Table of correlated SCPI and API commands on Red Pitaya (date: 21.1.2015)

|  |  |  |  |
| --- | --- | --- | --- |
| SCPI | OPTIONS | DESCRIPTION | API |
| STATE commands |  |  |  |
| **\*RST** |  | Reset RP to default settings. | **rp\_ScpiReset** |
| **\*IDN?**  Example:  \*IDN?  Query return:  RedPitaya,EDU\_v1.1,00:23:31:45:03:34 | <manufacturer>={Red Pitaya}  <model>={EDU\_v1.1}  <MAC> = Red Pitaya address | Returns current device information. | **rp\_ScpiGetID** |
| **\*SRE?**  Query return:  {**0x**XX**,** ERR} | <http://www.av.it.pt/medidas/data/Manuais%20&%20Tutoriais/40b%20-%20VNA%20-%20ZVB20/CD/documents/Help_Files/WebHelp_ZVA/Remote_Control/Status_Reporting_System/status_registers.htm> | Returns STATE BYTE      **PON** - Bit is always set  **URQ** - Bit is set when parser get command from user  **CME** - Bit is set if a command which is undefined or syntactically incorrect is received.  **EXE** - Bit is set if a received command is syntactically correct, but cannot be performed for other reasons  **DDE** - Bit is always 0  **QYE** - Bit is always 0  **RQC** - Bit is set when  **OPC** - Bit is set on receipt of the command \*OPC after all previous commands have been executed.  **Register is cleared after reading it!** | **rp\_ScpiGetSRE** |
| **\*CLS** |  | Delete **OPC** and **EXE** bit in **SRE** | **rp\_ScpiCls** |
| **\*OPC** |  | Set **OPC** bit in **SRE**  after all previous commands have been executed. | **rp\_ScpiOpc** |
| **\*OPC?**  Query return:  {0**,**1,ERR} |  | Returns OPC bit  0-operation in process  1-all operations complete | **rp\_ScpiGet** |
| LED diodes and GPIOs Red Pitaya |  |  |  |
| **DIG:PIN:DIR <dir>,<pin>**    Examples:  DIG:PIN:DIR OUTP,DIO0\_N  DIG:PIN:DIR INP,DIO1\_P | <dir> = {OUTP,INP}  <pin>={DIO**1**\_**P**...DIO**7**\_**P**, DIO**0**\_**N**...DIO**7**\_**N**}  OUTP = OUTPUT  IN = INPUT  Default: OUTP | Set direction of digital pins to output or input. | **rp\_DpinSetDirection** |
| **DIG:PIN <pin>,<state>**  Examples:  DIG:PIN DIO0\_N,1  DIG:PIN LED2,1 | <pin>={DIO**1**\_**P**...DIO**7**\_**P**, DIO**0**\_**N**...DIO**7**\_**N**, LED**1**...LED**8**}  <state>={0,1}  Default: 0 | Set state of digital outputs to 1(HIGH) or 0(LOW). | **rp\_DpinSetState** |
| **DIG:PIN? <pin>**  Examples:  DIG:PIN? DIO0\_N  DIG:PIN? LED2  Query return:  {0, 1, ERR} | <pin>={DIO**1**\_**P**...DIO**7**\_**P**, DIO**0**\_**N**...DIO**7**\_**N**, LED**1**...LED**8**} | Get state of digital inputs and outputs. | **rp\_DpinGetState** |
| Analog Inputs and Outputs |  |  |  |
| **ANALOG:PIN <pin>,<value>**  Examples:  ANALOG:PIN AOUT0,1  ANALOG:PIN AOUT2,1.34 | <pin>={AOUT**0**, AOUT**1**, AOUT**2**, AOUT**3**}  <value>={value in Volts}  Default: 0 | Set analog voltage on  slow analog outputs.  Voltage range of slow analog outputs is:  0 -1.8 V | **rp\_ApinSetValue** |
| **ANALOG:PIN? <pin>**  Examples:  ANALOG:PIN? AOUT0  ANALOG:PIN? AIN2  Query return:  {value in Volts, ERR} | <pin>={AIN**0**, AIN**1**, AIN**2**, AIN**3**, AOUT**0**, AOUT**1**, AOUT**2**, AOUT**3**} | Read analog voltage  from slow analog inputs.  Voltage range of slow analog inputs is:  0 -3.3 V | **rp\_ApinGetValue** |
| Signal Generator  **<n> = {1,2}** (set channel OUT1 or OUT2) |  |  |  |
| **OUTPUT<n>:STATE <par>**  Examples:  OUTPUT1:STATE ON  OUTPUT2:STATE OFF | <par>={ON,OFF}  Default: OFF | Disable or enable fast analog outputs. | **rp\_GenOutEnable**  **rp\_GenOutDisable** |
| **OUTPUT<n>:STATE?**  Examples:  OUTPUT1:STATE?  Query return:  {'ON', 'OFF'} |  | Returns the status of fast analog outputs | **rp\_GenOutIsEnabled** |
| **SOUR<n>:FREQ:FIX <value>**  Examples:  SOUR1:FREQ:FIX 1000  SOUR2:FREQ:FIX 100000 | <value>={frequency in Hz}  Default: 1000 | Set frequency of fast analog outputs. | **rp\_GenFreq** |
| **SOUR<n>:FREQ:FIX?**  Examples:  SOUR1:FREQ:FIX?  Query return:  {1,…,50e6} |  | Returns the current set frequency of fast analog outputs | **rp\_GenGetFreq** |
| **SOUR<n>:FUNC <par>**  Examples:  SOUR1:FUNC SINE  SOUR2:FUNC TRIANGLE | <par>={SINE, SQUARE, TRIANGLE, SAWU, SAWD PWM, ARBITRARY}  Default: SINE | Set waveform of fast analog outputs. | **rp\_GenWaveform** |
| **SOUR<n>:VOLT <value>**  Examples:  SOUR1:VOLT 1  SOUR2:VOLT 0.5 | <value>={voltage in V}  Default: 1  AMP+OFFS <= |1|V | Set amplitude voltage of fast analog outputs. Amplitude + offset value must be less than maximum output range +/- 1V | **rp\_GenAmp** |
| **SOUR<n>:VOLT?**  Examples:  SOUR1:VOLT?  Query return:  {0,…,1} |  | Returns the currently set amplitude voltage on fast analog outputs. | **rp\_GenGetAmp** |
| **SOUR<n>:VOLT:OFFS <value>**  Examples:  SOUR1:VOLT:OFFS 0.2  SOUR1:VOLT:OFFS 0.1 | <value>={voltage in V}  Default: 0  AMP+OFFS <= |1|V | Set offset voltage of fast analog outputs. Amplitude + offset value must be less than maximum output range +/- 1V | **rp\_GenOffset** |
| **SOUR<n>:VOLT:OFFS?**  Examples:  SOUR1:VOLT:OFFS?  Query return:  {0,…,1} |  | Returns the currently set offset voltage value on fast analog outputs | **rp\_GenGetOffset** |
| **SOUR<n>:PHAS <value>**  Examples:  SOUR2:PHAS 3.14  SOUR2:PHAS 180 DEG | <value>={phase shift in radians}  Default: 0 | Set phase of fast analog outputs. | **rp\_GenPhase** |
| **SOUR<n>:PHAS?**  Examples:  SOUR1:PHASE?  Query return:  {-360,…,360} |  | Returns the currently set phase of fast analog outputs | **rp\_GenGetPhase** |
| **SOUR<n>:DCYC <par>**  Examples:  SOUR1:DCYC 34  SOUR2:DCYC 50 | <value>={duty cycle 0-100}  Default: 50  Only for PWM | Set duty cycle of PWM waveform. | **rp\_GenDutyCycle** |
| **SOUR<n>:DCYC?**  Examples:  SOUR1:DCYC?  Query return:  {0,…,100} |  | Returns the currently set duty cycle of PWM waveform | **rp\_GenGetDutyCycle** |
| **SOUR<n>:TRAC:DATA:DATA <array>**  Examples:  SOUR1:TRAC:DATA:DATA 1,0.5,0.2 | <array>={value1, value2,...valueN}  max. 16k values  Values are floats in range from -1 to 1. | Import data for arbitrary waveform generation. | **rp\_GenArbWaveform** |
| **SOUR<n>:BURS:STAT <par>**  Examples:  SOUR1:BURS:STAT ON  SOUR1:BURS:STAT OFF | <par>={ON,OFF}  Default: OFF | Enable or disable  burst (pulse) mode.  Red Pitaya will generate N period of signal and then stop.. | **rp\_GenMode** |
| **SOUR<n>:BURS:STAT?**  Examples:  SOUR1:BURS:STAT?  Query return:  {'ON', 'OFF'} |  | Returns the currently set state of the burst(pulse) mode. | **rp\_GenGetMode** |
| **SOUR<n>:BURS:NCYC <value>**  Examples:  SOUR1:BURS:NCYC 1 | <value>={number of burst cycles; 1 – 50000, INF}    INF = infinity ­ continuous    Default: 1  Currently is 1 | Set N number of periods for burst mode. Currently N is 1. | **rp\_GenBurstCount** |
| **SOUR<n>:BURS:NCYC?**  Examples:  SOUR1:BURST:NCYC?  Query return:  {1,…,50000} |  | Returns the currently set number of periods for burst mode. | **rp\_GenBurstCount** |
| **SOUR<n>:BURS:NOR <value>**  **Examples:**  **SOUR1:BURS:NOR 1** | <value>={burst  repetitions 1-50000,  INF}    INF = infinity | Set R number of repeated  bursts | **rp\_GenBurstRepetitions** |
| **SOUR<n>:BURS:NOR?**  Examples:  SOUR1:BURS:NOR?  Query return:  {1,…,50000, INF} |  | Returns R number of currently set repeated bursts. | **rp\_GenGetBurstRepetitions** |
| **SOUR<n>:BURS:INT:PER <value>**  Examples:  SOUR1:BURS:INT:PER 100000 | <value>={burst period 1us-500us} | Set P total time of one burst  in micro seconds. This  includes the signal and  delay | **rp\_GenBurstPeriod** |
| **SOUR<n>:BURS:INT:PER?**  Examples:  SOUR1:BURS:INT:PER?  Query return:  {1us,…,500us} |  | Returns the currently set time of one burst in micro second. This includes the signal and delay. | **rp\_GenGetBurstPeriod** |
| **SOUR<n>:TRIG:SOUR <par>**  Examples:  SOUR1:TRIG:SOUR EXT | <par>={EXT\_PE,EXT\_NE,INT}  EXT = External  INT = Internal  Default: INT | Set trigger source for selected signal. | **rp\_GenTriggerSource** |
|  |  |  |  |
| **SOUR<n>:TRIG:IMM**  Examples:  SOUR1:TRIG:IMM |  | Triggers selected source immediately | **rp\_GenTrigger** |
| **TRIG:IMM**  Examples:  TRIG:IMM | Mask determines channel: 1->ch1, 2->ch2, 3->ch1&ch2 | Triggers both sources immediately | **rp\_GenTrigger** |
| **GEN:RST**  Examples:  GEN:RST |  | Reset generator to default settings. |  |
| Acquire  **<n> = {1,2}** (set channel IN1 or IN2) |  |  |  |
| Control |  |  |  |
| **ACQ:START**  Examples:  ACQ:START |  | Starts acquisition. | rp\_AcqStart |
| **ACQ:RST**  Examples:  ACQ:STOP |  | Stops acquisition and sets all parameters to default values. | rp\_AcqReset |
| Sampling rate & decimation |  |  |  |
| **ACQ:DEC <par>** | <par>={**1**,8,64,1024,8192,65536}  Default: 1 | Set decimation factor. | rp\_AcqSetDecimation |
| **ACQ:DEC?**  Example:  ACQ:DEC?  Query return:  {1,8,64,1024,8192,65536} |  | Get decimation factor. | rp\_AcqGetDecimation |
| **ACQ:SRAT?**  Example:  ACQ:SRAT?  Query return:  125000000 Hz |  | Get sampling rate in Hz. |  |
| **ACQ:AVG <par>** | <par>={OFF,**ON**}  Default: ON | Enable/disable averaging. | rp\_AcqSetAveraging |
| **ACQ:AVG?**  Example:  ACQ:AVG?  Query return:  {OFF,ON} |  | Get averaging status. | rp\_AcqGetAveraging |
| Trigger |  |  |  |
| **ACQ:TRIG <par>**  Example:  ACQ:TRIG CH1\_PE | <par>={**DISABLED**,NOW,CH1\_PE,CH1\_NE,CH1\_PE,CH1\_NE,EXT\_PE,EXT\_NE,AWG\_PE}  Default: DISABLED | Disable triggering, trigger immediately or set trigger source & edge. | rp\_AcqSetTriggerSrc |
| **ACQ:TRIG:STAT?**  Example:  ACQ:TRIG:STAT?  Query return:  {WAIT,TD} |  | Get trigger status. | rp\_AcqGetTrigger  if DISABLED -> TD  else WAIT |
| **ACQ:TRIG:DLY <par>**  Example:  ACQ:TRIG:DLY 2314 | <par>={value in samples}  Default: 0 | Set trigger delay in samples. | rp\_AcqSetTriggerDelay |
| **ACQ:TRIG:DLY?**  Example:  ACQ:TRIG:DLY?  Query return:  2314 |  | Get trigger delay in samples. | rp\_AcqGetTriggerDelay |
| **ACQ:TRIG:DLY:NS <par>**  Example:  ACQ:TRIG:DLY:NS 128 | <par>={value in ns}  Default: 0 | Set trigger delay in ns. | rp\_AcqSetTriggerDelayNs |
| **ACQ:TRIG:DLY:NS?**  Example:  ACQ:TRIG:DLY:NS?  Query return:  128 ns |  | Get trigger delay in ns. | rp\_AcqGetTriggerDelayNs |
| **ACQ:SOUR<n>:GAIN <par>**  Example:  ACQ:SOUR1:GAIN LV | <par>={LV,HV}  Default: LV | Set gain settings to HIGH or LOW. This gain is referring to jumper settings on Red Pitaya fast analog inputs. | rp\_AcqSetGain |
| **ACQ:TRIG:LEV <par>**  Example:  ACQ:TRIG:LEV 125 mV | <par>={value in V}  Default: 0 | Set trigger level in V. | rp\_AcqSetChannelThreshold  (tu se popravi ime) |
| **ACQ:TRIG:LEV?**  Example:  ACQ:TRIG:LEV?  Query return:  123 mV |  | Get trigger level in V. | rp\_AcqGetChannelThreshold  (tu se popravi ime) |
| Data pointers |  |  |  |
| **ACQ:WPOS?**  Example:  ACQ:WPOS?  Query return:  {write pointer position} |  | Returns current position of write pointer. | rp\_AcqGetWritePointer |
| **ACQ:TPOS?**  Example:  ACQ:TPOS?  Query return:  1234 |  | Returns position where trigger event appeared. | rp\_AcqGetWritePointerAtTrig |
| Data read |  |  |  |
| **ACQ:DATA:UNITS <PAR>**  Example:  ACQ:GET:DATA:UNITS RAW | <par>={RAW, **VOLTS**}  Default: VOLTS | Selects units in which acquired data will be returned. | / |
| **ACQ:[DATA:UNITS](data:UNITS)?**  **Example:**  **ACQ:GET:DATA:UNITS?** |  | Returns what units are currently selected as global units in SCPI. | / |
| **ACQ:DATA:FORMAT <PAR>**  Example:  ACQ:GET:DATA:FORMAT ASCII | <par>={**FLOAT**, ASCII}  Default: FLOAT | Selects format acquired data will be returned. | rp\_AcqScpiDataFormat |
| **ACQ:SOUR<n>:DATA:STA:END? <start\_pos>,<end\_pos>**  Example:  ACQ:SOUR1:GET:DATA 10,13  Query return:  {123,231,-231} | <start\_pos> ={0,1,...,16384}  <stop\_pos> ={0,1,...16384}  stop\_pos > start\_pos | Read samples from start to stop position. | rp\_AcqGetDataPosRaw  rp\_AcqGetDataPosV |
| **ACQ:SOUR<n>:DATA:STA:N? <start\_pos>,<m>**  Example:  ACQ:SOUR1:DATA? 10,3  Query return:  {1.2,3.2,-1.2} |  | Read m samples from start position on. | rp\_AcqGetDataRaw  rp\_AcqGetDataV |
| **ACQ:SOUR<n>:DATA?**  Example:  ACQ:SOUR2:DATA?  Query return:  {1.2,3.2,...,-1.2} |  | Read full buf. size starting from oldest sample in buffer (this is first sample after trigger delay). Trigger delay by default is set to zero (in samples or in seconds). If trigger delay is set to zero it will read full buf. size starting from trigger. | rp\_AcqGetOldestDataRaw  rp\_AcqGetOldestDataV  size=buf\_size ! |
| **ACQ:SOUR<n>:DATA:OLD:N? <m>**  Example:  ACQ:SOUR2:DATA:OLD? 3  Query return:  {1.2,3.2,-1.2} |  | Read **m** samples after trigger delay, starting from oldest sample in buffer (this is first sample after trigger delay). Trigger delay by default is set to zero (in samples or in seconds). If trigger delay is set to zero it will read **m** samples starting from trigger. | rp\_AcqGetOldestDataRaw  rp\_AcqGetOldestDataV |
| **ACQ:SOUR<n>:DATA:LAT:N? <m>**  Example:  ACQ:SOUR1:DATA:LAT? 3  Query return:  {1.2,3.2,-1.2} |  | Read **m** samples before trigger delay. Trigger delay by default is set to zero (in samples or in seconds). If trigger delay is set to zero it will read **m** samples before trigger | rp\_AcqGetLatestDataRaw  rp\_Ac qGetLatestDataV |
|  | | | |
| **ACQ:BUF:SIZE?**  Example:  ACQ:BUF:SIZE?  Query return:  16384 |  | Returns buffer size. | rp\_AcqGetBufSize |
| Oscilloscope TBD!!!  **<n> = {1,2}** (set channel IN1 or IN2) |  |  |  |
| **RUN**  Examples:  RUN |  | Start Acquisition in Auto and Normal mode. | **rp\_AcqStart** |
| **STOP**  Examples:  STOP |  | Stop Acquisition in all modes (Auto, Normal, Single) | **rp\_AcqStop** |
| **SING**  Examples:  SING |  | In single trigger mode, the acquisition triggers once the trigger conditions are met and then stops.  Once buffer is full acq. stops. | **rp\_AcqSingle** |
| **~~TFOR~~**  ~~Examples:~~  ~~TFOR~~ |  | ~~In single trigger mode, using the~~ **~~TFOR~~** ~~command can generate a trigger signal~~  ~~forcefully. Once buffer is full acq. stops.~~ | **~~rp\_TrigForce~~** |

|  |  |
| --- | --- |
| **rp\_ScpiReset** |  |
| **rp\_ScpiGetID** |  |
| **rp\_ScpiGetSRE** |  |
| **rp\_ScpiCls** |  |
| **rp\_ScpiOpc** |  |
| **rp\_ScpiGet** |  |
| LED diodes and GPIOs Red Pitaya |  |
| **rp\_DpinSetDirection** |  |
| **rp\_DpinSetState** |  |
| **rp\_DpinGetState** |  |
| Analog Inputs and Outputs |  |
| **rp\_ApinSetValue** |  |
| **rp\_ApinGetValue** |  |
| Signal Generator |  |
| **rp\_GenOutEnable**  **rp\_GenOutDisable** |  |
| **rp\_GenFreq** |  |
| **rp\_GenWaveform** |  |
| **rp\_GenAmp** |  |
| **rp\_GenOffset** |  |
| **rp\_GenPhase** |  |
| **rp\_GenDutyCycle** |  |
| **rp\_GenArbWaveform** |  |
| **rp\_GenMode** |  |
| **rp\_GenBurstCount** |  |
| **rp\_GenTriggerSource** |  |
| **rp\_GenTrigger** |  |
| **rp\_GenTrigger** |  |
| Acquire |  |
| **Control** |  |
| **rp\_AcqStart()** |  |
| **rp\_AcqStop()** |  |
| **rp\_AcqSingle()** |  |
| **rp\_TrigForce()** |  |
| **Acquire** |  |
| **rp\_AcqSetSamplingRate(rp\_acq\_sampling\_rate sampling\_rate);** |  |
| **rp\_AcqGetSamplingRate(float \* sampling\_rate);** |  |
| **rp\_AcqSetDecimation(float \* sampling\_rate);** |  |
| **rp\_AcqSetAveraging(ON|OFF);** |  |
| **Trigger** |  |
| **rp\_SetTriggerMode(rp\_trg\_mode\_t mode);** |  |
| **rp\_SetTriggerSrc(rp\_trig\_src\_t src);** |  |
| **rp\_SetTriggerSlope(rp\_trig\_slope\_t slope);** |  |
| **rp\_SetTriggerLevel(float level);** |  |
| **rp\_GetTriggerStatus(rp\_trig\_stat\_t \* stat);** |  |
| **Timebase** |  |
|  |  |
| **rp\_SetTimeDelayOffset(float offset);** |  |
| **rp\_GetTimeDelayOffset(float offset);** |  |
| **Wavefrom data** |  |
| **rp\_SetWaveformSource(rp\_ain\_t src);** |  |
| **rp\_SetWaveformFormat(rp\_wformat\_t format);** |  |
| **rp\_SetWaveformPoints(int points);** |  |
| **rp\_GetWaveform(float \* data);** |  |
| **rp\_GetWaveformTrigSample(int \* value);** |  |

