PiKoder/SSC PRO

User's Guide

Version 1. 2
dated 04/25/20

Gregor Schlechtriem

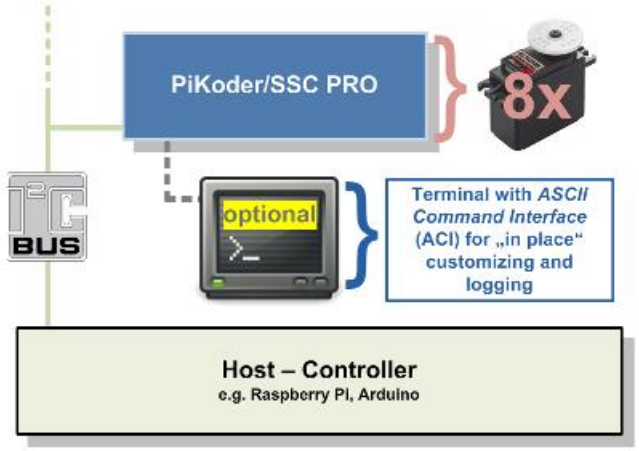
Content

1

Overview

The PiKoder/SSC PRO 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.

For smaller systems and for customizing, you can connect e.g. your Arduino or Raspberry Pi without any additional interface hardware directly to the PiKoder's UART. For larger applications you may want to consider using I2C because with the I2C bus the host controller can manage a complete network using only two pins. A simple protocol like the "miniSSC"-message allows for control of the servo channelsusing the additional UART to customize the PiKoder in place as needed.



This User's Guide covers the features, the programming and the serial interface of the PiKoder/SSC.

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/SSC PRO.

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/SSC PRO is setup in line with this reference schematic.**

Your next step is most likely to commission and test your PiKoder/SSC PRO. 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/SSC PRO to an Arduino using I2C.

This User's Guide is based on the most recent hard- and firmware version 1.2 available for the PiKoder/SSC PRO and the related programming software "PCC PiKoder Control Center" (release 3.1). 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/SSC PRO webpage.

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

This User's Guide is based on the most recent hard- and firmware available for the PiKoder/SSC PRO and the related programming software PCC (PiKoder Control Center). Please check for updated information and new software releases on www.pikoder.com.

Please share with me any comments, improvement ideas or errors you will find or encounter in working with your PiKoder/SSC PRO. 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/SSC PRO allowing you to customize the controller to your specific needs and requirements.

The PiKoder/SSC PRO 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) or the I2C bus. 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
- miniSSC protocol supported (de facto standard for RC servo controllers)
- bi-directional ASCII protocol enabling line monitoring
- I2C slave interface
- optional failsafe position when connection to host is lost
- 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 operating voltage from 3.3 to 5.0 Volts allows you to use an existing power source in your application rather than adding hardware.

In this capacity the SSC 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. The PiKoder/SSC PRO supports three 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 – fully backwards compatible to the PiKoder/SSC 2.09
- I2C in slave mode allowing the integration into larger system - such as robots - because with the I2C bus the host controller can manage a complete network using only two pins. A simple protocol like the "miniSSC"-message allows for control of the servo channels.

The PiKoder/SSC PRO would automatically detect the protocol used and no user interaction would be required.

With the full support of the miniSSC protocol, the PiKoder/SSC PRO would allow you to daisy chain controllers in order to control up to 255 servos as specified in the miniSSC protocol definition. The SSC Control Center provides for setting a miniSSC Offset and thereby defining the channel base address the PiKoder/SSC would respond to. 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.

A free graphical and intuitive configuration and control program, the "PCC PiKoder Control Center" is available for Windows 7, making it simple to test and program the controller over USB (please refer to section 5 for more details).

The PiKoder/SSC PRO also has 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.

The PiKoder/SSC PRO 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 neutral position.

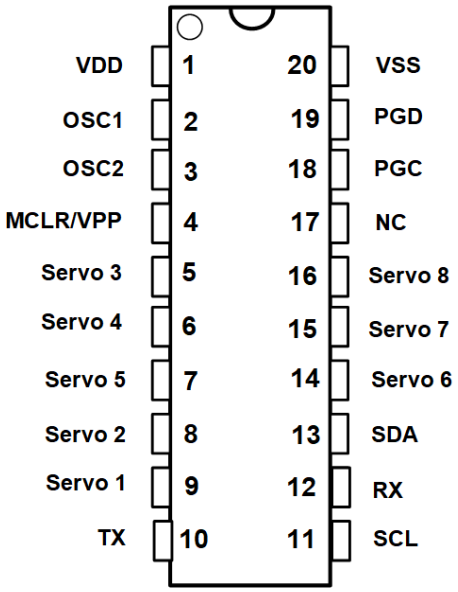
If you were to use this function you might want to implement a regular "ping" in your application to make sure that the failsafe function is not triggered by a period of inactivity.

3

Pin Description and Packaging

The PiKoder/SSC PRO comes in a 20 pin DIP package (see below). The device operates from 3.3 – 5 Volts. Please refer to the PIC 16F690 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.



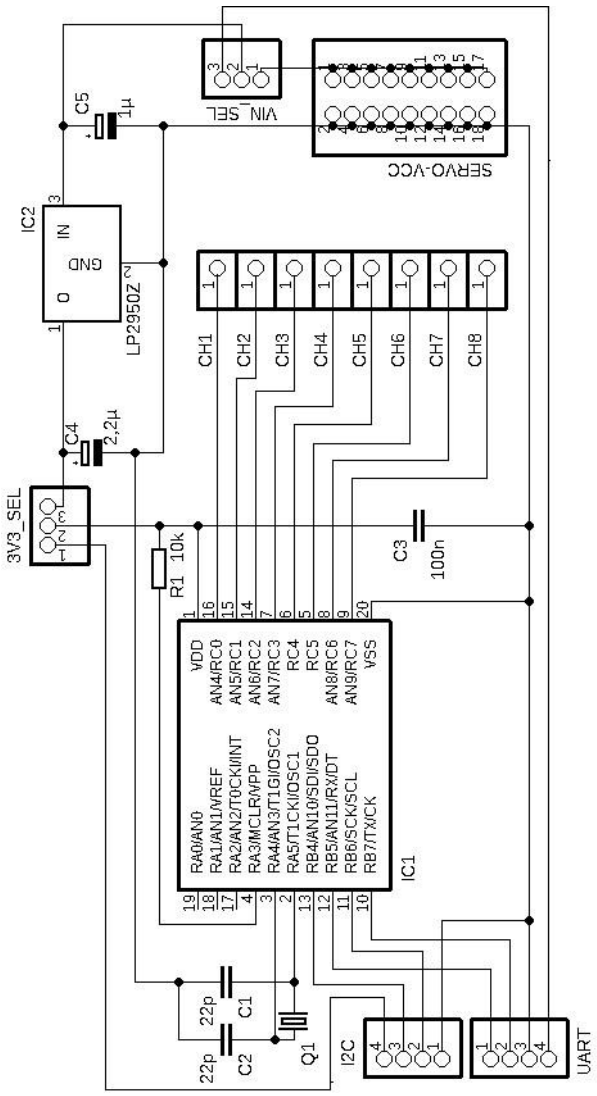
Description of pins

Pin	Symbol	Description
1	VDD	Supply voltage. Connect to 3,3 – 5 V DC
2	OSC1	Crystal 4 MHz. Please refer to Microchip documentation for details
3	OSC2	Crystal 4 MHz. Please refer to Microchip documentation for details
4	MCLR/VPP	Reset pin, active low. Connects directly to Vss for automatic reset at power up.
5	Servo 3	Output pin with PWM signal for servo channel 3
6	Servo 4	Output pin with PWM signal for servo channel 4
7	Servo 5	Output pin with PWM signal for servo channel 5
8	Servo 2	Output pin with PWM signal for servo channel 2
9	Servo 1	Output pin with PWM signal for servo channel 1
10	TX	Serial transmit output
11	SCL	I2C clock line
12	RX	Serial receive input
13	SDA	I2C data line
14	Servo 6	Output pin with PWM signal for servo channel 6
15	Servo 7	Output pin with PWM signal for servo channel 7
16	Servo 8	Output pin with PWM signal for servo channel 8
17	NC	No connection (reserved)
18	PGC	Clock pin for In-Circuit-Serial-Programming
19	PGD	Data pin for In-Circuit-Serial-Programming
20	VSS	Ground connection

4

Standard application

The following schematic shows the standard application of the PiKoder/SSC PRO.



The development board which is available as a kit on **www.pikoder.com** is a full representation of the above reference schematic.

Room for your notes

5

The PiKoder Control Center PCC

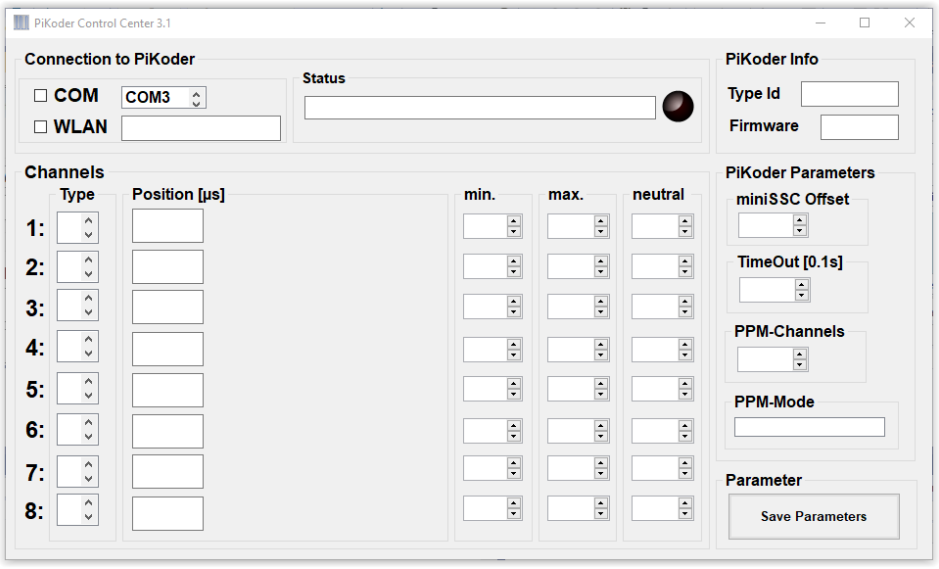
The PiKoder/SSC PRO's serial interface provides access to configuration options as well as support for real time control. The PCC (PiKoder Control Center) is a graphical tool that makes it easy for you to use this interface. For almost any project you will start by using the PCC to set up and test your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

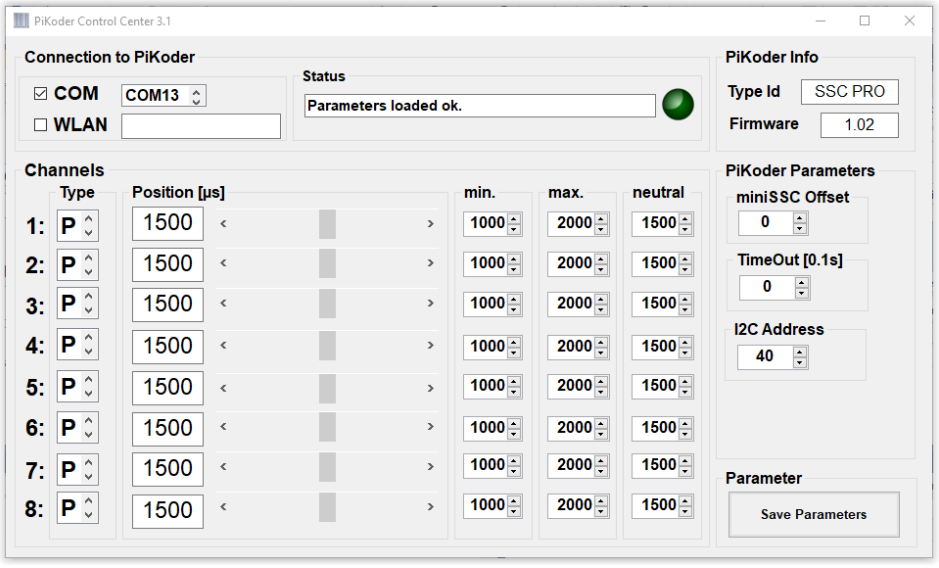
It is highly recommended that you download the latest version of the PCC software to enjoy the complete feature set of your PiKoder/SSC PRO. The PiKoder Control Center software is Open Source and released under an Apache License, Version 2.0. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0. The source code is made available through a github repository; the executable can be downloaded on the PiKoder PCC webpage. **Please note that the PiKoder/SSC PRO requires release 3.1 as a minimum – previous versions do not support this PiKoder.**

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/SSC PRO is connected to (in this example COM13, see below) and then check the box on the left to the COM list box and communication to the controller will be established.

The LED color would change to red while data are uploaded and then turn green for an online indication and the current parameters and settings of the PiKoder/SSC PRO would be loaded and displayed as shown on the following page. The firmware version of your SSC PRO would be displayed in the respective field.



You would now have full control of your SSC PRO: 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 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.

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 µs. A separate row of controls is displayed for each of servo position the PiKoder/SSC'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

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.

TimeOut [0.1s]

This field can be used to activate to program a time out fail-safe configuration. The time out value is shown in multiples of 0.1 s. The maximum input would be 999 resulting in a time out of 99.9 s.

As soon as you change the field value from zero the time out would be activated. From this point onwards the PiKoder/SSC would be monitoring the UART input and expect to receive at least one character within each timeout interval. As soon as a message is received the time out interval is restarted. Please note that the PCC PiKoder Control Center does line monitoring in the background making sure that the time out does not occur while you are programming the PiKoder.

If the PiKoder input would be timed out, then the PiKoder would copy the neutral values to the respective channel to create a predetermined output situation.

I2C Address

This filed contains the current I2C bus address o the PiKoder/SCC PRO (7 bit addressing scheme).

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/SSC to be retrieved when started up the next time.

Please note that saving the 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/SSC PRO's serial interface is based on the controller's UART. The parameters for the serial transmission are 9600 Baud, 8 data bits, one stop bit, with no parity.

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

- ACI - a two-way ASCII-control interface designed to support controlling the SSC with standard terminal programs such as (but not limited to) TeraTerm, Putty, hyperterm
- MiniSSC protocol representing a very common protocol for controlling SSCs

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

ASCII Command Interface

The ASCII Command Interface (ACI) is used to program the PiKoder/SSC PRO. 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 quering parameters and for setting parameters.

If a parameter is read the PiKoder/SSC PRO 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/SSC PRO 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 very limited at this point in time.

The following ACI commands are available:

- '?': query the PiKoder type information; the PiKoder/SSC PRO will respond with "T=SSC PRO".
- "k?': query the current pulse width for channel k (k = 1..8); PiKoder will respond 'CR' 'LF' 'xxxx' 'CR' 'LF' with xxxx representing the pulse width in µs
- 'k=xxxx': set the pulse width for channel k to xxxx µs (xxxx in decimal format, k = 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'
- 'Uk?': query the upper limit for pulse width for channel k; PiKoder will respond 'xxxx' with xxxx representing the pulse width in µs (xxxx in decimal format, k = 1..8) - the command is not case sensitive
- 'Uk=xxxx': set the upper limit for pulse width for channel k to xxxx µs (xxxx in decimal format, k = 1..8); the command is not case sensitive and PiKoder will acknowledge execution with a '!' framed by 'CR' 'LF'
- 'Lk?': query the lower limit for pulse width for channel k; PiKoder will respond 'xxxx' with xxxx representing the pulse width in µs (xxxx in decimal format, k = 1..8); the command is not case sensitive
- 'Lk=xxxx': set the lower limit for pulse width for channel k to xxxx µs (xxxx in decimal format, k = 1..8); the command is not case sensitive and PiKoder will acknowledge execution with a '!' framed by 'CR' 'LF'
- 'Nk?': query the pulse width for the neutral position for channel k; PiKoder will respond 'xxxx' with xxxx representing the pulse width in µs (xxxx in decimal format, k = 1..8) - the command is not case sensitive
- 'Nk=xxxx': set the pulse width for the neutral position for channel k to xxxx µs (xxxx in decimal format, k = 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.
- 'M=xxx' or 'm=xxx': set the channel offset for the miniSSC-protocol
- 'Z?' or 'z?' : query the current zero offset value
- 'Z=xx' or 'z=xx' : set the zero offset

- 'T=xxx' or 't=xxx': enables the fail-safe timer. 'xxx' is given in multiples of 0.1 s. Please note that input is always given in three decimal digits ranging from '001' (= 0.1 s) to '999' (= 99.9 s). In the factory default configuration the fail safe timer is not active. You would activate the monitoring by sending a time out value > 0, a value of 0 would disable the timer function. Command execution is acknowledged by the PiKoder with a '!'.
- 'T?'or 't?': query the current time out value in multiples of 0.1 s. A response of '000' indicates that the fail safe function is not activated
- 'E' or 'e': toggle copy of I2C commands to UART; PiKoder would send the current status after toggling ('+' for on, '-' for off); factory default would be off.
- 'I?'or 'i?': query the current I2C bus write address of the PiKoder; the response will be 'hh' representing the write-address in the I2C bus 7 bit addressing scheme in hex
- 'I=hh'or 'i=hh': set the I2C bus write address of the PiKoder with 'hh' representing the write-address in the I2C bus 7 bit addressing scheme in hex. Command execution is acknowledged by the PiKoder with a '!'
- '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.
- 'Ci?': query all parameters for channel i. The Output would be: nnnnllllluuuu with nnnn representing neutral, llll representing lower limit and uuuu representing the upper limit. The command is not case sensitive.
- SU]U], sU]U]: 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'

miniSSC Protocol

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 miniSSC protocol sequence starts with transmitting 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/SSC 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 = (Upper Limit – Neutral) / 127 Steps

This meas for example, that with an upper limit of 2008 µs and a neutal position of 1500 µs an increment of one in the servo target byte would result in:

du = (2008 – 1500) µs / 127 Steps = 4 µs/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 µs + 4 µs = 1504 µs.

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:

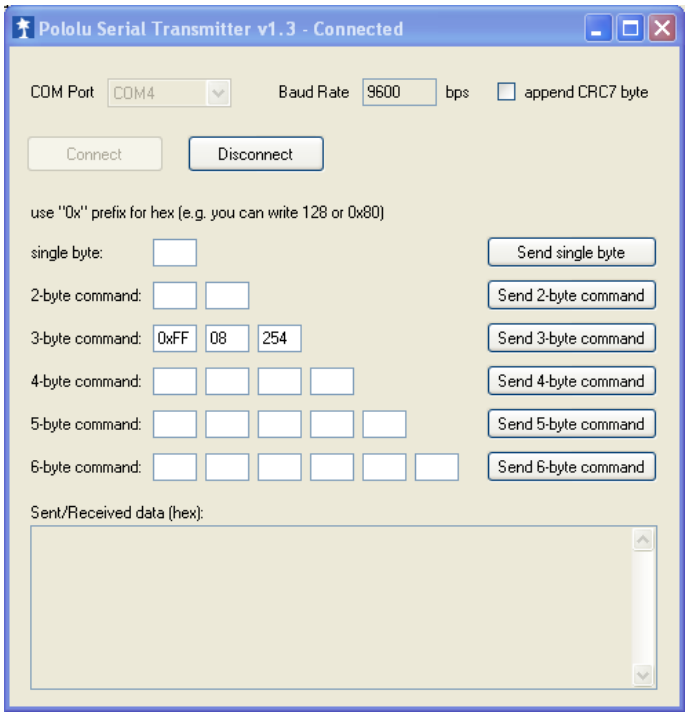
dl = (Neutral - Lower Limit) / 127 Steps

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

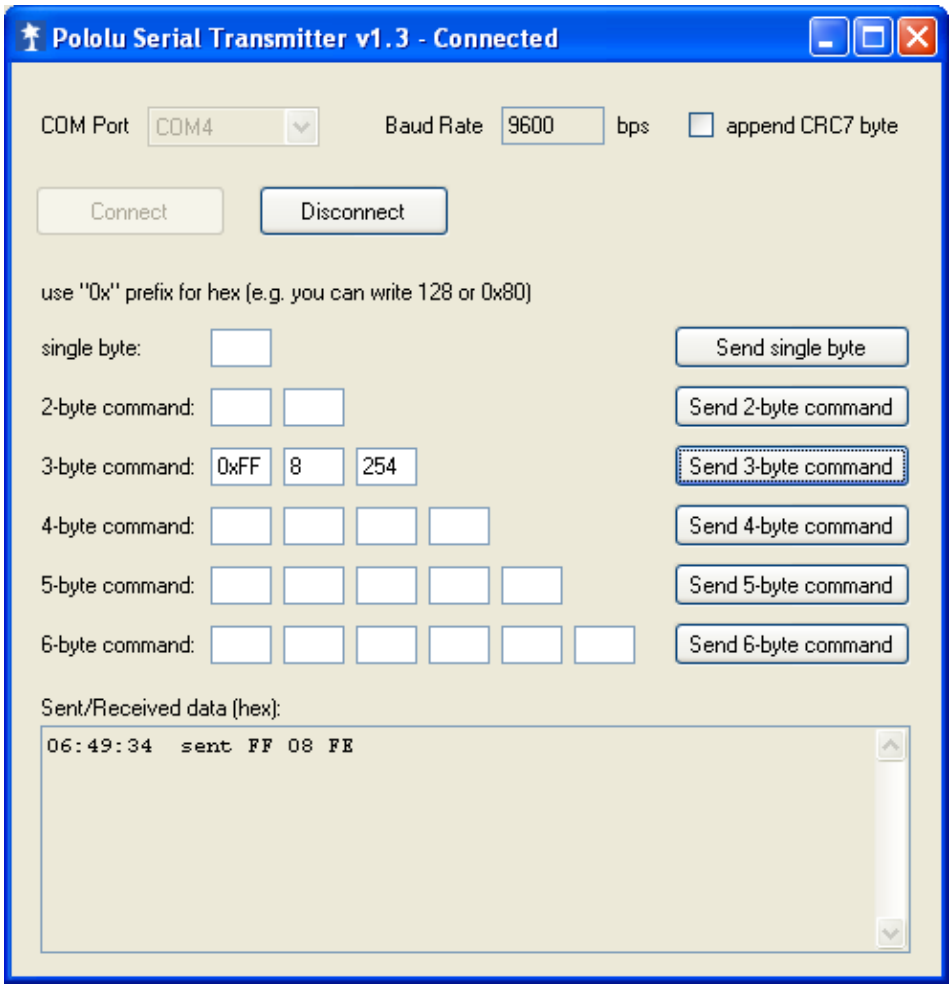
- *du* and *dl* are calculated per channel and may differ per channel allowing for asymmetric ranges and a neutral positon outside of the mechanical central position of the servo
- *du* and *dl* are calculated during controller startup and there is no update when you change the values of neutral, upper or lower limit without resetting the controller. This means, the intended operation would be to set the values, reset the PiKoder/SSC and then verify the setting.
- *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 PRO 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 SSC PRO but a servo connected to the respective channel would move as directed.


Pololu Serial Transmitter v1.3 - Connected
COM Port
COM4
Baud Rate
9600
bps
append CRC7 byte
Connect
Disconnect
use "0x" prefix for hex (e.g. you can write 128 or 0x80)
single byte:
Send single byte
2-byte command:
Send 2-byte command
3-byte command:
0xFF
8
254
Send 3-byte command
4-byte command:
Send 4-byte command
5-byte command:
Send 5-byte command
6-byte command:
Send 6-byte command
Sent/Received data (hex):
06:49:34 sent FF 08 FE

7

I2C interface

The PiKoder/SSC PRO features an I2C bus interface.

The controller acts as an I2C slave expecting the bus master to send commands for positioning servos. The PiKoder/SSC PRO supports a 7 bit addressing scheme with a bus speed of max. 100 kbit/s. The default I2C address is 0x40 but can be changed in the PCC or directly via the ACI command 'I'.

Using the I2C bus it only takes three serial bytes to set the target of one servo, which provides for a similar efficiency than the miniSSC protocol. The first byte to be send however would be the I2C bus address, followed by a servo number byte (1..8 as an offset addressing the channel of the PiKoder/SSC PRO at the I2C bus address), and then the 8-bit servo target byte for the servo position.

The processing ot the servo target byte is completely identical to the miniSSC-protocol handling. Therefore, please refer to this section for more information.

Sample sketch for Arduino

The sample sketch on the following page demonstrates how to interface the PiKoder/SSC PRO with an Arduino.

```
// Control of RC servos by I2C with PiKoder/SSC PRO
//
// Please refer to www.pikoder.com for more details on the PiKoder/SSC PRO
//
#include <Wire.h>

int x = 0;
int rc = 0;

void setup()
{
 Serial.begin(9600);                    // Setup of host communication
 Wire.begin();                          // Start I2C communication
}

void loop()
{
 do {                                   // make sure to sent message
  Serial.print("Servo position = ");
  Serial.print(x);
  Wire.beginTransmission(byte(0x40));       // Address Pikoder/SSC PRO
  Wire.write(byte(0x1));                    // Servo @ channel 1
  Wire.write(byte(x));                      // Transmit new servo position
  rc = Wire.endTransmission();
  Serial.print(" - Return code: ");
  Serial.println(rc);
 } while (rc != 0);
 x = (x + 20)% 255;
 delay(1000);
}
```