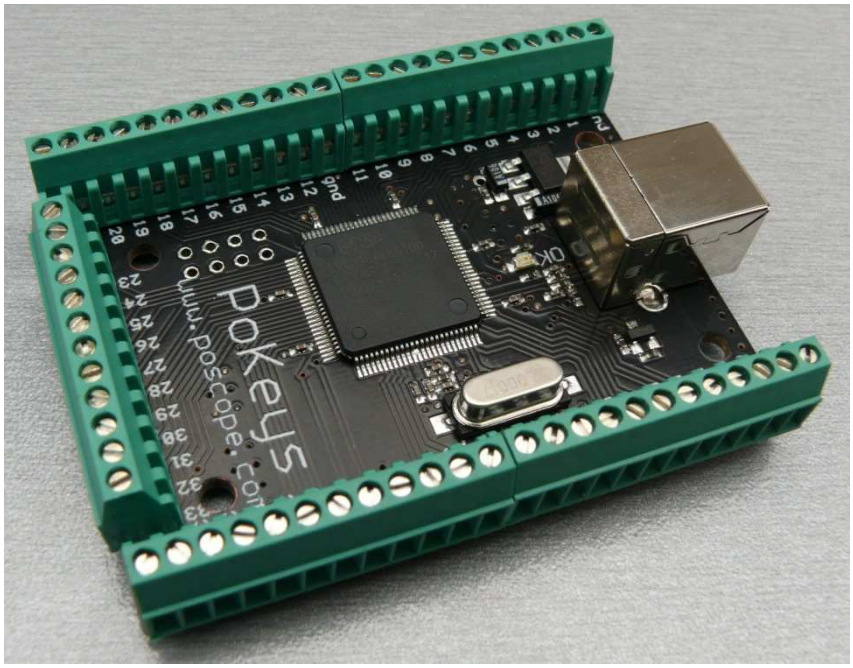
PoKeys 55

User's manual



PoKeys55 user's manual

1. Description

PoKeys55 is simple, easy-to-use USB device that combines standard keyboard and joystick simulation. PoKeys55 enables user to design specially built robust computer interface, comprising only the mechanical keys and PoKeys55 device. The device is highly adjustable and as such requires no complex knowledge on device programming.

If additional input and output capabilities are needed, the device provides 55 digital 5V tolerant inputs or outputs, 5 10-bit analog inputs and one 10-bit analog output. They are controlled via included software, which enables user either to use the highly intuitive graphical user interface or advanced console type interface. Chosen settings can be stored on device, so no special software is needed on target system.

NEW: From version 1.7 on, encoder support is available on any of the PoKeys55 inputs. User can freely choose two pins on which encoder A and B channel signals will be connected to. Configuration software allows assigning virtual keyboard keys separately for both directions.

There is also new capability to assign a special keyboard macro sequence instead of direct key mapping or encoder key mapping.

PoKeys devices are fully supported with all new software provided.

2. Features

- Compatible with USB 1.1/2.0 HID standard
- Standard USB keyboard simulation
- Standard USB joystick simulation
- 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
- 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
- Intuitive and user-friendly software

3. Requirements

- One available USB 1.1 or USB 2.0 port
- USB HID device driver enabled operating system (Windows 98 SE/ME/2000/XP/Vista, Linux, Mac OS)
- Included software requires Windows 2000/XP/Vista with .NET framework 2.0 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).

4. Installation

PoKeys55 is a USB 1.1/2.0 compliant device and as such requires no additional drivers for operation as a standard USB keyboard and joystick.

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.

5. Using PoKeys55 configuration application

PoKeys55 settings application is the utility for setting up the device for normal use. Upon starting the program, the main window (Figure 1) with connection dialog is displayed. If there are PoKeys55 devices detected to be attached to the system, the device selection box will be populated with all devices. To easily identify a specific device, the User ID number is appended to the default device name. To start editing device settings, click the 'Connect' button. After the current configuration is uploaded from the device, the user interface is enabled (Figure 2).

There is graphical representation for configuration of each PoKeys55 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.

There are 5 main pin functions possible: inactive, digital input, digital output, analog input and analog output.

5.1. Inactive

Any pin can be set as inactive. Inactive pin is put in high-Z state with internal pull-up resistors enabled.

5.2. Digital input

Any one of the 55 pins can be configured as digital input by selecting 'Digital input' option box. All input pins have a weak pull-up resistor enabled and are 5V tollerant. If the pin polarity is inverted, check the 'Invert pin' box.

There are several additional possibilities for digital input pin functions.

5.2.1. Direct key mapping

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 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.

Send ((opening bracket): This key kombination differs from your system regional settings. As the PoKeys55 emulates 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.

5.2.2. Keyboard macro

If there is a need for more than one key to be sent on pin activation, there is a possibility to assign a keyboard macro to a pin. Please see section Keyboard macros for more information on editing and assigning keyboard macros.

5.2.3. Encoder input

Rotational encoder switch can be used with PoKeys55 digital inputs. It is possible to connect up to 25 encoders to one PoKeys55 device. To enable encoder input, first select encoder index with numerical up-down selector, then select appropriate encoder channel. The last step is to check the box 'Encoder'.

Same as simple digital inputs, encoders can be assigned to direct key mapping or keyboard macro. This is possible for both directions (CW and CCW) separately. Simply set one mapping for channel A and another for channel B. To check for proper connection and settings, there is a special dialog that displays current encoder state. Please see section 'Displaying encoder RAW values'.

5.3. Digital output

Any one of the 55 pins can be configured as digital output by selecting 'Digital output' option box. Each 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. If the polarity of the pin is inverted, check the 'Invert pin' box.

5.4. Analog input

Analog input function is only available for pins 43 to 47. These analog inputs can also be freely mapped to any of the 5 joystick axes; X,Y, rotation X, rotation Y and throttle (Figure 3). To monitor current analog input value please see section Analog input values box.

5.5. Analog output

Analog output function is only available for pin 43. It is possible to set analog output voltage for this pin with 10-bit resolution. Console application or third party control application must be used to set a value for this pin!

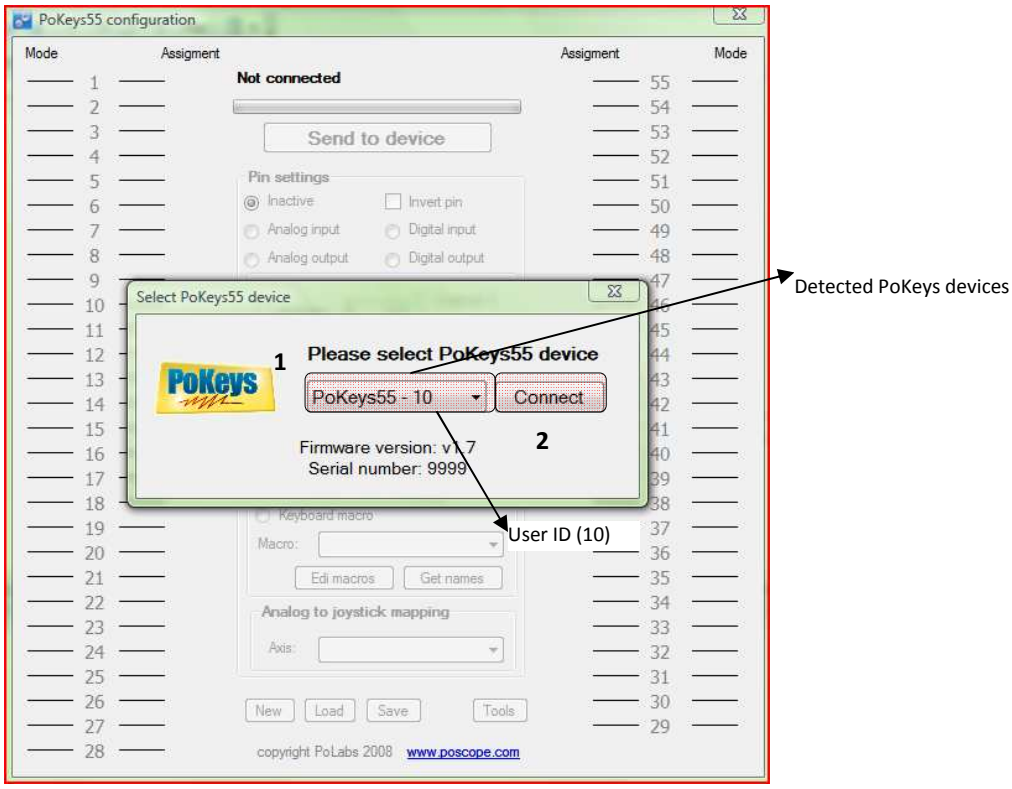


Figure 1: Main window

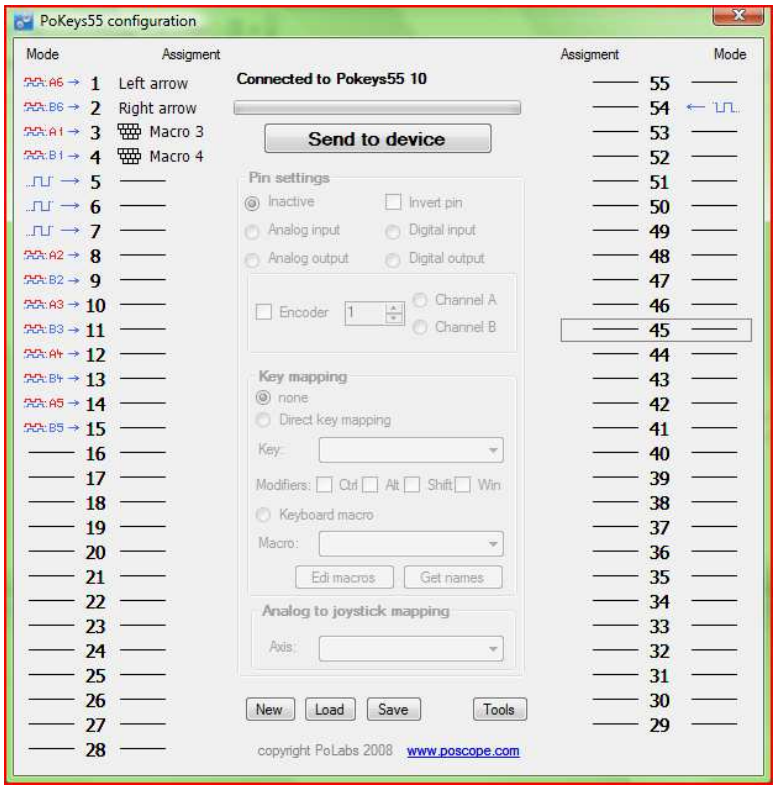


Figure 2: Connected to the PoKeys55 device with User ID 1

5.6. Keyboard macros

PoKeys55 device now supports 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 macro, click the 'Edit macros' button. The following dialog appears

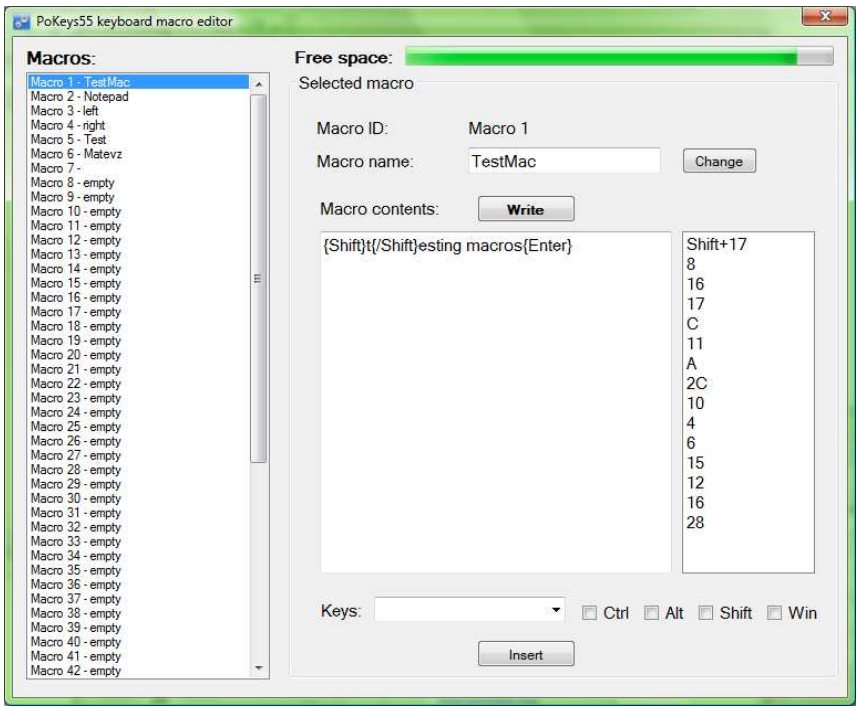


Figure 3: Macro editing dialog

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 digital macro content.

5.7. Displaying encoder RAW values

To open encoder RAW values dialog, click the 'Tools' command button on main window and select 'View encoder RAW values'. The following dialog below appears. It simply shows the list of all encoders and their current values.

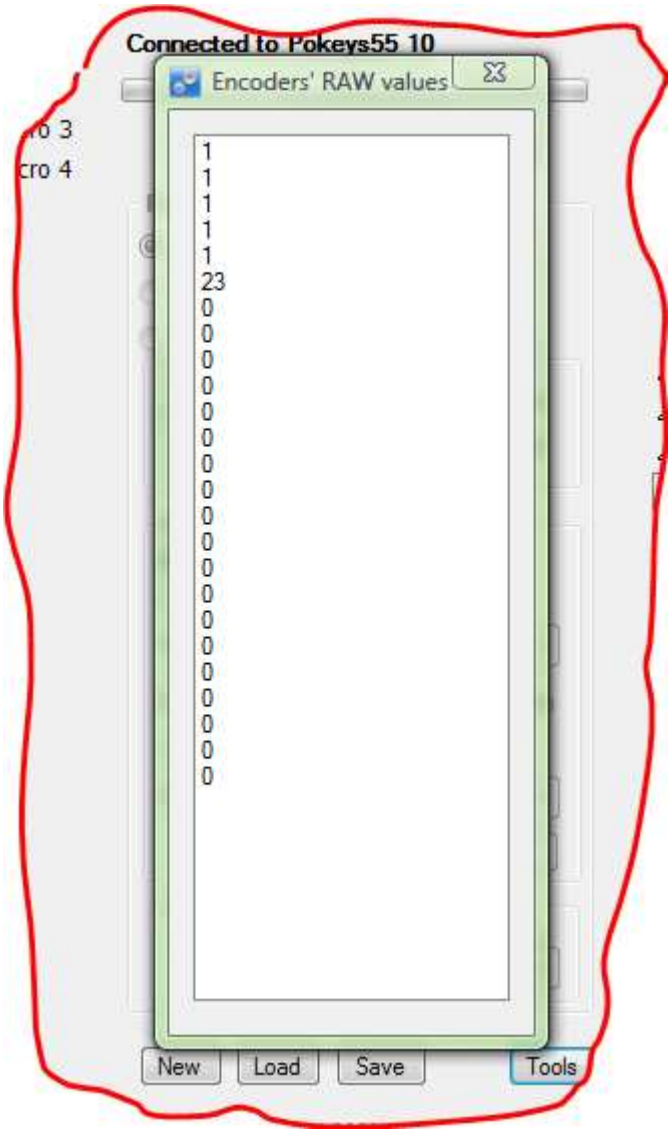


Figure 4: Encoders' RAW values

5.8. Analog inputs dialog

To open analog inputs dialog, click the 'Tools' command button on main window and select 'View analog inputs'. The following dialog below appears. To enable display of analog channel, check the appropriate check box. It is enabled only when the input is set up as analog input.

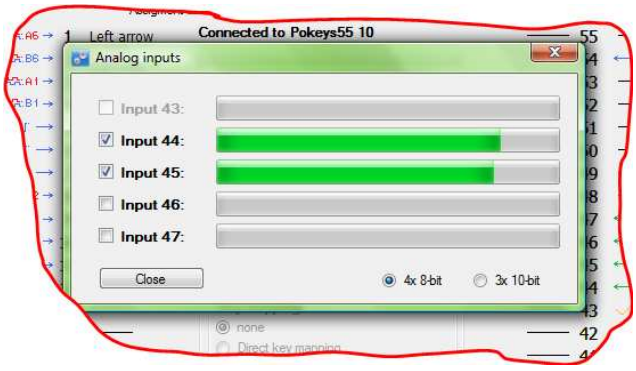


Figure 5: Analog inputs dialog

5.9. Changing User ID number

Users can freely assign their own User ID number that represents a specific PoKeys55 device (enables distinguishing between different PoKeys55 devices in case there is more than one connected to a single host PC). To change the User ID number, click the 'Tools' button and click ‘Show device details…’. The About dialog (Figure 6) will be displayed, giving detailed information about the device, such as Serial number, Firmware version and User ID. Simply enter any number between 0 and 255, and click the 'Change user ID' button.

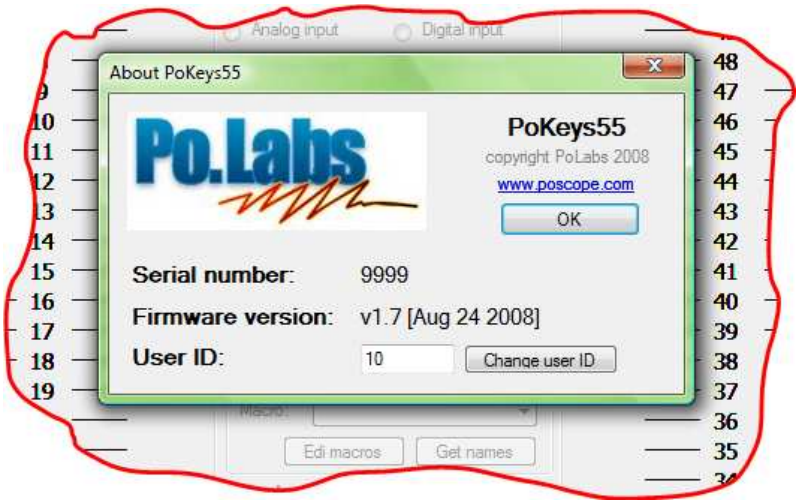


Figure 6: About dialog

5.10. Saving current configuration to file

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

5.11. Example: starting a program on Windows using PoKeys55 device

On a Windows operating system, users can assign a custom shortcut key to any program shortcut. Find the shortcut, 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 PoKeys55 application and connect to the desired PoKeys55 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.

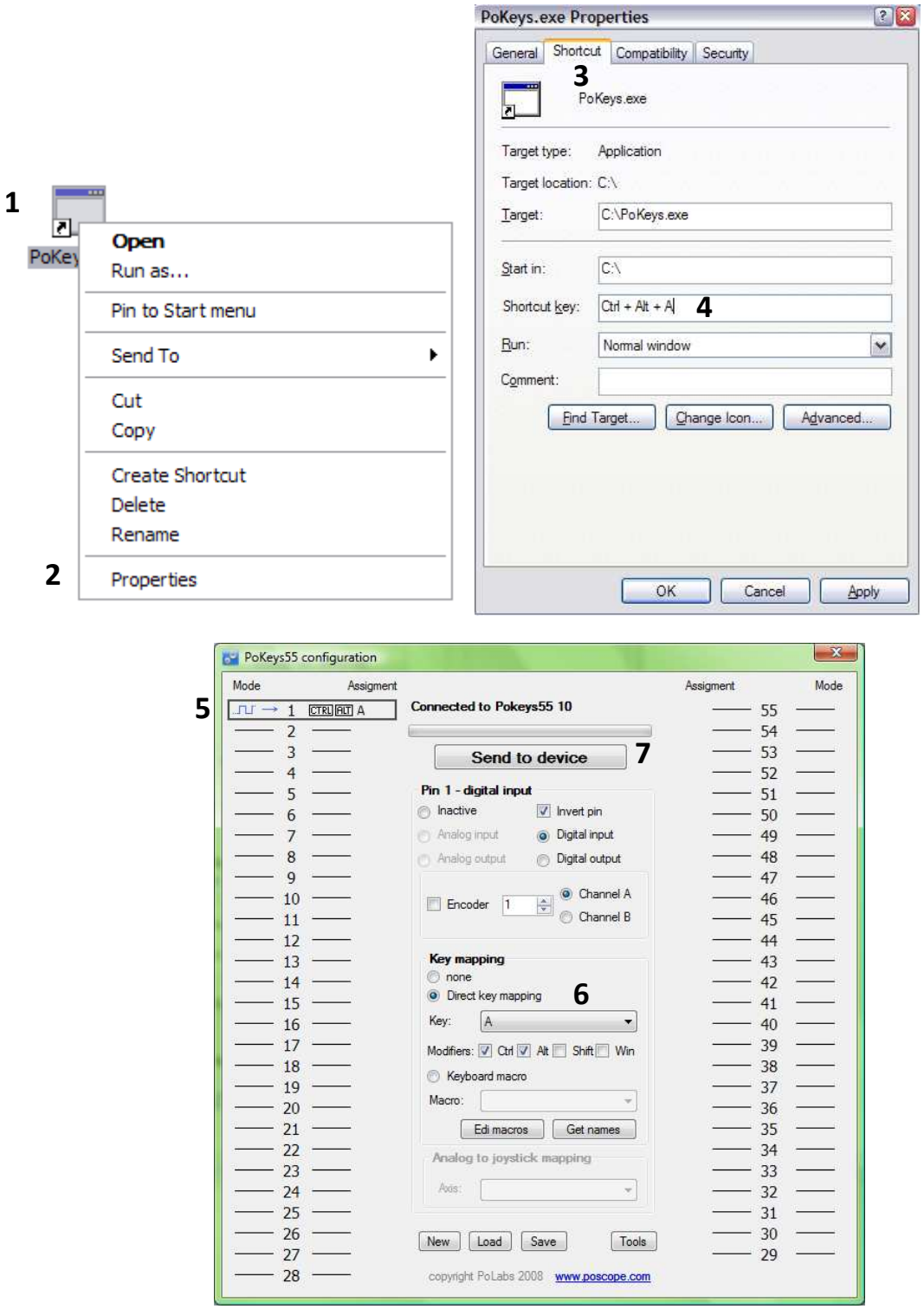


Figure 7: Setting up PoKeys55 device

6. Communicating with the device using console

In the software bundle included with the PoKeys55 device, there is also a console interface application, which enables command-line style communication with the device (figure 6). To start using the console application, go Start>Run…, type cmd and press Enter. Navigate to the folder, where PoKeysConsole.exe is located (usually C:\Program Files\PoLabs\).

6.1. Supported operations

Enumerate PoKeys55 devices

Command line: PoKeysConsole.exe -e

Description: Enumerates and prints out all the detected PoKeys55 devices with their User IDs.

Get details of the specific PoKeys55 device

Command line: PoKeysConsole.exe –d<user ID>

Description: Prints out the detailed description of the device, i.e. the device's serial number, firmware version and User ID.

Example: PoKeysConsole.exe –d1

Connect to PoKeys55 device

Command line: PoKeysConsole.exe –c<user ID>

Description: Before any operation can be executed, host software must connect to PoKeys55 device, using Connect to PoKeys55 device operation.

Example: PoKeysConsole.exe –c1

Save current configuration to flash memory

Command line: PoKeysConsole.exe –w

Description: After the settings have been changed, they need to be sent to the device. This is accomplished with via the Save configuration operation. 'Connect to PoKeys55 device' operation must be executed before this operation!

Example: PoKeysConsole.exe –c1 –w

Get current pin setting

Command line: PoKeysConsole.exe –g<pin ID>

Description: Prints out the current pin setting. If the pin ID parameter is omitted, settings for all the pins are printed out.

Example: PoKeysConsole.exe –c1 –g10

Set pin setting

Command line: PoKeysConsole.exe –s<pin ID>,<pin function>,+/-

Description: Enables the desired function on the selected pin.

Pin function	Value
Inactive	0
Digital input	2
Digital output	4
Analog input	8
Analog output	16

The last parameter is used to define polarity of digital input and output pins. It must be either + (non-inverted polarity) or - (inverted polarity).

Example: PoKeysConsole.exe –c1 –s10,2,-

Get digital input value

Command line: PoKeysConsole.exe –i<pin ID>

Description: Reads and prints out current digital input value on selected pin.

Example: PoKeysConsole.exe –c1 –i10

Set digital output value

Command line: PoKeysConsole.exe –o<pin ID>,0/1

Description: Sets the digital output to specified value.

Example: PoKeysConsole.exe –c1 –o11,1

Get analog input value

Command line: PoKeysConsole.exe –a<pin ID>

Description: Reads and prints out current analog input value on selected pin. Pin ID must be between 43 and 47, since only these pins support analog to digital conversion.

Example: PoKeysConsole.exe –c1 –a43

Set analog output value

Command line: PoKeysConsole.exe –b<pin ID>,value

Description: Sets the digital output to specified value. Value can be any number between 0 (0 V) and 255 (3.3 V).

Example: PoKeysConsole.exe –c1 –b43,50

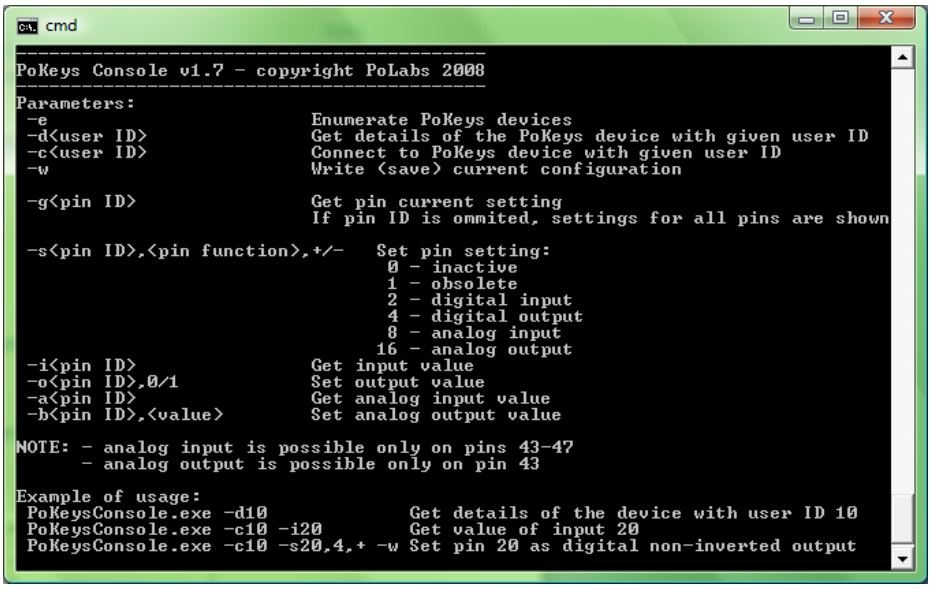
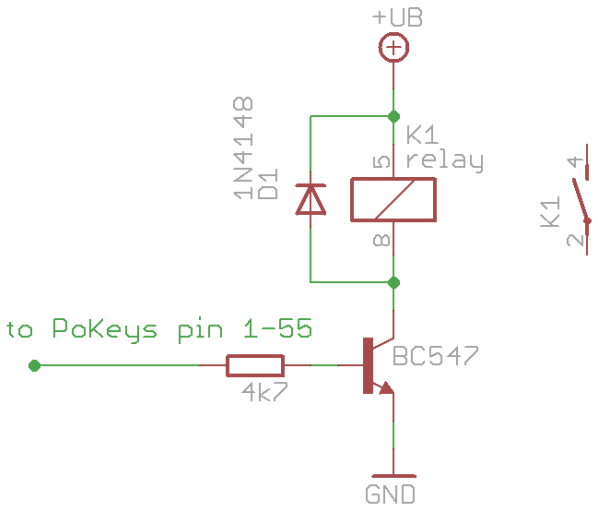


Figure 8: Console application window

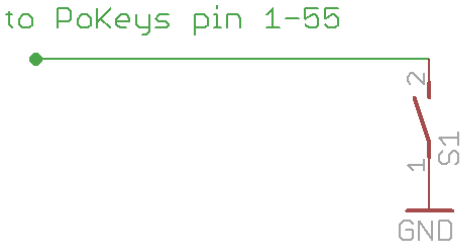
7. Connection common peripherals to PoKeys55 device

1. *Relays*



2. *LEDs*

3. Switches

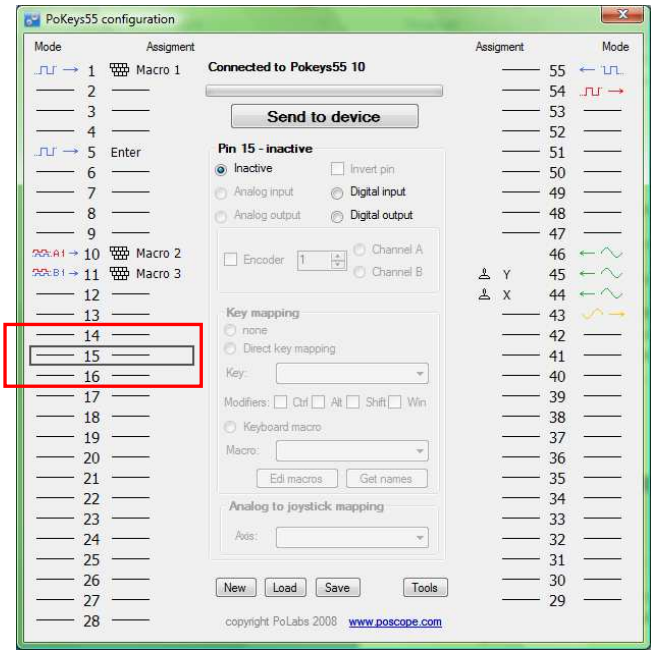


Example: Setting up key mapping

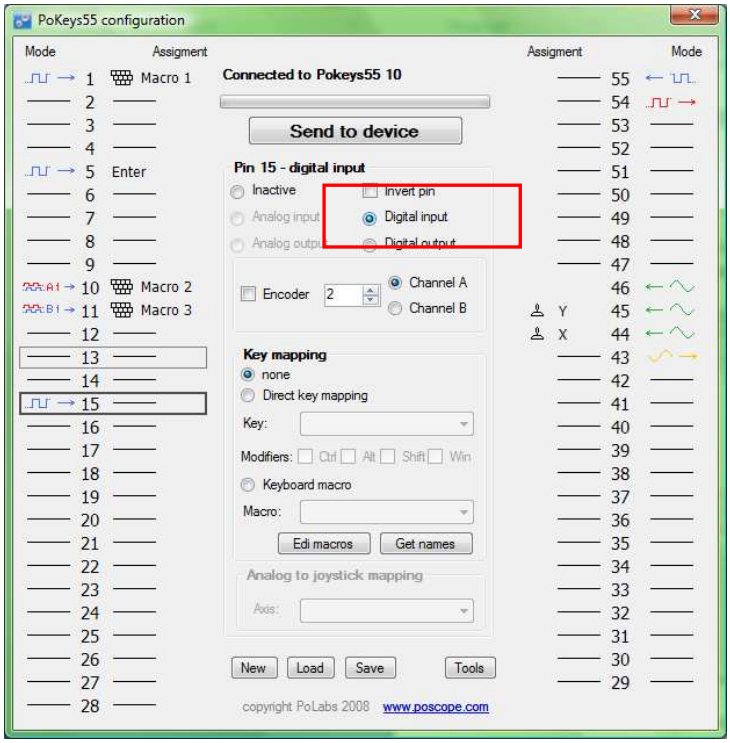
This example shows how easy is to set up a digital input pin for direct key mapping

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

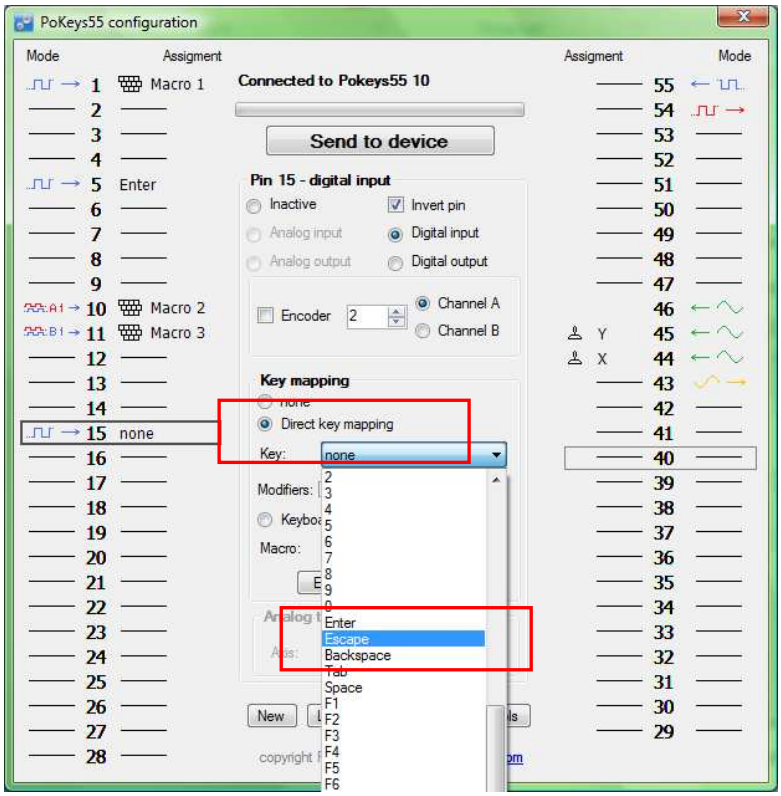
1. Connect a switch to PoKeys55 device as shown above
2. Open PoKeys55 configuration application
3. Select your PoKeys55 device from drop-down box and click ‘Connect’ button
4. Wait the application to load current configuration from PoKeys55 device
5. Click the same pin number as you connected a switch to (in this example pin 15)



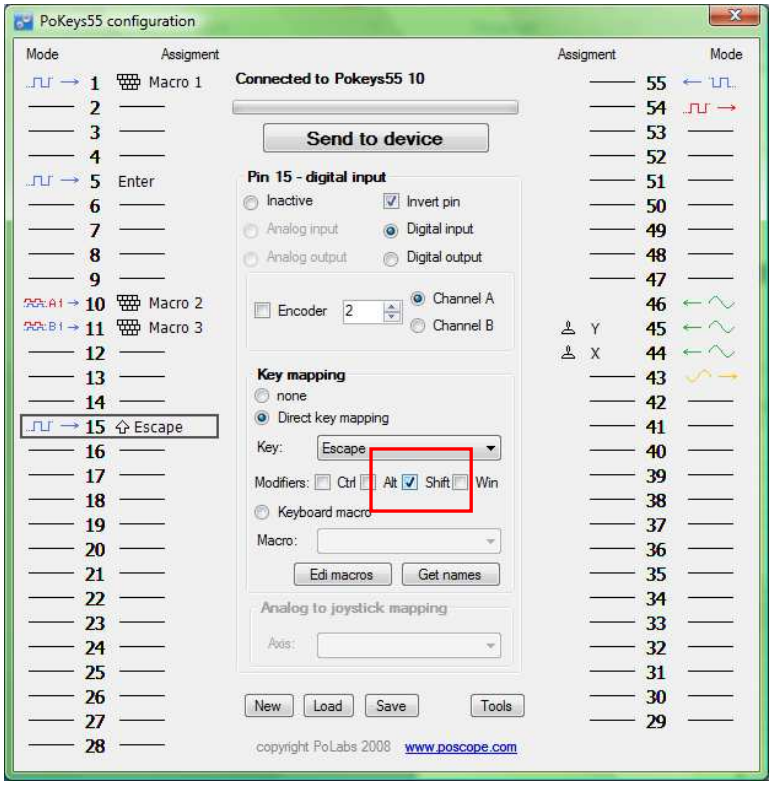
6. Set this pin as digital input



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

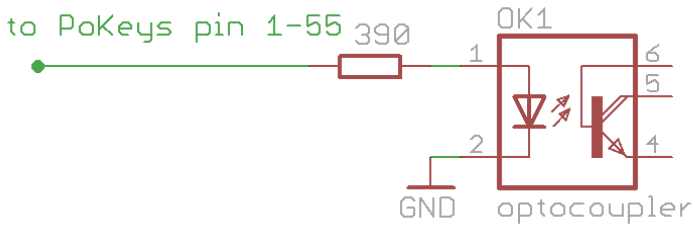


8. Click on the ‘Shift’ checkbox to enable Shift modifier

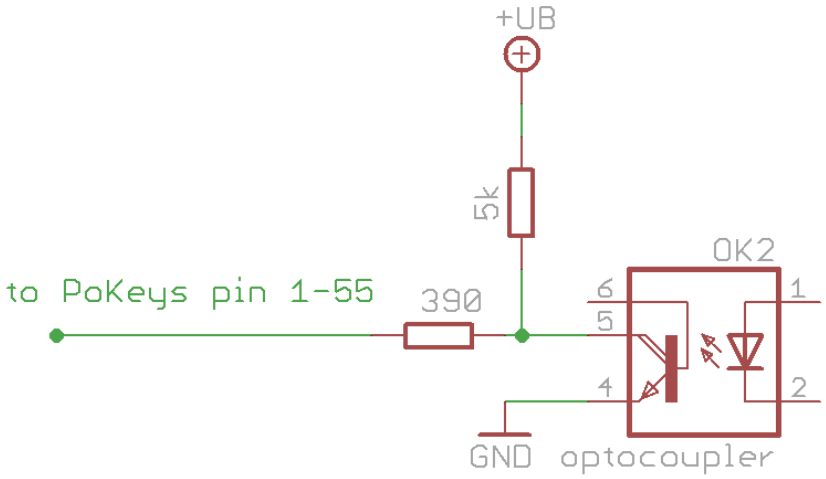


9. Send configuration to device by clicking ‘Send to device button’.

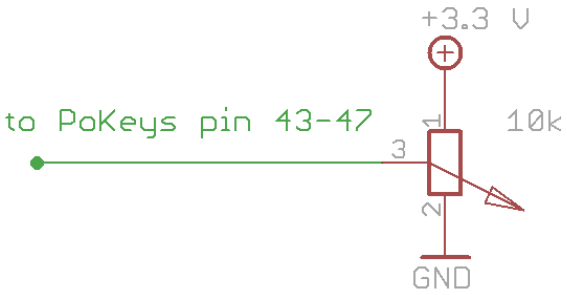
4. Optocoupled digital output



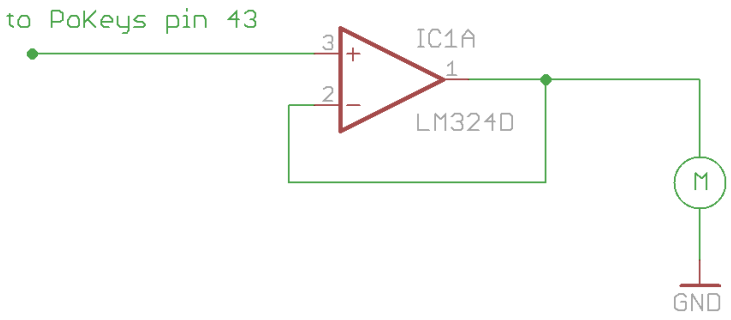
5. *Optocoupled digital input*



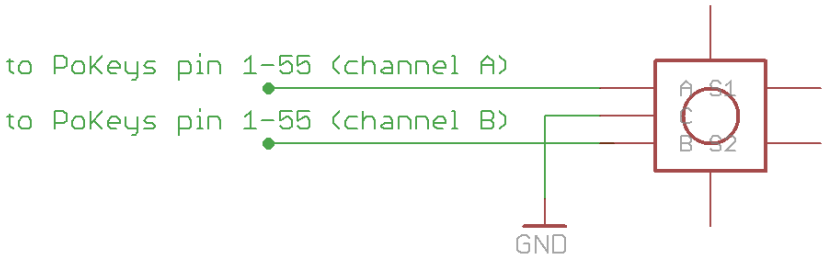
6. *Potentiometers (variable resistors)*



7. *Linear motor control*



8. *Rotational encoder switch*



8. Quick resetting the device configuration

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 PoKeys55 device from USB
2. Find pin labeled ‘Reset’ on the PoKeys55 device (otherwise use pin numbered 54)
3. Short this pin to ground (GND) and reconnect the PoKeys55 device to USB
4. Green light should start flashing rapidly
5. Wait approximately 5s, the light will stop flashing
6. PoKeys55 device will be reconnected with default settings

PoKeys55 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\PoKeys55\ on Windows 2000, XP or C:\Users\{username}\AppData\Local\PoKeys55\), named backup1.pkc, backup2.pkc and backup3.pkc with backup3.pkc being the oldest configuration.

9. 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.

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

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'.

How do I connect switch/relay/LED/... to PoKeys55 device?

Please see the section 'Connection common peripherals to PoKeys55 device.'

I have two (or more) PoKeys55 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 '5.9 Changing User ID number'.

10. Interfacing with PoKeys55 library – C# example

Preinitialization

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

Class initialization

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

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

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

The command returns the number of 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).

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

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"));
```

Joystick axis mapping

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.

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);
```

11. 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.

12. 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.

1. 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.

2. Usage

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

3. 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.

4. 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.

5. 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.

6. 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.

7. Viruses

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

8. 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.

9. 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.

10. Trademarks

Windows is a registered trademark of Microsoft Corporation. PoKeys, PoKeys55, PoScope, PoLabs and others are internationally registered trade marks.

support: www.poscope.com