Leon Serial Com Documentation

Release 0.1

Stavros Giannakopoulos

May 23, 2014

CONTENTS

Contents:

CHAPTER

ONE

INTRODUCTION

This is the introduction of the Leon-Serial documentation.

CHAPTER

TWO

AUTO GENERATED DOCUMENTATION

2.1 Serial Communication

The leonSer module includes functions for handling the communication through serial port on the most basic level.

The current setup automatically detects the OS and opens the port accordingly (see `platform`). It includes 4 methods for opening, communicating and closing serial ports.

Author: Stavros Giannakopoulos

`leonSer.`**`leonTx`**`(`*serport*, *strinp*`)`

This is the second function which is used to send data to the Communication Port. It takes a port object, in which it sends the string object also received after concatenating the 'endline' character to the end. It finally receives the answer from the board up to 100 bytes and it prints it to the terminal. It may be modified to return the answer as a string.

Parameters

- **serport** – Receives an open serial port object as an input.
- **strinp** – Receives a string to be sent to the com- port.

`leonSer.`**`leonsend`**`(`*serport*, *strinp*`)`

This is the second function which is used to send and receive data to the Communication Port. It takes a port object, in which it sends the string object also received after concatenating the 'endline' character to the end. It finally receives the answer from the board up to 100 bytes and it prints it to the terminal. It may be modified to return the answer as a string.

Parameters

- **serport** – Receives an open serial port object as an input.
- **strinp** – Receives a string to be sent to the com- port.

Returns s, which is a string read by the serial port.

`leonSer.`**`leonstart`**`()`

This is the first function which is used to initialize the Communication Port. Returns the opened port. The Com5 port is set to open for now, but it might be implemented as an argument- or detected automatically in a later date. The baudrate is set to 38343 and it might be implemented as an argument in a later build.

Returns serport, which is the serial port object that was opened.

`leonSer.`**`leonstop`**`(`*serport*`)`

This is the third function which is used to close the Communication Port supplemented as an argument while printing a debugging message on the terminal.

Parameters **serport** – Receives an open serial port object as an input.

The SeriL module includes functions for calling and managing the communication through use of the 'leonSer' module.

The current setup includes two functions, one for running the communication from the GUI, and one for user triggered communication for debugging purposes.

Author: Stavros Giannakopoulos

`SeriL.`**`GuiLeon`**`(`*inp*`)`

This function takes a input string and then calls 'leonstart' to open a serial port. Then it sends the input argument over with 'leonsend' on the port opened.

Parameters **inp** – Input string to be sent over the serial port.

`SeriL.`**`SeriLeon`**`(`*inp*`)`

This function takes a input string and then calls 'leonstart' to open a serial port. Then it sends the input argument over with 'leonsend' on the port opened. Finally it captures the response of the device.

Parameters **inp** – Input string to be sent over the serial port.

Returns y, a string containing the response of the Leon3 board.

The ahbSeri module is used to manage and debug the :mod:'SeriL' module for debugging reasons by simulating user triggered read and write functions on the Leon3 board AHB uart.

Author: Stavros Giannakopoulos

`ahbSeri.`**`ahbread`**`(`*addrr*`)`

This function is used to receive a 32-bit hex memory address in the form of a string by the user. And then it reads a 32 bit hex number from this address and prints it to the terminal. In a future build it will return the value of the memory.

Parameters **addrr** – Input string , address in the form '0x########'

Returns The data read by the memory address given.

`ahbSeri.`**`ahbwrite`**`(`*addrr*, *data*`)`

This function is used to write a 32-bit hex number in a 32-bit memory address in the form of a string given to the function as an argument.

Parameters

- **addrr** – Input string , address in the form '0x########'
- **data** – Input string, data to be written on the memory, in the form '0x########'

Returns 0 if the write was successful or -1 if there was a problem in the communication.

`ahbSeri.`**`userinput`**`()`

This function is used as the main function of this module if the function is called by the terminal. It simulates the input by the data handling function. The user inputs the mode:

R= Read W= Write

Depending on the mode, the respective function is called and executed with the user input as its arguments.

2.2 Data handling

The DataStrLeon module is used to parse and handle the input data by the GUI to the Serial Communication.

Author: Stavros Giannakopoulos

`DataStrLeon.`**`addresser`**(*addressinstring*, *words*, *wordlen*)

This function takes a startng address and a number of words, and generates a list of addresses sparated by the word length.

Parameters

- **addressinstring** – Is the starting hex address in string format.
- **words** – Is the number of words addresses that will be generated beyond the first.
- **wordlen** – Is the length of the word measured in bytes.

Returns addresslist a list of Hex addresses.

`DataStrLeon.`**`data4intformator`**(*FloatInt_list*, *Multipliers*, *Adders*)

This function receives a list of float and int numbers and applies the proper multipliers and additions in order to eliminate any decimals or negative numbers.

Parameters

- **FloatInt_list** – A list float and int numbers.
- **Multipliers** – A list with multiplier values for the various components.
- **Adders** – A list with adder values for the EQ calculation.

Returns Grouped_data, a list of float and int data properly formated.

`DataStrLeon.`**`guiparse`**(*guinput*)

This function receives a string from the GUI. Each component is separated by '#' and each parameter of the respective component is separated by ','. It separates the string into a list of strings with each cell being a coefficient.

Parameters **guinput** – String that has proper formating.

Returns datalist, a list of strings. The dimensions are defined by the input string separators.

`DataStrLeon.`**`hexconcatenator`**(*Inhexed_list*, *In_priority*)

This function receives a list of strings representing hex numbers and concatenates them into 32 bit hex numbers for sending over the serial medium.

Parameters

- **Inhexed_list** – A list of strings representing hex numbers.
- **In_priority** – A string of a hex number with the priority of the effects already encoded.

Returns packets, a list of strings ready to send over the *ahbSeri* module.

`DataStrLeon.`**`hexizer`**(*numbered_data*)

This function converts a mixed list of float and int numbers into a list of hex numbers.

Parameters **numbered_data** – List of Float and Int numbers.

Returns final_data, a list of string hex numbers. The dimensions are defined by the *lengthshouldbe* list.

`DataStrLeon.`**`kickoff`**(*StrGuiinput*)

This function starts the execution of the module. It handles and sends the values to the proper functions. In this build it is also used for debugging by forcing input and printing the output of the module and by sending the data to the *ahbSeri* module.f

`DataStrLeon.`**`numerizer`**`(`*instring*`)`

This function receives a number in the form of a string and it converts it in its proper numeric form respectively.

Parameters **instring** – A string representing a float or int number.

Returns the float or int equivalent of the input string respectively.

`DataStrLeon.`**`parsed2values`**`(`*parsed_data*`)`

This function receives a list of strings from the *guiparse* function. It parses the list of strings supplied and then outputs the int or float equivalents in a simlar list. The numbers are parsed through the *numerizer* function due to the distinction between string to float and string to int conversions, to avoid exceptions.

Parameters **parsed_data** – A list of strings.

Returns Outbound_data, a list of int and float numbers. The dimensions are equal to the input list.

`DataStrLeon.`**`prioritizer`**`(`*In_priority_list*, *EffectNum*`)`

This function receives a list of numbers that indicate the priority of each effect. The most significant decimal digit indicates the first effect, and so on up to 9 effects. A 0 for a priority means that the effect is disabled. The order of the effect is as follows: Delay, Chorus, Flanger, Tremolo, Vibrato, Modulating Wah wah, Auto Wah wah, Phaser and Distortion.

Parameters

- **In_priority_list** – a list of string decimal numbers from 0-9. Each number corresponds to one effect.
- **EffectNum** – Is the number of effects passed down by the kickoff function.

Returns hexoutput, a string converted hex number that includes the effects encoded.

2.3 Graphic User Interface

The GUI module is used create and handle the Graphic User Interface for setting the effect coefficients of the Soundbox.

Author: Mohammed Elghoz

`GUI.`**`AUTO_val`**`()`

Makes the Auto type selectable for the Wah Wah and displays it if selected. Calls the outp function and saves the selected type on the string.

`GUI.`**`BLUES_val`**`()`

Makes the Blues type selectable for the Distortion and displays it if selected. Calls the outp function and saves the selected type on the string.

`GUI.`**`Chorus`**`()`

This function makes sure that the only frame displayed is the Chorus frame. The sliders and Type menus placement in the frame is set.

`GUI.`**`Create_labels`**`()`

This function creates the labels where the priority order will be stored.

`GUI.`**`Delay`**`()`

This function makes sure that the only frame displayed is the Delay frame. The sliders and the entry slots placement in the frame is set.

GUI.**Distortion**()
This function makes sure that the only frame displayed is the Distortion frame. The sliders and menu types placement in the frame is set.

GUI.**EQ**()
This function makes sure that the only frame displayed is the Equalizer frame. The sliders and the entry slots placement in the frame is set.

GUI.**Flanger**()
This function makes sure that the only frame displayed is the Flanger frame. The sliders placement in the frame is set.

GUI.**Gains**()
This function makes sure that the only frame displayed is the Gains frame. The sliders placement in the frame is set.

GUI.**METAL_val**()
Makes the Metal type selectable for the Distortion and displays it if selected. Calls the outp function and saves the selected type on the string.

GUI.**MOD_val**()
Makes the Modulating type selectable for the Wah Wah and displays it if selected. Calls the outp function and saves the selected type on the string.

GUI.**NoiseGate**()
This function makes sure that the only frame displayed is the Noisegate frame. The sliders placement in the frame is set.

GUI.**Phaser**()
This function makes sure that the only frame displayed is the Phaser frame. The sliders placement in the frame is set.

GUI.**ROCK_val**()
Makes the Rock type selectable for the Distortion and displays it if selected. Calls the outp function and saves the selected type on the string.

GUI.**SetPriority**()
This function handles the creation of the 12 priority checkboxes. the effects are as follows: 1:'delay',2:'chorus',3:'flanger',4:'tremolo',5:'vibrato',6:'wah wah',7:'phaser',8:'distortion',9:'noise gate',10:'PreGain',11:'OutGain',12:'EQ'

GUI.**Tremolo**()
This function makes sure that the only frame displayed is the Tremolo frame. The sliders and the menu type placement in the frame is set.

GUI.**Vibrato**()
This function makes sure that the only frame displayed is the Vibrato frame. The sliders and the menu type placement in the frame is set.

GUI.**WahWah**()
This function makes sure that the only frame displayed is the Wah Wah frame. The sliders and the menu type placement in the frame is set.

GUI.**checkClicked**(*number*)
This function takes is called whenever a priority item is clicked. It takes the number of the priority item that called it as an input and places the appropriate effect on the priority list frame. :param name: Number that coresponds to a priority checkbox clicked.

GUI.**chorusRANDOM_val**()
Makes the Random type selectable for the Chorus and displays it if selected. Calls the outp function and saves the selected type on the string.

GUI.**chorusSAWTOOTH_val**()
Makes the Sawtooth type selectable for the Chorus and displays it if selected. Calls the outp function and saves the selected type on the string.

GUI.**chorusSINE_val**()
Makes the Sine type selectable and displays it if selected. Calls the outp function and saves the selected type on the string.

GUI.**chorusSQUARE_val**()
Makes the Square type selectable for the Chorus and displays it if selected. Calls the outp function and saves the selected type on the string.

GUI.**chorusTRIANGLE_val**()
Makes the Triangle type selectable for the Chorus and displays it if selected. Calls the outp function and saves the selected type on the string.

GUI.**exitGUI**()
When the EXIT button is clicked the master frame is closed, thus closing the entire GUI.

GUI.**hello**()
This function is called from the choosable options from the File, Edit and Help menu that has yet to be configured. This function just prints the word hello.

GUI.**labellistset**()
This function updates the Outputpriorities list that is then sent on the Leon3 via the communication code.

GUI.**main**()
Depending on the value of the global variable v, a function of an effect is called and can be modified.

GUI.**makeMenu**()
Creates a File menu, Edit menu, and Help menu. The File menu has the ability to save the current output string to a text file and reload it. The Edit and Help menu calls the hello function and prints hello.

GUI.**openFile**()
Displays the last saved string.

GUI.**outp**(*strprint*, *position*)
This function is called from every function where an effect value can be set and places the chosen values in the correct place. :param strprint: Places the values in the string. :param position: Places the values in the chosen position (in the string).

GUI.**radiobuttons**()
Creates a radiobutton for every effect, and assigns the variable v a value depending on which effect is selected.

GUI.**resetbtn**()
This function resets the effect priorities to their zero values and clears the priority list.

GUI.**saveFile**()
Saves the current string to a file.

GUI.**set_bc**()
This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Bass cutoff.

GUI.**set_bq**()
This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Bass Q.

GUI.**set_bv**()
This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Bass gain.

GUI.**set_chorusDepth**()
This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Chorus depths

GUI.**set_chorusLevel**()
This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Chorus level.

GUI.**set_chorusRate**()
This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Chorus rate.

GUI.**set_delayDryWet**()
This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Delay Dry/Wet.

GUI.**set_delayFeedback**()
This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Delay feedback.

GUI.**set_delayTime**()
This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Delay feedback.

GUI.**set_distortionLevel**()
This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Distortion level.

GUI.**set_distortionMastergain**()
This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Distortion master gain.

GUI.**set_distortionPregain**()
This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Distortion pregain.

GUI.**set_distortionTone**()
This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Distortion tone.

GUI.**set_flangerDelay**()
This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Flanger delay.

GUI.**set_flangerDepth**()
This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Flanger depth.

GUI.**set_flangerLevel**()
This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Flanger level.

GUI.**set_flangerRate**()
This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Flanger rate.

GUI.**set_gain1**()
This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Pregain.

GUI.**set_gain2**()
: This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Outgain.

GUI.**set_labels**()
: Places the variables in the correct order.

GUI.**set_noisegateThreshold**()
: This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Noisegate threshold.

GUI.**set_pc**()
: This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Peak cutoff.

GUI.**set_phaserDepth**()
: This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Phaser depth.

GUI.**set_phaserRate**()
: This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Phaser rate.

GUI.**set_phaserRes**()
: This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Phaser res.

GUI.**set_pq**()
: This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Peak Q.

GUI.**set_pv**()
: This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Peak gain.

GUI.**set_tc**()
: This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Trebble cutoff.

GUI.**set_tq**()
: This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Trebble Q.

GUI.**set_tremoloDepth**()
: This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Tremolo depth.

GUI.**set_tremoloLevel**()
: This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Tremolo level.

GUI.**set_tremoloRate**()
: This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Tremolo rate.

GUI.**set_tv**()
: This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Trebble gain.

GUI.**set_vibratoDepth**()
: This function is called when the PROGRAM button is pressed and calls in turn the output function where it

stores the chosen value for the Vibrato depth.

GUI.**set_vibratoRate**()
This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Vibrato rate.

GUI.**set_wahwahDepth**()
This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Wah Wah depth.

GUI.**set_wahwahRate**()
This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Wah Wah rate.

GUI.**set_wahwahRes**()
This function is called when the PROGRAM button is pressed and calls in turn the output function where it stores the chosen value for the Wah Wah res.

GUI.**strout2 = [‘0’, ‘0’, ‘0’, ‘#’, ‘0’, ‘0’, ‘0’, ‘#’, ‘0’, ‘0’, ‘0’, ‘#’, ‘0’, ‘0’, ‘0’, ‘#’, ‘0’, ‘0’, ‘0’, ‘0’, ‘#’, ‘0’, ‘0’, ‘0’, ‘0’, ‘#’, ‘0’, ‘0’**
This is the strings that are outputted from the GUI. The PriorityList keeps track of the order of the effects. The strout2 string keeps track of all the values set for the different effects. The ‘#’ is used to separate the effects from each other. The order of the effects is the following:

Bass gain, Bass Cutoff frequency, Bass Q and then a ‘#’. Peak gain, Peak Cutoff frequency, Peak Q and then a ‘#’. Trebble gain, Trebble Cutoff frequency, Trebble Q and then a ‘#’. Delay time, Delay feedback, Delay dry/wet and then a ‘#’. Chorus rate, Chorus depth, Chorus level, Chorus type and then a ‘#’. Flanger rate, Flanger depth, Flanger delay, Flanger level and then a ‘#’. Tremolo Rate, Tremolo depth, Tremolo level, Tremolo type and then a ‘#’. Vibrato rate, Vibrato depth, Vibrato type and then a ‘#’. Wah Wah rate, Wah Wah depth, Wah Wah res, Wah Wah type and then a ‘#’. Phaser rate, Phaser depth, Phaser res and then a ‘#’. Distortion pre gain, Distortion master gain, Distortion tone, Distortion level, Distortion type and then a ‘#’. Noise Gate threshold and then a ‘#’. Gain 1 and then a ‘#’. Gain 2 and then a ‘#’.

GUI.**tremoloRANDOM_val**()
Makes the Random type selectable for the Tremolo and displays it if selected. Calls the outp function and saves the selected type on the string.

GUI.**tremoloSAWTOOTH_val**()
Makes the Sawtooth type selectable for the Tremolo and displays it if selected. Calls the outp function and saves the selected type on the string.

GUI.**tremoloSINE_val**()
Makes the Sine type selectable for the Tremolo and displays it if selected. Calls the outp function and saves the selected type on the string.

GUI.**tremoloSQUARE_val**()
Makes the Square type selectable for the Tremolo and displays it if selected. Calls the outp function and saves the selected type on the string.

GUI.**tremoloTRIANGLE_val**()
Makes the Triangle type selectable for the Tremolo and displays it if selected. Calls the outp function and saves the selected type on the string.

GUI.**update_send**()
When the Program button is clicked, this function stores the values for all the sliders in the string and outputs them.

GUI.**vibratoRANDOM_val**()
Makes the Random type selectable for the Vibrato and displays it if selected. Calls the outp function and saves the selected type on the string.

GUI.**vibratoSAWTOOTH_val**()
Makes the Sawtooth type selectable for the Vibrato and displays it if selected. Calls the outp function and saves the selected type on the string.

GUI.**vibratoSINE_val**()
Makes the Sine type selectable for the Vibrato and displays it if selected. Calls the outp function and saves the selected type on the string.

GUI.**vibratoSQUARE_val**()
Makes the Square type selectable for the Vibrato and displays it if selected. Calls the outp function and saves the selected type on the string.

GUI.**vibratoTRIANGLE_val**()
Makes the Triangle type selectable for the Vibrato and displays it if selected. Calls the outp function and saves the selected type on the string.

CHAPTER

THREE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX