UNIVERSAL+

MULTI DAC/ADC/GPIO/PWM/SERVO HATs

SOFTWARE CONFIGURABLE RASPBERRY PI ZERO™ FORMAT ADD-ON CARD

UNIVERSAL PLUS

RPi Hats & Arduino shield

Python library methods

Croydon, United Kingdom
+447988481257
info@diymat.co.uk

`def SafeMode(self, mode):`

Sets SafeMode

Parameters:

mode - 0 - methods do not send ACK when completed
          1 - methods send ACK when completed

`def GPIOInit(self, Ports, mode=GPIO_INPUT, type=GPIO_TYPE_PP, popd=GPIO_PUPD_NO_PP, speed=GPIO_SPEED_HIGH, state=0):`

Sets up the GPIO

Parameters:

Ports -	list of the port numbers
mode -	GPIO_INPUT / GPIO_OUTPUT
type -	GPIO_TYPE_PP - push-pul
	GPIO_TYPE_OD - open drain
popd -	GPIO_PUPD_NO_PP - no push/pull
	GPIO_PUPD_PU - pull-up
	GPIO_PUPD_PD - pull-down
speed -	GPIO_SPEED_HIGH - port high speed
	GPIO_SPEED_MEDIUM - port medium speed
	GPIO_SPEED_LOW - port low speed

keep the speed high if RPi is not battery operated

state - initial GPIO level

`def GPIOSet(self, Ports, value):`

Sets the GPIO

Parameters:

Ports - list of the the port numbers

value = 0 or not zero. Sets the GPIO output level to low or high

`def GPIOToggle(self, Ports):`

Toggles the GPIO ports

Parameters:

Ports - list of the the port numbers

`def GPIORead(self, Ports):`

Reads the input GPIO ports

Parameters:

Ports - list of the the port numbers

returns the list of the read values on the coresponding positions in the list (GPIO19 is on the 19 position on the list)

`def DACInit(self, Port, obuff=1, generate=0, initialVoltage=0):`

Initialises the DAC Port

Parameters:

Port - DAC port number

obuff - output buffer on/off

generate - init in the generator mode

initalVoltage - initial voltage in the raw format(0-4095)

`def DACWrite(self, Port, Voltage):`

Sets the DAC Port voltage

Parameters:

Port - DAC port number

Voltage - initial voltage in the raw format(0-4095)

def DACGenerate(self, Port, nsamples, samples, frequency, period=0):

Initialises the DAC Waveform generator. Does not start the generation

Parameters:

Port - DAC port number

nsamples - number of samples (1 - 4096)

samples - list with the samples (number of the samples must be equal to nsamples value)

frequency - frequency of the generated signal in Hz

period - period of the generated signal in ns

`def DACFrequency(self, Port, frequency=0.0, period=0):`

Sets the DAC port generator signal frequency. If the generation was started it will change the frequency on the fly

Parameters:

Port - DAC port number

frequency - frequency of the generated signal in Hz

period - period of the generated signal in ns

`def DACStart(self, Port):`

Starts the DAC port generator. The port must be initialised by the DACInit & DACGenerate methods

Parameters:

Port - DAC port number

`def DACStop(self, Port, Voltage=0):`

Stops the DAC port generator

Parameters:

Port - DAC port number

Voltage - output voltage

`def PWMFrequencyDuty(self, Ports, frequency, period=-1, duty=0):`

Sets the PWM port frequency and duty ratio. If the PWM channel is started it changes the frequency and the duty on the fly

Parameters:

Ports - list of the PWM ports to set the frequency / duty ratio

frequency - frequency in Hz

period - period in ns

ratio - duty ratio in % (0 - 100%)

`def PWMFrequency(self, Ports, frequency, period=-1):`

Sets the PWM ports frequency

Parameters:

Ports - list of the PWM ports to set the frequency / duty ratio
frequency - frequency in Hz
period - period in ns

`def PWMDuty(self, Ports, duty=-1):`

Sets the PWM ports duty ratio. If the PWM channel is started it changes the duty on the fly

Parameters:

Ports - list of the PWM ports to set the frequency / duty ratio
ratio - duty ratio in % (0 - 100%)

`def PWMInit(self, Ports, frequency=0, period=0, duty=0):`

Initialises the PWM ports

Parameters:

Ports - list of the PWM ports to set the frequency / duty ratio
frequency - frequency in Hz
period - period in ns
ratio - duty ratio in % (0 - 100%)

`def PWMStart(self, Ports):`

Starts PWM generation on the selected ports

Parameters:

Ports - list of the PWM ports to set the frequency / duty ratio

`def SERVOInit(self, ports, frequency=50, minimum=1000000, maximum=2000000, centre=1500000, exponential=0):`

Initialises the servo port (or PWM port to generate th RC PWM servo signal)

Parameters:

Ports - list of the ports
frequency - frequency of the generated signal. Standard servos require 20ms time between the impluses. High speed servos may accept the higher rates
minimum - minimum impulse width in ns. For standard servos it is 1ms = 1000000ns
centre - centre position impulse width in ns. For standard servos it is 1.5ms = 1500000ns
maximum - minimum impulse width in ns. For standard servos it is 2ms = 2000000ns
expotential - expo ratio. For the explanation please visit: https://www.desmos.com/calculator/x3utvihals

def SERVOSetPos(self, ports, position):

Sets the servo position

Parameters:

Ports - list of the ports

osition - servo position in %. 0% - centre point, -100% minimum position, 100% maximum position

def ADCInit(self, Ports,speed = 0xff):

Initialises the ADC ports

Parameters:

Ports - list of the ports

speed - sample time:

S = sample time:

0 23.4ns

1 39.1ns

2 70.3ns

3 117.2ns

4 304.7ns

5 960.9ns

6 2835.9ns

7 9398.4ns

0xff (dflt) 117.2ns

def ADCReadVref(self):

Reads the Voltage reference

Parameters:

none

Return the reference voltage in mV

def ADCRead(self, ports, speed = 0xff):

Reads the ADC ports

Parameters:

Ports - list of the ports

speed - sample time:

S = sample time:

0 23.4ns

1 39.1ns

2 70.3ns

3 117.2ns

4 304.7ns

5 960.9ns

6 2835.9ns

7 9398.4ns

0xff (dflt) 117.2ns

returns list of lists. List 0 - raw values (0-4095), List 1 - voltage in mV

`def ADCReadData(self, ports, speed = 0xff, nsamples = 0, frequency = 0, period = 0):`

Reads set of data results from ADC ports

Parameters:

Ports - list of the ports

speed - sample time:

S = sample time:

0	23.4ns
1	39.1ns
2	70.3ns
3	117.2ns
4	304.7ns
5	960.9ns
6	2835.9ns
7	9398.4ns
0xff (dflt)	117.2ns

nsmaples - number of samples to be read in one period (1/frequency). The actual time between reads is period / nsamples

frequency - frequency in Hz

period - period in ns

returns list of lists of the list. List 0 - list of lists of results, list[port][0] - raw values (0-4095), list[port][0] - voltage in mV

`def IMPULSERead(self, Port, mode = 0, edge = 1):`

Gets impulse width, frequency of the signal, or frequency and duty ratio of the signal

Parameters:

Port - GPIO port number

mode - 0 - impullse width

1 - frequency and duty ratio of the PWM signal

2 - frequency

edge - initial edge of the signal (1 rising, 0 falling)

Return values:

[result1, result2]

result1 - in mode 0 - width of the impulse in the 1/64000000 s units

- in mode 1 - impulse width of the first part of the PWM signal in the 1/64000000 s units
- in mode 2 - period of the signal in the 1/64000000 s units

result2 - in mode 1 - impulse width of the second part of the PWM signal in the 1/64000000 s units