fmcw Documentation

Alexandre Bondoux

Aug 30, 2019

Contents:

/!PAGE STILL UNDER CONSTRUCTION /!

This library provides high level access to the FMCW3 radar. It provides the necessary building blocks to automatically handle configuration of the FPGA, reception of the data and post-processing of it. With reasonable settings, most users should even be able to be able to display a few plots in real time.

CHAPTER 1

Table Of Contents:

1.1 fmcw

Modules file calling for all the fmcw package data with a maxdepth of 4.

1.1.1 fmcw package

The fmcw package contains a few modules handling various aspects of it.

Submodules

fmcw.adc module

fmcw.display module

fmcw.ftdi module

pylibftdi - python wrapper for libftdi

Copyright (c) 2010-2017 Ben Bass <benbass@codedstructure.net> See LICENSE file for details and (absence of) warranty

pylibftdi: http://bitbucket.org/codedstructure/pylibftdi

libftdi can be found at: http://www.intra2net.com/en/developer/libftdi/

Neither libftdi nor Intra2net are associated with this project; if something goes wrong here, it's almost definitely my fault rather than a problem with the libftdi library.

fmcw.preprocessing module

fmcw.postprocessing module

Module contents

class fmcw.**FMCW3**(*ADC*, *encoding='latin1'*)

Bases: object

Creates the FTDI object that handles the communication.

clear_adc(*oe1=False*, *oe2=False*, *shdn1=False*, *shdn2=False*)

Create a packet signaling the ADC pins to clear on the FPGA. :param oe1: Clear Output Enable 1 :param oe2: Clear Output Enable 2 :param shdn1: Clear Shutdown 1 :param shdn2: Clear Shutdown 2 :return: Packet to be encapsulated

clear_buffer()

Clear some buffer :return: Packet to be encapsulated

clear_gpio(*led=False*, *pa_off=False*, *mix_enbl=False*, *adf_ce=False*)

Create a packet signaling the GPIO pins to clear on the FPGA. :param led: Clear the LED :param pa_off: Clear pa_off :param mix_enbl: Disable mixer :param adf_ce: Clear the adf_ce :return: Packet to be encapsulated

close()

Close the FTDI object :return:

send_packet(*x*, *cmd*)

Add a header to a packet, encode it and write to FTDI. :param x: data :param cmd: type of packet :return: write to FTDI

set_adc(*oe1=False*, *oe2=False*, *shdn1=False*, *shdn2=False*)

Create a packet signaling the ADC pins to set on the FPGA. :param oe1: Set Output Enable 1 :param oe2: Set Output Enable 2 :param shdn1: Set Shutdown 1 :param shdn2: Set Shutdown 2 :return: Packet to be encapsulated

set_channels(*a=True*, *b=True*)

WARNING: ONLY 2 CHANNELS SUPPORTED Set the channels to be activated (only two supported) :param a: State of channel a :param b: State of channel b :return: Packet to be encapsulated

set_downsampler(*enable=True*, *quarter=False*)

WARNING: THE FPGA CODE REQUIRES IT TO BE ENABLED. TO DO: ALLOW THE USER TO DEACTIVATE IT Set the downsampler. :param enable: Turn it on :param quarter: Divide the sampling rate by another factor of 2 :return: Packet to be encapsulated

set_gpio(*led=False*, *pa_off=False*, *mix_enbl=False*, *adf_ce=False*)

Create a packet signaling the GPIO pins to set on the FPGA. :param led: Set the LED :param pa_off: Set the pa_off :param mix_enbl: Enable the mixer :param adf_ce: Set the adf_ce :return: Packet to be encapsulated

set_sweep(*fstart*, *bw*, *length*, *delay*)

Set sweep parameters. :param fstart: [Hz] Start frequency of the chirp :param bw: [Hz] Bandwidth to use :param length: [s] Duration of the sweep :param delay: [s] Delay between two sweeps :return: real delay between two sweeps (just informational)

write_decimate(*decimate*)

Create a packet to configure the decimation factor at the FPGA level. :param decimate: Number of sweeps to skip. 0 means no sweeps are skipped. :return: Packet to be encapsulated

write_pa_off_timer(*length*)
Convert the duration pa_off_timer to a number of ADC clock cycles :param length: number of clock cycles that represent the pa_off_timer :return: Packet to be encapsulated

write_pll()
Call for the configuration of all the registers. :return: void

write_pll_reg(*n*)
Create a packet to configure a PLL register. :param n: Configuration parameter :return: Packet to be encapsulated

write_sweep_delay(*length*)
Convert the duration between two sweeps (sweep delay) to a number of ADC clock cycles :param length: number of clock cycles that represent the duration between two sweeps :return: Packet to be encapsulated

write_sweep_timer(*length*)
Convert the duration of the sweep to a number of ADC clock cycles :param length: number of clock cycles that represent the duration of a sweep :return: Packet to be encapsulated

class fmcw.**Writer**(*filename*, *queue*, *encoding='latin1'*, *timeout=0.5*)
Bases: threading.Thread

DEPRECATED Legacy Writer thread used to write to the binary log file

run()
Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

f

Index