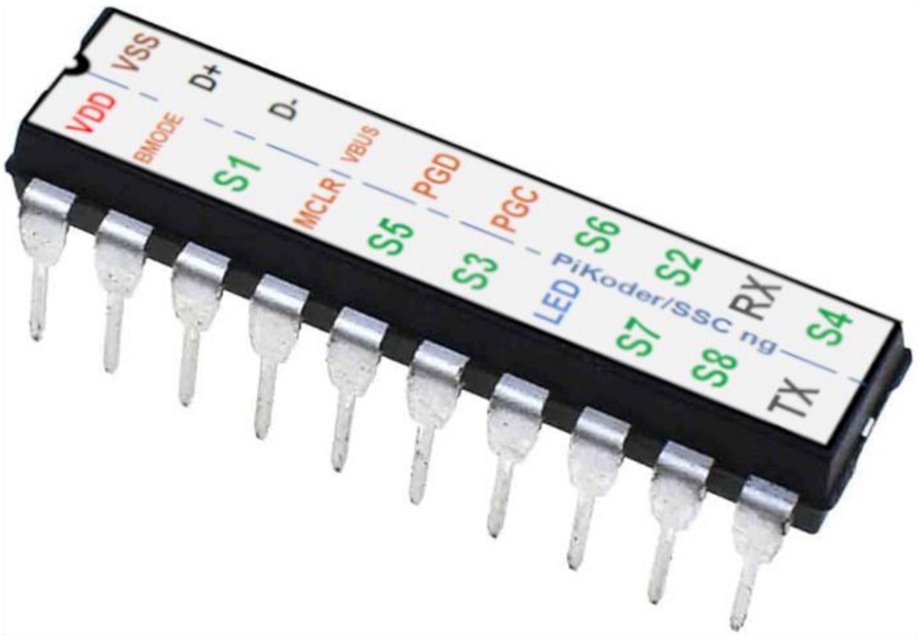


PiKoder/SSCng
User Manual

Version 1.0
dated 10.24

Gregor Schlechtriem
webmaster@pikoder.com

www.pikoder.com

Content

1

Overview

The PiKoder/SSCng 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 µs through a serial interface. Alternatively, the servo outputs can be configured as binary outputs (switch function). As a "next generation" (ng) controller, the PiKoder/SSCng features a USB input for programming and operating. This User Manual covers the features, the programming and the serial interface of the PiKoder/SSCng.

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/SSCng.

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/SSCng is setup in line with this reference schematic as the respective engineering board is also.**

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

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

This User's Guide is based on the most recent hard- and firmware version 1.0 available for the PiKoder/SSCng and the related programming software

"PCCng PiKoder Control Center". Please check for updated information and new software releases on www.pikoder.com.

Hyperlinks were integrated into the text for convenience. You would also find all downloads referenced on the PiKoder/SSCng webpage.

Please share with me any comments, improvement ideas or errors you will find or encounter in working with your PiKoder/SSCng. I can be reached at 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/SSCng allowing you to customize the controller to your specific needs and requirements.

The PiKoder/SSCng 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 boot mode). 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.

The key features are:

- resolution 1 µs with a precision of 0,5 µs or better
- operating voltage range 3.3-5.0 V
- non-volatile memory for application specific parameters
- bi-directional ASCII protocol enabling line monitoring
- Baud rate can be selected
- optional failsafe position when connection to host is lost
- native USB port (no USB-to-UART adaptor needed)
- native UART port to interface with Arduino or Raspberry Pi
- ICSP pins available for software upgrades
- Sample software and source code for PC application available

The PiKoder/SSCng supports two interfaces (UART and USB). Upon reset the PiKoder/SSCng would boot in USB-mode when power would be provided via the USB port and expect communication to run via USB. Otherwise, the PiKoder/SSCng would operate in UART mode.

Both interfaces vary slightly regarding the commands they would feature. For example, would the baud rate selection only be available when the controller operates in USB mode. Please refer to section 6 for more details.

The PiKoder/SSCng supports two protocols:

- 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
- a data-based protocol for high-speed applications

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

The PiKoder/SSC does support a failsafe position for the use in autonomous and RC applications. You can activate a timer for 0.1 s to 99.9 s to monitor the communication. If no message is received within this preset time frame, then the PiKoder/SSC would set all servo outputs to the failsafe position.

If you were to use this function you might want to implement a regular "ping" in your application to make sure that the failsafe function is not triggered by a period of inactivity.

The PiKoder/SSC also has non-volatile (EEPROM) data storage for retaining application specific operating parameters such as startup position after powering up and fail-safe values.

With the simple addition of a USB cable, you control the servo outputs directly from your computer. In addition, the controller's wide range of operating voltage from 3.3 to 5.0V allows you to use an existing power source in your application rather than adding hardware.

You could connect a controller board such as your Arduino or Raspberry Pi without any additional interface hardware directly to the PiKoder's UART. In this application the SSCng would free up the Arduino or a 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 (please refer to sections 7 and 8 for more details).

A free graphical and intuitive configuration and control program, the "PCC 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).

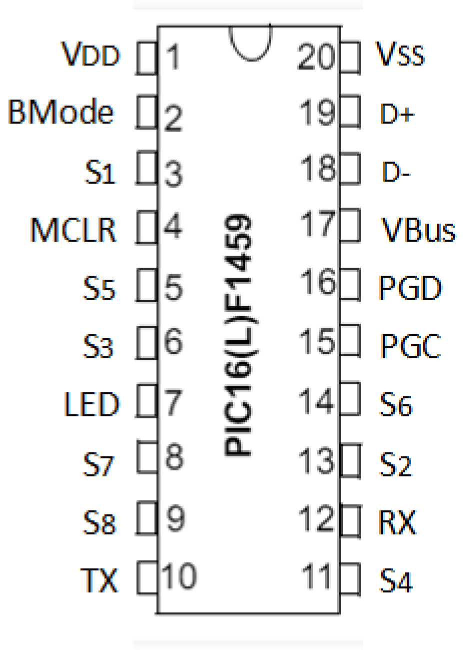
3

Pin Description and Packaging

The PiKoder/SSCng comes in a 20 pin DIP package (see below). The device operates from 3.3 – 5 Volts. Please refer to the PIC 16F1459 data sheet from Microchip (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 for more information.

For a detailed evaluation of the PiKoder/SSCng an engineering board kit is available at www.pikoder.com. This kit can make your development process more efficient and provides for modularity in your design.



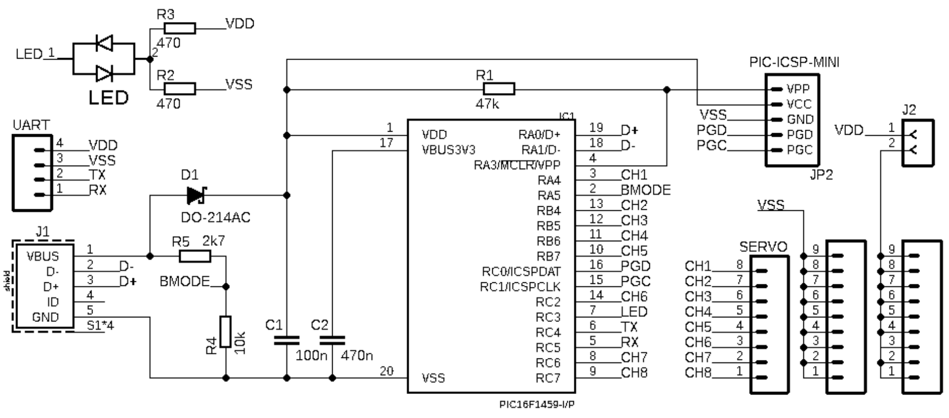
Description of pins

Pin	Symbol	Description
1	VDD	Ground connection
2	BMode	Bootmode – if this pin would be high during reset, then the controller will boot in USB mode
3	S1	Channel 1 output pin
4	MCLR/VPP	Reset pin, active low. Connects directly to Vss for automatic reset at power up.
5	S5	Channel 5 output pin
6	S3	Channel 3 output pin
7	LED	USB connection indicator (please refer to section 4 for more information)
8	S7	Channel 7 output pin
9	S8	Channel 8 output pin
10	TX	Serial transmit output
11	S4	Channel 4 output pin
12	RX	Serial receive input
13	S2	Channel 2 output pin
14	S6	Channel 6 output pin
15	PGC	Clock pin for In-Circuit-Serial-Programming
16	PGD	Data pin for In-Circuit-Serial-Programming
17	VBus	3.3 Output in USB mode -please refer to the controller data sheet formore information (470n capacitor required for operation)
18	D-	USB bus data line D-
19	D+	USB bus data line D+
20	VSS	Supply voltage. Connect to 3,3 – 5 V DC

4

Standard application

The following schematic shows the standard application of the PiKoder/SSCng. Please note that the bi-color LED indicates the USB status. The LED is optional and will remain red in UART mode.



LED status indication (in USB mode)

This following table shows the status of the USB connection ("DEVICE STATE") based on the LED signaling state ("LED RESPONSE").

DEVICE STATE	LED RESPONSE
DETACHED	OFF
ATTACHED	RED
POWERED	RED
DEFAULT	RED & GREEN
ADRESS	RED & GREEN
CONFIGURED	GREEN

Given the simplicity of the schematic, the PiKoder/SSCng 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.**

5

The PiKoder Control Center

The PiKoder/SSCng's USB interface provides access to configuration options as well as support for real time control. The *PCCng PiKoder Control Center* is a Windows 10 based graphical tool that makes it easy for you to use this interface. For almost any project you will start by using the *PCCng PiKoder Contol Center* to set up and test your PiKoder. The app is distributed free of charge through the Microsoft app store. This section explains the features of the *PCCng PiKoder Control Center*.

Getting started

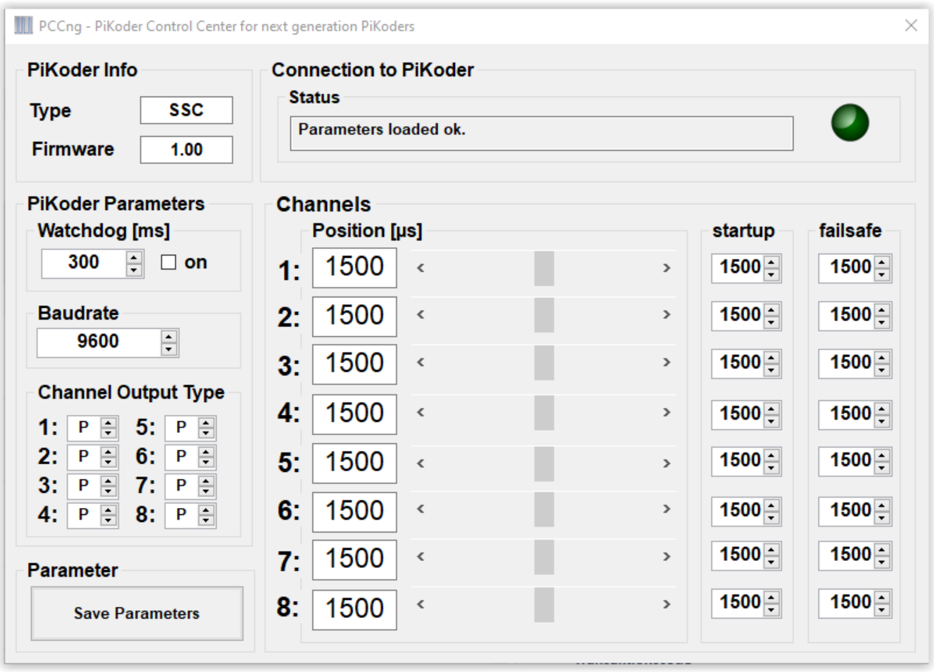
The hardware setup is simple and straight forward with a PiKoder/SSCng engineering board or your own prototype. You must connect your PiKoder/SSCng with the USB port of your Windows PC using a suitable cable. This cable will provide also for the power supply of the board while programming. **Please make sure that you remove jumper 2 because a USB port would normally not provide sufficient power to drive servos.**

If you want to perform testing with servos in USB mode, then provide an independent power source for the servos connected to the respective servo power input. Please refer to the pinning information for the engineering board provided in the construction manual for more information.

When you connect your PiKoder/SSCng for the first time, Windows will automatically install the drivers needed. On the engineering board the LED would be blinking red in fast mode during the installation process. Please refer to section 4 for more information regarding the LED indicator.

Please make sure to connect your PiKoder to a USB port before starting the PCCng as the PCC will scan your computer's USB ports to automatically connect to the PiKoder.

The connection is made and the actual parameters of the PiKoder are displayed.



Type: SSC
Firmware: 1.00
Status: Parameters loaded ok.
Watchdog [ms]: 300 ☐ on
Baudrate: 9600

Channel Output Type

1:	P	5:	P
2:	P	6:	P
3:	P	7:	P
4:	P	8:	P

	Position [µs]	startup	failsafe
1:	1500	1500	1500
2:	1500	1500	1500
3:	1500	1500	1500
4:	1500	1500	1500
5:	1500	1500	1500
6:	1500	1500	1500
7:	1500	1500	1500
8:	1500	1500	1500

You would now have full control of your PiKoder/SSCng: either for real-time control by the sliders or for changing the settings with respect to output types.

Real-time control

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 µs within a range of 1000 – 2000 µs. A separate row of controls is displayed for each of the active channels.

Watchdog

The watchdog would be used to make sure that the failsafe values for all channels are set when the connection to the host would be interrupted. Activating the watchdog is comprised of setting the value (default is 300 ms) and activating the watchdog itself by checking the respective box.

UART baudrate

The PiKoder/SSCng offers baudrates of 9600, 19200, 38400, 57600 and 115200. The setting would become effective after the next reset of the PiKoder.

Channel Output Type

The Channel Output Type fields indicate 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 µs. 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.

Startup

In many applications the pulse width for the neutral value would be 1.5 ms. However, some ESCs (Electronic Speed Controllers) for drones might need 1.0 ms as a start-up value to make sure that they are not spinning immediately after turning the PiKoder/SSCng on. To address these applications, you would set the initial value and safe the parameters by hitting the respective button on the lower right of the screen.

Save Parameters

Changes made while using the *PCC G2* 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/SPE 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.

6

Serial Interface

The PiKoder/SSCng supports a two-way ASCII-Protocol named Ascii Command Interface (ACI) designed to support controlling PiKoders with standard terminal programs such as (but not limited to) Tera Term.

ASCII Command Interface

The ASCII Command Interface (ACI) is probably the most versatile way to program any PiKoder without any specific host software such as the *PCCng PiKoder Control Center*. All commands are simple ASCII and are sent using a Windows based terminal program such as 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 getting parameter values and commands for setting parameters values. The commands are not case sensitive unless noted.

If a parameter is read the PiKoder 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 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 extremely limited now.

The following ACI commands are available:

- '?': query the PiKoder type information; PiKoder/SSCng will respond in a format 'SSCng' framed by 'CR' 'LF'

- '0': query the firmware version; PiKoders will respond in a format 'n.nn' framed by 'CR' 'LF'
- 'i?': query the current pulse width in µs for channel i (i = 1..8)
- 'i=xxxx': set the pulse width for channel i to xxxx µs (xxxx in decimal format, i = 1..8)
- 'B?': query the current UART baud rate. PiKoder/SSC will respond with the current Baudrate 9600, 19200, 38400, 57600 or 115200.
- 'B=k': set the UART baud rate with k = [0..4] translating into: 0 = 9600, 1 = 19200, 2 = 38400, 3 = 57600 and 4 = 115200
- 'Fk?': get the fail-safe value for channel k. PiKoders will respond 'xxxx' with xxxx representing the pulse width in µs (xxxx in decimal format, k = 1..8)
- 'Fk=xxxx': set the startup value for channel k to xxxx µs (xxxx in decimal format, k = 1..8)
- 'Nk?': query the startup value for channel k in µs (k = 1..8)
- 'Nk=xxxx': set the startup value for channel k to xxxx µs (xxxx in decimal format, k = 1..8); the PiKoders will acknowledge execution with a '!' framed by 'CR' 'LF'.
- SU]U], sU]U]: will save the current parameters to the controller's non-volatile memory making the current servo positions the start up positions after powering up; returns a '!' upon successful completion framed by 'CR' 'LF'
- 'T?': query the current watchdog value. PiKoders will respond 'xxx' with xxx representing the time in increments of 20 ms (frames missed) - the command is not case sensitive.
- 'T=xxx': set the timeout value in increments of 20 ms; - the command is not case sensitive and the PiKoder/SPE will acknowledge execution with a '!' framed by 'CR' 'LF'.
- 'W?': query the current watchdog status. The PiKoder/SSC will respond either with '1' indicating an active watchdog or '0' for watchdog not activated. - the command is not case sensitive.
- 'W=b': activate / deactivate watchdog. For b=1 the watchdog would be activated, for b=0 the watchdog would be turned off; - the command is not case sensitive and the PiKoder/SPE will acknowledge execution with a '!' framed by 'CR' 'LF'.
- '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 µs will translate into a logical 1). PiKoder will acknowledge execution with a '!' framed by 'CR' 'LF'. The command is not case sensitive.

Data protocol

The PiKoder/SSCng also supports a data-based protocol for applications with a high-performance need. The protocol is simple, and it only takes four serial bytes to set the target position of one servo.

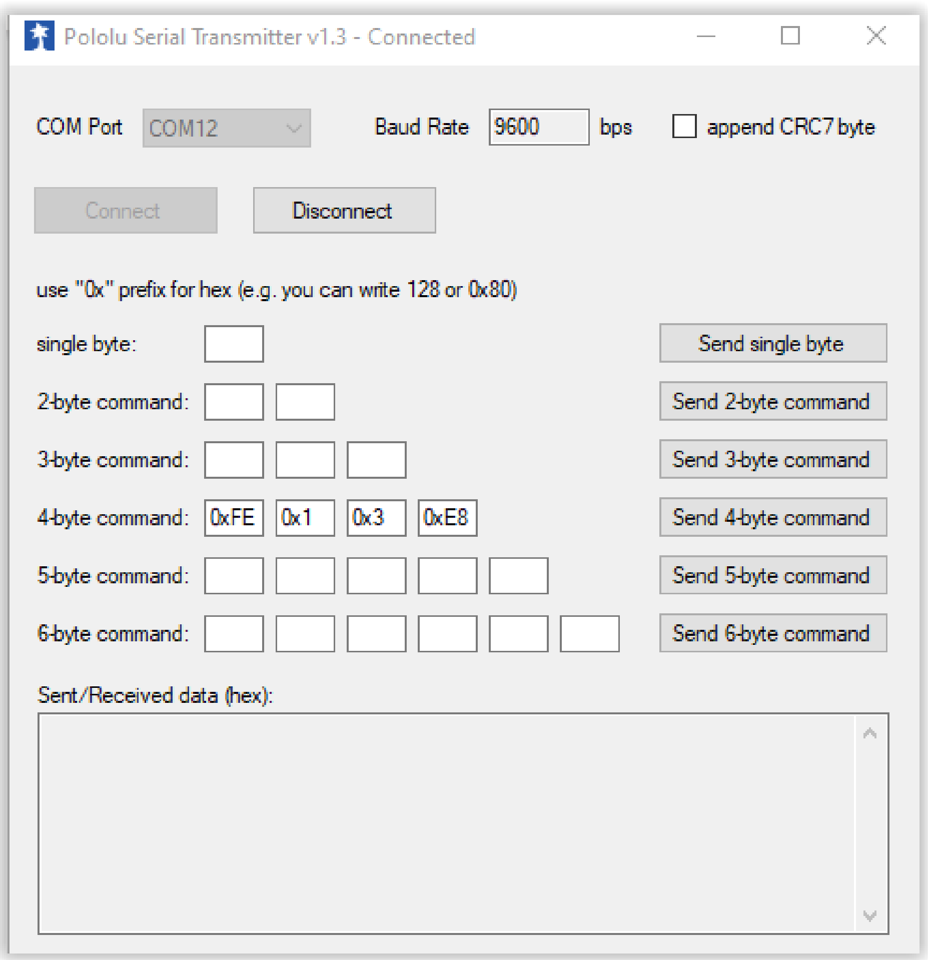
The data-based protocol would transmit 0xFE as the first (command) byte, followed by a servo number byte (1 .. 8), and then the 16-bit servo target bytes (higher byte first) for the servo position in µs. This means that the protocol covers the full range of 1000 – 2000 µs with a resolution of 1 µs allowing for 1000 positions.

For testing the data-based protocol you may want to use a byte oriented tool such as the "Pololu Serial Transmitter" utility for Windows (see https://www.pololu.com/docs/0J23 for more details).

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

You would start of with the header byte (0xFE), followed by a channel number arbitrarily chosen to be 1 in this example. A channel value of e.g. 1000 µs would be in hex 0x3E8. The higher byte 0x3 goes first and then the lower byte 0xE8 (see next page).

Once you hit the "Send 4-byte command"-button the bytes are sent to the PiKoder; as per protocol definition there is no response by the SSC.



COM Port COM12

Baud Rate 9600 bps

append CRC7 byte

use "0x" prefix for hex (e.g. you can write 128 or 0x80)

single byte:

2-byte command:

3-byte command:

4-byte command: 0xFE 0x1 0x3 0xE8

5-byte command:

6-byte command:

Sent/Received data (hex):

Please note that the miniSSC protocol is not supported anymore.

7

Connect SSCng to a Raspberry Pi

The PiKoder/SSCng 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/SSCng 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 to avoid unstable power potentially creating reboots.

The schematic is shown on the next page.

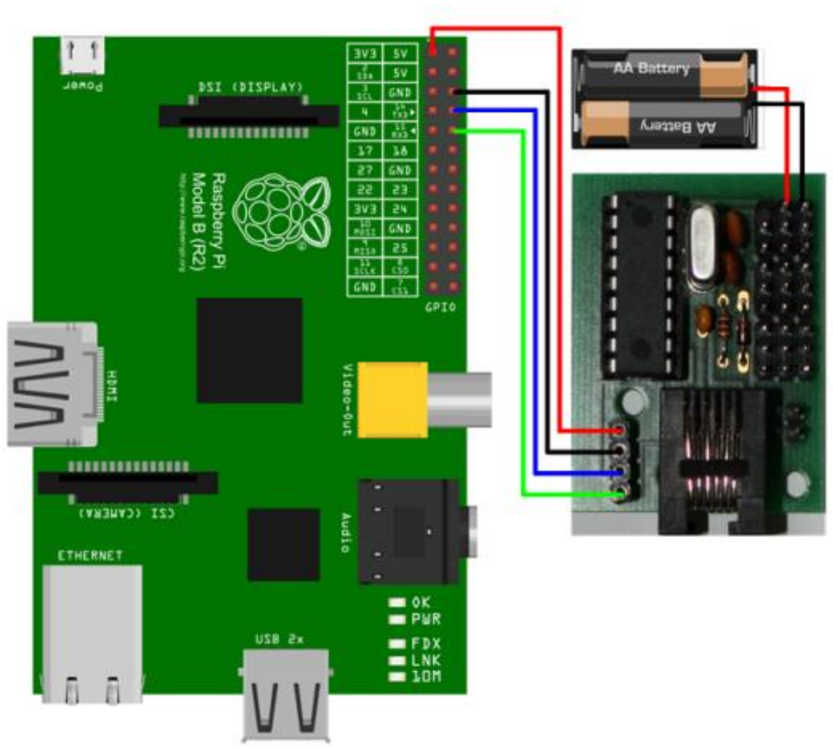
Software configuration and hardware setup

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 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/SSC 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.

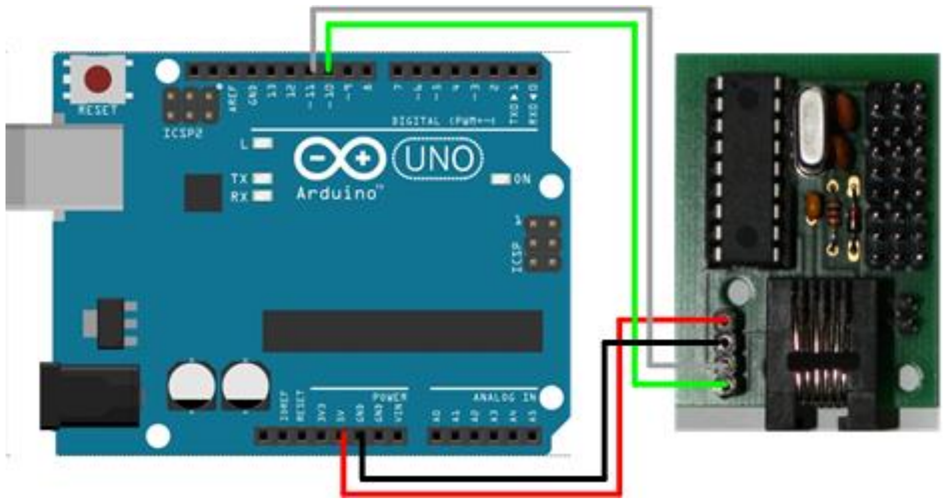
Installing terminal software such as Minicom

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/SSCng using the ASCII commands listed in section 6 of this User's Guide.

8

Connect the SSCng to an Arduino

The PiKoder/SSCng 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/SSCng releases valuable digital pins because up two 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 on digital pin for eight servos. In this instance the tx-line of the PiKoder/SSC is not connected (green cable).



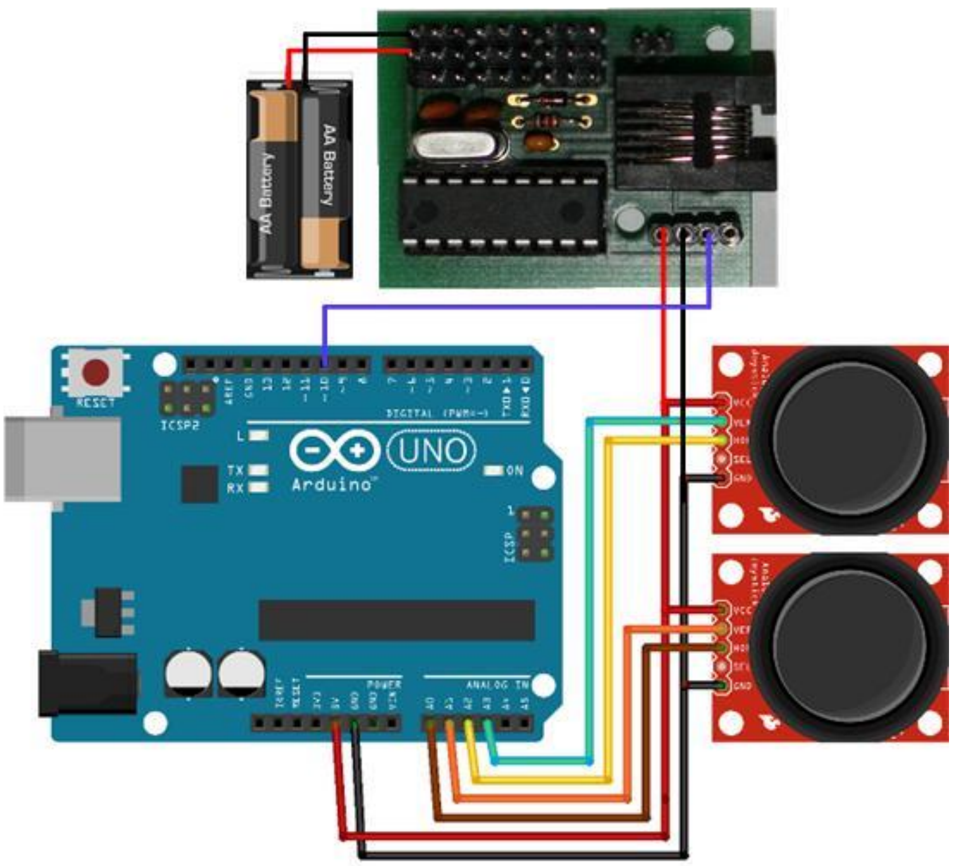
Interface Software

ASCII-based interface

The following sketch would allow you to test the servo shield by entering PiKoder/SSCng 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.

Servo control by thumb joysticks

An interesting project for the combination of an Arduino and a PiKoder/SSCng would be a remote control of four servos by wire. Two thumb joysticks would be evaluated by the Arduino through analog inputs. The joystick position is translated into the respective servo position and communicated to the PiKoder/SSCng using the miniSSCII protocol. The complete schematic is shown below.



The Arduino Sketch is straight forward. The program would be reading each analogue input and compare the value acquired with the previous position. Would the values deviate from the previous reading then the Arduino would send the new position to the PiKoder/SSCng. The Sketch Four Channel Encoder for SSCng (.ino-Datei) can be downloaded here.

This sketch is Open Source and is released under a GNU General Public License Version 3.