
- Additional 1-wire devices support
- Custom device name

To access the temperature measurements, these commands are available in PoKeys communication dll:

GetAllSensorValuesString

Retrieves all measurements in one string. Each measurements is separated with | sign.

```
string GetAllSensorValuesString()
```

Arguments:

none

Remarks:

none

GetAllSensorValues

Retrieves all measurements in one structure.

```
bool GetAllSensorValues(ref sPoTLogDevice logDevice)
```

Arguments:

none

Remarks:

C++ alternative: COM_GetAllSensorValues(IntPtr logDevice)

GetAllSensorIDs

Retrieves all sensor 64-bit IDs in one structure.

```
bool GetAllSensorIDs(ref sPoTLogDevice logDevice)
```

Arguments:

none

Remarks:

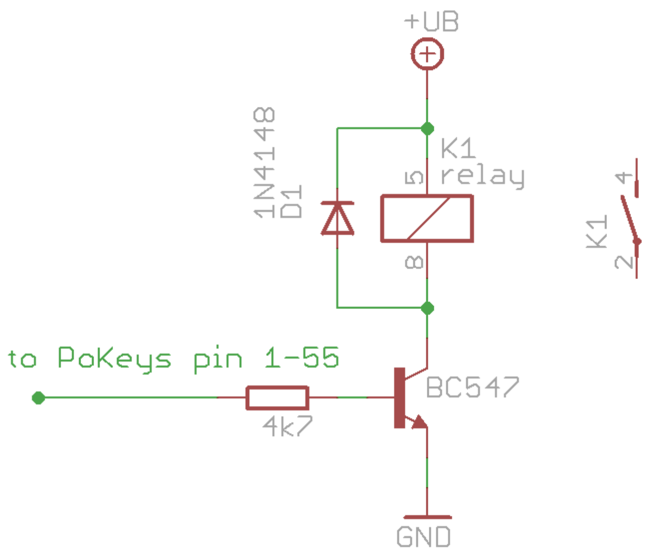
C++ alternative: COM_GetAllSensorIDs(IntPtr logDevice)

Example code (C#):

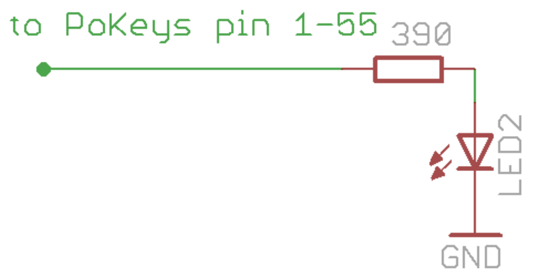
```
sPoTLogDevice logger = new sPoTLogDevice();

MyDevice.GetAllSensorValues(ref logger);
MyDevice.GetAllSensorIDs(ref logger);

for (int i = 0; i < 27; i++)
{
    byte sType = logger.sensors1Wire[i].sensorID[0];
    double value = (double)logger.sensors1Wire[i].sensorValue / 100;
    string displayValue = "";

    if (sType == 0x10)
    {
        displayValue = value + " °C";
        listBox1.Items.Add("Pin " + (i + 1) + ":\t" + displayValue + "\t[DS1820]");
    }
    else if (sType == 0x28)
    {
        displayValue = value + " °C";
        listBox1.Items.Add("Pin " + (i + 1) + ":\t" + displayValue + "\t[DS18B20]");
    }
    else
    {
        listBox1.Items.Add("Pin " + (i + 1) + ":\tNo sensor");
    }
}
```

10. Connecting common peripherals to PoKeys devices

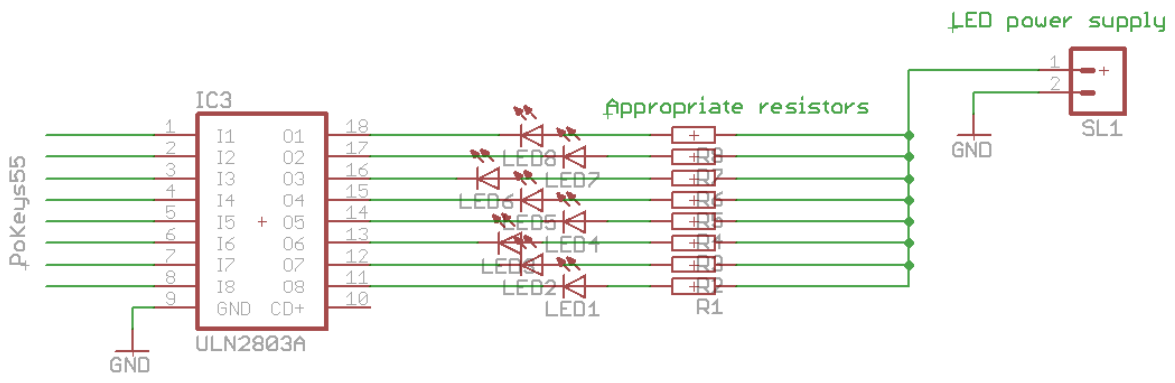
Relays



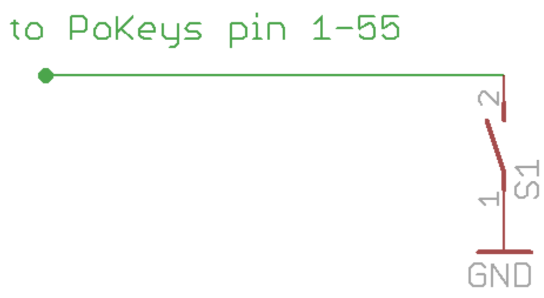
LEDs



High-current LEDs



Switches

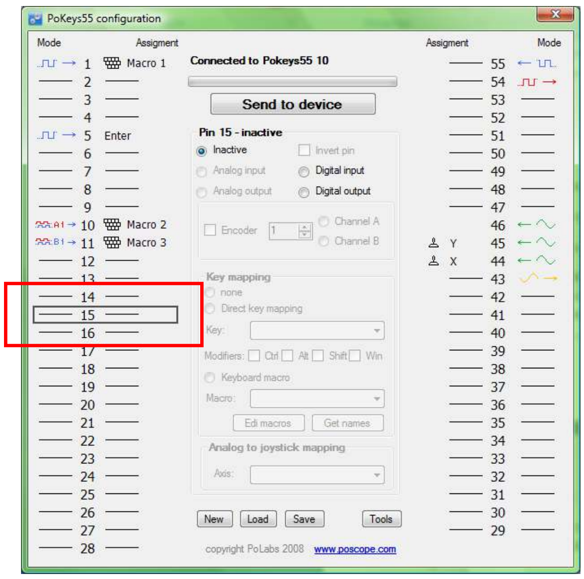


Example: Setting up key mapping

This example shows how easy is to set up a digital input pin for direct key mapping

We will set up a Shift-Escape combination for pin 15.

1. Connect a switch to your PoKeys device as shown above
2. Open PoKeys configuration application
3. Select your PoKeys device from drop-down box and click ‘Connect’ button
4. Wait the application to load current configuration from PoKeys device
5. Click the same pin number as you connected a switch to (in this example pin 15)

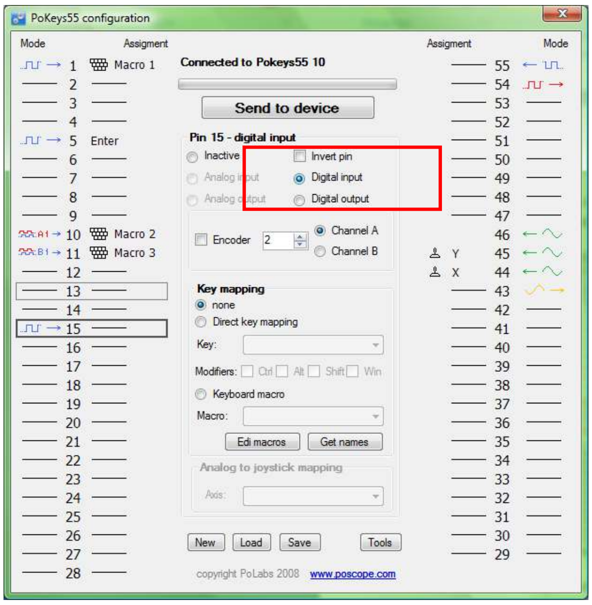

PoKeys55 configuration

Connected to Pokeys55 10

Pin 15 - inactive

[✓ Inactive]

6. Set this pin as digital input



Connected to Pokeys55 10

Pin 15 - digital input

Inactive
Invert pin
Analog input
[✓ Digital input]
Analog output
Digital output

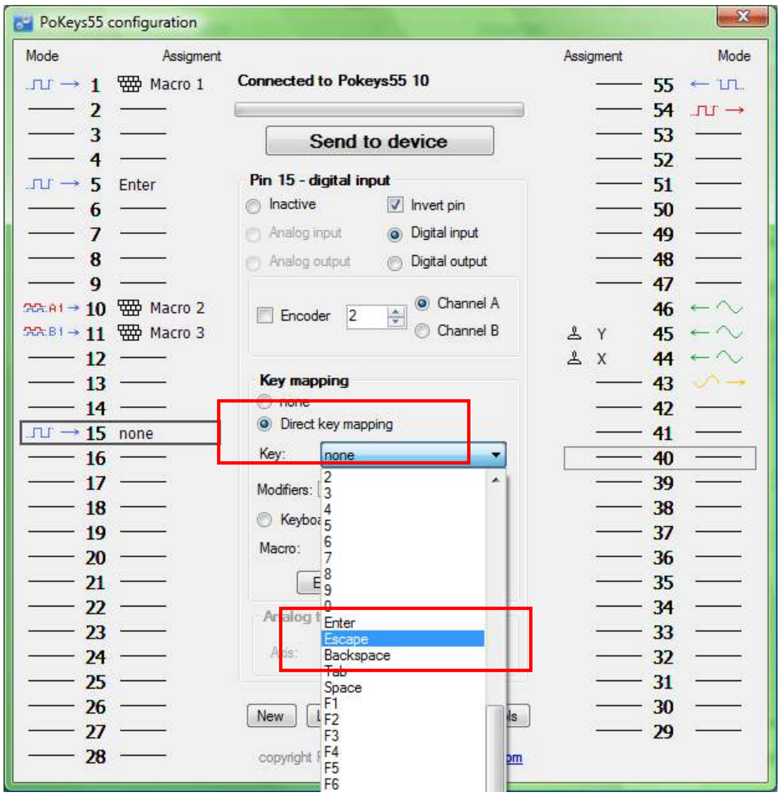
Encoder 2
[✓ Channel A]
Channel B

Key mapping
[✓ none]
Direct key mapping
Key:
Modifiers: Ctrl Alt Shift Win
Keyboard macro
Macro:

Analog to joystick mapping
Axis:

Pin 1 Macro 1; Pin 5 Enter; Pin 10 Macro 2; Pin 11 Macro 3; Pin 45 Y; Pin 44 X

7. Select ‘Direct key mapping’ and from drop-down box select Escape



Connected to Pokeys55 10

Pin 15 - digital input

Inactive
[✓ Invert pin]
Analog input
[✓ Digital input]
Analog output
Digital output

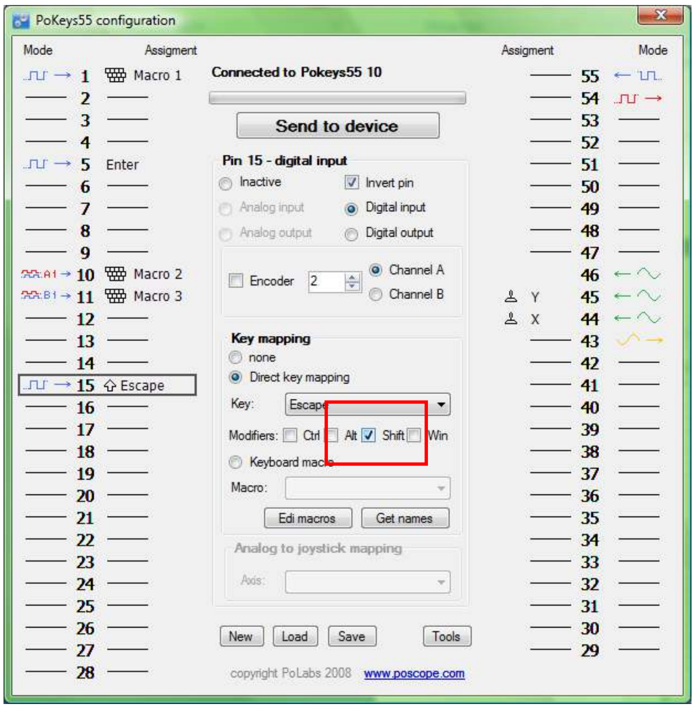
Encoder 2
[✓ Channel A]
Channel B

Key mapping
none
[✓ Direct key mapping]
Key: none
Modifiers:
Keyboard macro
Macro:

2
3
4
5
6
7
8
9
0
Enter
[✓ Escape]
Backspace
Tab
Space
F1
F2
F3
F4
F5
F6

Pin 1 Macro 1; Pin 5 Enter; Pin 10 Macro 2; Pin 11 Macro 3; Pin 15 none; Pin 45 Y; Pin 44 X

8. Click on the ‘Shift’ checkbox to enable Shift modifier



PoKeys55 configuration

Mode	Pin	Assigment
⎍ →	1	Macro 1
	2	
	3	
	4	
⎍ →	5	Enter
	6	
	7	
	8	
	9	
A1 →	10	Macro 2
B1 →	11	Macro 3
	12	
	13	
	14	
⎍ →	15	⇧ Escape
	16–28	

Assigment	Pin	Mode
	55	← ⎍
	54	⎍ →
	53–47	
	46	← ∿
Y	45	← ∿
X	44	← ∿
	43	∿ →
	42–29	

Connected to Pokeys55 10

Send to device

Pin 15 - digital input
- Inactive
- [✓ Invert pin]
- Analog input
- [✓ Digital input]
- Analog output
- Digital output
- Encoder 2 — [✓ Channel A] Channel B

Key mapping
- none
- [✓ Direct key mapping]
- Key: Escape
- Modifiers: Ctrl Alt [✓ Shift] Win
- Keyboard macro
- Macro:

Edi macros | Get names

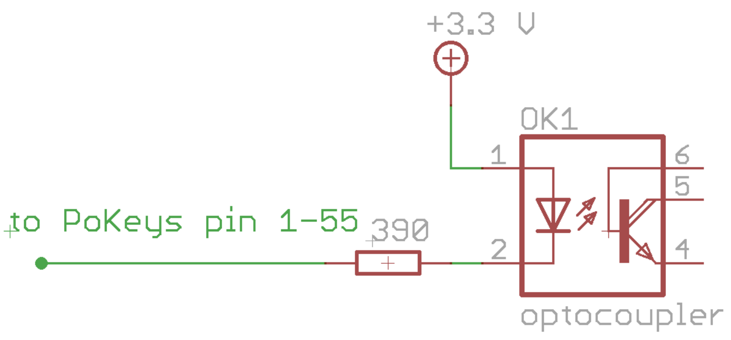
Analog to joystick mapping
- Axis:

New | Load | Save | Tools

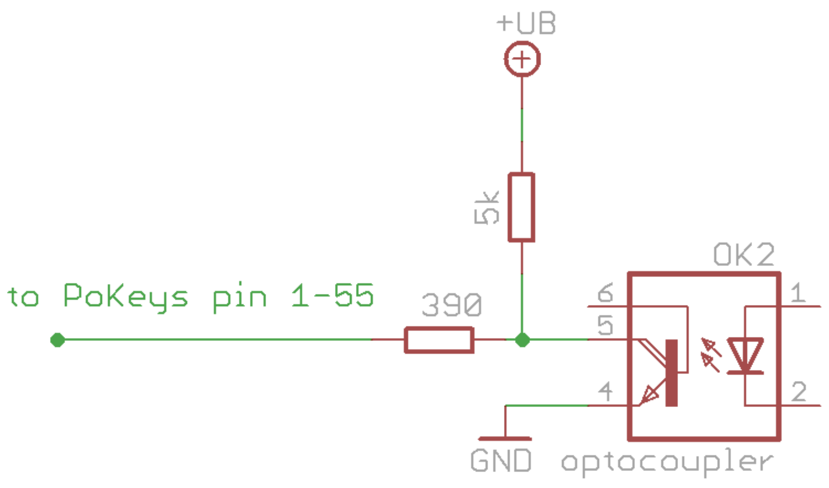
copyright PoLabs 2008 www.poscope.com

9. Send configuration to device by clicking ‘Send to device button’.

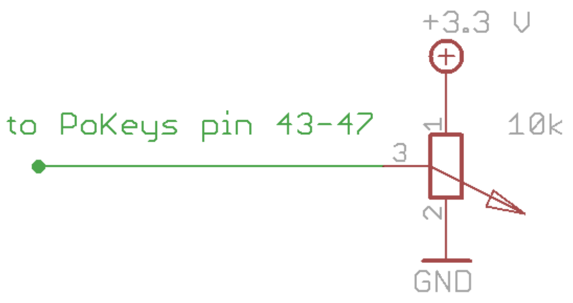
Optocoupled digital output



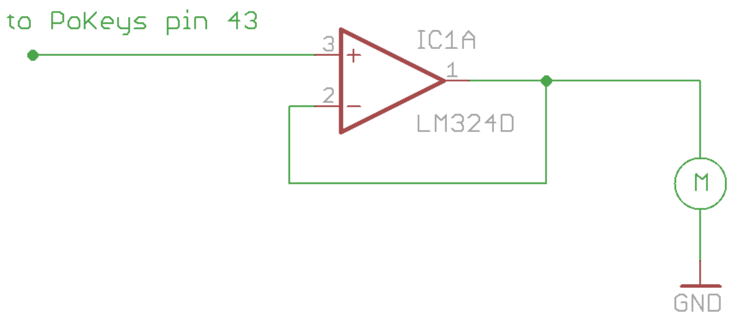
Optocoupled digital input



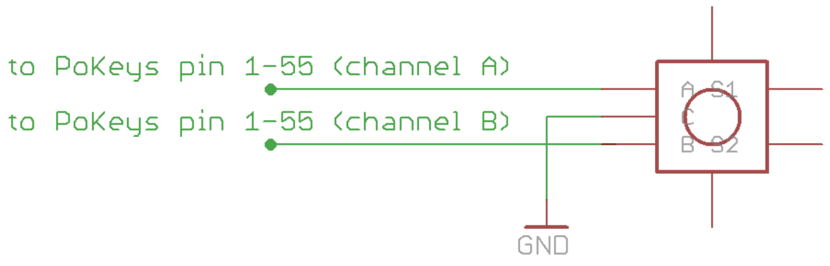
Potentiometers (variable resistors)



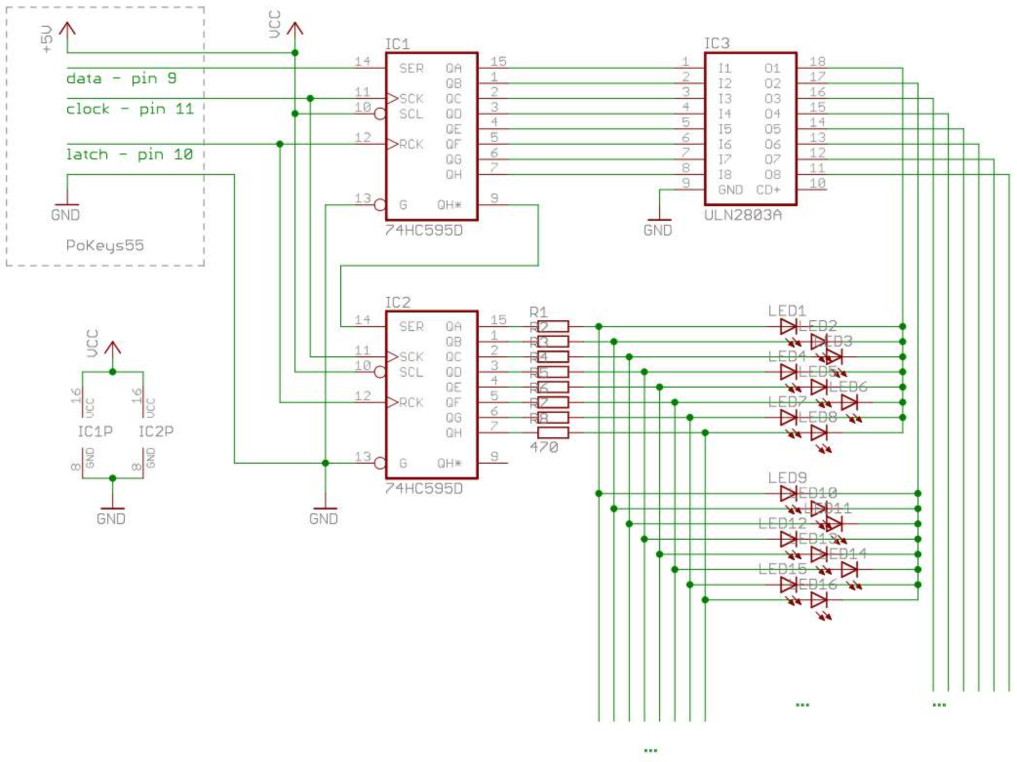
Linear motor control



Rotational encoder switch


to PoKeys pin 1-55 (channel A)
to PoKeys pin 1-55 (channel B)
A S1
C
B S2
GND

LED displays/arrays



11. Restoring factory defaults - quick resetting the device configuration (or recovering bad firmware update)

If configuration editor cannot be used to reconfigure the device because of endless key presses from the device, simple reset procedure should be executed.

1. Disconnect PoKeys device from USB (unplug the power for the PoKeys56E/57E device)
2. Find pin labeled 'RST' on the PoKeys device (otherwise use pin numbered 54)
3. Short this pin to ground (GND) and reconnect the PoKeys device to USB (or reconnect power)
4. Green light should start flashing rapidly and PoKeys device will connect in recovery mode
5. Open PoKeys configuration application
6. PoKeys configuration application should detect PoKeys device in recovery mode.
    a. To reset the device configuration, use the option 'Clear settings'. By clicking this button and confirming your decision on the next dialog, settings will be erased.
    b. To recover from bad firmware update, click 'Recover'.
7. After completing the operation, unplug PoKeys device and disconnect 'RST' pin from ground
8. If resetting the configuration, ensure the device is properly cleared – replug PoKeys device, connect to it and execute Device > Clear settings in device.

PoKeys configuration software is backing up current configuration state (except keyboard macro sequences) on each connection start.

These configuration files can be found in the local application folder (system folder – usually c:\Documents and settings\{username}\Local Settings\Application Data\PoKeys\ on Windows 2000, XP or C:\Users\{username}\AppData\Local\PoKeys\), named backup1.pkc, backup2.pkc and backup3.pkc with backup3.pkc being the oldest configuration.

12. Frequently asked questions

What software must be installed to operate the device?

On first use or when reconfiguring the device, the supplied software must be installed. There are no device drivers needed. They are already supplied with your operating system. Once the device has been configured, the settings are stored on-board. Device can then be freely used on any machine (see requirements for USB HID device driver enabled operating system) without any additional installation.

What is the difference between PoKeys56E and PoKeys57E?

PoKeys57E has (compared to PoKeys56E) more memory space for future updates and PoIL code.

I misconfigured the device. Now the device starts pressing virtual keys before I can do anything. What can I do?

If you misconfigured the device in such a way that configuration utility cannot be used to repair the configuration, see the section 'Quick resetting the device configuration' in this manual.

How do I connect switch/relay/LED/… to PoKeys device?

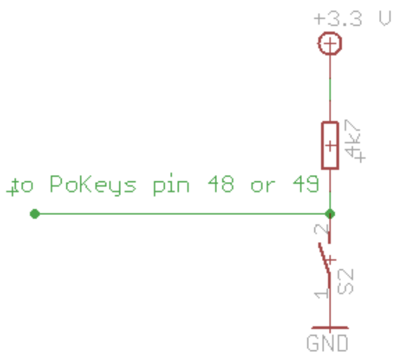
Please see the section 'Connection common peripherals to PoKeys device' in this manual.

I have two (or more) PoKeys devices connected on one system and cannot differentiate the devices to set the configurations.

It is advised that the users assign different UserID numbers to each of the device connected to a system. Please see the section Changing User ID of this manual.

It appears that pins 48 and 49 are floating. What should I do? (PoKeys55 only)

Due to device design, pins 48 and 49 should be equipped with external 5-10 kΩ pull-up resistor as shown bellow.



Note: on PoKeys55 boards with serial number above 11500, this problem is removed.

I have connected a switch to pin 4 and now PoKeys55 is not recognized by the computer anymore.

You must have connected normally-closed switch to pin 4 and therefore connected pin 4 to ground. At boot (connecting PoKeys55 to USB) this means that PoKeys55 is entering system boot and therefore cannot be used from the computer. Please use another pin for normally-closed switches. ***Note: on PoKeys55 boards with serial number above 11500, this problem is removed.***

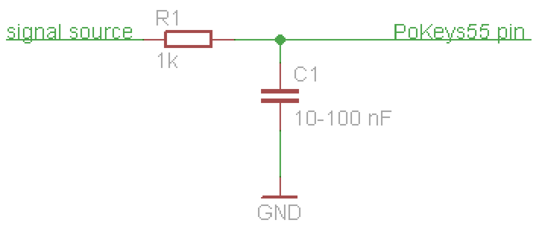
I have connected a switch to pin 54 and now PoKeys is connecting in recovery mode on every boot.

You must have connected normally-closed switch to pin 54 and therefore connected pin 4 to ground. At boot (connecting PoKeys device to USB) this means that PoKeys device is entering recovery mode and therefore cannot be used from the third-party software. Please use another pin for normally-closed switches. Also see chapter 9: Quick resetting the device configuration.

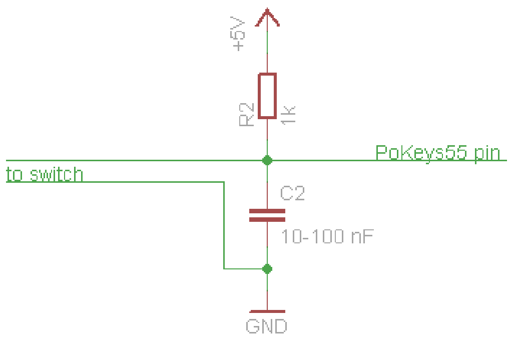
There is spontaneous triggering of some of the pins. What is wrong?

You might have connected long cable from the PoKeys board to the switch. If this cable crosses any power cables, it can trigger a false signal on PoKeys board input due to interference or coupling. Use twisted pair wires if possible. If this not help and cables cannot be routed elsewhere, use shielded cable.

If cables are routed inside electrically ‘dirty’ environment, use simple RC filter on those signals as illustrated below.



If connecting switches to PoKeys device with long cables, use the schematics below. When using matrix keyboard connected to PoKeys device with long cables, it is also advised to use the schematics below for digital input pins. This filter should not be used on digital output pins!



My problem is that when I start the program, everything works good for about 3 to 5 seconds, then the CPU “lock up” and the only way to recover is to unplug the PoKeys device.

The problem occurs because you are writing 'dirty' code. You create object every time you need to use it in a loop, but you forget do properly dispose it. Best way to use PoKeys DLL in an application that read or writes data in a loop, is to create a global object and initialize it once at the start of application, and use its functions to read or write in a loop. This way the communication is much faster.

If you are using Visual Basic development environment, add a reference to PoKeysDevice DLL and use object browser to find proper declaration. Via object browser you can also access the list of all supported functions, which will also be used by Intelli sense in editor.

Some outputs on PoExtBus boards are not responding.

If you encounter a problem with multiple PoExtBus boards not functioning normally, please check if the boards have the following 1 nF capacitor soldered between pins 2 (GND) and 3 (DATA) of the ExtBusOUT connector (see figure below).

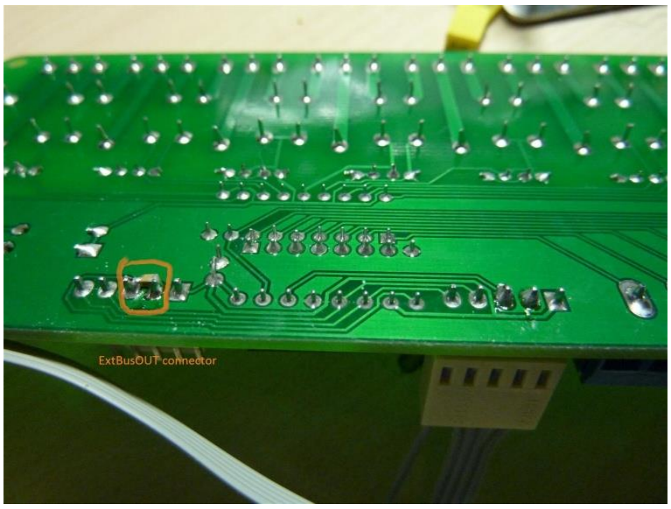
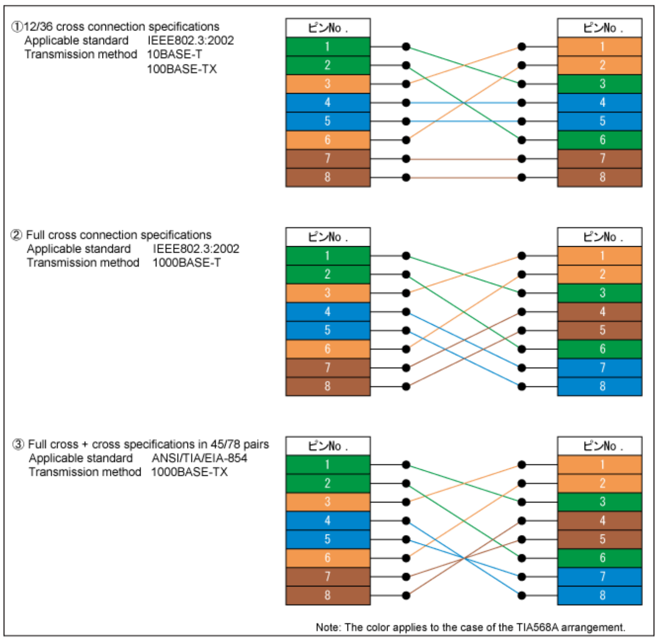


Figure 53: Capacitor on PoExtBus board

The connection with the PoKeys56E/57E device cannot be established with 10BASE-T router

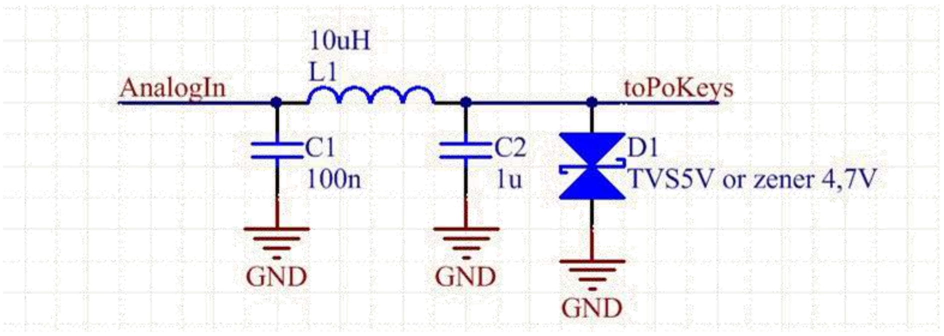
Please check that you are using proper cable that is supported by 10BASE-T standard. The following diagram shows the correct wiring for different connection standards.


①12/36 cross connection specifications
Applicable standard IEEE802.3:2002
Transmission method 10BASE-T 100BASE-TX
② Full cross connection specifications
Applicable standard IEEE802.3:2002
Transmission method 1000BASE-T
③ Full cross + cross specifications in 45/78 pairs
Applicable standard ANSI/TIA/EIA-854
Transmission method 1000BASE-TX
ピンNo.
Note: The color applies to the case of the TIA568A arrangement.

(source: http://www.okidensen.co.jp/en/prod/cable/lan/img/cate6_n_fig05.gif)

How should PoKeys inputs be protected for the use in (electrically) noisy environments?

Please use the following circuitry for the input protection. Is higher speeds are required, capacitors C1 and C2 can be exchanged with 10 nF parts.



13. PoKeys library functions

The following list contains only the most commonly used library functions.

For the list and description of all library functions, open the PoKeys DLL help file found in Start Menu > PoKeys > > Development > PoKeys DLL help.

13.1. ConnectToDevice (universal for all PoKeys devices)

Connect to the device with the device's serial number specified.

```
bool ConnectToDevice(int serialNumber, int checkEthernet)
```

Arguments:

serialNumber

Serial number of device to connect to

checkEthernet

If 1, PoKeys56E devices will be also included in the search, if 2, and UDP connection will be used for connecting to PoKeys56E

Remarks:

The command is a universal command to find the PoKeys device with the specified serial number and open the connection with it. Function returns True if connection is established or False if there were errors.

13.2. EnumerateDevices (for USB-based PoKeys devices)

Enumerate the PoKeys devices and return number of found PoKeys devices.

```
int EnumerateDevices()
```

Arguments:

none

Remarks:

This function must be called on every class initialization

13.3. ConnectToDevice (for USB-based PoKeys devices)

Connect to the device with the index specified.

```
bool ConnectToDevice(int deviceIndex)
```

Arguments:

deviceIndex

Index of the PoKeys device

Remarks:

Index is not UserID of the PoKeys device and therefore can change if more than one PoKeys device is used at a time. Function returns True if connection is established or False if there were errors.

13.4. DisconnectDevice

Terminate the connection with the device.

```
void DisconnectDevice()
```

Arguments:

none

Remarks:

This function should be called before class disposal or changing of the device.

13.5. StartEthernetDiscovery

Starts the automatic discovery of PoKeys56E devices.

```
void StartEthernetDiscovery ()
```

Arguments:

none

Remarks:

See example section for usage example

13.6. StopEthernetDiscovery

Stops the automatic discovery of PoKeys56E devices.

```
void StopEthernetDiscovery ()
```

Arguments:

none

Remarks:

See example section for usage example

13.7. GetNumberOfDetectedNetworkDevices

Returns the number of detected PoKeys56E devices.

```
int GetNumberOfDetectedNetworkDevices()
```

Arguments:

none

Remarks:

See example section for usage example

13.8. ConnectToNetworkDevice

Connects to PoKeys56E device with the specified IPv4 address in format xxx.xxx.xxx.xxx (e.g., 192.168.0.105, 192.168.050.002, etc.). If connection is established successfully, returns True.

```
bool ConnectToNetworkDevice(string address)
```

Arguments:

address

IPv4 address of the selected device

Remarks:

See example section for usage example

13.9. GetDeviceIDEx

Retrieve device ID data, i.e. serial number and firmware version.

```
bool GetDeviceIDEx(ref int serialNumber, ref int firmwareVersionMajor, ref
int firmwareVersionMinor)
```

Arguments:

serialNumber

Variable in which serial number will be saved to

firmwareVersionMajor

Variable in which firmware major version will be saved to

firmwareVersionMinor

Variable in which firmware minor version will be saved to

Remarks:

Returns False on error.

13.10. GetUserID

Retrieve user ID.

```
bool GetUserID(ref byte userID)
```

Arguments:

userID

Variable in which user ID will be saved to

Remarks:

Returns False on error.

13.11. SetUserID

Set user ID.

```
bool SetUserID(byte newUserID)
```

Arguments:

newUserID

New user ID

Remarks:

It is advised that each PoKeys device on a system should have its unique user ID. Returns False on error.

13.12. SetPinData

Set pin data – pin's function and options.

```
bool  SetPinData(byte  pinID,  byte  pinFunction,  byte  pullUpDownResistor,
byte invertPin)
```

Arguments:

pinID

Pin ID is zero-based pin index on the device (output marked as 1 therefore has index 0)

pinFunction

pinFunction has the following structure

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Pin invert	reserved	reserved	A output	A input	D output	D input	reserved

where A stands for analog and D for digital.

pullUpDownResistor

No function at the moment

invertPin

No function at the moment, see bit 7 of pinFunction argument

Remarks:

It is advised to use function SetPinData with only 2 parameters (pinID and pinFunction). Returns False on error.

13.13. SetPinData

Set pin data – pin's function and options.

```
bool SetPinData(byte pinID, byte pinFunction)
```

Arguments:

pinID

Pin ID is zero-based pin index on the device (output marked as 1 therefore has index 0)

pinFunction

pinFunction has the following structure

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Pin invert	reserved	reserved	A output	A input	D output	D input	reserved

where A stands for analog and D for digital.

Remarks:

Returns False on error.

13.14. GetPinData

Get pin data – pin's function and options.

```
bool   GetPinData(byte   pinID,    ref   byte   pinFunction,   ref   byte
pullUpDownResistor, ref byte invertPin, ref byte pinPossibleFunctions)
```

Arguments:

pinID

Pin ID is zero-based pin index on the device (output marked as 1 therefore has index 0)

pinFunction

pinFunction has the following structure

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Pin invert	reserved	reserved	A output	A input	D output	D input	reserved

where A stands for analog and D for digital.

pullUpDownResistor

No function at the moment

invertPin

No function at the moment, see bit 7 of pinFunction argument

pinPossibleFunctions

No function at the moment

Remarks:

It is advised to use function GetPinData with only 2 parameters (pinID and pinFunction). Returns False on error.

13.15. GetPinData

Get pin data – pin's function and options.

```
bool GetPinData(byte pinID, byte pinFunction)
```

Arguments:

pinID

Pin ID is zero-based pin index on the device (output marked as 1 therefore has index 0)

pinFunction

pinFunction has the following structure

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Pin invert	reserved	reserved	A output	A input	D output	D input	reserved

where A stands for analog and D for digital.

Remarks:

Returns False on error.

13.16. GetInput

Get digital input status.

```
bool GetInput(byte pinID, ref bool inputState)
```

Arguments:

pinID

Pin ID is zero-based pin index on the device (output marked as 1 therefore has index 0)

inputState

Variable in which input state will be saved to

Remarks:

Pin must be set as digital input for this command to function properly. Returns False on error.

13.17. SetOutput

Set digital output status.

```
bool SetOutput(byte pinID, bool outputState)
```

Arguments:

pinID

Pin ID is zero-based pin index on the device (output marked as 1 therefore has index 0)

outputState

New output state

Remarks:

Pin must be set as digital output before this operation will function properly. On device initialization, pins are NOT set as outputs even if they are configured so. Before using the pins as digital outputs SetPinData must be called to set the direction of the pin. This must be repeated on every startup. Returns False on error.

13.18. GetAnalogInput

Get analog input status.

```
bool GetAnalogInput(byte pinID, ref [int,byte] inputValue)
```

Arguments:

pinID

Pin ID is zero-based pin index on the device (output marked as 1 therefore has index 0)

inputValue

Variable in which analog value will be saved to. Use byte type for 8-bit analog values and int type for 10-bit (or 12-bit for PoKeys56 devices) analog values

Remarks:

The returned value is between 0 and 255 (8-bit resolution) , between 0 and 1023 (10-bit resolution on PoKeys55 devices) or between 0 and 4095 (12-bit resolution on PoKeys56 devices). 0 means 0 V on input, while 255, 1023, respectively 4095 means Vdd (approximately 3.3V) on input. Returns False on error. Use the command GetAnalogValueMax to get this value automatically.

GetAnalogValueMax

Gets the maximum value of the analog-to-digital conversion. Returns 1024 for PoKeys55 devices and 4096 for PoKeys56 devices.

```
int GetAnalogValueMax()
```

Arguments:

none

13.19. SaveConfiguration

Saves current device configuration to non-volatile flash memory.

```
bool SaveConfiguration()
```

Arguments:

none

Remarks:

This function takes some time to complete. Is meantime, communication with the device is not possible. Returns False on error.

13.20. GetMatrixConfiguration

Get complete matrix keyboard configuration.

```
bool  GetMatrixKeyboardConfiguration(ref  byte  configuration,  ref  byte
width, ref byte height, ref byte[] row_pins, ref byte[] column_pins, ref
bool[] macro_mapping, ref byte[] keycodes, ref byte[] keymodifiers);
```

Arguments:

configuration

If bit 0 is set, matrix keyboard is enabled. Other bits are reserved

width, height

Number of columns and rows of the matrix keyboard

row_pins

An array of 8 bytes, each having an index of a pin that is associated with the row. Row pins must be set as digital outputs.

column_pins

An array of 8 bytes, each having an index of a pin that is associated with the column. Column pins must be set as digital inputs.

macro_mapping

An array of 64 boolean values (see below for numbering hint). If the value is set to true, instead of key press simulation, macro is run.

keycodes

An array of 64 byte values (see below for numbering hint). If appropriate macro_mapping value is set to true, each value can contain index of a macro else it contains code of a key.

keymodifiers

An array of 64 byte values (see below for numbering hint). It contains key modifiers.

Keys indexing:

No matter what dimensions the matrix keyboard has, the following scheme is used for keys indexing. A1 is always 0, B1 1, A2 8, ... For example, if user connects a 3x3 matrix keyboard, keys have indexes: 0, 1, 2, 8, 9, 10, 16, 17, 18.

	A	B	C	D	E	F	G	H
1	0	1	2	3	4	5	6	7
2	8	9	10	11	12	13	14	15
3	16	17	18	19	20	21	22	23
4	24	25	26	27	28	29	30	31
5	32	33	...					

Remarks:

Row pins must be set as digital outputs and column pins as digital inputs respectively. Returns False on error.

13.21. SetMatrixConfiguration

Set complete matrix keyboard configuration.

```
bool  SetMatrixKeyboardConfiguration(ref  byte  configuration,  ref  byte
width, ref byte height, ref byte[] row_pins, ref byte[] column_pins, ref
bool[] macro_mapping, ref byte[] keycodes, ref byte[] keymodifiers);
```

Arguments:

configuration

If bit 0 is set, matrix keyboard is enabled. Other bits are reserved

width, height

Number of columns and rows of the matrix keyboard

row_pins

An array of 8 bytes, each having an index of a pin that is associated with the row.

column_pins

An array of 8 bytes, each having an index of a pin that is associated with the column.

macro_mapping

An array of 64 boolean values (see below for numbering hint). If the value is set to true, instead of key press simulation, macro is run.

keycodes

An array of 64 byte values (see below for numbering hint). If appropriate macro_mapping value is set to true, each value can contain index of a macro else it contains code of a key.

keymodifiers

An array of 64 byte values (see below for numbering hint). It contains key modifiers.

Keys indexing:

No matter what dimensions the matrix keyboard has, the following scheme is used for keys indexing. A1 is always 0, B1 1, A2 8, ... For example, if user connects a 3x3 matrix keyboard, keys have indexes: 0, 1, 2, 8, 9, 10, 16, 17, 18.

	A	B	C	D	E	F	G	H
1	0	1	2	3	4	5	6	7
2	8	9	10	11	12	13	14	15
3	16	17	18	19	20	21	22	23
4	24	25	26	27	28	29	30	31
5	32	33	…					

Remarks:

Returns False on error.

13.22. GetPWMOutputs

Get complete PWM outputs configuration

```
bool GetPWMOutputs(ref bool[] channels, ref uint period, ref uint[]
duty_values);
```

Arguments:

channels

An array of 6 boolean values, each representing one PWM channel. Channel is enabled if this value is set to true.

period

32-bit PWM period value. PWM module of a PoKeys55 device runs at 12 Mhz, so a value of 12 000 000 produces a period of 1 second. PWM module on PoKeys56 devices runs at 25 MHz, so a value of 25 000 000 produces a period of 1 second.

duty_values

An array of 6 32-bit unsigned integers, each representing the value of PWM duty for each channel. Minimum value is 0, maximum value is the same as period.

Channel to pin mapping:

Channels are mapping according to this table:

Channel	PoKeys pin
0	22
1	21
2	20
3	19
4	18
5	17

Remarks:

When using PWM enabled pins, digital inputs and outputs are inactive. Returns False on error.

13.23. SetPWMOutputs

Set complete PWM outputs configuration

```
bool  SetPWMOutputs(ref  bool[]  channels,  ref  uint  period,  ref  uint[]
duty_values);
```

Arguments:

channels

An array of 6 boolean values, each representing one PWM channel. Channel is enabled if this value is set to true.

period

32-bit PWM period value. PWM module of a PoKeys55 device runs at 12 Mhz, so a value of 12 000 000 produces a period of 1 second. PWM module on PoKeys56 devices runs at 25 MHz, so a value of 25 000 000 produces a period of 1 second.

duty_values

An array of 6 32-bit unsigned integers, each representing the value of PWM duty for each channel. Minimum value is 0, maximum value is the same as period.

Channel to pin mapping:

Channels are mapping according to this table:

Channel	PoKeys pin
0	22
1	21
2	20
3	19
4	18
5	17

Remarks:

When using PWM enabled pins, digital inputs and outputs are inactive. Returns False on error.

14. Interfacing with PoKeys library – C# example

Preinitialization

C#

1. Add a reference to a PoKeysDevice_DLL.dll, located in installation folder
2. Use the class `PoKeysDevice` from the PoKeysDevice_DLL namespace

Visual Basic 6.0

1. Add a reference to a PoKeysDevice_DLL.tlb, located in installation folder
2. Use the class `PoKeysDevice_DLL.PoKeysDevice`

Class initialization

```
PoKeysDevice_DLL.PoKeysDevice cPoKeys = new PoKeysDevice_DLL.PoKeysDevice();
```

Connecting to USB devices – PoKeys55/PoKeys56U

Enumerating the USB devices (this step must be taken even if we know exact device user ID!)

```
int iNumDevices = cPoKeys.EnumerateDevices();
```

The command returns the number of USB PoKeys devices detected on the system.

Getting device's serial number, user ID, firmware version and pin count:

```
int iSerialNumber = 0;
int iFirmwareVersion = 0;
int iPinNum = 0;
byte iUserID = 0;


for (int n = 0; n < iNumDevices; n++)
{
   cPoKeys.ConnectToDevice(n);
   cPoKeys.GetDeviceID(ref iSerialNumber, ref iFirmwareVersion, ref iPinNum);
   cPoKeys.GetUserID(ref iUserID);
   cPoKeys.DisconnectDevice();

   Console.WriteLine(n + ". device: Serial: " + iSerialNumber + "   Firmware: " +
iFirmwareVersion + " User ID: " + iUserID);
}
```

Before any data can be read from or written to the device, the command ConnectToDevice must be executed. It's parameter is a device's index and not the userID! (therefore can be changed when multiple devices are connected at a time).

Connecting to Ethernet devices – PoKeys56E

Enumerating the ethernet devices, connecting and retrieving basic data

The example below enumerates PoKeys56E devices, connects to every one of them and displays its user ID and type in the listbox named lstEthDevices.

```
// Enumerate and list ethernet devices
MyDevice.StartEthernetDiscovery();
byte userID = 0;

lstEthDevices.Items.Clear();
lstEthDevices.Items.Add("Searching....");
lstEthDevices.Enabled = false;
lstEthDevices.Refresh();
Application.DoEvents();
Thread.Sleep(2000);

MyDevice.StopEthernetDiscovery();

if (MyDevice.GetNumberOfDetectedNetworkDevices() > 0)
{
   lstEthDevices.Enabled = true;
   lstEthDevices.Items.Clear();
   cmdEthConnect.Enabled = true;
}
else
{
   lstEthDevices.Items.Clear();
   lstEthDevices.Items.Add("No device detected");
   cmdEthConnect.Enabled = false;
}
for (int k = 0; k < MyDevice.GetNumberOfDetectedNetworkDevices(); k++)
{
     // Three possibilities to connect to network device...
    MyDevice.ConnectToNetworkDevice(MyDevice.EthDeviceIP[k]);

    // Also
    //MyDevice.ConnectToNetworkDevice(MyDevice.GetNetworkDeviceAddress(k));

    // or even (with pointers)
    //PoKeysDevice_DLL.sPoKeysNetworkDeviceData netData = new
PoKeysDevice_DLL.sPoKeysNetworkDeviceData();
    //IntPtr ptr =  Marshal.AllocHGlobal(200);
    //Marshal.StructureToPtr(netData, ptr, false);

    //MyDevice.COM_GetNetworkDeviceData(ptr, 0);

    //netData = (PoKeysDevice_DLL.sPoKeysNetworkDeviceData)Marshal.PtrToStructure(ptr,
typeof(PoKeysDevice_DLL.sPoKeysNetworkDeviceData));
    //MyDevice.ConnectToNetworkDevice(netData.IPAddress);

   MyDevice.GetUserID(ref userID);
   lstEthDevices.Items.Add(MyDevice.GetDeviceTypeName() + " (" + userID + ")");

   MyDevice.DisconnectDevice();
}
```

Remarks: the array `MyDevice.EthDeviceIP` holds the IP addresses of all detected devices in the process of automatic discovery.

PoKeys56E has a short connection timeout setting of 3 seconds (default, but can be altered in the Device>Network device settings... menu. After no data is received for this period, the connection with the host is terminated.

Reading pin configuration

```
byte iPinFunction = 0;

cPoKeys.GetPinData(0, ref iPinFunction);
```

In this example 0 (Pin 1) is used for a pin ID. Pin IDs are 0 based.

iPinFunction has the following structure

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Pin invert	reserved	reserved	A output	A input	D output	D input	reserved

where A stands for analog and D for digital.

Reading pin key mapping (only on USB PoKeys devices – PoKeys55 and PoKeys56U)

Let us presume that pin 2 is defined as keyboard digital input with direct key mapping

```
byte iPinKey = 0;
byte iPinModifier = 0;
byte iMappingType = 0;

cPoKeys.GetPinKeyMapping(1, ref iMappingType, ref iPinKey, ref iPinModifier);
```

iPinKey is a key code as described in USB HID standard
iPinModifier is a modifier for a key (Ctrl, Alt, Shift, Win key) and can be used with these masks:

```
const byte CtrlMask = 1;
const byte ShiftMask = 2;
const byte AltMask = 4;
const byte WinMask = 8;
const byte AltGrMask = 64;
```

iMappingType has the following structure

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
reserved	reserved	reserved	reserved	reserved	macro	direct	enable

bit 0 – Enable key mapping: to enable key mapping for a specific pin, this bit must be set to 1
bit 1 – enable direct key mapping: when this bit is set to 1, pin actions are directly reflected as a keyboard key
bit 2 – enable keyboard macro mapping: when this bit is set to 1, special macro sequence is sent on pin activation

There can be only one of the bits 1 or 2 set!

Reading input value

Let us presume that pin 3 is configured as digital input.

```
bool bInputVal = false;
cPoKeys.GetInput(2, ref bInputVal);
Console.WriteLine("Input 3 is " + (bInputVal?"On":"Off"));
```

Reading analog input value

Let us presume that pin 43 is configured as analog input.

```
// Read analog value
int value = 0;
MyDevice.GetAnalogInput(42, ref value);

MessageBox.Show("Analog value: " + (3.3 * (double)value /
     MyDevice.GetAnalogValueMax()).ToString("0.00") + " V");
```

Joystick axis mapping (only on USB PoKeys devices – PoKeys55 and PoKeys56U)

This is only possible on pins 43-47. If this is used on any other pin, the function will fail or be ignored.

```
cPoKeys.SetJoystickAxisMapping(42, iJoystickAxis);
```

iJoystickAxis can be used as follows:

0	None
1	Rx
2	Ry
3	X
4	Y
5	Throttle

Block read - digital

It is possible to poll 32 input pins with one request. All 55 pins can be read with two request joined in single command.

```
// Read pins 1 to 32
bool[] values_1_32 = new bool[32];

myDevice.BlockGetInput1(ref values_1_32);

// Read pins 33 to 55
bool[] values_33_55 = new bool[23];

myDevice.BlockGetInput2(ref values_33_55);

// Read all pins (1-55)
bool[] values = new bool[55];

myDevice.BlockGetInputAll55(ref values);
```

Block read - analog

It is possible to poll 4 8-bit or 3 10-bit analog inputs with one command.

```
// 8-bit mode
byte[] channels = { 42, 43, 0, 45 };
byte[] values = new byte[channels.Length];

myDevice.BlockGetAnalogInput8bit(ref channels, ref values);

byte value1 = values[0];
byte value2 = values[1];
byte value3 = values[3];

// 10-bit mode
byte[] channels = { 42, 43, 45};
int[] values = new int[channels.Length];
```

```
myDevice.BlockGetAnalogInput10bit(ref channels, ref values);

int value1 = values[0];
int value2 = values[1];
int value3 = values[2];
```

Block write - digital

It is possible to set 32 output pins with one request. All 55 pins can be set with two request joined in single command.

```
// Simple 8-bit binary counter
bool[] states = new bool[32];

for (int n = 0; n < 255; n++)
{
  for (int i = 0; i < 8; i++)
  {
    if ((n & (1 << i)) > 0) states [i] = false; else states [i] = true;
  }
  MyDevice.BlockSetOutput1(ref states); // Update pins 1 to 32
}
```

Reading encoder RAW values

RAW values from the encoder inputs can be read with following command.

```
byte iEncoderValue = 0;

cPoKeys.GetEncoderValue(1, ref iEncoderValue);
```

iEncoderValue is a value between 0 and 255.

Macro operations (only on USB PoKeys devices)

Create new macro

This command creates macro in first free position. It returns macro index.

```
byte iMacroID = 0;
byte iMacroLen= 10;

cPoKeys.MacroCreate(iMacroLen, ref iMacroID);
```

Modify macro length

This command modifes macro length.

```
byte iMacroID = 0;
byte iMacroNewLen = 50;

cPoKeys.MacroModifyLength(iMacroID, iMacroNewLen);
```

Delete macro

This command deletes specific macro.

```
byte iMacroID = 0;

cPoKeys.MacroDelete(iMacroID);
```

Save macro configuration to flash

This command saves the current macro configuration to flash.

```
cPoKeys.MacroSaveConfiguration();
```

Change macro name

This command changes the macro name. Name property supports up to 7 characters.

```
byte iMacroID = 0;
cPoKeys.MacroSetName(iMacroID, "Macro1");
```

Set macro key

This command sets one macro key at the position iIndex. This index must be between 0 and iMacroLen - 1.

```
byte iIndex = 5;
byte iKeyCode = 10;
byte iKeyModifier = 0;

cPoKeys.MacroSetKey(iMacroID, iIndex, iKeyCode, iKeyModifier);
```

Get free space for macros

Space for saving macros is limited. To find out how much free space exists, use the following command.

```
int iFreeSpace = 0;

cPoKeys.MacroGetFreeSpace(ref iFreeSpace);
```

Get the list of macros' states

If the macro has the length of 0 it is designated as inactive. This command retrieves the list of states for all the macros. If specific macro is active, bActiveMacros has the value True.

```
bool[] bActiveMacros = new bool[64];

cPoKeys.MacroGetActiveMacros(ref bActiveMacros);
```

Display current time on LCD

```
// Initialize library
PoKeysDevice_DLL.PoKeysDevice dev = new PoKeysDevice_DLL.PoKeysDevice();

// Enumerate devices and connect to first (we have only 1 connected)
dev.EnumerateDevices();
dev.ConnectToDevice(0);

// Set settings for 4x20 LCD
dev.LCDSetSettings(1, 4, 20);
// Initialize LCD
dev.LCDInit();
// Clear LCD
dev.LCDClear();
// Move cursor to home
dev.LCDGotoXY(1, 1);
// Print Hello, world!
dev.LCDPrint("Hello, world!");

// 1s delay
System.Threading.Thread.Sleep(1000);
```

```
while (true)
{
  Application.DoEvents();
  System.Threading.Thread.Sleep(100);

  // Move cursor to home
  dev.LCDGotoXY(1, 1);
  // Print current date and time
  dev.LCDPrint(DateTime.Now.ToString());
}
```

Configuring PWM outputs

Configures pin 17 as PWM output with 20 ms period and 1.5 ms (7.5%) duty cycle. If a model RC servo signal input is connected to this pin, servo motor horn should position itself in the middle position.

```
bool[] channel = new bool[6];
uint[] duty = new uint[6];

// PWM base clock is 12 MHz (or 25 MHz on PoKeys56 devices), so 1 ms takes 12000 (25000)
cycles
float ms = MyDevice.GetPWMFrequency() / 1000;
uint period = (uint)(ms * 20);   // 20 ms period

channel[5] = true; // Pin 17 = channel 5 (Pin 18 = channel 4, ...)
duty[5] = (uint)(ms * 1.5); // Set duty cycle to 1.5 ms (7.5 %)

MyDevice.SetPWMOutputs(ref channel, ref period, ref duty);
```

Configuring matrix keyboard

Configures a 4x4 matrix keyboard, connected to first eight pins of PoKeys device.

```
byte[] rows = new byte[16]; // This must always be the size of 16
byte[] cols = new byte[8];  // This must always be the size of 8

bool[] macros = new bool[128];      // This must always be the size of 128
byte[] keys = new byte[128];         // This must always be the size of 128
byte[] modifiers = new byte[128];    // This must always be the size of 128

for (int i=0; i<4; i++)
{
    cols[i] = (byte)i;
    rows[i] = (byte)(i+4);
}

byte configuration = 1;
byte numrows = 4;
byte numcols = 4;

// Set column pins as digital inputs
MyDevice.SetPinData(0, 1 << 1);
MyDevice.SetPinData(1, 1 << 1);
MyDevice.SetPinData(2, 1 << 1);
MyDevice.SetPinData(3, 1 << 1);

// Set row pins as digital outputs
MyDevice.SetPinData(4, 1 << 2);
MyDevice.SetPinData(5, 1 << 2);
MyDevice.SetPinData(6, 1 << 2);
MyDevice.SetPinData(7, 1 << 2);

MyDevice.SetMatrixKeyboardConfiguration(ref configuration, ref numcols, ref numrows, ref
rows, ref cols, ref macros, ref keys, ref modifiers);
```

Reading matrix keyboard status

Reads all matrix keyboard keys statuses.

```
bool[] KeyStates = new bool[128];

MyDevice.GetMatrixKeyboardKeyStatus(ref KeyStates);

string status = "";
for (int row = 0; row < 4; row++)
{
    for (int col = 0; col < 4; col++)
    {
        status += KeyStates[row * 8 + col] ? "1 " : "0 ";
    }
    status += "\n";
}

MessageBox.Show("Key statuses:\n" + status);
```

Write data to PoExtBus device

Sends the byte to the first PoExtBus device. As the devices can be daisy-chained, the first device connected to PoKeys device has the index 9. If any device is daisy-chained to this device, it would have the index 8, etc.

```
byte[] dataOut = new byte[10];

// Set some value
dataOut[9] = 0xAA;

MyDevice.AuxilaryBusSetData(1, dataOut);
```

Write data to matrix LED display

This example activates the matrix LED display 1 as 8x8 display, clears it and draws a + sign on it.

```
// Enable LED1
MyDevice.MatrixLEDSetSettings(true, 8, 8, false, 8, 8);

// Invert the whole matrix
MyDevice.MatrixLED1ClearAll(true);

// Draw a dark + sign
MyDevice.MatrixLED1SetPixel(2, 1, false);
MyDevice.MatrixLED1SetPixel(1, 1, false);
MyDevice.MatrixLED1SetPixel(0, 1, false);
MyDevice.MatrixLED1SetPixel(1, 0, false);
MyDevice.MatrixLED1SetPixel(1, 2, false);
```

Read tick counter

Every PoKeys device has a tick counter, a counter that internally counts milliseconds.

```
uint ticks = 0;

MyDevice.GetTickCounter(ref ticks);

MessageBox.Show("Tick counter: " + ticks);
```

Read temperature from the LM75 sensor, connected to I²C bus

```
byte stat = 0;
byte[] data = new byte[32];

// Write 0 to the device at the address 0x90 (LM75 command: set address)
data[0] = 0;
MyDevice.I2CStartWrite(0x90, 1, data);
MyDevice.I2CGetWriteStatus(ref stat);
```

```
if (stat != 1)
{
    MessageBox.Show("Error writing");
    return;
}

// Read temperature - read two bytes from the device at the address of 0x90
MyDevice.I2CStartRead(0x90, 2);
// This operation was quite fast, so no wait is needed...
MyDevice.I2CGetReadStatus(ref stat, ref data);

if (stat == 1)
{
    MessageBox.Show("Temperature: " + data[0] + "°C");
}
else
{
    MessageBox.Show("Error reading");
}
```

Read temperature from DS18B20 sensor, connected to 1-wire bus

```
byte stat = 0;
byte[] data = new byte[16];

MyDevice.prot1WireSetStatus(true);

data[0] = 0xCC;
data[1] = 0x44;
MyDevice.prot1WireStartWriteAndRead(2, 0, data);

Thread.Sleep(1000);

data[0] = 0xCC;
data[1] = 0xBE;
MyDevice.prot1WireStartWriteAndRead(2, 9, data);

Thread.Sleep(10);

byte len = 0;
MyDevice.prot1WireGetReadStatus(ref stat, ref data, ref len);

if (stat == 1)
{
    MessageBox.Show("Temperature: " + (((data[1] << 8) + data[0]) >> 4) + "°C");
}
else
{
    MessageBox.Show("Error reading");
}
```

15. Major changes from 1.x to 1.7:

To move the PoKeys55 device to a new level some major changes to interface were imminent.

Pi n function 1 was removed (this was directly key mapped pin function). Instead, key mapping functionality was added to any digital input pin. Key mapping type (none/direct/macro) can be set via changed Key mapping command as shown in above example.

15.1. Pin 13 not functioning appropriately

On PoKeys55 devices with serial numbers greater than 10133 and lower than 11500 there is a flawed connection for pin 13. Please do not use this pin.

Note: on PoKeys55 boards with serial number above 11500, this problem is removed.

15.2. Putting pin 4 low on startup disables PoKeys55 device from booting.

Avoid connecting switches that can be closed on startup to this pin.

Note: on PoKeys55 boards with serial number above 11500, this problem is removed.

16. Errata information

This section describes special limitations of the device.

Pins 5 and 6 cannot be separately set as outputs/inputs

Pins 5 and 6 must be both set to either input or output

Affected: PoKeys56E, PoKeys57E

Wrong power supply was specified for PoNET connector in the documentation

PoNET connector has 5V power available at pin 1 and not 3.3V as previously stated.

Stepper motor outputs of the internal pulse generator don't generate proper step signals for axes y and z.

Pins 48 and 49 require an external 470Ω pull-up resistor for Pulse engine operation.

Affected: PoKeys56U

17. Grant of license

support: www.poscope.com