TINKERplate Command Reference

This reference can also be accessed using the `Help()` command from the TINKERplate Python module. For example:

```
import piplates.TINKERplate as TINK
TINK.help()
```

Definitions:

addr - TINKERplates have jumpers on the board that allow their address to be set to a value between 0 and 7. Almost all commands will require an *addr* argument.

chan - most commands will require that a channel be specified. The analog terminal block has channels 1 through 4 while the Digital I/O port has channels numbered 1 through 8.

System Functions

help() - prints this reference to the screen - typically used from the command line.
setMODE(addr,chan,mode) - this function is used with the Digital I/O block to control its operating mode. When used in a script, it only has to be issued once. The *mode* argument is a string that can be upper or lower case. The following arguments are supported:

- 'DIN' - Data In - defaults at powerup.
- 'DOUT' - Data Out
- 'BUTTON' - Button
- 'PWM' - Pulse Width Modulator
- 'RANGE' - Ultrasonic Range Measurement
- 'TEMP' - Temperature
- 'SERVO' - Servo Motor
- 'RGBLED' - Neopixel® Driver

setDEFAULTS(addr) - returns all Digital I/O ports to simple inputs
getFWrev(addr) - returns the revision of the TINKERplate firmware
getHWrev(addr) - returns the revision of the TINKERplate hardware
getVersion() - returns the revision of the TINKERplate Python 3 module
getID(addr) - returns the string "Pi-Plate TINKERplate"
RESET(addr) - resets the TINKERplate to the power on state

Analog Functions

getADC(addr, chan) – returns a 12-bit measurement of the voltage appearing on the designated chan argument. Returned values will be in the range of 0.0 to 12.0 volts.

getADCall(addr) – returns a list of measurements from all four channels on the TINKERplate located at addr. Using this function is slightly faster than reading the channels one at a time.

getPOT(addr,chan,*range) – this function will return a number from 0.0 to 100.0 that represents the percentage value of the wiper location on the potentiometer. When this command is used without the optional range argument, the TINKERplate assumes that the potentiometer is connected to a +5V voltage source. If you choose to connect your potentiometer to any other voltage source, then *range* should be set to the value of that source.

RELAY Functions

Note that most RELAY commands require the following two arguments:
addr – the address of the TINKERplate. Valid values are 0 through 7
relay – the desired relay on the TINKERplate. Valid values are 1 and 2.

relayON(addr,relay) – energizes the designated relay: closes the NOx contact and opens the NCx contact

relayOFF(addr,relay) – de-energizes the designated relay: opens the NOx contact and closes the NCx contact

relayTOGGLE(addr,relay) – toggles the state of the designated relay. If the relay referenced in the relay argument is de-energized before this command is issued, it will be energized afterwards. If the relay is energized before this command is issued, it will be de-energized afterwards.

relayALL(addr,relays) – uses a single argument to control the state of both relays. See the table below for an explanation of each value. Valid arguments for relay are 0 through 3.

relays Value	NO1	NC1	NO2	NC2
0	open	closed	open	closed
1	closed	open	open	closed
2	open	closed	closed	open
3	closed	open	closed	open

relaySTATE(addr,relay) – returns the status of the relay designated in the relay argument. Returns a 0 if the designated relay is not energized: NCx is closed and NOx is open. Returns a 1 if the designated relay is energized: NCx is open and NOx is closed.

LED Functions

setLED(addr,chan,*brightness) – this command will simply turn an LED off and on when used with chan values of 0 through 8. The optional brightness argument can be used to set the brightness of the attached LED to a range from 0 to 100%. The brightness argument can only be used with channels 1 through 6.

clrLED(addr,chan)– turns off the LED connected to the channel called out in the chan argument.

toggleLED(addr,chan)– toggles the state of the LED connected to the channel called out in the chan argument. If the LED was OFF before this command was issued then it will be turned ON afterwards. Conversely, if the LED was ON before this command was issued then it will be turned OFF afterwards.

RGB LED Functions

setMODE(addr,chan,'rgbled') – this is the mode command for configuring a channel as an RGB LED output driver. It only needs to be issued once. Valid chan values are 1, 2, 3, 4, 5, 6, 7, and 8.

setRGB(addr,chan,red,grn,blu) – this command will set the first LED connected to the specified channel to the RGB color combination called out by the three red, grn, and blue arguments. The values of these arguments can range from 0,0,0 to 255,255,255. Some basic colors include:

red: 255,0,0
green: 0,255,0
blue: 0,0,255
yellow: 255,255,0
orange: 255,165,0

magenta: 255,0,255
cyan: 0,255,255
white: 255,255,255
off: 0,0,0

setRGBSTRING(addr,chan,string) - this command will control a string of up to 64 RGB LEDs. The string argument is a list of sequential rgb values and has valid lengths of 3 to 192 (3*64) values. The values should be arranged as r1,g1,b1,r2,g2,b2...rn,gn,bn.

SERVO Functions

setMODE(addr,chan,'servo') – this is the mode command for configuring a channel as a servo controller. It only needs to be issued once. When issued, any attached servo will be initialized to an angle of 90 degrees.

setSERVO(addr,chan,angle) – this command instructs the servo connected to chan to move to the angular position passed in the angle argument. Note that the angle variable assumes units of degrees and can only accept values between 0.0 and 180.0

setSERVOlow(addr,value) – while developing the TINKERplate, we found that not all servos behave the same way when it comes to timing. If you don't believe your servo motor is returning to the 0 degree position. you can adjust the low side value and "dial it in" until your are satisfied with its position. Set value to a decimal number towards the low end of 0.5 to 2.45. Note that the default servo settings are turned for the SG90.

setSERVOhigh(addr,value) – you can dial in your 180 degree position with this command. Set value to a decimal number towards the high end of 0.5 to 2.45 until you are satisfied with maximum rotation angle. Note that these values are not stored on the TINKERplate and will have to be set everytime you choose to use a servo that requires custom values.

PWM Functions

setMODE(addr,chan,'pwm') – this is the mode command for configuring a channel as a PWM output. It only needs to be issued once. Valid chan values are 1, 2, 3, 4, 5, and 6.

setPWM(addr,chan,dutyCycle) – Once a channel has been configured as a PWM output, use this command to control the duty cycle. The dutyCycle argument is in units of percent and has a valid range of 0.0 to 100.0. Valid chan values are 1, 2, 3, 4, 5, and 6.

Digital I/O

Input Functions

setMODE(addr,chan,'din') – this is the mode command for configuring a channel as a digital input. It only needs to be issued once.

getDIN(addr,chan) – returns the input status of the channel configured as a digital input. Returned values are 0 for a low voltage (<1V) and 1 for a high voltage (>2.3V). Input voltages between 1 and 2.3V will return unpredictable results.

getDINall(addr) – returns the value of all the digital inputs regardless of their mode. The returned value is an 8-bit number with the following mapping:

```
 -------------------------------------------------
|chan 8|chan 7|chan 6|chan 5|chan 4|chan 3|chan 2|chan 1|
 -------------------------------------------------
```

Output Functions

setMODE(addr,chan,'dout') – this is the mode command for configuring a channel as a digital output. It only needs to be issued once.

setDOUT(addr,chan) – sets the spcified channel to a high voltage

clrDOUT(addr,chan) – sets the spcified channel to a low voltage

toggleDOUT(addr,chan) – inverts the current output. If the output was low before this command was issued then it will be set to a high value afterwards. Conversely, if the output was high before this command was issued then it will be set to a low value afterwards.

setDOUTall(addr,byte) – controls all of the DOUT-configured channels with a single command. The following 8-bit mapping again applies:

```
 -------------------------------------------------
|chan 8|chan 7|chan 6|chan 5|chan 4|chan 3|chan 2|chan 1|
 -------------------------------------------------
```

Temperature

setMODE(addr,chan,'temp') – this is the mode command for configuring a channel as a temperature sensor. It only needs to be issued once. Valid chan values are 1, 2, 3, 4, 5, 6, 7, and 8.

getTEMP(addr,chan,*scale) – this function returns the last value measured on the DS18B20 connected to the port designated by the chan argument. This command supports an optional scale argument that overrides the global settings. Valid arguments for scale are 'f' for Fahrenheit, 'c' for Celcius, and 'k' for Kelvin.

setSCALE(scale) – the TINKERplate sets the default temperature scale to Fahrenheit. However, this can be overriden with this command. Valid arguments for scale are 'f' for Fahrenheit, 'c' for Celcius, and 'k' for Kelvin.

getSCALE() – this command returns a single character that denotes the current global temperature scale. Returned values will be 'f' for Fahrenheit, 'c' for Celcius, and 'k' for Kelvin.

Distance Measurement

setMODE(addr,channelpair,'range') – this is the mode command for configuring a channel pair for a range sensor. It only needs to be issued once. Valid channelpair values are 12, 34, 56, and 78.

getRANGE(addr,channelpair,*units) – this function returns the average of the last 8 range measurements. Because of this, the returned values will be be more consistent but will change slowly if the measurement distance changes. Again, a channel pair has to be specified. An optional units argument can be used to return values in units other than the current default. Note that this command will return with a "sensor failure" message if:

- no pinger is attached to the input
- the mode for this channel pair has not been set to 'range'
- the pinger is blocked

getRANGEfast(addr,channelpair,*units) – this function returns the most recent range measurement. Returned values will have more variation but it will show a much faster response to changes in distance. Again, a channel pair has to be specified. An optional units argument can be used to return values in units other than the current default.

setUNITS(units) – As mentioned above, the TINKERplate defaults to inches as a measurement of distance. Use this command to change the default. Valid values for units are ‘cm’ and ‘in’. Note that this command only has to be issued once.

getUNITS() – This command will return the current units of distance used by the TINKERplate. Returned values will be one two character strings: ‘in’ or ‘cm’

Button

setMODE(addr,chan,’button’) – this is the mode command for configuring a channel as a button. It only needs to be issued once.

getBUTTON(addr,chan) – this will return a debounced button input.

Returns a 0 (zero) if the attached button is not closed

Returns a 1 (one) if there is closure between the input and ground

Meter

openMETER(*linesel) – before using a meter, call this function first. If no argument is included, a 1 line meter will be created. If more than 1 line of data is required, pass a number between 1 and 12 in the options linesel argument.

setMETER(value,unit,descriptor,*linesel) – update the data on the optional line of the meter with the specified value, unit, and descriptor.

setTITLE(title) – changes the default title of the METER window to the string contained in the title argument

setCOLOR(newcolor) – use this function to change the color of the meter text from the default of green. The newcolor argument is a tuple of the three colors: (red, blue, green). Each of these are 8-bit values their range is 0 to 255. To display dark yellow for example you would pass a tuple of (128, 128, 0) in the newcolor argument.

closeMETER() – closes the open on-screen meter