

Claw Device Controller

Generated by Doxygen 1.16.0

1 claw	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 stepper_state Struct Reference	7
4.1.1 Detailed Description	7
5 File Documentation	9
5.1 claw.c File Reference	9
5.1.1 Detailed Description	11
5.1.2 Function Documentation	11
5.1.2.1 main()	11
5.1.2.2 pico_led_init()	11
5.1.2.3 pico_set_led()	12
5.1.2.4 process_command()	12
5.1.2.5 process_led_tick()	12
5.1.2.6 process_stdin_input()	13
5.1.2.7 process_stepper_movement()	13
5.1.2.8 stepper_enable()	13
5.1.2.9 stepper_get_status()	14
5.1.2.10 stepper_init()	14
5.1.2.11 stepper_set_step_period()	14
5.1.2.12 stepper_set_target_position()	15
5.1.2.13 stepper_stop()	15
5.1.2.14 timer_callback()	15
5.1.3 Variable Documentation	16
5.1.3.1 help_message	16
5.1.3.2 led_period	16
5.1.3.3 ms_ticks_count	16
5.1.3.4 ten_us_ticks_count	16
Index	17

Chapter 1

claw

Raspberry pico 2 code to drive stepper motor driven claw

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

stepper_state	Structure to hold stepper motor state	7
-------------------------------	---	-------------------

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

claw.c	Firmware to control a claw device with stepper motors via USB serial commands	9
------------------------	---	---

Chapter 4

Class Documentation

4.1 stepper_state Struct Reference

Structure to hold stepper motor state.

Public Attributes

- int **current_position**
Current position in steps.
- int **target_position**
Target position in steps.
- int **step_period**
Step period in TIMER_INTERVAL_US units.
- bool **moving**
Is the stepper currently moving.
- bool **enabled**
Is the stepper enabled.

4.1.1 Detailed Description

Structure to hold stepper motor state.

The documentation for this struct was generated from the following file:

- [claw.c](#)

Chapter 5

File Documentation

5.1 claw.c File Reference

Firmware to control a claw device with stepper motors via USB serial commands.

```
#include <stdio.h>
#include "pico/stdlib.h"
#include "hardware/timer.h"
#include <string.h>
#include "pico/assert.h"
#include <stdlib.h>
#include "hardware/gpio.h"
#include <ctype.h>
#include <math.h>
```

Classes

- struct [stepper_state](#)

Structure to hold stepper motor state.

Macros

- #define LED_DELAY_MS 1000
- #define TIMER_INTERVAL_US 10
- #define DEFAULT_STEPPER_PERIOD 4
- #define MIN_STEPPER_PERIOD 4
- #define STEPPER_STEP_PIN 2
- #define STEPPER_DIR_PIN 3
- #define STEPPER_ENABLE_PIN 4
- #define STEPPER_ENABLE_PIN_INVERTED true
- #define STEPPER_DIRECTION_FORWARD 1
- #define STEPPER_DIRECTION_BACKWARD 0
- #define STEPPER_STEPS_PER_REV 3200
- #define STEPPER_MAX_REVOLUTIONS 12
- #define STEPPER_BUMP_STEPS (STEPPER_STEPS_PER_REV / 4)
- #define MAX_STEPPER_POSITION (STEPPER_STEPS_PER_REV * STEPPER_MAX_REVOLUTIONS)

- #define **MIN_STEPPER_POSITION** 0
- #define **MAX_COMMAND_LENGTH** 50
- #define **CLAW_SET_POSITION_COMMAND** "claw_set "
- #define **LED_PERIOD_COMMAND** "led_period "
- #define **SET_STEPPER_PERIOD_COMMAND** "set_stepper_period "
- #define **SET_STEPPER_ZERO_COMMAND** "set_stepper_zero"
- #define **MOVE_STEPPER_ABSOLUTE_COMMAND** "move_stepper_absolute "
- #define **MOVE_STEPPER_RELATIVE_COMMAND** "move_stepper_relative "
- #define **MOVE_STEPPER_ROTATIONS_COMMAND** "move_stepper_rotations "
- #define **MOVE_STEPPER_BUMP_DOWN_COMMAND** "move_stepper_bump_down"
- #define **STOP_STEPPER_COMMAND** "stop_stepper"
- #define **GET_STEPPER_STATUS_COMMAND** "get_stepper_status"
- #define **ENABLE_STEPPER_COMMAND** "enable_stepper"
- #define **DISABLE_STEPPER_COMMAND** "disable_stepper"

Typedefs

- typedef struct **stepper_state stepper_state_t**

Functions

- bool **stepper_init** (stepper_state_t *stepper, int initial_position, int step_period)

Initialize the stepper state.
- bool **stepper_set_target_position** (stepper_state_t *stepper, int target_position)

Set the target position for the stepper motor.
- bool **stepper_set_step_period** (stepper_state_t *stepper, int step_period_us)

Set the step period for the stepper motor.
- bool **stepper_stop** (stepper_state_t *stepper)

Stop the stepper motor, setting target position to current position.
- bool **stepper_get_status** (stepper_state_t *stepper)

Get the current status of the stepper motor, printing to stdout.
- bool **stepper_enable** (stepper_state_t *stepper, bool enable)

Enable the stepper motor.
- int **pico_led_init** (void)

Initialise the LED.
- void **pico_set_led** (bool led_on)

Turn the LED on or off.
- bool **process_steerer_movement** (stepper_state_t *stepper)

Process stepper movement.
- void **process_led_tick** (void)

Process LED timing tick.
- bool **timer_callback** (struct repeating_timer *)

Millisecond timer callback.
- int **process_command** (const char *cmd, stepper_state_t *stepper)

Process a command string.
- char * **process_stdin_input** (void)

Process stdin input.
- int **main** ()

Main function.

Variables

- volatile int `led_period` = LED_DELAY_MS
LED blink period in milliseconds.
- volatile int `ten_us_ticks_count` = 0
Global ten microsecond ticks count.
- volatile int `ms_ticks_count` = 0
Global millisecond ticks count.
- const char * `help_message`
Help message.

5.1.1 Detailed Description

Firmware to control a claw device with stepper motors via USB serial commands.

Author

Jon Wade

Date

19 Dec 2025

Copyright

(c) 2025 Jon Wade. Standard MIT License applies. See LICENSE file.

Command interface for controlling stepper motors and other functions associated with the claw device. This file implements a simple command interface over USB serial to control the claw device.

5.1.2 Function Documentation**5.1.2.1 main()**

int main ()

Main function.

Parameters

<input type="checkbox"/>	none
--------------------------	------

Returns

: none

5.1.2.2 pico_led_init()
Generated by Doxygen

```
int pico_led_init (
    void )
```

Parameters

<input type="checkbox"/>	none
--------------------------	------

Returns

: PICO_OK on success error code on failure

5.1.2.3 pico_set_led()

```
void pico_set_led (
    bool led_on)
```

Turn the LED on or off.

Parameters

<i>led_on</i>	true to turn on, false to turn off
---------------	------------------------------------

Returns

: none

5.1.2.4 process_command()

```
int process_command (
    const char * cmd,
    stepper_state_t * stepper)
```

Process a command string.

Parameters

<i>cmd</i>	pointer to command string
<i>stepper</i>	pointer to stepper state structure

Returns

: 0 on success, error code on failure

5.1.2.5 process_led_tick()

```
void process_led_tick (
    void )
```

Process LED timing tick.

Parameters

<input type="checkbox"/>	none
--------------------------	------

Returns

: none

5.1.2.6 process_stdin_input()

```
char * process_stdin_input (
    void )
```

Process stdin input.

Note

: This function reads characters from stdin, builds commands, and processes them when a newline is received.

This function has a simple lock to prevent re-entrancy.

Parameters

<input type="checkbox"/>	none
--------------------------	------

Returns

: none

5.1.2.7 process_stepper_movement()

```
bool process_stepper_movement (
    stepper_state_t * stepper)
```

Process stepper movement.

Parameters

<i>stepper</i>	pointer to stepper state structure
----------------	------------------------------------

Returns

: true if stepper is still moving, false if it has reached target

5.1.2.8 stepper_enable()

Generated by Doxygen

```
bool stepper_enable (
    stepper_state_t * stepper,
    bool enable)
```

Parameters

<i>stepper</i>	pointer to stepper state structure, must not be NULL
<i>enable</i>	true to enable, false to disable

Returns

: true on success, false on failure

5.1.2.9 stepper_get_status()

```
bool stepper_get_status (
    stepper_state_t * stepper)
```

Get the current status of the stepper motor, printing to stdout.

Parameters

<i>stepper</i>	pointer to stepper state structure, must not be NULL
----------------	--

Returns

: true on success, false on failure

5.1.2.10 stepper_init()

```
bool stepper_init (
    stepper_state_t * stepper,
    int initial_position,
    int step_period)
```

Initialize the stepper state.

Parameters

<i>stepper</i>	pointer to stepper state structure to initialize, must not be NULL
<i>initial_position</i>	initial position in steps must be between MIN_STEPPER_POSITION and MAX_STEPPER_POSITION
<i>step_period</i>	step period in TIMER_INTERVAL_US must be greater than 1 ms

Returns

: true on success, false on failure

5.1.2.11 stepper_set_step_period()

```
bool stepper_set_step_period (
    stepper_state_t * stepper,
    int step_period_us)
```

Parameters

<i>stepper</i>	pointer to stepper state structure, must not be NULL
<i>step_period</i>	step period in microseconds must be greater than MIN_STEPPER_PERIOD

Returns

: true on success, false on failure

5.1.2.12 stepper_set_target_position()

```
bool stepper_set_target_position (
    stepper_state_t * stepper,
    int target_position)
```

Set the target position for the stepper motor.

Parameters

<i>stepper</i>	pointer to stepper state structure, must not be NULL
<i>target_position</i>	target position in steps must be between MIN_STEPPER_POSITION and MAX_STEPPER_POSITION

Returns

: true on success, false on failure

5.1.2.13 stepper_stop()

```
bool stepper_stop (
    stepper_state_t * stepper)
```

Stop the stepper motor, setting target position to current position.

Parameters

<i>stepper</i>	pointer to stepper state structure, must not be NULL
----------------	--

Returns

: true on success, false on failure

5.1.2.14 timer_callback()

```
bool timer_callback (
    struct repeating_timer * t)
```

Generated by Doxygen

Millisecond timer callback.

Parameters

<i>t</i>	pointer to repeating_timer struct
----------	-----------------------------------

Returns

: true to keep repeating, false to stop

5.1.3 Variable Documentation

5.1.3.1 help_message

```
const char* help_message
```

Initial value:

```
=
"\n"
"Available commands:\n"
"  claw_set <position>          - Set the claw position 0 to 100\n"
"  led_period <ms>              - Set the LED blink period in milliseconds\n"
"  set_stepper_period <us>      - Set the stepper motor step period in us\n"
"  set_stepper_zero             - Set the current position to zero\n"
"  move_stepper_absolute <steps> - Move the stepper to an absolute position\n"
"  move_stepper_relative <steps> - Move the stepper by a relative number of steps\n"
"  move_stepper_rotations <rotations> - Move the stepper by a number of rotations\n"
"  move_stepper_bump_down       - Move the stepper down by a small fixed amount\n"
"  stop_stepper                - Stop the stepper motor\n"
"  get_stepper_status          - Get the current status of the stepper motor\n"
"  enable_stepper              - Enable the stepper motor\n"
"  disable_stepper             - Disable the stepper motor\n"
"  help                        - Show this help message\n"
"----\n"
```

Help message.

This message is displayed when the user requests help or enters an unknown command.

5.1.3.2 led_period

```
volatile int led_period = LED_DELAY_MS
```

LED blink period in milliseconds.

This variable can be modified via command interface to change the LED blink rate.

5.1.3.3 ms_ticks_count

```
volatile int ms_ticks_count = 0
```

Global millisecond ticks count.

This variable is incremented by the timer callback every 100 calls (1 ms = 100 * 10 us) and decremented in the main loop to track when the millisecond tasks should run.

5.1.3.4 ten_us_ticks_count

```
volatile int ten_us_ticks_count = 0
```

Global ten microsecond ticks count.

This variable is incremented by the timer callback and decremented in the main loop to track when the ten microsecond tasks should run.

Index

claw, 1
claw.c, 9
 help_message, 16
 led_period, 16
 main, 11
 ms_ticks_count, 16
 pico_led_init, 11
 pico_set_led, 12
 process_command, 12
 process_led_tick, 12
 process_stdin_input, 13
 process_stepper_movement, 13
 stepper_enable, 13
 stepper_get_status, 14
 stepper_init, 14
 stepper_set_step_period, 14
 stepper_set_target_position, 15
 stepper_stop, 15
 ten_us_ticks_count, 16
 timer_callback, 15

help_message
 claw.c, 16

led_period
 claw.c, 16

main
 claw.c, 11

ms_ticks_count
 claw.c, 16

pico_led_init
 claw.c, 11
pico_set_led
 claw.c, 12

process_command
 claw.c, 12

process_led_tick
 claw.c, 12

process_stdin_input
 claw.c, 13

process_stepper_movement
 claw.c, 13

stepper_enable
 claw.c, 13

stepper_get_status
 claw.c, 14

stepper_init
 claw.c, 14

stepper_set_step_period
 claw.c, 14

stepper_set_target_position
 claw.c, 15

stepper_state, 7

stepper_stop
 claw.c, 15

ten_us_ticks_count
 claw.c, 16

timer_callback
 claw.c, 15