

UART2PPM

User's Guide

Version 2.04
dated 02/20/16

Gregor Schlechtriem
www.pikoder.com

Content

Overview	3
PCC PiKoder Control Center	5
Getting started.....	5
Real-time Control.....	7
miniSSC Offset.....	7
TimeOut [0.1s].....	7
Serial Interface	9
Mini SSC Protocol	9
ASCII Command Interface	12
Pin Description and Packaging	17
Description of pins	18
Standard application	19

Overview

The UART2PPM (aka PiKoder/COM) is a single chip converter for serial data (UART) to an eight channel PPM signal with a resolution of 1 μ s. This User's Guide covers the features, the programming and the serial interface of the UART2PPM.

Just add an off-the-shelf "USB to UART converter" to connect your computer's USB port to the PiKoder/COM's input and the PPM stream generated can be used by many R/C transmitters for flying with computer joystick, mouse, etc. by simply connecting the PiKoder output to the trainer connector of the R/C transmitter.

Also a microcontroller such as e.g. your Arduino or Raspberry Pi could take over control: both controllers can connect directly to the PiKoder's UART. 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.

The UART2PPM supports two protocols:

- MiniSSC protocol representing a very common protocol for controlling SSCs and
- a two-way ASCII-Protocol designed to support controlling the UART2PPM with standard terminal programs such as (but not limited to) Tera Term and TTY

The UART2PPM would automatically detect the protocol used and no user interaction would be required.

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 through a USB-port.

The UART2PPM 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.

This User's Guide is based on the most recent hard- and firmware available for the UART2PPM and the related PCC programming software. 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 UART2PPM. I can be reached at webmaster@pikoder.com. Thank you very much!

2

PCC PiKoder Control Center

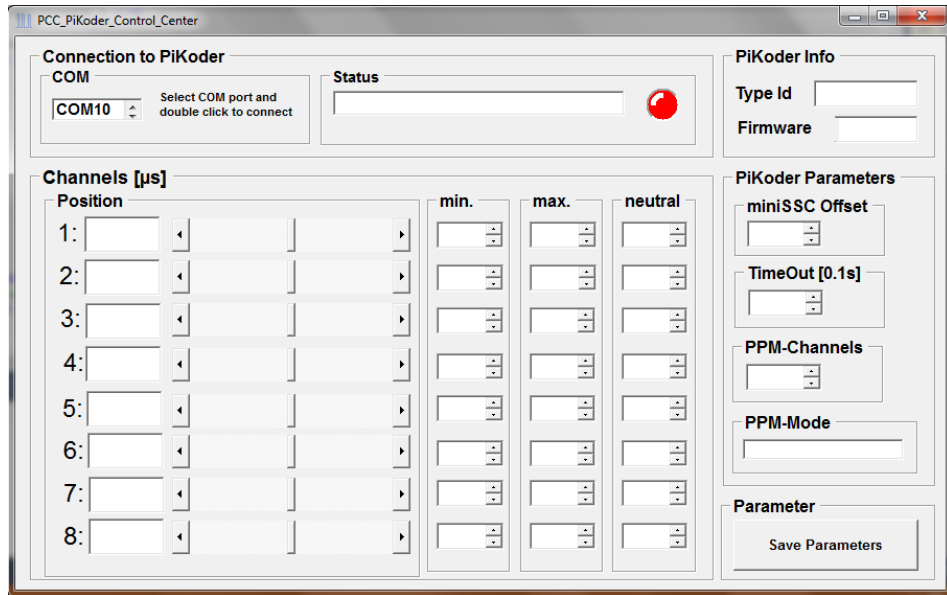
The UART2PPM's serial interface provides access to configuration options as well as support for real time control. The PCC PiKoder Control Center is a graphical tool that makes it easy for you to use this interface. For almost any project you will start by using the PCC PiKoder Control Center to set up and test your PiKoder. This section explains the features of the PCC PiKoder Control Center.

Getting started

The hardware setup for the interface is simple and straight forward: You have to connect your UART2PPM with the USB port of your PC using a suitable USB converter and cable. This cable will provide also for the power supply of the UART2PPM.

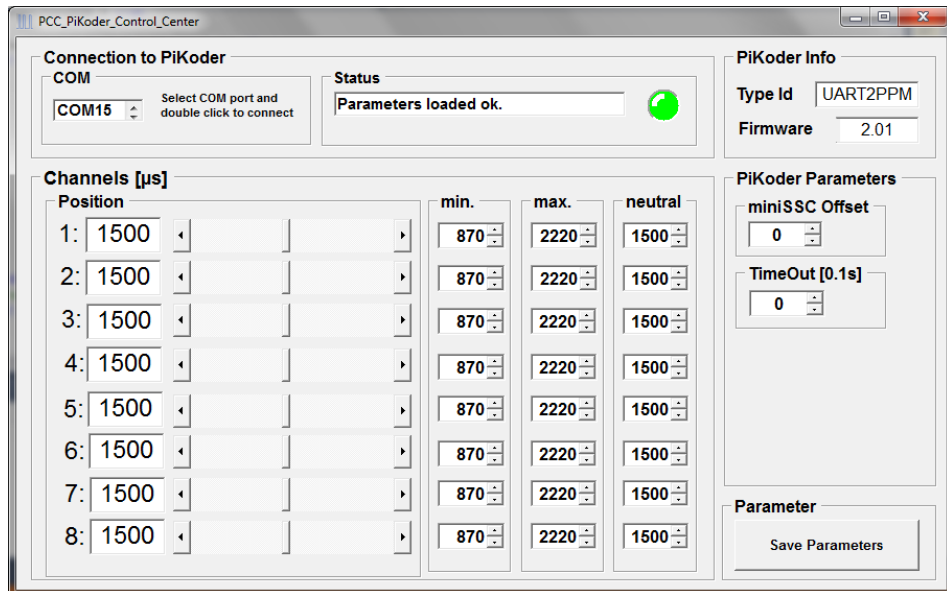
It is highly recommended that you download the latest version of the PCC PiKoder Control Center to enjoy the complete feature set of your PiKoder. The PCC PiKoder Control Center software is Open Source and released under the Apache License version 2.0. The source code is available through github (https://github.com/Pikoder/PCC_PiKoder_Control_Center), the executable is available at http://www.pikoder.com/PiKoder_PCC_EN.html.

If you downloaded the executable then please unzip the downloaded file. The program does not require any installation. After starting it you will see the window as shown on the following page.



The PCC PiKoder Control Center application would show COM10 to be the first available port in your computer. Select the COM port your UART2PPM is connected to (in this example COM15, see below) and then click on the box to indicate your selection and communication to the controller will be established.

The LED color would change to green for an online indication and the current parameters and settings of the UART2PPM would be loaded and displayed as shown on the following page. The firmware version of your PiKoder would be displayed in the respective field.



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

Real-time Control

The sliders are used for controlling the UART2PPM'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 the PiKoder'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 PCC PiKoder 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 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 Chanining" 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/COM 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 actually be timed out then the PiKoder would copy the neutral values to the respective channel to create a predetermined output situation.

- **Room for notes** -

3

Serial Interface

The PiKoder'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 UART2PPM are commands which allow you to control the program and control the PiKoder. The UART2PPM supports two protocols:

- MiniSSC protocol representing a very common protocol for controlling Serial Servo Controllers
- a two-way ASCII-Protocol designed to support controlling the UART2PPM with standard terminal programs such as (but not limited to) TerraTerm, Putty, hyperterm

The PiKoder does automatically detect the terminal protocol; no user interaction would be required.

Mini SSC 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 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 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 μs and a neutral position of 1500 μs 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:

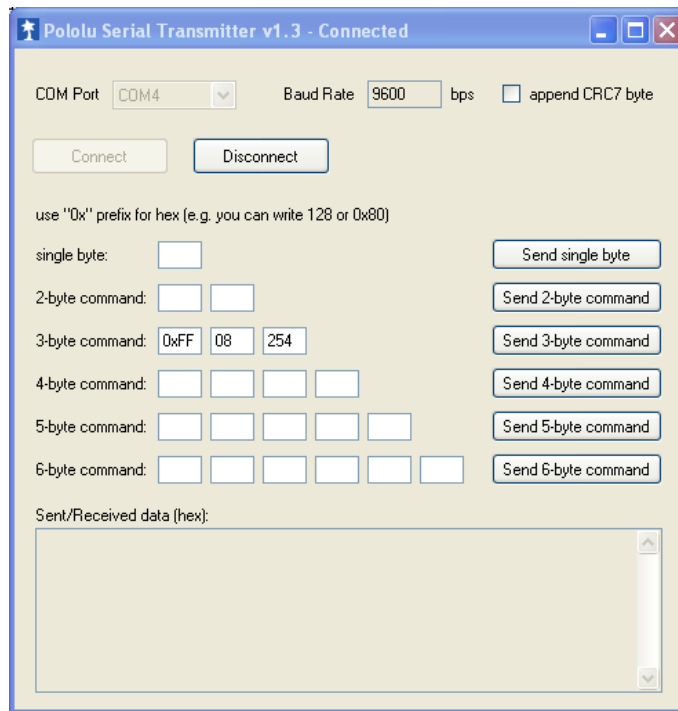
$$dl = (\text{Neutral} - \text{Lower Limit}) / 127 \text{ Steps}$$

Please note the following with respect to du and dl :

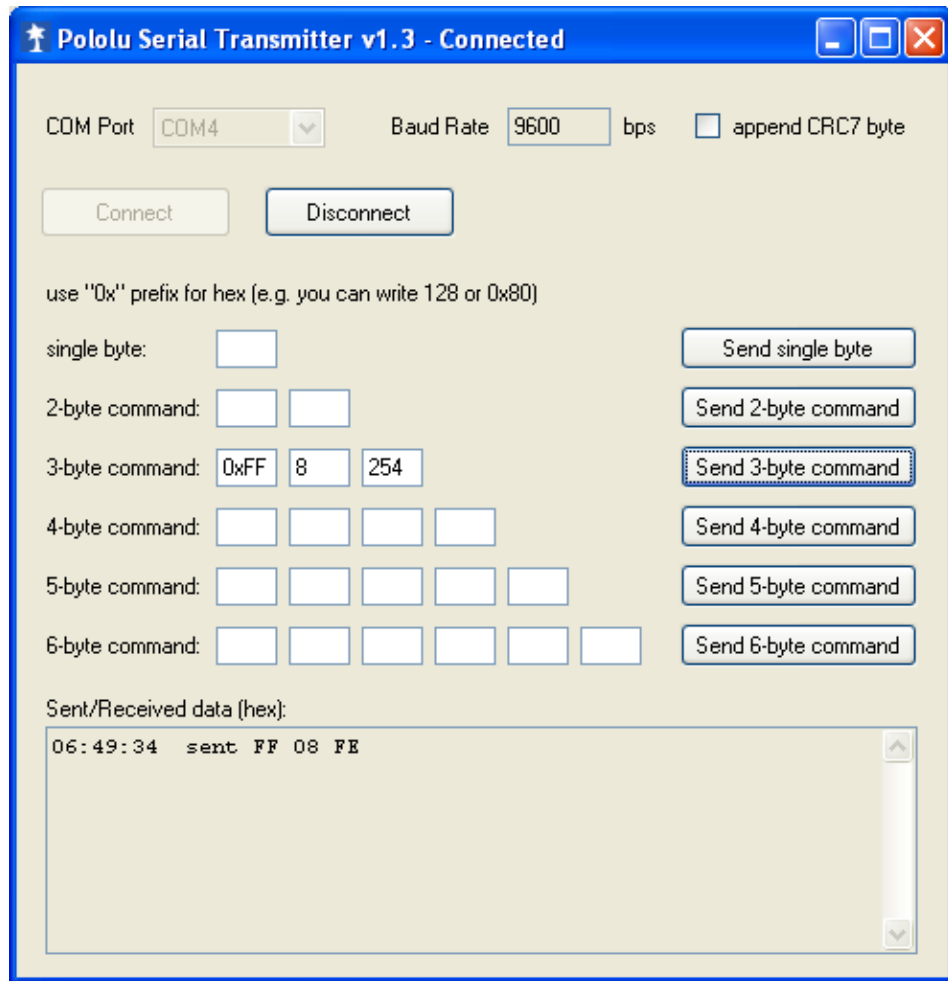
- du and dl are calculated per channel and may differ per channel allowing for asymmetric ranges and a neutral position outside of the mechanical central position of the servo
- du and dl are calculated during controller startup. With firmware release 2.4 these parameters are updated when you change 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 UART2PPM 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 UART2PPM.



ASCII Command Interface

The ASCII Command Interface (ACI) is probably the most versatile way to program the PiKoder without any specific host software such as the “PCC PiKoder Control Center”. 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 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 very limited at this point in time.

The following ACI commands are available:

- '?': query the PiKoder type information; the PiKoder/COM will respond with "T=UART2PPM".
- '0': query the firmware version; PiKoder will respond in a format 'n.nn' framed by 'CR' 'LF'
- '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 μs
- 'i=xxxx': set the pulse width for channel i to xxxx μs (xxxx in decimal format, i = 1..8); PiKoder will acknowledge execution of the program by sending an 'CR' 'LF' '!' 'CR' 'LF'
- 'S','s': 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'
- 'Ui?': query the upper limit for pulse width for channel i; PiKoder will respond 'xxxx' with xxxx representing the pulse width in μs (xxxx in decimal format, i = 1..8) - the command is not case sensitive; the output is formatted with 'CR' 'LF'
- 'Ui=xxxx': set the upper limit for pulse width for channel i to xxxx μs (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 μs (xxxx in decimal format, i = 1..8) - the command is not case sensitive; the output is formatted with 'CR' 'LF'
- 'Li=xxxx': set the lower limit for pulse width for channel i to xxxx μs (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 μs (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 μs (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.
- 'M=iii' or 'm=iii': set the channel offset for the miniSSC-protocol.
- 'Z?' : query the current zero offset value
- 'Z=ii' : set the zero offset
- 'T=iii','t=iii': enables the fail safe timer. 'iii' 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?','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.

- **Room for notes** -

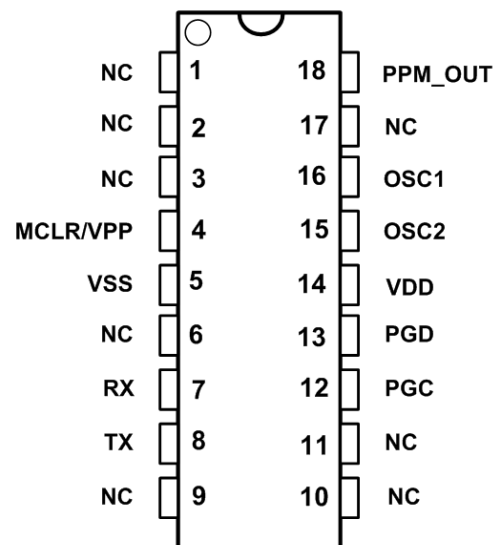
A

Pin Description and Packaging

The PiKoder/COM 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) 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 evaluating the UART2PPM a pcb and a complete 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	NC	Not connected (reserved for later use)
2	NC	Not connected (reserved for later use)
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	NC	Not connected (reserved for later use)
7	RX	Serial receive input
8	TX	Serial transmit output
9	NC	Not connected (reserved for later use)
10	NC	Not connected (reserved for later use)
11	NC	Not connected (reserved for later use)
12	PGC	Clock pin for In-Circuit-Serial-Programming
13	PGD	Data pin for In-Circuit-Serial-Programming
14	VDD	Ground connection
15	OSC2	Crystal 4 MHz. Please refer to Microchip documentation for details
16	OSC1	Crystal 4 MHz. Please refer to Microchip documentation for details
17	NC	Not connected (reserved for later use)
18	PPM_OUT	Output pin with PPM signal

B

Standard application

The following schematic shows the standard application of the UART2PPM.

