Gitcon

Generated by Doxygen 1.9.5

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

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

Chapter 3

Data Structure Documentation

3.1 fft_config_t Struct Reference

Data Fields

- int **size**
- float * **input**
- float * **output**
- float * **twiddle_factors**
- fft_type_t **type**
- fft_direction_t **direction**
- unsigned int **flags**

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

- C:/Users/Smon/source/MTAP-MIDI-Guitar-Converter/firmware/MIDI-Testing/lib/fft/fft.h

3.2 gitcon_context_t Struct Reference

Gitcon Configuration.

`#include <gitcon.h>`

Collaboration diagram for gitcon_context_t:

Data Fields

- i2s_sampler_t * **sampler**
- midi_handle_t **midi_handle**
- QueueHandle_t **midi_queue**

3.2.1 Detailed Description

Gitcon Configuration.

Parameters

mcp3201	MCP3201 ADC
midi_handle	MIDI Context Handler
midi_queue	MIDI Queue

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

- C:/Users/Smon/source/MTAP-MIDI-Guitar-Converter/firmware/MIDI-Testing/src/gitcon.h

3.3 i2s_sampler_t Struct Reference

Sampler Configuration.

`#include <i2s_sampler.h>`

Data Fields

- QueueHandle_t **dma_queue**
- QueueHandle_t **dsp_queue**
- size_t * **buffer**
- size_t **buffer_pos**
- size_t **buffer_size**

3.3.1 Detailed Description

Sampler Configuration.

Parameters

dma_queue	Samples are sent to this queue by the DMA
dsp_queue	Queue to send samples to
buffer	Buffer to store samples in
buffer_pos	Current position in buffer
buffer_size	Size of the buffer in samples

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

- C:/Users/Smon/source/MTAP-MIDI-Guitar-Converter/firmware/MIDI-Testing/lib/audio/i2s_sampler.h

3.4 mcp3201_config_t Struct Reference

Data Fields

- spi_host_device_t **host**

- gpio_num_t **cs_io**
- gpio_num_t **miso_io**
- gpio_num_t **mosi_io**
- int **dma_chan**

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

- C:/Users/Smon/source/MTAP-MIDI-Guitar-Converter/firmware/MIDI-Testing/lib/mcp3201/mcp3201.h

3.5 mcp3201_context_t Struct Reference

MCP3201 Context struct for internal use.

Collaboration diagram for mcp3201_context_t:

Data Fields

- mcp3201_config_t **cfg**
- spi_device_handle_t **spi**
- spi_transaction_t * **ongoing_transaction**

3.5.1 Detailed Description

MCP3201 Context struct for internal use.

Parameters

cfg	MCP3201 Configuration
spi	SPI Device Handle
ongoing_transaction	Ongoing SPI Transaction

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

- C:/Users/Smon/source/MTAP-MIDI-Guitar-Converter/firmware/MIDI-Testing/lib/mcp3201/mcp3201.c
- C:/Users/Smon/source/MTAP-MIDI-Guitar-Converter/firmware/MIDI-Testing/lib/mcp3201/mcp3201_↩ sampler.c

3.6 mcp3201_sampler_t Struct Reference

Collaboration diagram for mcp3201_sampler_t:

Data Fields

- mcp3201_handle_t **mcp_handle**
- QueueHandle_t **dma_queue**
- QueueHandle_t **dsp_queue**
- size_t * **buffer**
- size_t **buffer_pos**
- size_t **buffer_size**

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

- C:/Users/Smon/source/MTAP-MIDI-Guitar-Converter/firmware/MIDI-Testing/lib/mcp3201/mcp3201.h

3.7 midi_config_t Struct Reference

MIDI UART Configuration.

`#include <midi.h>`

Data Fields

- uart_port_t **uart_num**
- uint **baudrate**
- gpio_num_t **rx_io**
- gpio_num_t **tx_io**

3.7.1 Detailed Description

MIDI UART Configuration.

Parameters

uart_num	UART Port
baudrate	UART Baudrate
rx_io	UART RX Pin
tx_io	UART TX Pin

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

- C:/Users/Smon/source/MTAP-MIDI-Guitar-Converter/firmware/MIDI-Testing/lib/midi/midi.h

3.8 midi_context_t Struct Reference

MIDI Context (internal! not to be accessed externally, use midi_handle_t instead)

Collaboration diagram for midi_context_t:

Data Fields

- midi_config_t **cfg**

3.8.1 Detailed Description

MIDI Context (internal! not to be accessed externally, use midi_handle_t instead)

Parameters

cfg	MIDI Config

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

- C:/Users/Smon/source/MTAP-MIDI-Guitar-Converter/firmware/MIDI-Testing/lib/midi/midi.c

3.9 midi_message_t Struct Reference

MIDI Message.

`#include <midi.h>`

Data Fields

- uint8_t **param1**
- midi_status_t **status**
- uint8_t **channel**
- uint8_t **param2**

3.9.1 Detailed Description

MIDI Message.

Parameters

status	MIDI Status Byte
channel	MIDI Channel
param1	MIDI Parameter 1
param2	MIDI Parameter 2

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

- C:/Users/Smon/source/MTAP-MIDI-Guitar-Converter/firmware/MIDI-Testing/lib/midi/midi.h

Chapter 4

File Documentation

4.1 C:/Users/Smon/source/MTAP-MIDI-Guitar-Converter/firmware/MIDI-Testing/include/config.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "freertos/timers.h"
#include "driver/gpio.h"
#include "driver/spi_master.h"
#include "driver/uart.h"
#include "driver/adc.h"
#include "driver/i2s.h"
#include "esp_adc_cal.h"
#include "esp_log.h"
#include "fft.h"
#include "i2s_sampler.h"
#include "mcp3201.h"
#include "midi.h"
```

Include dependency graph for config.h:

4.2 config.h

Go to the documentation of this file.

```
1
11 #ifndef CONFIG_H
12 #define CONFIG_H
13
14 #include <stdio.h>
15 #include <stdlib.h>
16 #include <string.h>
17 #include <math.h>
18
19 #include "freertos/FreeRTOS.h"
20 #include "freertos/task.h"
21 #include "freertos/queue.h"
22 #include "freertos/timers.h"
23
```

```
24 #include "driver/gpio.h"
25 #include "driver/spi_master.h"
26 #include "driver/uart.h"
27 #include "driver/adc.h"
28 #include "driver/i2s.h"
29 #include "esp_adc_cal.h"
30 #include "esp_log.h"
31
32 #include "fft.h"
33 #include "i2s_sampler.h"
34 #include "mcp3201.h"
35 #include "midi.h"
36
37 #define SPI_MOSI (GPIO_NUM_23)
38 #define SPI_MISO (GPIO_NUM_19)
39 #define SPI_SCLK (GPIO_NUM_18)
40 #define SPI_CS (GPIO_NUM_5)
41 #define SPI_DEV (VSPI_HOST)
42
43 #define MIDI_UART (UART_NUM_1)
44 #define MIDI_BAUD (115200)
45 #define MIDI_TX (GPIO_NUM_26)
46 #define MIDI_RX (GPIO_NUM_27)
47
48 #define DMA_CHAN 1
49 #define ADC_RES_BITS 12
50 #define ADC_RES (1 « ADC_RES_BITS)
51 #define INTERNAL_ADC_UNIT (ADC_UNIT_1)
52 #define INTERNAL_ADC_CHANNEL (ADC_CHANNEL_0)
53 #define INTERNAL_ADC_IO (GPIO_NUM_4)
54
55 #define AUDIO_BUFFER_SIZE 512 // Size of buffer for FFT and sampler
56 #define F_SAMPLE_HZ 44100     // Sample rate of FFT and sampler
57 #define FFT_WINDOW_SIZE 2     // Amount of buffers to take for FFT
58 #define FFT_SIZE 4096         //(AUDIO_BUFFER_SIZE * FFT_WINDOW_SIZE)  Amount of samples to take for FFT
59
60 // leave this commented out to use internal ADC
61 // #define USE_MCP3201
62
63 #endif // CONFIG_H
```

4.3 processed-data.h

```
1 float test_buffer[] = {5.656499999999999,5.667,5.6739999999999995,5.672499999999999,5.6655,5.6525,5.638,5.6259999999999994,5
```

4.4 i2s_sampler.h

```
1 #ifndef SAMPLING_H
2 #define SAMPLING_H
3
4 #include <stdint.h>
5 #include "driver/i2s.h"
6 #include "freertos/FreeRTOS.h"
7 #include "freertos/task.h"
8 #include "freertos/queue.h"
9 #include "freertos/timers.h"
10 #include "esp_log.h"
11
20 typedef struct
21 {
22     QueueHandle_t dma_queue;
23     QueueHandle_t dsp_queue;
24     size_t *buffer;
25     size_t buffer_pos;
26     size_t buffer_size;
27 } i2s_sampler_t;
28
38 i2s_sampler_t *i2s_sampler_start(adc_channel_t adc1_channel, QueueHandle_t recv_queue, size_t
       buffer_size, size_t f_sample);
39
45 esp_err_t i2s_sampler_stop(i2s_sampler_t *sampler);
46
47 #endif // SAMPLING_H
```

4.5 C:/Users/Smon/source/MTAP-MIDI-Guitar-Converter/firmware/MIDI-Testing/lib/audio/scale.h File Reference

Basic Functions for Frequency calculation.

```
#include "inttypes.h"
#include <math.h>
#include <stdio.h>
```

Include dependency graph for scale.h:

Macros

- #define **CONCERT_PITCH** 440.0
- #define **ADC_MAX_VAL** 255.0
- #define **A0_NUM** 49
- #define **MODAL_COUNT** 4

Enumerations

- enum **key_name_t** {
  KEY_A = 0 , **KEY_AS** = 1 , **KEY_B** = 2 , **KEY_C** = 3 ,
  KEY_CS = 4 , **KEY_D** = 5 , **KEY_DS** = 6 , **KEY_E** = 7 ,
  KEY_F = 8 , **KEY_FS** = 9 , **KEY_G** = 10 , **KEY_GS** = 11 }
- enum modal_name_t { **MAJ** = 0 , **MIN** = 1 , **HMJ** = 2 , **HMN** = 3 }

  Modal Names.

Functions

- double get_pitch_hz (uint8_t key_num)

  Convert Keynumber to Frequency.
- uint8_t get_key_num (double freq)

  Convert Frequency to Keynumber.
- void print_key_name (uint8_t key_num)

  print Keyname
- double adc_to_pitch (uint8_t adc_val, uint8_t oct_offset)

  Convert ADC Value to Frequency.
- uint8_t adc_to_num (uint8_t adc_val, uint8_t oct_offset)

  Convert ADC Value to Keynumber.
- char * get_key_name (uint8_t key_num)

  Get a key name.
- char * get_modal_name (uint8_t modal_num)

  Get a modal name.

4.5.1 Detailed Description

Basic Functions for Frequency calculation.

Author

@h-ihninger

@s-grundner

Version

0.1

Date

2022-05-05

Copyright

4.5.2 Enumeration Type Documentation

4.5.2.1 modal_name_t

```
enum modal_name_t
```

Modal Names.

Parameters

MAJ	Major
MIN	Minor
HMJ	Harmonic Major
HMN	Harmonic Minor

4.5.3 Function Documentation

4.5.3.1 adc_to_num()

```
uint8_t adc_to_num (
            uint8_t adc_val,
            uint8_t oct_offset )
```

Convert ADC Value to Keynumber.

Parameters

adc_val	ADC Value
oct_offset	Octave Offset

Returns

uint8_t Keynumber

4.5.3.2 adc_to_pitch()

```
double adc_to_pitch (
            uint8_t adc_val,
            uint8_t oct_offset )
```

Convert ADC Value to Frequency.

Parameters

adc_val	ADC Value
oct_offset	Octave Offset

Returns

double Frequency

4.5.3.3 get_key_name()

```
char * get_key_name (
            uint8_t key_num )
```

Get a key name.

Parameters

key_num	Key number

Returns

char∗ Keyname

4.5.3.4 get_key_num()

```
uint8_t get_key_num (
            double freq )
```

Convert Frequency to Keynumber.

Parameters

freq	Frequency

Returns

uint8_t Keynumber

4.5.3.5 get_modal_name()

```
char * get_modal_name (
            uint8_t modal_num )
```

Get a modal name.

Parameters

modal_num	Number of the modal

Returns

char∗

4.5.3.6 get_pitch_hz()

```
double get_pitch_hz (
            uint8_t key_num )
```

Convert Keynumber to Frequency.

Parameters

key_num	Keynumber

Returns

double Frequency

4.5.3.7 print_key_name()

```
void print_key_name (
            uint8_t key_num )
```

print Keyname

Parameters

key_num	Keynumber

4.6 scale.h

Go to the documentation of this file.

```
1
12 #ifndef SCALE_H_
13 #define SCALE_H_
14
15 #include "inttypes.h"
16 #include <math.h>
17 #include <stdio.h>
18
19 #define CONCERT_PITCH 440.0
20 #define ADC_MAX_VAL 255.0
21 #define A0_NUM 49
22
23 typedef enum
24 {
25     KEY_A = 0,
26     KEY_AS = 1,
27     KEY_B = 2,
28     KEY_C = 3,
29     KEY_CS = 4,
30     KEY_D = 5,
31     KEY_DS = 6,
32     KEY_E = 7,
33     KEY_F = 8,
34     KEY_FS = 9,
35     KEY_G = 10,
36     KEY_GS = 11,
37 } key_name_t;
38
39 #define MODAL_COUNT 4
40
48 typedef enum
49 {
50     MAJ = 0,
51     MIN = 1,
52     HMJ = 2,
53     HMN = 3,
54 } modal_name_t;
55
62 double get_pitch_hz(uint8_t key_num);
63
70 uint8_t get_key_num(double freq);
71
77 void print_key_name(uint8_t key_num);
78
86 double adc_to_pitch(uint8_t adc_val, uint8_t oct_offset);
87
95 uint8_t adc_to_num(uint8_t adc_val, uint8_t oct_offset);
96
103 char *get_key_name(uint8_t key_num);
104
111 char *get_modal_name(uint8_t modal_num);
112 #endif // SCALE_H_
```

4.7 fft.h

```
1
26 #ifndef __FFT_H__
27 #define __FFT_H__
28
29 typedef enum
30 {
31   FFT_REAL,
32   FFT_COMPLEX
33 } fft_type_t;
34
35 typedef enum
36 {
37   FFT_FORWARD,
38   FFT_BACKWARD
39 } fft_direction_t;
40
41 #define FFT_OWN_INPUT_MEM 1
42 #define FFT_OWN_OUTPUT_MEM 2
43
44 typedef struct
45 {
46   int size;                  // FFT size
47   float *input;              // pointer to input buffer
48   float *output;             // pointer to output buffer
49   float *twiddle_factors;    // pointer to buffer holding twiddle factors
50   fft_type_t type;           // real or complex
51   fft_direction_t direction; // forward or backward
52   unsigned int flags;        // FFT flags
53 } fft_config_t;
54
55 fft_config_t *fft_init(int size, fft_type_t type, fft_direction_t direction, float *input, float
       *output);
56 void fft_destroy(fft_config_t *config);
57 void fft_execute(fft_config_t *config);
58 void fft(float *input, float *output, float *twiddle_factors, int n);
59 void ifft(float *input, float *output, float *twiddle_factors, int n);
60 void rfft(float *x, float *y, float *twiddle_factors, int n);
61 void irfft(float *x, float *y, float *twiddle_factors, int n);
62 void fft_primitive(float *x, float *y, int n, int stride, float *twiddle_factors, int tw_stride);
63 void split_radix_fft(float *x, float *y, int n, int stride, float *twiddle_factors, int tw_stride);
64 void ifft_primitive(float *input, float *output, int n, int stride, float *twiddle_factors, int
       tw_stride);
65 void fft8(float *input, int stride_in, float *output, int stride_out);
66 void fft4(float *input, int stride_in, float *output, int stride_out);
67
68 #endif // __FFT_H__
```

4.8 C:/Users/Smon/source/MTAP-MIDI-Guitar-Converter/firmware/MIDI-Testing/lib/mcp3201/mcp3201.c File Reference

`#include "mcp3201.h"`
Include dependency graph for mcp3201.c:

Data Structures

- struct mcp3201_context_t

  MCP3201 Context struct for internal use.

Typedefs

- typedef struct mcp3201_context_t **mcp3201_context_t**

Functions

- esp_err_t **mcp3201_init** (mcp3201_context_t **out_ctx, const mcp3201_config_t *cfg)
- esp_err_t **mcp3201_read** (mcp3201_context_t *ctx, uint16_t *out_value)
- esp_err_t mcp3201_exit (mcp3201_handle_t mcp_handle)

  Exits the MCP3201 ADC device and frees all resources.

4.8.1 Detailed Description

Author

@s-grundner

Version

0.1

Date

2022-12-24

Copyright

4.8.2 Function Documentation

4.8.2.1 mcp3201_exit()

```
esp_err_t mcp3201_exit (
            mcp3201_handle_t mcp_handle )
```

Exits the MCP3201 ADC device and frees all resources.

Parameters

mcp_handle	MCP3201 Device to exit

Returns

ESP_OK on success

Here is the caller graph for this function:

4.9 C:/Users/Smon/source/MTAP-MIDI-Guitar-Converter/firmware/MIDI-Testing/lib/mcp3201/mcp3201.h File Reference

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
```

```
#include "driver/gpio.h"
#include "driver/spi_master.h"
#include "hal/spi_types.h"
#include "esp_log.h"
```

Include dependency graph for mcp3201.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct mcp3201_config_t
- struct mcp3201_sampler_t

Macros

- #define **ADC_CLK** SPI_MASTER_FREQ_8M

Typedefs

- typedef struct mcp3201_context_t * **mcp3201_handle_t**

Functions

- esp_err_t mcp3201_init (mcp3201_handle_t *out_handle, const mcp3201_config_t *cfg)

  Initializes the MCP3201 ADC device.
- esp_err_t mcp3201_exit (mcp3201_handle_t mcp_handle)

  Exits the MCP3201 ADC device and frees all resources.
- esp_err_t mcp3201_read (mcp3201_handle_t handle, uint16_t *out_value)

  Reads a single value from the MCP3201 ADC.
- mcp3201_sampler_t * mcp3201_sampler_start (mcp3201_handle_t mcp_handle, QueueHandle_t recv_queue, const size_t buffer_size, const size_t f_sample)

  Starts the MCP3201 Sampler, which samples continuously and puts the samples into a queue.
- void mcp3201_sampler_stop (mcp3201_sampler_t *sampler)

  Stops the MCP3201 Sampler.

4.9.1 Detailed Description

Author

@s-grundner

Version

0.1

Date

2022-12-24

Copyright

4.9.2 Function Documentation

4.9.2.1 mcp3201_exit()

```
esp_err_t mcp3201_exit (
            mcp3201_handle_t mcp_handle )
```

Exits the MCP3201 ADC device and frees all resources.

Parameters

mcp_handle	MCP3201 Device to exit

Returns

ESP_OK on success

Here is the caller graph for this function:

4.9.2.2 mcp3201_init()

```
esp_err_t mcp3201_init (
            mcp3201_handle_t * out_handle,
            const mcp3201_config_t * cfg )
```

Initializes the MCP3201 ADC device.

Parameters

out_handle	MCP3201 Handler to store initialization data
cfg	MCP3201 Configuration

Returns

4.9.2.3 mcp3201_read()

```
esp_err_t mcp3201_read (
            mcp3201_handle_t handle,
            uint16_t * out_value )
```

Reads a single value from the MCP3201 ADC.

Parameters

	handle	MCP3201 Device
out	*out_value*	Value

Returns

ESP_OK on success

4.9.2.4 mcp3201_sampler_start()

```
mcp3201_sampler_t * mcp3201_sampler_start (
            mcp3201_handle_t mcp_handle,
            QueueHandle_t recv_queue,
            const size_t buffer_size,
            const size_t f_sample )
```

Starts the MCP3201 Sampler, which samples continuously and puts the samples into a queue.

Parameters

mcp_handle	MCP3201 Device Handler
recv_queue	queue to send the samples into
buffer_size	size of the audio buffer
f_sample	Sample rate

Returns

mcp3201_sampler_t* Sampler Handler

Here is the call graph for this function: Here is the caller graph for this function:

4.9.2.5 mcp3201_sampler_stop()

```
void mcp3201_sampler_stop (
            mcp3201_sampler_t * sampler )
```

Stops the MCP3201 Sampler.

Parameters

sampler	Sampler Handler

Here is the call graph for this function: Here is the caller graph for this function:

4.10 mcp3201.h

Go to the documentation of this file.

```
1
12 #ifndef MCP3201_H
13 #define MCP3201_H
14
15 #include <stdio.h>
16 #include "freertos/FreeRTOS.h"
17 #include "freertos/task.h"
18 #include "freertos/queue.h"
19 #include "driver/gpio.h"
20 #include "driver/spi_master.h"
21 #include "hal/spi_types.h"
22 #include "esp_log.h"
23
24 #define ADC_CLK SPI_MASTER_FREQ_8M
25
36 typedef struct
37 {
38     spi_host_device_t host;
39     gpio_num_t cs_io;
40     gpio_num_t miso_io;
41     gpio_num_t mosi_io;
42     int dma_chan;
43 } mcp3201_config_t;
44
46 typedef struct mcp3201_context_t *mcp3201_handle_t;
47
58 typedef struct
59 {
60     mcp3201_handle_t mcp_handle;
61     QueueHandle_t dma_queue;
62     QueueHandle_t dsp_queue;
63     size_t *buffer;
64     size_t buffer_pos;
65     size_t buffer_size;
66 } mcp3201_sampler_t;
67
75 esp_err_t mcp3201_init(mcp3201_handle_t *out_handle, const mcp3201_config_t *cfg);
76
83 esp_err_t mcp3201_exit(mcp3201_handle_t mcp_handle);
84
92 esp_err_t mcp3201_read(mcp3201_handle_t handle, uint16_t *out_value);
93
103 mcp3201_sampler_t *mcp3201_sampler_start(mcp3201_handle_t mcp_handle, QueueHandle_t recv_queue, const
        size_t buffer_size, const size_t f_sample);
104
110 void mcp3201_sampler_stop(mcp3201_sampler_t *sampler);
111
112 #endif // MCP3201_H
```

4.11 C:/Users/Smon/source/MTAP-MIDI-Guitar-Converter/firmware/MIDI-Testing/lib/midi/midi.c File Reference

MIDI driver for ESP32.

`#include "midi.h"`
Include dependency graph for midi.c:

Data Structures

- struct midi_context_t

  MIDI Context (internal! not to be accessed externally, use midi_handle_t instead)

Macros

- #define **MIDI_BYTE_SIZE_DEFAULT** 3
- #define **MIDI_BYTE_SIZE_SHORT** 2

Typedefs

- typedef struct midi_context_t **midi_context_t**

Functions

- esp_err_t **midi_init** (midi_context_t **out_ctx, midi_config_t *out_cfg)
- esp_err_t midi_exit (midi_handle_t midi_handle)

  Exits MIDI and frees all resources.
- esp_err_t midi_write (midi_handle_t handle, midi_message_t *msg)

  Writes MIDI Message to UART.
- esp_err_t midi_read (midi_handle_t midi_handle, midi_message_t *msg, TickType_t timeout)

  Reads MIDI Message from UART.

4.11.1 Detailed Description

MIDI driver for ESP32.

Author

@s-grundner

Version

0.1

Date

2022-12-23

Copyright

4.11.2 Function Documentation

4.11.2.1 midi_exit()

```
esp_err_t midi_exit (
            midi_handle_t midi_handle )
```

Exits MIDI and frees all resources.

Parameters

midi_handle	MIDI Handle to be freed

Returns

esp_err_t

Here is the caller graph for this function:

4.11.2.2 midi_read()

```
esp_err_t midi_read (
            midi_handle_t midi_handle,
            midi_message_t * msg,
            TickType_t timeout )
```

Reads MIDI Message from UART.

Parameters

midi_handle	MIDI Handle to pass parameters
msg	MIDI Message to be read

Returns

esp_err_t

4.11.2.3 midi_write()

```
esp_err_t midi_write (
            midi_handle_t midi_handle,
            midi_message_t * msg )
```

Writes MIDI Message to UART.

Parameters

midi_handle	MIDI Handle to pass parameters
msg	MIDI Message to be sent

Returns

esp_err_t

4.12 C:/Users/Smon/source/MTAP-MIDI-Guitar-Converter/firmware/MIDI-Testing/lib/midi/midi.h File Reference

midi driver for esp32

```
#include <stdio.h>
#include <stdint.h>
#include "driver/gpio.h"
#include "driver/uart.h"
#include "esp_log.h"
```

Include dependency graph for midi.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct midi_message_t

  MIDI Message.
- struct midi_config_t

  MIDI UART Configuration.

Macros

- #define **MIDI_LOG_LEVEL** ESP_LOG_ERROR
- #define **MIDI_BYTE_SIZE_DEFAULT** 3
- #define **MIDI_BYTE_SIZE_SHORT** 2
- #define **MIDI_PITCH_BEND_MIN** (0)
- #define **MIDI_PITCH_BEND_MAX** (16383)
- #define **MIDI_PITCH_BEND_CENTER** (8192)

Typedefs

- typedef struct midi_context_t * **midi_handle_t**

Enumerations

- enum midi_status_t {
  MIDI_STATUS_NOTE_OFF = 0x80 , **MIDI_STATUS_NOTE_ON** = 0x90 , **MIDI_STATUS_POLYPHONIC_↩KEY_PRESSURE** = 0xA0 , **MIDI_STATUS_CONTROL_CHANGE** = 0xB0 ,
  MIDI_STATUS_PROGRAM_CHANGE = 0xC0 , **MIDI_STATUS_CHANNEL_PRESSURE** = 0xD0 , **MIDI_↩STATUS_PITCH_BEND** = 0xE0 }

  MIDI Status Bytes.

Functions

- esp_err_t midi_init (midi_handle_t *out_handle, midi_config_t *out_cfg)

  initializes MIDI and allocates driver resources
- esp_err_t midi_exit (midi_handle_t midi_handle)

  Exits MIDI and frees all resources.
- esp_err_t midi_write (midi_handle_t midi_handle, midi_message_t *msg)

  Writes MIDI Message to UART.
- esp_err_t midi_read (midi_handle_t midi_handle, midi_message_t *msg, TickType_t timeout)

  Reads MIDI Message from UART.
- midi_message_t **note_off** (uint8_t channel, uint8_t key_num, uint8_t velocity)
- midi_message_t **note_on** (uint8_t channel, uint8_t key_num, uint8_t velocity)
- midi_message_t **poly_key_pressure** (uint8_t channel, uint8_t key_num, uint8_t value)
- midi_message_t **ctrl_change** (uint8_t channel, uint8_t controller_num, uint8_t value)
- midi_message_t **prg_change** (uint8_t channel, uint8_t program)
- midi_message_t **channel_pressure** (uint8_t channel, uint8_t value)
- midi_message_t **pitch_bend** (uint8_t channel, uint16_t value)

4.12.1 Detailed Description

midi driver for esp32

Author

@s-grundner

Version

0.1

Date

2022-12-23

Copyright

4.12.2 Enumeration Type Documentation

4.12.2.1 midi_status_t

```
enum midi_status_t
```

MIDI Status Bytes.

Parameters

MIDI_STATUS_NOTE_OFF	0x80, requires param2
MIDI_STATUS_NOTE_ON	0x90, requires param2
MIDI_STATUS_POLYPHONIC_KEY_PRESSURE	0xA0, param2 is not used
MIDI_STATUS_CONTROL_CHANGE	0xB0, requires param2
MIDI_STATUS_PROGRAM_CHANGE	0xC0, param2 is is not used
MIDI_STATUS_CHANNEL_PRESSURE	0xD0, param2 is is not used
MIDI_STATUS_PITCH_BEND	0xE0, requires param2

4.12.3 Function Documentation

4.12.3.1 midi_exit()

```
esp_err_t midi_exit (
            midi_handle_t midi_handle )
```

Exits MIDI and frees all resources.

Parameters

midi_handle	MIDI Handle to be freed

Returns

esp_err_t

Here is the caller graph for this function:

4.12.3.2 midi_init()

```
esp_err_t midi_init (
            midi_handle_t * out_handle,
            midi_config_t * out_cfg )
```

initializes MIDI and allocates driver resources

Parameters

out	*out_handle*	MIDI Handle to be initialized
out	*out_cfg*	MIDI Configuration

Returns

esp_err_t

4.12.3.3 midi_read()

```
esp_err_t midi_read (
            midi_handle_t midi_handle,
            midi_message_t * msg,
            TickType_t timeout )
```

Reads MIDI Message from UART.

Parameters

midi_handle	MIDI Handle to pass parameters
msg	MIDI Message to be read

Returns

esp_err_t

4.12.3.4 midi_write()

```
esp_err_t midi_write (
            midi_handle_t midi_handle,
            midi_message_t * msg )
```

Writes MIDI Message to UART.

Parameters

midi_handle	MIDI Handle to pass parameters
msg	MIDI Message to be sent

Returns

esp_err_t

4.13 midi.h

Go to the documentation of this file.

```
1
12 #ifndef MIDI_H
```

```
13 #define MIDI_H
14
15 #include <stdio.h>
16 #include <stdint.h>
17 #include "driver/gpio.h"
18 #include "driver/uart.h"
19 #include "esp_log.h"
20
21 #define MIDI_LOG_LEVEL ESP_LOG_ERROR
22
23 #define MIDI_BYTE_SIZE_DEFAULT 3
24 #define MIDI_BYTE_SIZE_SHORT 2
25
26 #define MIDI_PITCH_BEND_MIN (0)
27 #define MIDI_PITCH_BEND_MAX (16383)
28 #define MIDI_PITCH_BEND_CENTER (8192)
29
41 typedef enum
42 {
43     MIDI_STATUS_NOTE_OFF = 0x80,
44     MIDI_STATUS_NOTE_ON = 0x90,
45     MIDI_STATUS_POLYPHONIC_KEY_PRESSURE = 0xA0,
46     MIDI_STATUS_CONTROL_CHANGE = 0xB0,
47     MIDI_STATUS_PROGRAM_CHANGE = 0xC0,
48     MIDI_STATUS_CHANNEL_PRESSURE = 0xD0,
49     MIDI_STATUS_PITCH_BEND = 0xE0,
50 } midi_status_t;
51
60 typedef struct
61 {
62     uint8_t param1;
63     midi_status_t status;
64     uint8_t channel;
65     uint8_t param2;
66 } midi_message_t;
67
76 typedef struct
77 {
78     uart_port_t uart_num;
79     uint baudrate;
80     gpio_num_t rx_io;
81     gpio_num_t tx_io;
82 } midi_config_t;
83
85 typedef struct midi_context_t *midi_handle_t;
86
94 esp_err_t midi_init(midi_handle_t *out_handle, midi_config_t *out_cfg);
95
102 esp_err_t midi_exit(midi_handle_t midi_handle);
103
111 esp_err_t midi_write(midi_handle_t midi_handle, midi_message_t *msg);
112
120 esp_err_t midi_read(midi_handle_t midi_handle, midi_message_t *msg, TickType_t timeout);
121
122 // functions to configure midi messages
123 midi_message_t note_off(uint8_t channel, uint8_t key_num, uint8_t velocity);
124 midi_message_t note_on(uint8_t channel, uint8_t key_num, uint8_t velocity);
125 midi_message_t poly_key_pressure(uint8_t channel, uint8_t key_num, uint8_t value);
126 midi_message_t ctrl_change(uint8_t channel, uint8_t controller_num, uint8_t value);
127 midi_message_t prg_change(uint8_t channel, uint8_t program);
128 midi_message_t channel_pressure(uint8_t channel, uint8_t value);
129 midi_message_t pitch_bend(uint8_t channel, uint16_t value);
130
131 #endif // MIDI_H
```

4.14 C:/Users/Smon/source/MTAP-MIDI-Guitar-Converter/firmware/MIDI-Testing/src/gitcon.c File Reference

```
#include "gitcon.h"
#include "processed-data.h"
```

Include dependency graph for gitcon.c:

Functions

- esp_err_t **gitcon_init** (gitcon_context_t **out_handle)
- esp_err_t gitcon_exit (gitcon_handle_t handle)

  frees all resources

4.14.1 Detailed Description

Author

@s-grundner @Laurenz03

Version

0.1

Date

2022-12-23

Copyright

Copyright (c) 2022

4.14.2 Function Documentation

4.14.2.1 gitcon_exit()

```
esp_err_t gitcon_exit (
            gitcon_handle_t handle )
```

frees all resources

Parameters

handle	gitcon context handler

Returns

ESP_OK on success

Here is the call graph for this function:

4.15 C:/Users/Smon/source/MTAP-MIDI-Guitar-Converter/firmware/MIDI-Testing/src/gitcon.h File Reference

```
#include "config.h"
```

Include dependency graph for gitcon.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct gitcon_context_t
  Gitcon Configuration.

Macros

- #define **GITCON_LOG_LEVEL** ESP_LOG_ERROR

Typedefs

- typedef gitcon_context_t * **gitcon_handle_t**

Functions

- esp_err_t gitcon_init (gitcon_handle_t *out_handle)
  initializes gitcon device
- esp_err_t gitcon_exit (gitcon_handle_t handle)
  frees all resources

4.15.1 Detailed Description

Author

@s-grundner @Laurenz03

Version

0.1

Date

2022-12-23

Copyright

Copyright (c) 2022

4.15.2 Function Documentation

4.15.2.1 gitcon_exit()

```
esp_err_t gitcon_exit (
            gitcon_handle_t handle )
```

frees all resources

Parameters

handle	gitcon context handler

Returns

ESP_OK on success

Here is the call graph for this function:

4.15.2.2 gitcon_init()

```
esp_err_t gitcon_init (
            gitcon_handle_t * out_handle )
```

initializes gitcon device

Parameters

out	*out_handle*	gitcon context handler

Returns

esp_err_t ESP_OK on success, ESP_ERR_NO_MEM on memory allocation error

4.16 gitcon.h

Go to the documentation of this file.

```
1
12 #ifndef GITCON_H
13 #define GITCON_H
14
15 #include "config.h"
16
17 #define GITCON_LOG_LEVEL ESP_LOG_ERROR
18
25 typedef struct
26 {
27 #ifdef USE_MCP3201
28   mcp3201_sampler_t *sampler;
29 #else
30   i2s_sampler_t *sampler;
31 #endif
32   midi_handle_t midi_handle;
33   QueueHandle_t midi_queue; // TODO: merge with midi_handle
34 } gitcon_context_t;
35 typedef gitcon_context_t *gitcon_handle_t;
36
43 esp_err_t gitcon_init(gitcon_handle_t *out_handle);
44
51 esp_err_t gitcon_exit(gitcon_handle_t handle);
52
53 #endif // GITCON_H
```

4.17 C:/Users/Smon/source/MTAP-MIDI-Guitar-Converter/firmware/MIDI-Testing/src/main.c File Reference

main file for gitcon project

```
#include "gitcon.h"
```
Include dependency graph for main.c:

Macros

- #define **USER_LOCAL_LEVEL** ESP_LOG_ERROR
- #define **PROTOTYPE** 0

Functions

- void **app_main** (void)

4.17.1 Detailed Description

main file for gitcon project

Author

@s-grundner

Version

0.1

Date

2022-12-23

Copyright

Index