

# *PiKoder/SSCe* User's Guide

---

Version 1.0  
dated 10/18/20

Gregor Schlechtriem  
[webmaster@pikoder.com](mailto:webmaster@pikoder.com)

[www.pikoder.com](http://www.pikoder.com)

# Content

<b>Overview</b>	<b>3</b>
<b>Features</b>	<b>5</b>
<b>Pin Description and Packaging</b>	<b>7</b>
Description of pins.....	8
<b>Standard application</b>	<b>9</b>
<b>The PiKoder Control Center PCC</b>	<b>11</b>
Getting started.....	11
Type .....	13
Position .....	13
miniSSC Offset (not available in free version) .....	13
Save Parameters .....	13
<b>Serial Interface</b>	<b>15</b>
Mini SSC Protocol (limited support in free version) .....	15
ASCII Command Interface.....	18
<b>Connect the SSCe to a Raspberry Pi</b>	<b>21</b>
Software configuration and hardware setup .....	21
<b>Connect the SSCe to an Arduino</b>	<b>23</b>
Interface Software.....	23

---

# 1

---

## Overview

The PiKoder/SSCe is a single chip solution for implementing a serial servo controller (SSC) which gives you full control of up to eight servos or electronic speed controls with a resolution of 1  $\mu$ s through a serial interface. Alternatively, the servo outputs can be configured as binary outputs (switch function). This User's Guide covers the features, the programming, and the serial interface of the PiKoder/SSCe.

In section 2 you will find a brief description of key features, the overall function and an overview of the interfaces supported. It is recommended to carefully read this section to get a good basic understanding of the PiKoder/SSCe.

The next two sections 3 and 4 deal with the controller hardware. Section 3 provides the pinning and section 4 describes the reference schematic for the PiKoder. **Please note that the applications and interfaces described in the following sections assume that the PiKoder/SSCe is setup in line with this reference schematic.**

Your next step is most likely to commission and test your PiKoder/SSCe. Rather than using the “bits and bytes”-serial interfaces directly, which are laid out in section 6, you may consider using the more elegant PCC (PiKoder Control Center) Windows 10 software with a graphical user interface.

Section 7 demonstrates how you would interface your PiKoder/SSCe to a Raspberry PI and Section 8 focusses on connecting the PiKoder/SSCe to an Arduino.

This User's Guide is based on the most recent hard- and firmware version 1.0 available for the PiKoder/SSCe and the related programming software “PCC PiKoder Control Center”. Please check for updated information and new software releases on [www.pikoder.com](http://www.pikoder.com).

Hyperlinks were integrated into the text for convenience. You would also find all downloads referenced on the [PiKoder/SSCe webpage](#).

For your evaluation of the PiKoder/SSCe a demo version of the firmware is provided free of charge for downloading. Please note that the demo firmware has a limited feature set compared to the pre-programmed controller: Saving any parameters and daisy chaining is not supported.

Please share with me any comments, improvement ideas or errors you will find or encounter in working with your PiKoder/SSC. I can be reached at [webmaster@pikoder.com](mailto:webmaster@pikoder.com). Thank you very much!

---

# 2

---

## *Features*

This section will familiarize you with the feature set and a high-level overview of the intended use of the PiKoder/SSCe allowing you to customize the controller to your specific needs and requirements.

The PiKoder/SSCe Serial Servo Controller allows you to control up to eight servos or electronic speed controls from a serial port (either USB or UART depending on the protocol converter). The key features are:

- resolution 1  $\mu$ s with a precision of 0,5  $\mu$ s or better
- operating voltage range 3.3 V ... 5.0 V
- non-volatile memory for application specific parameters (not available in free demo version)
- miniSSC protocol supported as a quite common standard for RC servo controllers (only supported for channels 0 ...7 in free demo firmware)
- bi-directional ASCII protocol enabling line monitoring
- ICSP pins available for software upgrades
- Sample software and source code for PC application available

Alternatively, the servo outputs can be configured individually as binary outputs (switch function) allowing for a variety of additional special functions not requiring a pwm signal.

You can connect a controller board such as your Arduino or Raspberry Pi without any additional interface hardware directly to the PiKoder's UART (please refer to sections 7 and 8 for more details). With the addition of a simple off-the-shelf "USB to UART converter" you control the servos directly from your computer. In addition, the controller's wide range of oper-

ating voltage from 3.3 to 5.0V allows you to use an existing power source in your application rather than adding hardware.

In this capacity the SSCe would free up the Arduino or an Raspberry Pi of responding to real-time events such as generating pulses for servos in a given time frame and also free up resources such as internal timers and pwm generators ('Set and Forget'-function). Thereby intermittent problems due to internal collisions are avoided. And, finally: you can control up to eight servos consuming only two pins.

A free graphical and intuitive configuration and control program, the "PCC PRO" (PiKoder Control Center) is available for Windows 10, making it simple to test and program the controller over USB (please refer to section 5 for more details).

The PiKoder/SSCe supports two protocols:

- MiniSSC protocol representing a very common protocol for controlling SSCs and
- a two-way ASCII-Protocol designed to support controlling the SSC with standard terminal programs such as (but not limited to) Tera Term and TTY

The PiKoder/SSCe Serial Servo Controller would automatically detect the protocol used and no user interaction would be required.

With the full support of the miniSSC protocol, the PiKoder/SSCe would allow you to daisy chain controllers to control up to 255 servos as specified in the miniSSC protocol definition. The PCC provides for setting a miniSSC Offset and thereby defining the channel base address the PiKoder/SSC would respond to (not available in demo firmware). This means, that the PiKoder/SSC can control any contiguous block of eight servo numbers from 0 to 254. Please refer to the [\\_respective application note](#) for more details.

The PiKoder/SSCe also features non-volatile (EEPROM) data storage for retaining application specific operating parameters such as startup position after powering up, neutral position and upper as well as lower limits for servo pulse width (not available in demo firmware).

**Please note that the PiKoder/SSCe does not support any fail-safe function. This means that this PiKoder should not be deployed in receivers or other autonomous applications unless this feature is not needed. In all other applications, please consider upgrading your application to the PiKoder/SSC.**

---

# 3

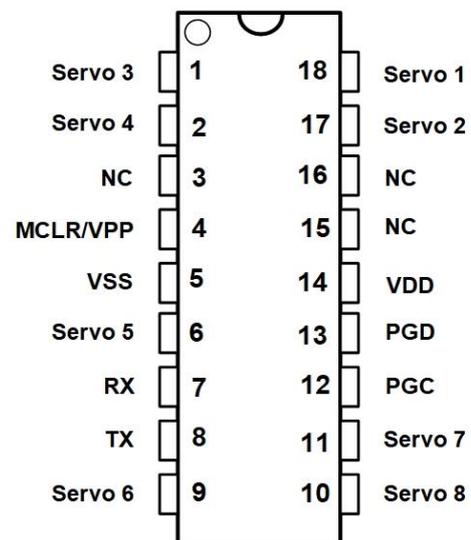
---

## *Pin Description and Packaging*

The PiKoder/SSC comes in an 18 pin DIP package (see below). The device operates from 3.3 – 5 Volts. Please refer to the PIC 16F628A data sheet from Microchip ([www.microchip.com](http://www.microchip.com)) for complete electrical and physical specifications.

A complete description of the pins is given on the following page. If a different package would be required for your application then please contact [sales@pikoder.com](mailto:sales@pikoder.com) for more information.

For evaluating the PiKoder/SSC a PCB as well as a complete kit is available at [www.pikoder.com](http://www.pikoder.com). This kit can make your development process more efficient and provides for modularity in your design.



## Description of pins

Pin	Symbol	Description
1	Servo 3	Output pin with PWM signal for servo channel 3
2	Servo 4	Output pin with PWM signal for servo channel 4
3	NC	Not connected (reserved for later use)
4	MCLR/VPP	Reset pin, active low. Connects directly to Vss for automatic reset at power up.
5	VSS	Supply voltage. Connect to 3,3 – 5 V DC
6	Servo 5	Output pin with PWM signal for servo channel 5
7	RX	Serial receive input
8	TX	Serial transmit output
9	Servo 6	Output pin with PWM signal for servo channel 6
10	Servo 8	Output pin with PWM signal for servo channel 8
11	Servo 7	Output pin with PWM signal for servo channel 7
12	PGC	Clock pin for In-Circuit-Serial-Programming
13	PGD	Data pin for In-Circuit-Serial-Programming
14	VDD	Ground connection
15	NC	Not connected (reserved for later use)
16	NC	Not connected (reserved for later use)
17	Servo 2	Output pin with PWM signal for servo channel 2
18	Servo 1	Output pin with PWM signal for servo channel 1

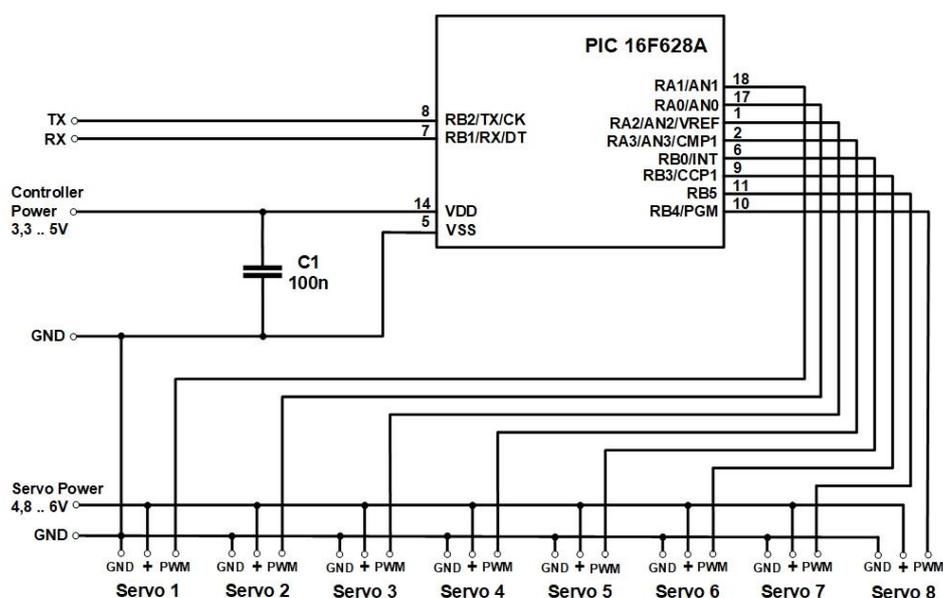
---

# 4

---

## *Standard application*

The following schematic shows the standard application of the PiKoder/SSCe.



Given the simplicity of the schematic, the PiKoder/SSCe can easily be evaluated on a prototype board. If you are looking for a more permanent prototype then please consider the evaluation board which is available as a kit on [www.pikoder.com](http://www.pikoder.com).

- Room for notes -

---

# 5

---

## ***The PiKoder Control Center PCC***

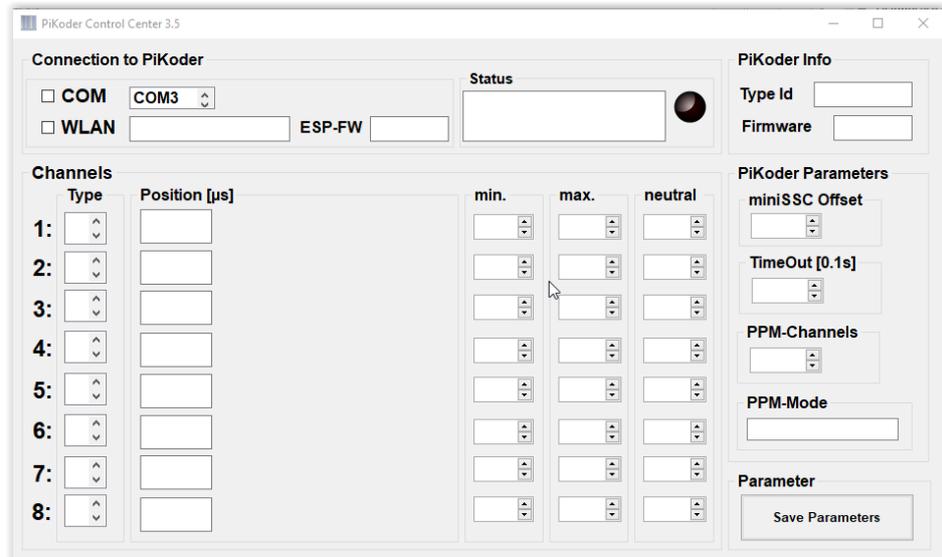
The PiKoder/SSCe's serial interface provides access to configuration options as well as support for real time control. The PCC (PiKoder Control Center) is a graphical tool that makes it easy for you to use this interface. For almost any project you will start by using the PCC to set up and test your PiKoder/SSCe. This section explains the features of the PiKoder Control Center.

### **Getting started**

The hardware setup for the interface is simple and straight forward: You would connect your PiKoder/SSCe with the USB port of your PC using a suitable USB adaptor and cable. This cable will provide also for the power supply of the PiKoder/SSCe.

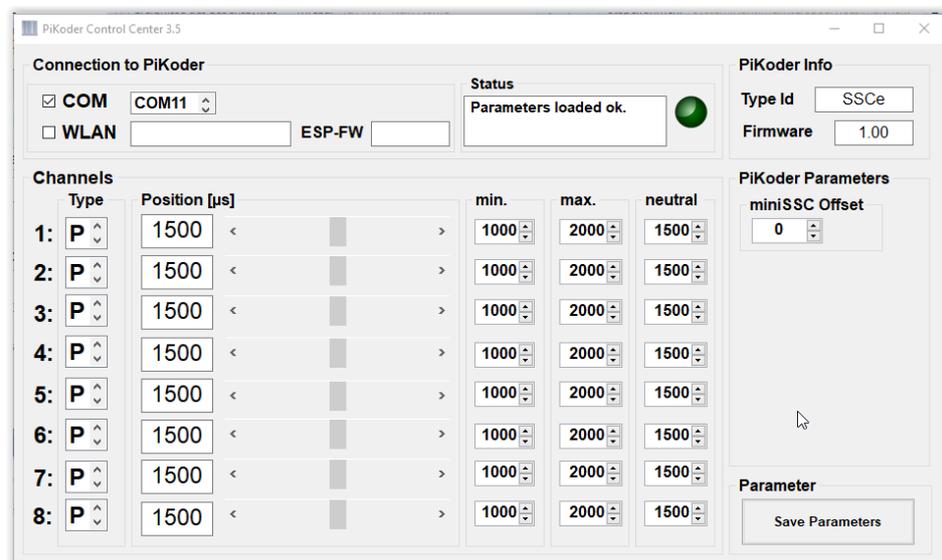
It is highly recommended that you download the latest version of the PCC software to enjoy the complete feature set of your PiKoder/SSC. The PiKoder Control Center software is Open Source and released under an Apache License, Version 2.0. You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>. The source code is made available through a [github repository](#); the [executable](#) can be downloaded on the PiKoder webpage.

Please unzip the downloaded file and start the program – there is no installation required.



The PiKoder Control Center application would show COM3 to be the first available port in your computer. Select the COM port your PiKoder/SSCe is connected to (in this example COM11, see below) and then click on the box to indicate your selection and communication to the controller will be established.

The LED would light up in red color indicating that the data upload started and change to green once all data were received correctly. The firmware version of your SSC would be displayed in the respective field.



You would now have full control of your SSC: either for real-time control by the sliders or for changing the settings and save the new parameters.

## Type

The Type field indicates whether the channel output would be pwm for controlling a servo (value="P") or binary switch (value="S"). The value of a switch channel will be logical 1 (voltage depending on actual Vcc) for a servo position larger than 1800  $\mu$ s. Any value below this threshold will translate into a 0 value at the respective output. The type of a channel is changed by selecting the desired type through the channel's combo box and then click on the box which will send the respective parameter to the PiKoder.

## Position

The sliders are used for controlling the PiKoder's outputs and the respective numerical fields monitor the status in real time displaying the current channel value in  $\mu$ s. A separate row of controls is displayed for each of servo position the PiKoder/SSCe's channels.

The key parameters for each channel such as min. and max. pulse width and neutral position can be set individually in the respective row. The SSC Control Center will limit the slider value to the min. and max. value shown. This feature however is implemented in the Control Center. The PiKoder/SSC itself does not perform a parameter check and would therefore accept channel values outside of the shown boundaries.

## miniSSC Offset (not available in free version)

This field is used in combination with the miniSSC protocol to determine the base offset for the actual controller. Please refer to "AN02 Daisy Chaining" for more information.

## Save Parameters

All changes made while using the PCC will not be permanent unless you select to save the parameters. Hitting this button transfers all settings into the non-volatile memory of the PiKoder/SSCe to be retrieved when started up the next time.

Please note that saving new values may take some time and requires disabling some internal interrupt logic. This may result in erroneous servo behavior.

- Room for notes -

---

# 6

---

## ***Serial Interface***

The PiKoder/SSCe's serial interface is based on TX and RX lines, which allow the controller to send and receive non-inverted, TTL (0 – 5V) serial bytes. The parameters for the serial transmission are 9600 Baud, 8 data bits, one stop bit, with no parity.

The bytes sent to the PiKoder/SSCe are commands which allow you to control the program and control the SSCe. The PiKoder/SSCe supports two protocols:

- MiniSSC protocol representing a very common protocol for controlling SSCs
- a two-way ASCII-Protocol designed to support controlling the SSCe with standard terminal programs such as (but not limited to) Tera Term, Putty, hyperterm

The PiKoder/SSCe Serial Servo Controller does automatically detect the terminal protocol; no user interaction would be required.

### **Mini SSC Protocol (limited support in free version)**

This protocol allows you to control up to 255 different servos by chaining multiple servo controllers together. It only takes three serial bytes to set the target of one servo, so this protocol is good if need to send many commands rapidly. The Mini SSC protocol is to transmit 0xFF as the first (command) byte, followed by a servo number byte, and then the 8-bit servo target byte for the servo position.

A servo target byte of 127 (0x7F) will always indicate the neutral position maintained as a PiKoder/SSCe parameter.

The actual position taken by sending a servo target byte to the controller is calculated depending on the actual parameters. If you wanted to move the servo from neutral towards the upper limit, then the increment in pulse length  $du$  is calculated based on:

$$du = (\text{Upper Limit} - \text{Neutral}) / 127 \text{ Steps}$$

This means for example, that with an upper limit of 2008  $\mu\text{s}$  and a neutral position of 1500  $\mu\text{s}$  an increment of one in the servo target byte would result in:

$$du = (2008 - 1500) \mu\text{s} / 127 \text{ Steps} = 4 \mu\text{s}/\text{Step}$$

Therefore, a servo target byte of 0x80, which is an increment of one to the neutral position would result in a pulse length of  $1500 \mu\text{s} + 4 \mu\text{s} = 1504 \mu\text{s}$ .

The same procedure is applied when you wanted to move the servo from neutral towards the lower limit; then the decrement in pulse length  $dl$  is calculated based on:

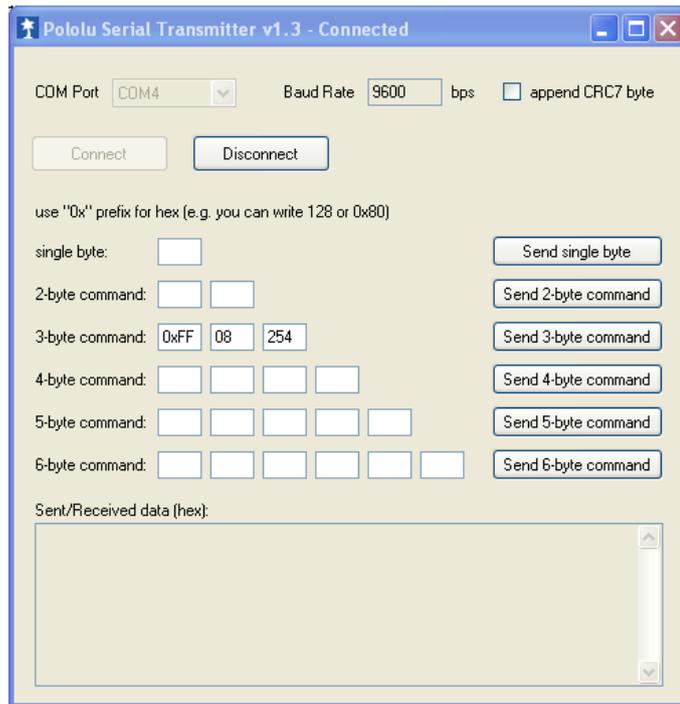
$$dl = (\text{Neutral} - \text{Lower Limit}) / 127 \text{ Steps}$$

Please note the following with respect to  $du$  and  $dl$ :

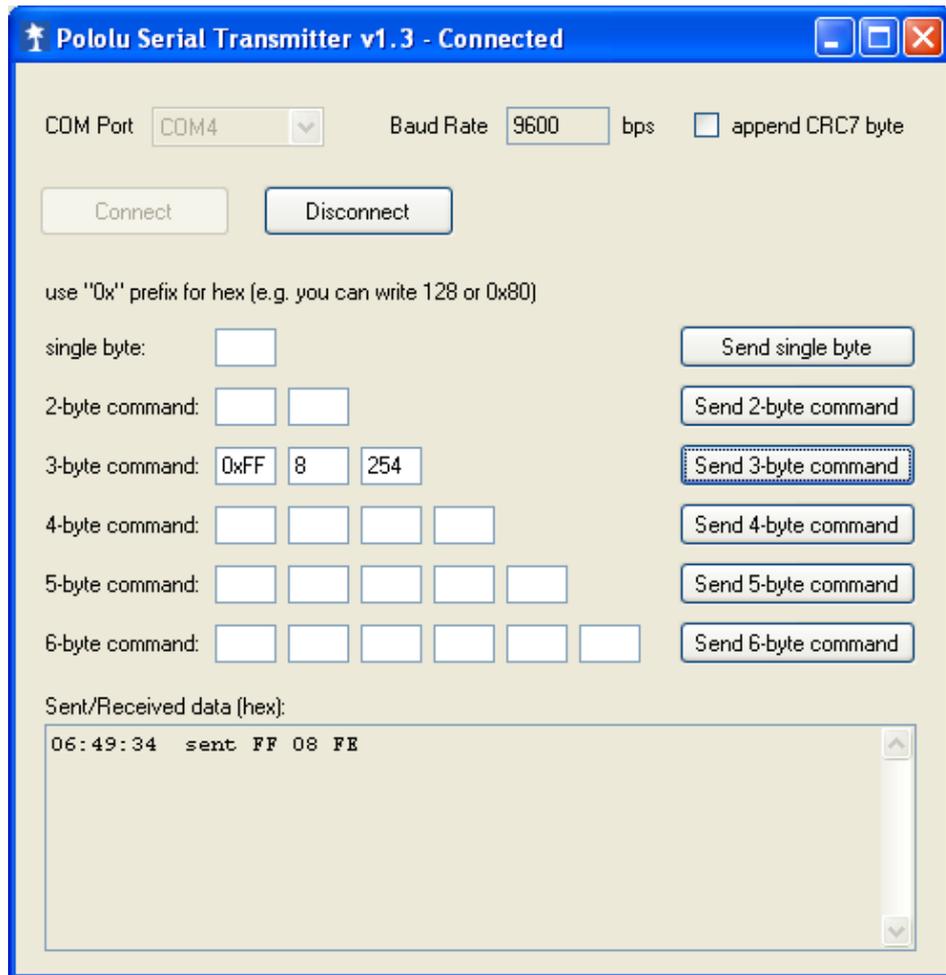
- $du$  and  $dl$  are calculated per channel and may differ per channel allowing for asymmetric ranges and a neutral position outside of the mechanical central position of the servo
- $du$  and  $dl$  are calculated during controller startup. With firmware release 2.3 these parameters are updated when you change the values of neutral, upper or lower limit without resetting the controller.
- $du$  and  $dl$  are calculated based on 8 bit arithmetic. Depending on the numbers a slight overrun over the upper limit resp. a slight underrun of the lower limit may occur.

For testing the miniSSC protocol you may want to use a byte oriented tool such as the “Pololu Serial Transmitter” utility for Windows (see <http://www.pololu.com/docs/0j23> for more details).

After installing and starting the software you would have to connect to the PiKoder/SSC by selecting the COM port and pushing the connect button. The miniSSC-protocol is a three-byte command and should be entered in the respective column as shown below.



Once you hit the “Send 3-byte command”-button the bytes are sent to the PiKoder (see below); as per protocol definition there is no response by the SSCe.



## ASCII Command Interface

The ASCII Command Interface (ACI) is probably the most versatile way to program the PiKoder/SSCe without any specific host software such as the PCC. All commands are simple ASCII and are sent using a Windows based terminal program such as Hyperterm or Tera Term. The commands can be typed in right away and the response of the controller is readable without referring to any specific code tables. Please note that neither 'CR' nor 'LF' is needed to send the command to the controller.

There are two basic types of commands: commands for querying parameters and for setting parameters.

If a parameter is read the PiKoder/SSCe will provide for proper formatting by sending a "CRLF" prior to sending the parameter value and support readability by sending another "CRLF" after the parameter value.

If a parameter is set the PiKoder/SSCe will acknowledge the proper execution by sending an "!" framed by "CRLF".

If a command could not be interpreted at all then a question mark '?' framed by 'CR' 'LF' would be echoed. Please note that protocol syntax checking is currently limited.

The following ACI commands are available:

- '?': query the PiKoder type information; the PiKoder/SSC will respond with "T=SSC".
- 'i?': query the current pulse width for channel i (i = 1..8); PiKoder will respond 'CR' 'LF' 'xxxx' 'CR' 'LF' with xxxx representing the pulse width in  $\mu\text{s}$
- 'i=xxxx': set the pulse width for channel i to xxxx  $\mu\text{s}$  (xxxx in decimal format, i = 1..8); PiKoder will acknowledge execution of the program by sending an 'CR' 'LF' '!' 'CR' 'LF'
- '0': query the firmware version; PiKoder will respond in a format 'n.nn' framed by 'CR' 'LF'
- 'Ui?': query the upper limit for pulse width for channel i; PiKoder will respond 'xxxx' with xxxx representing the pulse width in  $\mu\text{s}$  (xxxx in decimal format, i = 1..8) - the command is not case sensitive
- 'Ui=xxxx': set the upper limit for pulse width for channel i to xxxx  $\mu\text{s}$  (xxxx in decimal format, i = 1..8); PiKoder will acknowledge execution with a '!' framed by 'CR' 'LF'.
- 'Li?': query the lower limit for pulse width for channel i; PiKoder will respond 'xxxx' with xxxx representing the pulse width in  $\mu\text{s}$  (xxxx in decimal format, i = 1..8) - the command is not case sensitive
- 'Li=xxxx': set the lower limit for pulse width for channel i to xxxx  $\mu\text{s}$  (xxxx in decimal format, i = 1..8); PiKoder will acknowledge execution with a '!' framed by 'CR' 'LF'.
- 'Ni?': query the pulse width for the neutral position for channel i; PiKoder will respond 'xxxx' with xxxx representing the pulse width in  $\mu\text{s}$  (xxxx in decimal format, i = 1..8) - the command is not case sensitive
- 'Ni=xxxx': set the pulse width for the neutral position for channel i to xxxx  $\mu\text{s}$  (xxxx in decimal format, i = 1..8); - the command is not case sensitive and the PiKoder will acknowledge execution with a '!' framed by 'CR' 'LF'.
- 'M?' or 'm?': query the current channel offset for the miniSSC-protocol. – **not available in free mode.**
- 'M=iii' or 'm=iii': set the channel offset for the miniSSC-protocol – **not available in free mode.**
- 'Oi?': query the type of output / channel i; PiKoder will respond 'X' with X representing 'P' for pwm and 'S' for switch. The command is not case sensitive.
- 'Oi=X': set the type of output / channel i with 'P' or 'p' setting the output to pwm and 'S' or 's' setting the output to switch type. The

binary output value will depend of the channel pulse length (any value larger than 1800  $\mu$ s will translate into a logical 1). PiKoder will acknowledge execution with a '!' framed by 'CR' 'LF'. The command is not case sensitive.

- 'Ci?': query all parameters for channel i. The Output would be: nnnnlllluuuu with nnnn representing neutral, llll representing lower limit and uuuu representing the upper limit. The command is not case sensitive.
  
- SUJU], sUJU]: will save the current parameters to the controller's EEPROM making the current servo positions the start up positions after powering up; returns a '!' upon successful completion framed by 'CR' 'LF' – Please note that saving new values may take some time and requires disabling some internal interrupt logic. This may result in erroneous servo behavior. – ***not available in free mode.***
  
-

---

# 7

---

## ***Connect the SSCe to a Raspberry Pi***

The PiKoder/SSC releases your Raspberry Pi from generating real-time pulses for controlling servos ('Set and forget'-function).

This is advantageous when using elaborate operating system such as LINUX, because their real-time capabilities are limited due to the number of concurrent tasks. This might limit the precision of the signals generated.

The PiKoder/SSC connects to your Raspberry's UART. This frees up valuable I/O pins which can be used for additional peripherals in your application, since only two pins are needed for controlling eight servos (or in a daisy chain configuration even up to 255 servos).

In this application the PiKoder/SSC is operating with 3.3 Volts, which is supplied by your Raspberry Pi. The servo power supply of 4.8 ... 6 Volts must be kept separate in order to avoid unstable power potentially creating reboots.

The schematic is shown on the next page.

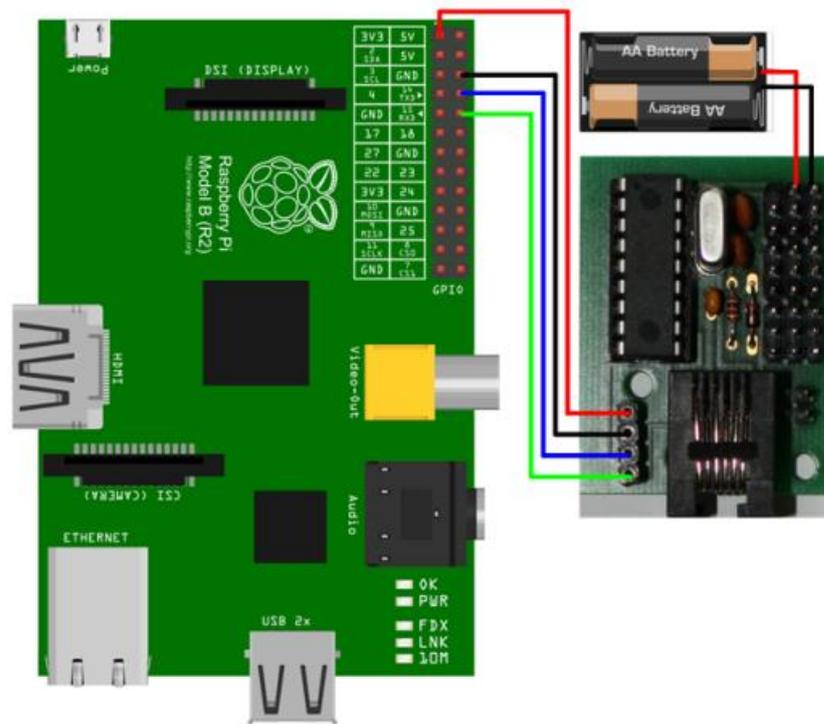
### **Software configuration and hardware setup**

The software configuration and the hardware setup require three simple steps:

- Turning of the UART as a serial console
- Wiring
- Installing terminal software such as Minicom

## Turning of the UART as a serial console

In your Raspberry Pi's default configuration, the UART is used as a serial console. This function must be deactivated in order to use this port to control the PiKoder/SSC with a terminal software. The process of turning of the serial console is described in various blogs and may vary slightly depending on your distribution.



Wiring scheme for connecting your PiKoder/SSCe to a Raspberry Pi

## Wiring

The wiring is shown in the schematic at the top of this page. Please turn of all components to avoid damage by unintended shorts.

## Installing terminal software such as Minicom

You will also find more information about the installation of Minicom in various Raspberry Pi related blogs. When starting the software please make sure to set the baud rate of the UART to 9600 Baud required by the PiKoder. For your convenience it is recommended that you activate the local echo (command "E"). After setting these parameters you can control your PiKoder/SSCe using the ASCII commands listed in section 6 of this User's Guide.

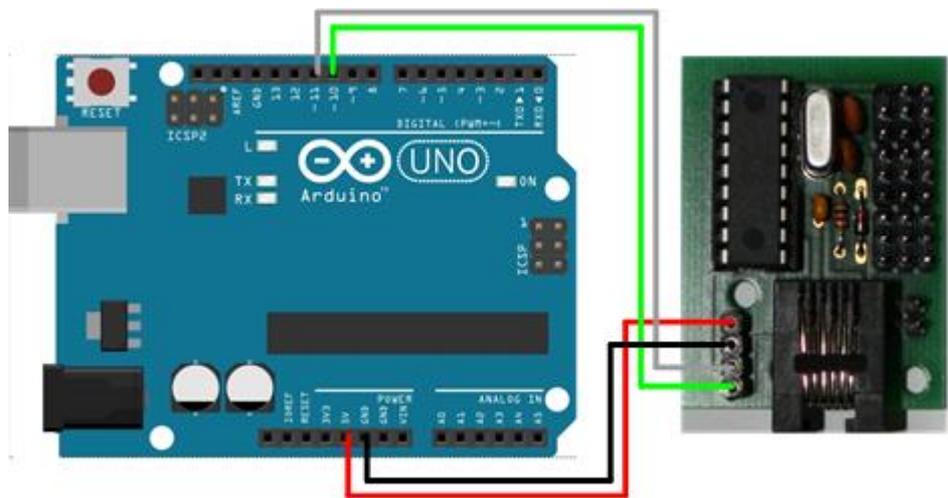
---

# 8

---

## Connect the SSCe to an Arduino

Connecting an PiKoder/SSCe to an Arduino releases the Arduino from calling the servo library at least every 20ms ('Set and Forget-function') which is required to refresh the servo signals. Additionally, the PiKoder/SSCe releases valuable digital pins because up to eight servos can be controlled by only two signals. The schematic is presented below. If a connection monitoring would not be required, the resource requirement can be reduced to just one digital pin for eight servos. In this instance the tx-line of the PiKoder/SSCe is not connected (green cable).



### Interface Software

#### ASCII-based interface

The following sketch would allow you to test the servo shield by entering PiKoder/SSCe commands (please refer to section 6 for a full interface description). The [Sketch Interface Test \(.ino-Datei\)](#) can be downloaded here. This sketch is Open Source and is released under a [GNU General Public License Version 3](#).

