Move38-Arduino-Platform Documentation

Release 1.0

Move38-Arduino-Platform

Oct 25, 2017

Contents

Welcome welcome welcome!

CHAPTER 1

Getting Started

This is going to be so much fun!

CHAPTER 2

Blink Lib

This defines a high-level interface to the blinks tile hardware.

Memory

void **setState**(*byte newState*)
Set our state to newState.

This state is repeatedly broadcast to any neighboring tiles. Note that setting our state to 0 make us stop broadcasting and effectively disappear from the view of neighboring tiles.

Parameters

- `newState`: The new state that should be set.

bool **neighborChanged**()
Did the state on any face change since last called? Get the neighbor states with *getNeighborState()*

byte **getNeighborState**(*byte face*)
Returns the last received state of the indicated face, or 0 if no messages received recently on indicated face.

Parameters

- `face`: Which side of the Blink from which to get the neighbor's state

Button

bool **buttonPressed**(void)
Returns true if the button has been pressed since the last time it was called.

bool **buttonLifted**(void)
Returns true if the button has been lifted since the last time it was called.

bool **buttonSingleClicked**()
Was the button single, double , or multi clicked since we last checked? Note that there is a delay after the button is first pressed before a click is registered because we have to wait to see if another button press is coming.

A multiclick is 3 or more clicks Remember that these click events fire a short time after the button is lifted on the final click If the button is held down too long on the last click, then click interaction is aborted.

bool **buttonDoubleClicked**()

bool **buttonMultiClicked**()

byte **buttonClickCount**(void)
The number of clicks in the longest consecutive valid click cycle since the last time called.

bool **buttonLongPressed**(void)
Remember that a long press fires while the button is still down.

bool **buttonDown**(void)
Returns true if the button is currently pressed down.

Color Helpers

BRIGHTNESS_LEVELS
Number of brightness levels in each channel of a color.

GET_R(color)

GET_G(color)

GET_B(color)

MAKECOLOR_RGB(r, g, b)

RED

YELLOW

GREEN

CYAN

BLUE

MAGENTA

WHITE

OFF

LEDs

Color **makeColorRGB**(*byte red*, *byte green*, *byte blue*)
Make a new color from RGB values.

Each value can be 0-31.

Parameters

- `red`:
- `green`:
- `blue`:

Color **dim**(*Color color*, *byte brightness*)
Dim the specified color.

Brightness is 0-31 (0=off, 31=don't dim at all-keep original color)

Parameters

- `color`:
- `brightness`:

Color **makeColorHSB**(*byte hue*, *byte saturation*, *byte brightness*)
Make a new color in the HSB colorspace.

All values are 0-255.

void **setColor**(*Color newColor*)
Change the tile to the specified color.

Parameters

- `newColor`:

void **setFaceColor**(*byte face*, *Color newColor*)
Set the pixel on the specified face (0-5) to the specified color.

Parameters

- `face`:
- `newColor`:

Comms

void **irSendData**(uint8_t *face*, uint8_t *data*)
Send data on a single face.

Data is 7-bits wide, top bit is ignored.

Parameters

- `face`:
- `data`:

void **irBroadcastData**(uint8_t *data*)
Broadcast data on all faces.

Data is 7-bits wide, top bit is ignored.

Parameters

- `data`:

bool **irIsReadyOnFace**(uint8_t *face*)
Is there a received data ready to be read on the indicated face? Returns 0 if none.

Parameters

- `face`:

uint8_t **irGetData**(uint8_t *led*)
Read the most recently received data.

Value 0-127. Blocks if no data ready.

Parameters

- `led`:

Error Codes

ERRORBIT_PARITY
There was an RX parity error.

ERRORBIT_OVERFLOW
A received byte in lastValue was overwritten with a new value.

ERRORBIT_NOISE
We saw unexpected extra pulses inside data.

ERRORBIT_DROPOUT
We saw too few pulses, or two big a space between pulses.

ERRORBIT_DUMMY

Sketch

void **setup**(void)
Called when this sketch is first loaded and then every time the tile wakes from sleep.

void **loop**()
Called repeatedly just after the display pixels on the tile face are updated.

Typedefs

typedef uint8_t **byte**

typedef unsigned **Color**

Functions

void **delay**(unsigned long *millis*)
Delay the specified number of milliseconds (1,000 millisecond = 1 second)

Parameters

- `millis`: The number of milliseconds to delay

unsigned long **millis**(void)
Number of milliseconds since we started (since last time setup called).

Note that this can increase by more than 1 between calls, so always use greater than and less than rather than equals for comparisons

byte **getSerialNumberByte**(*byte n*)

Read the unique serial number for this blink tile There are 9 bytes in all, so n can be 0-8.

Parameters

- `n`:

uint8_t **irGetErrorBits**(uint8_t *face*)

Read the error state of the indicated LED Clears the bits on read.

Parameters

- `face`:

CHAPTER 3

Blink Core

Low level button functionality.

Functions

void **button_init** (void)

void **button_enable** (void)

uint8_t **button_down** (void)

void **button_disable** (void)

void **button_callback_onChange** (void)

void **button_ISR_on** (void)

void **button_ISR_off** (void)

Index

Y