CMI Digital Signal Processor

Note: this document describes the technical details of the CMI DSP and its internal API. For description of the client-server connection to it see the document “CMI SPM controller interface”.

Overview

The electronics consists internally of two independent parts: a **feedback loop module** that is implemented in the Red Pitaya (RP) and primarily run on the FPGA (and can run even on a standalone RP, without the rest of electronics) and a **data acquisition and generation framework** that is formed by two electronics boards connected to it via SPI interface and includes 3x 20-bit DAC, 2x 18-bit 8-channel ADC, 1x 16-bit 16-channel DAC and RS232 interface. Block scheme is provided in Fig. 1. All the functionality is controlled by a single C API, that can be located in the gwyhwserver_gwyscope folder in the root directory. For running a Scanning Probe Microscope, also a server is provided, called gwyhwserver, that is supposed to be controlled from some client outside of the RP, connected to it via Ethernet.

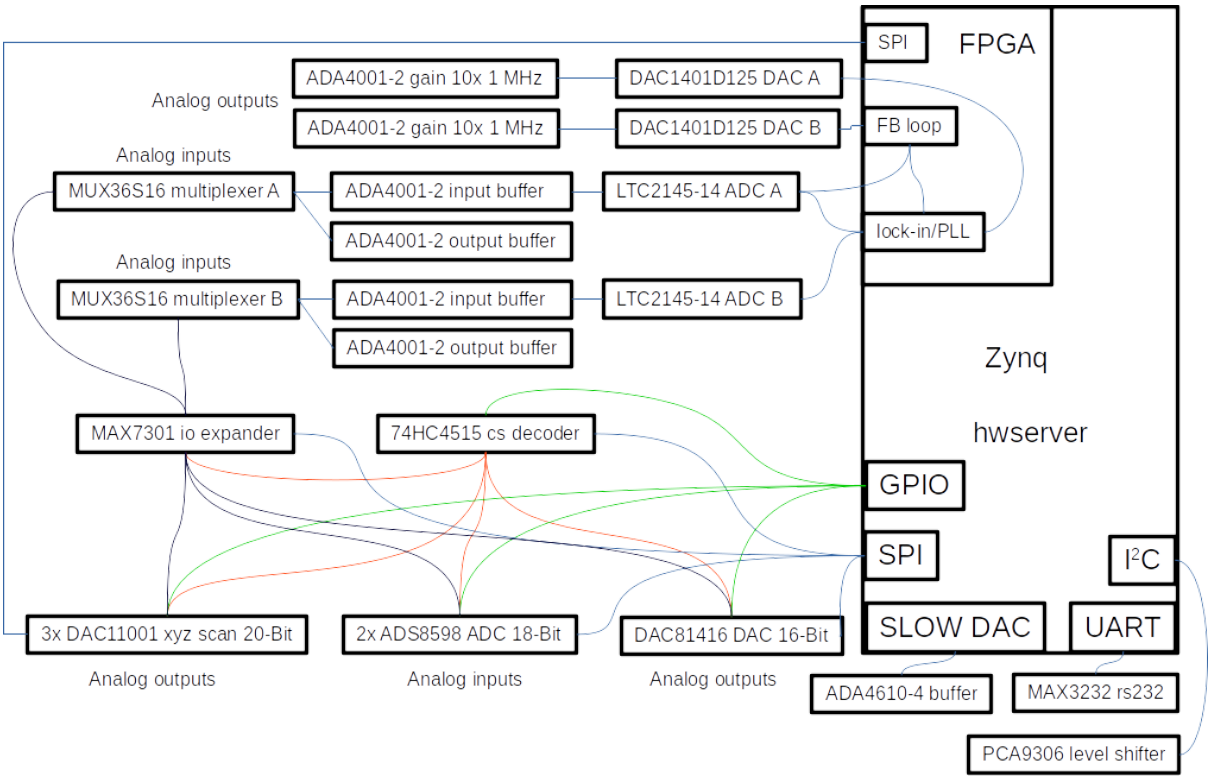

Fig. 1: Block scheme of the DSP

Feedback loop module

The feedback loop module consists of two independent lock-in amplifiers and different digital feedback loops implemented on Red Pitaya's FPGA. Its main purpose is to serve as a source of z feedback for SPM measurements (main feedback loop) and eventually provide PLL feedback for frequency modulated SPM (PLL feedback loop). Some other feedback loops (amplitude, KPFM, DART) are available as well. Schematics of the internal organisation on FPGA is shown in Fig. 2:

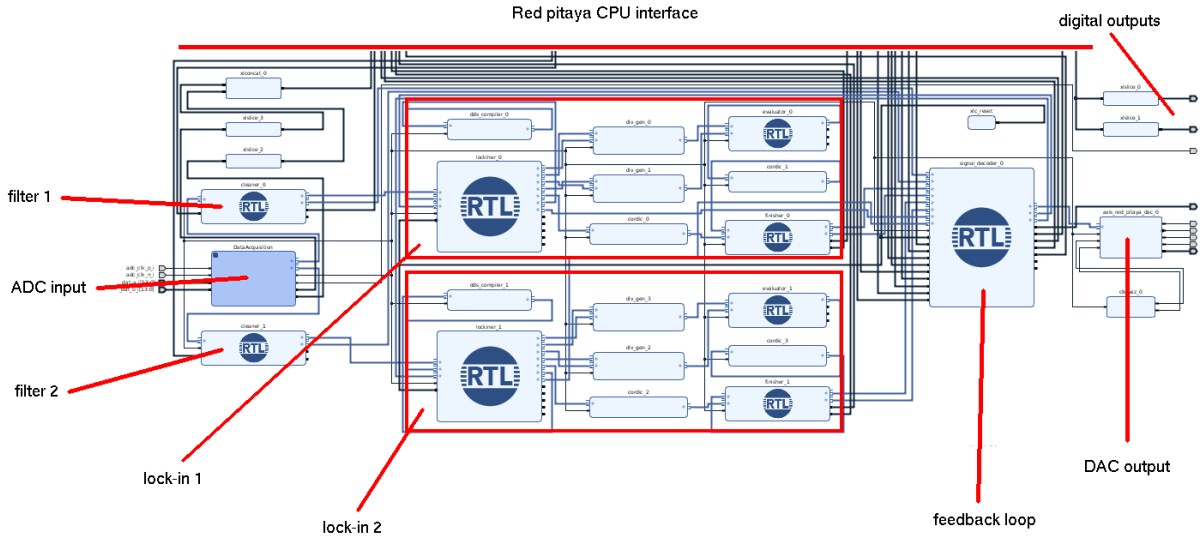

Fig. 2: internal organisation of the feedback loop module implemented on FPGA

The feedback loop module consists of the following components:

Data acquisition block: it samples the two high speed inputs at 125 MHz frequency and passes the data to the filter.

Filter: a simple IIR filter can be used to pretreat the values coming from ADC (each channel separately). It is based on calculating an average value that is then subtracted if passed to the lock-in or used if passed to the feedback loop directly. Use of the filter in lock-in operation is important for removing the DC offset, for this the decimation factor 12 (lowest cutoff frequency) should be used to prevent unwanted reduction of the detected amplitude. In the direct feedback loop it can be used to remove high frequency noise. The only control parameter, decimation factor, controls the average value calculation, except the value 0 that switches the filter off (signal is bypassed to the output). Other values are used to control the averaging time in an exponential way. The effect of the filter in the lock-in mode is shown in Fig. 3, displaying the detected decay of amplitude of generated and detected signal in a wide frequency range.

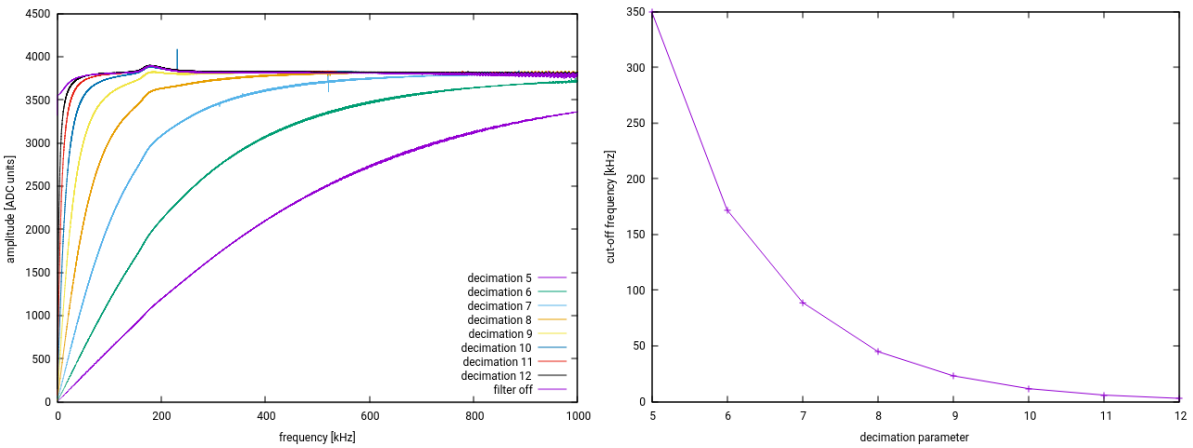

Fig 3: filter applied during lock-in measurement of the reference signal amplitude: frequency response and cutoff frequency dependence on the decimation factor.

Lock-in:

The feedback loop module contains two independent lock-ins, both capable of creating a signal in the range of 500 Hz - 1.995 MHz with amplitude of 0-1 V. Driving signal (output) goes to the feedback loop function and is passed to some of the DAC outputs, depending on the regime. As an input, the signal coming from the filter is used and the filter should be on, removing the DC component (unless the DC component is known to be negligible). Lock-in 1 works with ADC and filter 1, runs DAC1 and provides amplitude 1 and phase 1, operation of lock-in 2 is analogous. The settable frequency resolution is 30 Hz, the settable amplitude resolution is 0.0159301587 V. In the default regime the lock-in is designed to work with already amplified signals (amplitudes in hundreds of mV), so the amplitude resolution is not good to catch signals with small amplitudes (e.g. below 10 mV). In the high resolution regime the lock-in can work with smaller signals (providing 64 times higher sensitivity), however it should not be used with low frequency signals combined with large amplitude (e.g. frequency of 500 Hz combined with >0.2 V amplitude). The digital outputs are amplitude and phase, to be used either within the feedback loop 1 or 2, or for the additional FM feedback loop providing e.g. the phase locked loop.

Digital outputs:

Red Pitaya's GPIO pins are used to control the external AD and DA converters, line decoder and for fast SPI communication with the 20-bit DACs. Nevertheless, there are additional four digital outputs denoted as RP1_SLOW_DAC[1..4] in the schematics that are user settable. Note that the logic level is 3.3 V.

Analog outputs:

The two high speed 14-bit analog outputs are controlled by the user or by the feedback loop, depending on the feedback loop regime (see below). Their range is +-10 V (after amplification). Moreover, the 20-bit high resolution DACs can be connected directly to the FPGA to provide fast and high resolution data for the scanner.

Feedback loop:
The feedback loop operation is controlled by its regime, set in the hwserver.ini file. The following sources of error signal can be fed into the main feedback loop:

RP Input1, RP Input2: feedback loop is based on an average signal coming from the filter (use filter decimation 1 if you want to pass the raw data as an average).

Amplitude1, Amplitude2: feedback loop is based on the amplitude signal coming from one of the lock-ins.

Phase1, Phase2: feedback loop is based on the phase signal coming from one of the lock-ins.

Frequency: feedback loop is coming from the frequency of PLL that can be run on generator 1 and lock-in 1. The frequency shift needed to keep the phase constant is used as a source of feedback loop. The phase that should be kept constant should be set before the FM part of the feedback loop is started (setting fmsettings setpoint before fmsettings feedback is on). This starts tracking the phase by altering generated frequency. Actual frequency shift is provided as an error signal. After that feedback loop for z piezo can be started (setting feedback 1 on, using desired frequency shift as value of setpoint1).

The direction of the error signal can be changed by the *librp_SetState* command (parameter swap_in). The default settings are in hwserver.ini and should reflect the hardware (e.g. the scanner orientation)

Data acquisition and generation framework

The data acquisition and generation framework consists of the following parts:

- 3x DAC11001A (3x single channel, 20-bit DAC, 1-LSB DNL (max), 4-LSB INL (max))
- 1x DAC81416 (1x 16-channel, 16-bit DAC)
- 2x ADS8598H (2x 8-channel, 18-bit ADC)
- RS232 interface

There is also an IO expander included, however this is used primarily for controlling the converters in the present version.

Gwyhwserver

The server for controlling the SPM operation is called gwyhwserver and should be already installed on the Red Pitaya, in the gwyhwserver_gwyscope folder, together with many smaller test applications. To run the server, provide a port, e.g. calling it as ./hwserver 50100. An ini file can be provided to set the hardware parameters that are system specific (e.g. scan ranges), a default ini file is in the gwyhwserver directory.
The server should run always when the client (e.g. GwyScope) is supposed to do any SPM control work. Only one client should be connected to the server and should be connected for the entire SPM operation, otherwise the behaviour might be unpredictable. Gwyhwserver is described in detail in a separate document.

Application Programming Interface

All the functionality of both the feedback loop module and data acquisition and generation framework is controlled via API implemented in C. All the functions are located in cmirp.h and cmirp.c files. These functions are also used in all the test applications and in gwyhwserver. Note that some of the function parameters should match the physical hardware settings.

The following functions are available:

int **librp_Init**(librpSet *rpset, int hrdac_regime, int hrdac1_range, int hrdac2_range, int hrdac3_range, int input_range1, int input_range2, int dds1_range, int dds2_range, int oversampling, int rp1_input_hv, int rp2_input_hv, int rp_bare_output, int rp_bare_input);

This function initializes all the functions and connections, and sets all the default values. Parameter hrdac_regime controls how the 20-bit DACs will be used (0: entirely from FPGA, 1: entirely from CPU). The next three parameters set the 20-bit output range and should match the jumpers on the board (LIBRP_HRDAC_RANGE_FULL: +-10 V, LIBRP_HRDAC_RANGE_SMALL: 0-10V). Parameters input_range1 and input_range2 are related to RP inputs: 0 means +-1V operation, 1 is +- 10 V. Some of these settings are turned on when the SetState command is issued. Parameters dds1_range and dds2_range can be used to override the default range of synthesised harmonic signal coming out of the lock-in. If set to 1, the amplitude range is only 0.5 V and the signal is shifted towards positive values. This can be useful when output should go directly to some piezo amplifier. Otherwise the full range +-1 V is used, which is the default behaviour. Oversampling is applied to ADS8598H ADCs connected via SPI communication. Allowed values range from 0 (no averaging) to 6 (64 values are averaged), which is also the default value. See the ADC datasheet for more information.
Parameter rp1_input_hv and rp1_input_hv should be set to 1 if the fast ADC jumpers on RP are on HV range. Parameter rp_bare_output should be set to 1 if we don't use the auxiliary amplifiers to convert the bare RP outputs from +-1 V to +- 10 V. Similarly, parameter rp_bare_input should be used only if the inputs are connected directly to fast RP ADC.

int **librp_LoadCalData**(librpSet *rpset, double rpadc1_bare_lv_offset, double rpadc1_bare_lv_slope, double rpadc1_bare_hv_offset, double rpadc1_bare_hv_slope, double rpadc2_bare_lv_offset, double rpadc2_bare_lv_slope, double rpadc2_bare_hv_offset, double rpadc2_bare_hv_slope, double rpadc1_divhigh_lv_offset, double rpadc1_divhigh_lv_slope, double rpadc1_divhigh_hv_offset, double rpadc1_divhigh_hv_slope, double rpadc1_divlow_lv_offset, double rpadc1_divlow_lv_slope, double rpadc1_divlow_hv_offset, double rpadc1_divlow_hv_slope, double rpadc2_divhigh_lv_offset, double rpadc2_divhigh_lv_slope, double rpadc2_divhigh_hv_offset, double rpadc2_divhigh_hv_slope, double rpadc2_divlow_lv_offset, double rpadc2_divlow_lv_slope, double rpadc2_divlow_hv_offset, double rpadc2_divlow_hv_slope);

Sets the conversion factors between fast RP inputs and real values for all the potential combinations of input settings. This includes LV and HV options on the RP inputs, use of 1:10 divider on the auxiliary board, use of this board without division and use of bare inputs without board. The used equation has form: real_world_value = (integer_value + offset)*slope.

int **librp_SetInputRange**(librpSet *rpset, int input1_range, int input2_range)
Changes the input range settings (see librp_Init).

int **librp_SetDDSRange**(librpSet *rpset, int dds1_range, int dds2_range)
Changes the DDS range settings (see librp_Init).

int **librp_SetOversampling**(librpSet *rpset, int oversampling)
Changes the oversampling settings (see librp_Init).

int **librp_Close** (librpSet *rpset)
Stops all the functions and closes all connections.

int **librp_ReadADC** (librpSet *rpset, double *adc1, double *adc2)
Reads actual FPGA raw ADC values for the two simultaneous high-speed inputs, converted from raw values to the +-1V input range.

int **librp_ReadResults** (librpSet *rpset, double *error, double *dac, int raw)
Reads actual main FPGA feedback loop results. Meaning of the error signal depends on the feedback loop regime. Values are converted from FPGA numbers depending on the regime of operation, unless the raw option is set. Note that the result value (dac) can be in practice routed somewhere else than the particular channel dac output, depending on mode of operation. If a 20-bit external DAC is connected and the zpiezo_hrdac option is set, librp_ReadZSlow provides the correct dac value.

int **librp_ReadLockin** (librpSet *rpset, int channel, double *amplitude, double *phase)
Reads actual FPGA lock-in results for the selected channel, regardless if there is anything generated in that channel or if it has any meaning at the moment.

int **librp_ReadLockinAxes** (librpSet *rpset, int channel, double *x, double *y);
Reads the sine/cosine demodulation data from a lockin.

int **librp_ReadHRDAC** (librpSet *rpset, double *zslow)
Reads the FPGA based z value that goes to 20-bit DAC for z axis, if requested so during the init phase. In contrast to ReadResults dac value, which is scaled for the RP 14-bit output, this value has higher bit depth.

int **librp_ReadPLLResult** (librpSet *rpset, double *pllresult, double *ampresult)
Reads actual FPGA value of PLL results, used in FM mode (tracked frequency and amplitude).

int **librp_ReadDebug** (librpSet *rpset, double *debug)
Reads actual FPGA debug value, this might be whatever is connected to it on the FPGA side. It is not intended for general use.

int **librp_SetState**(librpSet *rpset, int state, int feedback, int out1, int out2, int outhr, int swap_in, int swap_out, int pidskip)
Sets the FPGA feedback loop, feedback loop status (on/off) and other parameters: output channels for two fast RP outputs and for the 20-bit DAC if enabled (e.g. excitation, dac value, pid result, see cmirp.h for definition) and input/output signal swaps. Pidskip can be used to slow down the feedback loop (0: 125 MHz, 1: 1 MHz, 2: 120 kHz, 3: 15 kHz).

int **librp_SetFeedback**(librpSet *rpset, int feedback)
Convenience function that calls librp_SetState with actual parameters except feedback, to switch it on or off.

int **librp_SetSignalProcessing**(librpSet *rpset, int channel, int decimation, int loopback, int hr, int phaseshift, int debugsource, int filter_phase, int nwaves, int lfr, int intosc)
Sets different parameters of the channel. Parameter decimation sets the FPGA IIR filter for the selected channel. Decimation=0 means that the filter is switched off, the reasonable values are 1-12. Parameter loopback allows the use of an internal value (e.g. output of another calculation) as an input to the lockin. Which parameter is used internally is controlled by the SetState() command. Parameter hr increases the resolution of the lockin about 64 times, at costs of bad operation for higher voltages and lower frequencies. Parameter debugsource passes either sine or cosine signal to debug output of the lockin. Filter_phase adds a simple IIR low pass filter at about 4 kHz to phase output of the lock-in. Parameter nwaves determines from how many individual waves each lock-in output is calculated, 0-5 means 1-32 waves (2^N). Parameter lfr sets the generator range: 0 means full range (0-2 MHz, about 0.02), 1 means the limited range. Parameter intosc asks the lock-in related to this channel to use auxiliary signal generator (see SetAuxGen) as the reference frequency for lock-in detection, instead of its own frequency set by SetGen (that is then used only for output).

int **librp_SetGen** (librpSet *rpset, int channel, double frequency, double amplitude, double offset)
Sets the FPGA frequency and amplitude of the lock-in frequency generator. Frequency (in Hertz) can be set in the range from 500 Hz to 1990000 Hz, amplitude is in Volts and its range is +-10 V. Depending on the feedback loop regime this value is output to the DAC or not. Offset is added to the value (provided in Volts, range is +- 10 V).

int **librp_SetSetpoint** (librpSet *rpset, double setpoint, int raw)
Sets the FPGA setpoint for the main feedback loop. The value is converted to appropriate FPGA numbers depending on the regime of operation, unless a raw option is set.

int **librp_SetPid** (librpSet *rpset, double p, double i, double d, int errorshift)
Sets the FPGA PID parameters for the main feedback loop. Parameters can be in range 0-1, higher value typically means faster response. Note that changing PID parameters on the fly

might lead to some interrupted operation, which should be avoided - a possible workaround is to stop the feedback before setting PID values and to start it after that again. Parameter errorshift performs a bit shift of the error signal internally in the feedback loop.

int **librp_SetPidOffset** (librpSet *rpset, double value)
Sets the offset of zpiezo, i.e. value that is added to the feedback loop output.

int **librp_GetPidOffset** (librpSet *rpset, double value)
Gets the last set value of piezo offset.

int **librp_SetKPFM** (librpSet *rpset, int mode, double frequency, double amplitude, double p, double i, double d, double offset)
Sets all the parameters of KPFM operation from one place, calling different functions to set the desired frequency, amplitude, offset. All this can be done also manually. Modes are defined in cmirp.h and can be either manual (only setting the voltage) or “kpfm”, with a feedback loop. In the FM regime the phase output of lock-in1 is driven to lock-in2 (parameter loopback in SetSignalProcessing). Note that using KPFM requires use of external 20bit z-piezo, in order to free the 14-bit output for KPFM operation.

int **librp_SetDAC** (librpSet *rpset, int channel, double value)
Sets the FPGA high speed DAC outputs directly. This is done only when the regime and feedback loop state allows it. Range is +-1V. If this is set up during the init phase, the channel 2 output can also be set via 20-bit DAC using this function, in this case the range is 0-10 V or +- 10 V (also set in librp_Init function).

int **librp_SetHRDAC** (librpSet *rpset, double value)
Sets the value for the 20-bit DAC connected to the FPGA, the range is the same as for the librp_SetDAC.

int **librp_SetPLL** (librpSet *rpset, int on, int amon, double phase, double amplitude, double p, double i, double ap, double ai, double ad, int phase_limit, int frequency_limit, int pllskip, int pll_input)
Sets the PLL operation. Phase is the setpoint value (in radians), p and i are parameters of the feedback loop, which is switched on and off using the on value. The amplitude stabilisation can be switched using the “amon” parameter and second set of PID loop parameters. Settable phase limit (integer values 0-7 correspond to about 1/32 to full scale) can be used prior to FM feedback loop application, as well as maximum frequency shift limit (integer values 0-7 correspond to about 7.5, 15, 30, 60, 120, 240 and 480 Hz). Pllskip is the slowdown factor for the PLL feedback loop, having the same meaning as for the main PID.

int **librp_DPinSetState** (librpSet *rpset, int pin, int on)
Sets the FPGA digital output pin on (high) or off (low). Pins are numbered 0-15, representing first dio_n [0-7] and then dio_p [8-15] on the Red Pitaya extension connector. Note that when the whole electronics is used (including the generation and data acquisition framework), the pins 0-15 should not be controlled by the user as they serve for driving the additional DA and AD converters.

int **librp_SPICycle** (librpSet *rpset, double hrdac1, double hrdac2, double hrdac3, double *lrdac, double *adc, int read_adc1, int read_adc2, int mux1, int mux2)
Runs the whole SPI communication cycle: three high resolution DAC values hrdac1, hrdac2 and hrdac3 are set (range is +-10 V) , sixteen low resolution DAC values are set, provided by array lrdac[16] (range is +-10 V) and sixteen ADCs are read and stored to adc[16] array (range is +-10 V) . The ADC readout can be further tuned by reading the first eight channels, the second eight channels, or all. All the DAC values are set only if they differ from previous settings. Mux1 and mux2 values control the multiplexers (values in the range 0-15), i.e. they control which analog signal is connected to Red Pitaya's input1 or input2, respectively.