Ein Bild, das Logo enthält.

Automatisch generierte Beschreibung

Gitcon Driver V1.0

Firmware Reference Manual - [EN]

Generated with Doxygen 1.9.5

Author Simon Grundner

[1 Data Structure Index 6](#_Toc130886401)

[1.1 Data Structures 6](#_Toc130886402)

[2 File Index 7](#_Toc130886403)

[2.1 File List 7](#_Toc130886404)

[3 Data Structure Documentation 8](#_Toc130886405)

[3.1 gitcon\_context\_t Struct Reference 8](#_Toc130886406)

[3.1.1 Detailed Description 8](#_Toc130886407)

[3.1.2 Field Documentation 8](#_Toc130886408)

[3.1.2.1 midi\_handle 8](#_Toc130886409)

[3.1.2.2 midi\_queue 8](#_Toc130886410)

[3.1.2.3 sampler 9](#_Toc130886411)

[3.2 i2s\_sampler\_t Struct Reference 9](#_Toc130886412)

[3.2.1 Detailed Description 9](#_Toc130886413)

[3.2.2 Field Documentation 9](#_Toc130886414)

[3.2.2.1 buffer 9](#_Toc130886415)

[3.2.2.2 buffer\_pos 10](#_Toc130886416)

[3.2.2.3 buffer\_size 10](#_Toc130886417)

[3.2.2.4 dma\_queue 10](#_Toc130886418)

[3.2.2.5 dsp\_queue 10](#_Toc130886419)

[3.3 midi\_config\_t Struct Reference 11](#_Toc130886420)

[3.3.1 Detailed Description 11](#_Toc130886421)

[3.3.2 Field Documentation 11](#_Toc130886422)

[3.3.2.1 baudrate 11](#_Toc130886423)

[3.3.2.2 rx\_io 11](#_Toc130886424)

[3.3.2.3 tx\_io 11](#_Toc130886425)

[3.3.2.4 uart\_num 11](#_Toc130886426)

[3.4 midi\_context\_t Struct Reference 12](#_Toc130886427)

[3.4.1 Detailed Description 12](#_Toc130886428)

[3.4.2 Field Documentation 12](#_Toc130886429)

[3.4.2.1 cfg 12](#_Toc130886430)

[3.5 midi\_message\_t Struct Reference 12](#_Toc130886431)

[3.5.1 Detailed Description 12](#_Toc130886432)

[3.5.2 Field Documentation 13](#_Toc130886433)

[3.5.2.1 channel 13](#_Toc130886434)

[3.5.2.2 param1 13](#_Toc130886435)

[3.5.2.3 param2 13](#_Toc130886436)

[3.5.2.4 status 13](#_Toc130886437)

[4 Structural Diagram 14](#_Toc130886438)

[5 File Documentation 15](#_Toc130886439)

[5.1 config.h File Reference 15](#_Toc130886440)

[5.1.1 Detailed Description 16](#_Toc130886441)

[5.1.2 Macro Definition Documentation 16](#_Toc130886442)

[5.1.2.1 ADC\_RES 16](#_Toc130886443)

[5.1.2.2 ADC\_RES\_BITS 16](#_Toc130886444)

[5.1.2.3 AUDIO\_BUFFER\_SIZE 16](#_Toc130886445)

[5.1.2.4 DMA\_CHAN 16](#_Toc130886446)

[5.1.2.5 F\_SAMPLE\_HZ 16](#_Toc130886447)

[5.1.2.6 FFT\_SIZE 16](#_Toc130886448)

[5.1.2.7 FFT\_WINDOW\_SIZE 17](#_Toc130886449)

[5.1.2.8 INTERNAL\_ADC\_CHANNEL 17](#_Toc130886450)

[5.1.2.9 INTERNAL\_ADC\_IO 17](#_Toc130886451)

[5.1.2.10 INTERNAL\_ADC\_UNIT 17](#_Toc130886452)

[5.1.2.11 MIDI\_BAUD 17](#_Toc130886453)

[5.1.2.12 MIDI\_RX 17](#_Toc130886454)

[5.1.2.13 MIDI\_TX 17](#_Toc130886455)

[5.1.2.14 MIDI\_UART 17](#_Toc130886456)

[5.1.2.15 SPI\_CS 17](#_Toc130886457)

[5.1.2.16 SPI\_DEV 18](#_Toc130886458)

[5.1.2.17 SPI\_MISO 18](#_Toc130886459)

[5.1.2.18 SPI\_MOSI 18](#_Toc130886460)

[5.1.2.19 SPI\_SCLK 18](#_Toc130886461)

[5.2 config.h 19](#_Toc130886462)

[5.3 i2s\_sampler.c File Reference 20](#_Toc130886463)

[5.3.1 Detailed Description 20](#_Toc130886464)

[5.3.2 Macro Definition Documentation 20](#_Toc130886465)

[5.3.2.1 READER\_TIMEOUT\_MS 20](#_Toc130886466)

[5.3.2.2 READER\_TIMEOUT\_TICKS 20](#_Toc130886467)

[5.3.2.3 RESAMPLE\_DENOMINATOR 21](#_Toc130886468)

[5.3.3 Function Documentation 21](#_Toc130886469)

[5.3.3.1 *static* sampler\_task() 21](#_Toc130886470)

[5.3.3.2 i2s\_sampler\_start() 21](#_Toc130886471)

[5.3.3.3 i2s\_sampler\_stop() 22](#_Toc130886472)

[5.4 i2s\_sampler.c 23](#_Toc130886473)

[5.5 i2s\_sampler.h File Reference 26](#_Toc130886474)

[5.5.1 Detailed Description 26](#_Toc130886475)

[5.5.2 Function Documentation 26](#_Toc130886476)

[5.5.2.1 i2s\_sampler\_start() 26](#_Toc130886477)

[5.5.2.2 i2s\_sampler\_stop() 27](#_Toc130886478)

[5.6 i2s\_sampler.h 28](#_Toc130886479)

[5.7 midi.c File Reference 29](#_Toc130886480)

[5.7.1 Detailed Description 29](#_Toc130886481)

[5.7.2 Macro Definition Documentation 29](#_Toc130886482)

[5.7.2.1 MIDI\_BYTE\_SIZE\_DEFAULT 29](#_Toc130886483)

[5.7.2.2 MIDI\_BYTE\_SIZE\_SHORT 30](#_Toc130886484)

[5.7.3 Typedef Documentation 30](#_Toc130886485)

[5.7.3.1 midi\_context\_t 30](#_Toc130886486)

[5.7.4 Function Documentation 30](#_Toc130886487)

[5.7.4.1 midi\_exit() 30](#_Toc130886488)

[5.7.4.2 midi\_init() 30](#_Toc130886489)

[5.7.4.3 midi\_read() 31](#_Toc130886490)

[5.7.4.4 midi\_write() 31](#_Toc130886491)

[5.8 midi.c 32](#_Toc130886492)

[5.9 midi.h File Reference 36](#_Toc130886493)

[5.9.1 Detailed Description 36](#_Toc130886494)

[5.9.2 Macro Definition Documentation 36](#_Toc130886495)

[5.9.2.1 MIDI\_BYTE\_SIZE\_DEFAULT 36](#_Toc130886496)

[5.9.2.2 MIDI\_BYTE\_SIZE\_SHORT 36](#_Toc130886497)

[5.9.2.3 MIDI\_LOG\_LEVEL 36](#_Toc130886498)

[5.9.2.4 MIDI\_PITCH\_BEND\_CENTER 36](#_Toc130886499)

[5.9.2.5 MIDI\_PITCH\_BEND\_MAX 36](#_Toc130886500)

[5.9.2.6 MIDI\_PITCH\_BEND\_MIN 36](#_Toc130886501)

[5.9.3 Typedef Documentation 36](#_Toc130886502)

[5.9.3.1 midi\_handle\_t 36](#_Toc130886503)

[5.9.4 Enumeration Type Documentation 36](#_Toc130886504)

[5.9.4.1 midi\_status\_t 36](#_Toc130886505)

[5.9.5 Function Documentation 36](#_Toc130886506)

[5.9.5.1 channel\_pressure() 36](#_Toc130886507)

[5.9.5.2 ctrl\_change() 36](#_Toc130886508)

[5.9.5.3 midi\_exit() 36](#_Toc130886509)

[5.9.5.4 midi\_init() 36](#_Toc130886510)

[5.9.5.5 midi\_read() 36](#_Toc130886511)

[5.9.5.6 midi\_write() 36](#_Toc130886512)

[5.9.5.7 note\_off() 36](#_Toc130886513)

[5.9.5.8 note\_on() 36](#_Toc130886514)

[5.9.5.9 pitch\_bend() 36](#_Toc130886515)

[5.9.5.10 poly\_key\_pressure() 36](#_Toc130886516)

[5.9.5.11 prg\_change() 36](#_Toc130886517)

[5.10 midi.h 37](#_Toc130886518)

[5.11 midi\_utils.c File Reference 39](#_Toc130886519)

[5.11.1 Detailed Description 39](#_Toc130886520)

[5.11.2 Function Documentation 39](#_Toc130886521)

[5.11.2.1 channel\_pressure() 39](#_Toc130886522)

[5.11.2.2 ctrl\_change() 39](#_Toc130886523)

[5.11.2.3 note\_off() 39](#_Toc130886524)

[5.11.2.4 note\_on() 39](#_Toc130886525)

[5.11.2.5 pitch\_bend() 39](#_Toc130886526)

[5.11.2.6 poly\_key\_pressure() 39](#_Toc130886527)

[5.11.2.7 prg\_change() 39](#_Toc130886528)

[5.12 midi\_utils.c 40](#_Toc130886529)

[5.13 gitcon.c File Reference 42](#_Toc130886530)

[5.13.1 Detailed Description 42](#_Toc130886531)

[5.13.2 Macro Definition Documentation 42](#_Toc130886532)

[5.13.2.1 CONCERT\_A 42](#_Toc130886533)

[5.13.2.2 CONCERT\_A\_NOTE 42](#_Toc130886534)

[5.13.2.3 FLOAT\_TO\_UINT16 42](#_Toc130886535)

[5.13.2.4 MIDI\_HIGHEST\_NOTE 42](#_Toc130886536)

[5.13.2.5 MIDI\_KEY\_BOUNDARY 42](#_Toc130886537)

[5.13.2.6 MIDI\_LOWEST\_NOTE 42](#_Toc130886538)

[5.13.2.7 SENSITIVITY 42](#_Toc130886539)

[5.13.2.8 UINT16\_TO\_FLOAT 42](#_Toc130886540)

[5.13.3 Function Documentation 42](#_Toc130886541)

[5.13.3.1 *static* dsp\_task() 42](#_Toc130886542)

[5.13.3.2 *static* midi\_task() 42](#_Toc130886543)

[5.13.3.3 gitcon\_exit() 42](#_Toc130886544)

[5.13.3.4 gitcon\_init() 42](#_Toc130886545)

[5.14 gitcon.c 43](#_Toc130886546)

[5.15 gitcon.h File Reference 49](#_Toc130886547)

[5.15.1 Detailed Description 49](#_Toc130886548)

[5.15.2 Macro Definition Documentation 49](#_Toc130886549)

[5.15.2.1 GITCON\_LOG\_LEVEL 49](#_Toc130886550)

[5.15.3 Typedef Documentation 49](#_Toc130886551)

[5.15.3.1 gitcon\_handle\_t 49](#_Toc130886552)

[5.15.4 Function Documentation 49](#_Toc130886553)

[5.15.4.1 gitcon\_exit() 49](#_Toc130886554)

[5.15.4.2 gitcon\_init() 49](#_Toc130886555)

[5.16 gitcon.h 50](#_Toc130886556)

[5.17 main.c File Reference 51](#_Toc130886557)

[5.18 main.c 52](#_Toc130886558)

[5.19 test\_fft.c File Reference 54](#_Toc130886559)

[5.20 test\_fft.c 54](#_Toc130886560)

[5.21 test\_midi.c File Reference 54](#_Toc130886561)

[5.22 test\_midi.c 54](#_Toc130886562)

[6 Index 55](#_Toc130886563)

# Data Structure Index

## Data Structures

Here are the data structures with brief descriptions:

gitcon\_context\_t

Driver passing parameters 6

i2s\_sampler\_t

Sampler Data 7

midi\_config\_t

MIDI Bus configuration 9

midi\_context\_t

MIDI configuration wrapper 10

midi\_message\_t

MIDI message data 10

# File Index

## File List

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

MTAP-MIDI-Guitar-Converter/firmware/Gitcon\_Driver/include/config.h

Gitcon Driver Configuration and Pin-Out File and dependencies 13

MTAP-MIDI-Guitar-Converter/firmware/Gitcon\_Driver/lib/audio/i2s\_sampler.c

I2S Sampler Driver Source for ESP32 18

MTAP-MIDI-Guitar-Converter/firmware/Gitcon\_Driver/lib/audio/i2s\_sampler.h

I2S Sampler Driver for ESP32 22

MTAP-MIDI-Guitar-Converter/firmware/Gitcon\_Driver/lib/midi/midi.c

MIDI Driver Source for ESP32 22

MTAP-MIDI-Guitar-Converter/firmware/Gitcon\_Driver/lib/midi/midi.h

MIDI UART Driver for ESP32 22

MTAP-MIDI-Guitar-Converter/firmware/Gitcon\_Driver/lib/midi/midi\_utils.c

MIDI Message Utilities 22

MTAP-MIDI-Guitar-Converter/firmware/Gitcon\_Driver/src/gitcon.c

Gitcon Driver Source 22

MTAP-MIDI-Guitar-Converter/firmware/Gitcon\_Driver/src/gitcon.h

Gitcon Driver 22

MTAP-MIDI-Guitar-Converter/firmware/Gitcon\_Driver/src/main.c

Main File for Gitcon Project 22

MTAP-MIDI-Guitar-Converter/firmware/Gitcon\_Driver/test/test\_fft/test\_fft.c

Unit Test for FFT 22

MTAP-MIDI-Guitar-Converter/firmware/Gitcon\_Driver/test/test\_midi/test\_midi.c

Unit Test for MIDI Driver 22

# Data Structure Documentation

## gitcon\_context\_t Struct Reference

Driver passing parameters.

#include <gitcon.h>

Data Fields

* i2s\_sampler\_t \*sampler
* midi\_handle\_t midi\_handle
* QueueHandle\_t midi\_queue

### Detailed Description

This data structure is passed around to all tasks to reconfigure driver parameters as needed.

Parameters

|  |  |
| --- | --- |
| *sampler* | I2S Sampler Handler |
| *midi\_handle* | MIDI Driver Context (MIDI over UART) |
| *midi\_queue* | MIDI Queue Handler |

* Definition at line 25 of file gitcon.h.

### Field Documentation

#### midi\_handle

midi\_handle\_t midi\_handle

MIDI driver context containing UART bus- and pin configuration.

* Definition at line 32 of file gitcon.h.

#### midi\_queue

QueueHandle\_t midi\_queue

MIDI-Messages are queued here to be written onto the UART bus by the MIDI driver.

* Definition at line 33 of file gitcon.h.

#### sampler

i2s\_sampler\_t\* sampler

Audio sampling parameters

* Definition at line 30 of file gitcon.h.

## i2s\_sampler\_t Struct Reference

Sampler Data.

#include <i2s\_sampler.h>

Data Fields

* QueueHandle\_t dma\_queue
* QueueHandle\_t dma\_queue
* QueueHandle\_t dsp\_queue
* size\_t ∗ buffer
* size\_t buffer\_pos
* size\_t buffer\_size

### Detailed Description

This struct contains the relevant data to sample audio correctly.

Parameters

|  |  |
| --- | --- |
| *dma\_queue* | Samples are sent to this queue by the DMA |
| *dsp\_queue* | Sampling result is sent to this queue |
| *buffer* | Buffer to store samples in |
| *buffer\_pos* | Current position in buffer |
| *buffer\_size* | Size of the buffer in samples |

* Definition at line 20 of file i2s\_sampler.h.

### Field Documentation

#### buffer

size\_t∗ buffer

The buffer stores sampled audio data with a size of 16 bit per sample due to the ADC resolution of 12 bit.

* Definition at line 24 of file i2s\_sampler.h.

#### buffer\_pos

size\_t buffer\_pos

To append incoming data correctly, a buffer position variable stores the index of where the DMA Controller left off.

* Definition at line 25 of file i2s\_sampler.h.

#### buffer\_size

size\_t buffer\_size

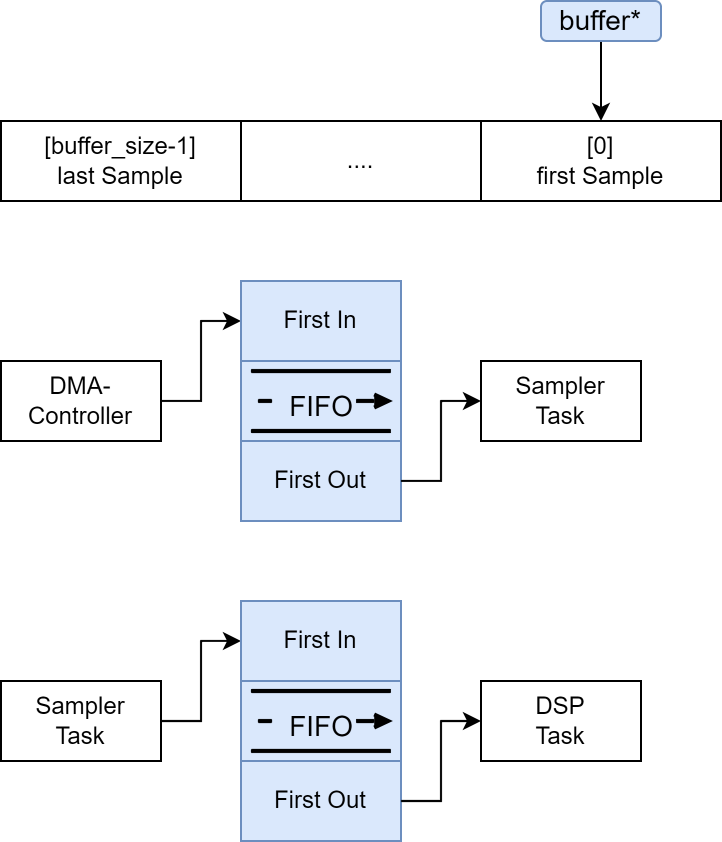
Stores the size of the buffer in Samples. This configuration is usually set during the initialization phase and not changed afterwards.

* Definition at line 26 of file i2s\_sampler.h.

#### dma\_queue

QueueHandle\_t dma\_queue

Raw data sent by the Direct-Memory-Access-Controller is read from a queue by the sampler.

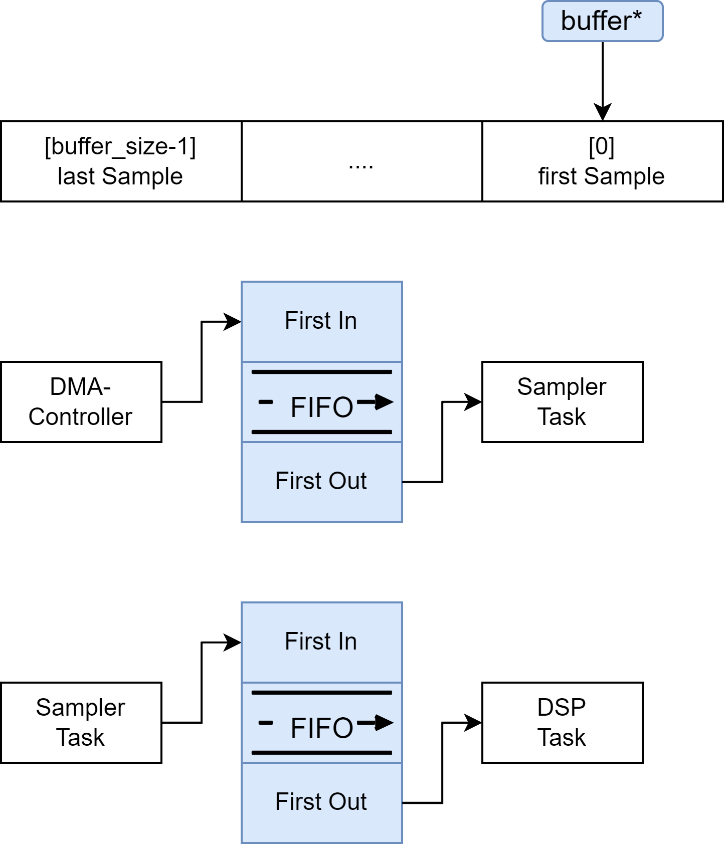


* Definition at line 22 of file i2s\_sampler.h.

#### dsp\_queue

QueueHandle\_t dsp\_queue

The buffer is sent into a queue for further processing.



* Definition at line 23 of file i2s\_sampler.h.

## 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

### Detailed Description

UART Driver Configuration for MIDI transmission

Parameters

|  |  |
| --- | --- |
| uart\_num | UART Port |
| Baudrate | UART Baudrate |
| rx\_io | UART RX Pin |
| tx\_io | UART TX Pin |

* Definition at line 76 of file midi.h.

### Field Documentation

#### baudrate

uint baudrate

Baud rate configuration for UART

* Definition at line 79 of file midi.h.

#### rx\_io

gpio\_num\_t rx\_io

Receive GPIO pin number.

* Definition at line 80 of file midi.h.

#### tx\_io

gpio\_num\_t tx\_io

Transmit GPIO pin number.

* Definition at line 81 of file midi.h.

#### uart\_num

uart\_port\_t uart\_num

UART port id of the ESP32

* Definition at line 78 of file midi.h.

## midi\_context\_t Struct Reference

Wrapper for midi\_config\_t

Data Fields

* midi\_config\_t cfg

### Detailed Description

gitcon\_context\_t references the wrapper struct.

Parameters:

|  |  |
| --- | --- |
| cfg | MIDI Config |

* Definition at line 24 of file midi.c.

### Field Documentation

#### cfg

midi\_config\_t cfg

* Definition at line 26 of file midi.c.

## midi\_message\_t Struct Reference

MIDI Message data.

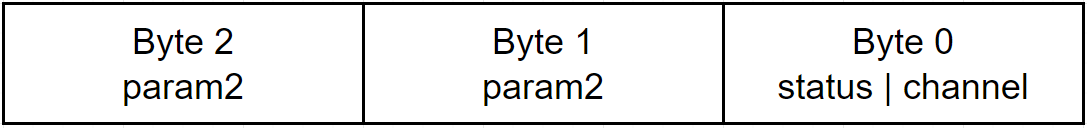
#include <midi.h>

Data Fields

* uint8\_t param1
* midi\_status\_t status
* uint8\_t channel
* uint8\_t param2

### Detailed Description

MIDI Message Data Struct



Parameters

|  |  |
| --- | --- |
| *status* | MIDI Status Byte |
| *channel* | MIDI Channel |
| *param1* | MIDI Parameter 1 |
| *param2* | MIDI Parameter 2 |

* Definition at line 60 of file midi.h.

### Field Documentation

#### channel

uint8\_t channel

Addresses a specific Channel on the MIDI-receiver side.

* Definition at line 64 of file midi.h.

#### param1

uint8\_t param1

Different functions depending on the status.

* Definition at line 62 of file midi.h.

#### param2

uint8\_t param2

Different functions depending on the status.

* Definition at line 65 of file midi.h.

#### status

midi\_status\_t status

* Definition at line 63 of file midi.h.

# Structural Diagram

Ein Bild, das Diagramm enthält.

Automatisch generierte Beschreibung

The Structural Flowchart Diagram above describes the **initialization setup** (in red), as well as the **repetitive** **process tasks** (in yellow). Once the firmware has been reset, an initialization is executed and all initial parameters are loaded into the driver parameter handler, which serves as a central data struct. When fully booted up and the real-time operating system has started, a continuous stream of data is sampled and processed, until a MIDI note is ready to be sent onto the UART bus.

# File Documentation

## config.h File Reference



Macros



### Detailed Description

Gitcon Driver Configuration and Pin-Out File and dependencies.

|  |  |  |  |
| --- | --- | --- | --- |
| Author | Version | Date | Copyright |
| @s-grundner | 0.1 | 2022-12-24 | Copyright (c) 2022 |

* Definition in file config.h.

### Macro Definition Documentation

#### ADC\_RES

#define ADC\_RES (1 << ADC\_RES\_BITS)

ADC Resolution ()

* Definition at line 50 of file config.h.

#### ADC\_RES\_BITS

#define ADC\_RES\_BITS 12

ADC Resolution in bit

* Definition at line 49 of file config.h.

#### AUDIO\_BUFFER\_SIZE

#define AUDIO\_BUFFER\_SIZE 512

Maximum number of samples stored in the audio buffer.

* Definition at line 55 of file config.h.

#### DMA\_CHAN

#define DMA\_CHAN 1

Direct memory access channel

* Definition at line 48 of file config.h.

#### F\_SAMPLE\_HZ

#define F\_SAMPLE\_HZ 15000

Sample rate at which the sampler reads audio data.

* Definition at line 56 of file config.h.

#### FFT\_SIZE

#define FFT\_SIZE (AUDIO\_BUFFER\_SIZE ∗ FFT\_WINDOW\_SIZE)

Number of samples the Fast Fourier Transform can process.

* Definition at line 58 of file config.h.

#### FFT\_WINDOW\_SIZE

#define FFT\_WINDOW\_SIZE 4

Factor by which the Fast Fourier Transform buffer is larger than the audio buffer.

* Definition at line 57 of file config.h.

#### INTERNAL\_ADC\_CHANNEL

#define INTERNAL\_ADC\_CHANNEL (ADC\_CHANNEL\_5)

Internal ADC channel. Corresponds to a hardware GPIO on the ESP32 chip.

* Definition at line 52 of file config.h.

#### INTERNAL\_ADC\_IO

#define INTERNAL\_ADC\_IO (GPIO\_NUM\_33)

Internal Analog Digital Converter GPIO Pin number.

* Definition at line 53 of file config.h.

#### INTERNAL\_ADC\_UNIT

#define INTERNAL\_ADC\_UNIT (ADC\_UNIT\_1)

Internal Analog Digital Converter unit.

* Definition at line 51 of file config.h.

#### MIDI\_BAUD

#define MIDI\_BAUD (115200)

Baud rate of the MIDI UART driver.

* Definition at line 44 of file config.h.

#### MIDI\_RX

#define MIDI\_RX (GPIO\_NUM\_27)

MIDI Driver receive GPIO.

* Definition at line 46 of file config.h.

#### MIDI\_TX

#define MIDI\_TX (GPIO\_NUM\_26)

MIDI Driver receive GPIO.

* Definition at line 45 of file config.h.

#### MIDI\_UART

#define MIDI\_UART (UART\_NUM\_1)

* Definition at line 43 of file config.h.

#### SPI\_CS

#define SPI\_CS (GPIO\_NUM\_5)

* Definition at line 40 of file config.h.

#### SPI\_DEV

#define SPI\_DEV (VSPI\_HOST)

* Definition at line 41 of file config.h.

#### SPI\_MISO

#define SPI\_MISO (GPIO\_NUM\_19)

* Definition at line 38 of file config.h.

#### SPI\_MOSI

#define SPI\_MOSI (GPIO\_NUM\_23)

* Definition at line 37 of file config.h.

#### SPI\_SCLK

#define SPI\_SCLK (GPIO\_NUM\_18)

* Definition at line 39 of file config.h.

## config.h



## i2s\_sampler.c File Reference

#include "i2s\_sampler.h"

Macros



Functions

* static void IRAM\_ATTR sampler\_task(void \*arg)
  + *Sampler Task for I2S*
* i2s\_sampler\_t ∗i2s\_sampler\_start (adc\_channel\_t adc1\_channel, QueueHandle\_t recv\_queue, size\_t buffer\_size, size\_t f\_sample)
  + *Starts a sampler Task that samples from the given ADC1 channel and sends the samples to the given Queue.*
* esp\_err\_t i2s\_sampler\_stop (i2s\_sampler\_t ∗sampler)
  + *Deletes sampler Task, frees memory, and uninstalls ADC as well as i2s driver.*

### Detailed Description

I2S Sampler Driver Source for ESP32

|  |  |  |  |
| --- | --- | --- | --- |
| Author | Version | Date | Copyright |
| @s-grundner | 0.1 | 2022-12-24 | Copyright (c) 2022 |

* Definition in file i2s\_sampler.c.

### Macro Definition Documentation

#### READER\_TIMEOUT\_MS

#define READER\_TIMEOUT\_MS 10

Time in milliseconds for the I2S reader to timeout.

* Definition at line 16 of file i2s\_sampler.c.

#### READER\_TIMEOUT\_TICKS

#define READER\_TIMEOUT\_TICKS (READER\_TIMEOUT\_MS / portTICK\_PERIOD\_MS)

Converts milliseconds into RTOS ticks.

* Definition at line 17 of file i2s\_sampler.c.

#### RESAMPLE\_DENOMINATOR

#define RESAMPLE\_DENOMINATOR 2

Buffer divider for double buffering.

* Definition at line 15 of file i2s\_sampler.c.

### Function Documentation

#### *static* sampler\_task()



This is a RTOS Task, which accepts data from the DMA Controller.

This function is executed in IRAM, due to its direct communication with the DMA-Controller

Parameters

|  |  |
| --- | --- |
| *arg* | Takes in i2s\_sampler\_t \* as datatype. |

* Definition at line 27 of file i2s\_sampler.c.

#### i2s\_sampler\_start()



Starts a sampler Task that samples from the given ADC1 Channel and sends the samples to the given Queue.

Parameters

|  |  |
| --- | --- |
| *adc1\_channel* | ADC1 Channel to use (Only ADC1 Channels are supported) |
| *recv\_queue* | Queue to send samples to |
| *buffer\_size* | Size of the buffer in samples |
| *f\_sample* | Sample rate at which the sampler samples the audio |

Returns

i2s\_sampler\_t∗ Sampler context or NULL if failed.

* Definition at line 53 of file i2s\_sampler.c.

#### i2s\_sampler\_stop()



Deletes sampler Task, frees memory, and uninstalls ADC as well as I2S driver.

Parameters

|  |  |
| --- | --- |
| *sampler* | Sampler to stop. |

Returns

ESP\_OK if successful

* Definition at line 92 of file i2s\_sampler.c.

## i2s\_sampler.c







## i2s\_sampler.h File Reference

I2S audio sampler for ESP32.



Data Structures

* struct i2s\_sampler\_t
  + *Sampler Data.*

Functions

* i2s\_sampler\_t ∗i2s\_sampler\_start (adc\_channel\_t adc1\_channel, QueueHandle\_t recv\_queue, size\_t buffer\_size, size\_t f\_sample)
  + *Starts a sampler Task that samples from the given ADC1 channel and sends the samples to the given Queue.*
* esp\_err\_t i2s\_sampler\_stop (i2s\_sampler\_t ∗sampler)
  + *Deletes sampler Task, frees memory and uninstalls ADC as well as i2s driver.*

### Detailed Description

I2S audio sampler for ESP32.

|  |  |  |  |
| --- | --- | --- | --- |
| Author | Version | Date | Copyright |
| @s-grundner | 0.1 | 2022-12-24 | Copyright (c) 2022 |

* Definition in file i2s\_sampler.h.

### Function Documentation

#### i2s\_sampler\_start()



Starts a sampler Task that samples from the given ADC1 Channel and sends the samples to the given Queue.

Parameters

|  |  |
| --- | --- |
| *adc1\_channel* | ADC1 Channel to use (Only ADC1 Channels are supported) |
| *recv\_queue* | Queue to send samples to |
| *buffer\_size* | Size of the buffer in samples |
| *f\_sample* | Sample rate |

Returns

i2s\_sampler\_t∗ Sampler context or NULL if failed.

* Definition at line 53 of file i2s\_sampler.c.

#### i2s\_sampler\_stop()



Deletes sampler Task, frees memory, and uninstalls ADC as well as I2S driver.

Parameters

|  |  |
| --- | --- |
| *sampler* | Sampler to stop. |

Returns

ESP\_OK if successful

* Definition at line 92 of file i2s\_sampler.c.

## i2s\_sampler.h





## midi.c File Reference

MIDI UART Driver Source for ESP32.

#include "midi.h"

Data Structures

* struct midi\_context\_t
  + MIDI config Wrapper

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)
  + *Initializes MIDI driver.*
* 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.*

### Detailed Description

MIDI UART Driver Source for ESP32.

|  |  |  |  |
| --- | --- | --- | --- |
| Author | Version | Date | Copyright |
| @s-grundner | 0.1 | 2022-12-23 | Copyright (c) 2022 |

* Definition in file midi.c.

### Macro Definition Documentation

#### MIDI\_BYTE\_SIZE\_DEFAULT

#define MIDI\_BYTE\_SIZE\_DEFAULT 3

* Definition at line 13 of file midi.c.

#### MIDI\_BYTE\_SIZE\_SHORT

#define MIDI\_BYTE\_SIZE\_SHORT 2

* Definition at line 14 of file midi.c.

### Typedef Documentation

#### midi\_context\_t

typedef struct midi\_context\_t midi\_context\_t

* Definition at line 24 of file midi.c.

### Function Documentation

#### 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 ESP\_OK on success.

* Definition at line 80 of file midi.c.

#### midi\_init()

esp\_err\_t midi\_init (

midi\_context\_t ∗∗ out\_ctx,

midi\_config\_t ∗out\_cfg)

Initialized MIDI driver.

Parameters

|  |  |
| --- | --- |
| out\_ctx | [out] Outer handler object identifier |
| out\_cfg | MIDI Configuration |

Returns

esp\_err\_t ESP\_OK on success,

ESP\_ERR\_NO\_MEM if failed to allocate memory.

* Definition at line 33 of file midi.c.

#### midi\_read()

Reads MIDI Message from UART.

Parameters

|  |  |
| --- | --- |
| midi\_handle | MIDI Handle to pass parameters. |
| msg | MIDI Message to be read. |

Returns

esp\_err\_t ESP\_OK on success.

#### 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 ESP\_OK if message was read,

ESP\_ERR\_TIMEOUT if UART bus times out,

ESP\_ERR\_INVALID\_ARG if status byte was invalid,

ESP\_FAIL if UART failed to send data.

* Definition at line 92 of file midi.c.

## midi.c









## midi.h File Reference

MIDI UART Driver for ESP32.



Data Structures

* struct midi\_message\_t
  + *MIDI message.*
* struct midi\_config\_t
  + *MIDI UART configuration.*

Macros



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)

### Detailed Description

MIDI UART Driver for ESP32.

|  |  |  |  |
| --- | --- | --- | --- |
| Author | Version | Date | Copyright |
| @s-grundner | 0.1 | 2022-12-23 | Copyright (c) 2022 |

Definition in file midi.h.

### Macro Definition Documentation

#### MIDI\_BYTE\_SIZE\_DEFAULT

#define MIDI\_BYTE\_SIZE\_DEFAULT 3

* Definition at line 23 of file midi.h.

#### MIDI\_BYTE\_SIZE\_SHORT

#define MIDI\_BYTE\_SIZE\_SHORT 2

* Definition at line 24 of file midi.h.

#### MIDI\_LOG\_LEVEL

#define MIDI\_LOG\_LEVEL ESP\_LOG\_ERROR

* Definition at line 21 of file midi.h.

#### MIDI\_PITCH\_BEND\_CENTER

#define MIDI\_PITCH\_BEND\_CENTER (8192)

* Definition at line 28 of file midi.h.

#### MIDI\_PITCH\_BEND\_MAX

#define MIDI\_PITCH\_BEND\_MAX (16383)

* Definition at line 27 of file midi.h.

#### MIDI\_PITCH\_BEND\_MIN

#define MIDI\_PITCH\_BEND\_MIN (0)

* Definition at line 26 of file midi.h.

### Typedef Documentation

#### midi\_handle\_t

typedef struct midi\_context\_t∗ midi\_handle\_t

* Definition at line 85 of file midi.h.

### Enumeration Type Documentation

#### midi\_status\_t

|  |  |
| --- | --- |
| *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 |

Enumerator

|  |  |
| --- | --- |
| *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 |

* Definition at line 41 of file midi.h.

### Function Documentation

#### channel\_pressure()

#### ctrl\_change()

#### midi\_exit()

#### midi\_init()

#### midi\_read()

#### midi\_write()

#### note\_off()

#### note\_on()

#### pitch\_bend()

#### poly\_key\_pressure()

#### prg\_change()

## midi.h







## midi\_utils.c File Reference

### Detailed Description

### Function Documentation

#### channel\_pressure()

#### ctrl\_change()

#### note\_off()

#### note\_on()

#### pitch\_bend()

#### poly\_key\_pressure()

#### prg\_change()

## midi\_utils.c





## gitcon.c File Reference

### Detailed Description

### Macro Definition Documentation

#### CONCERT\_A

#### CONCERT\_A\_NOTE

#### FLOAT\_TO\_UINT16

#### MIDI\_HIGHEST\_NOTE

#### MIDI\_KEY\_BOUNDARY

#### MIDI\_LOWEST\_NOTE

#### SENSITIVITY

#### UINT16\_TO\_FLOAT

### Function Documentation

#### *static* dsp\_task()

#### *static* midi\_task()

#### gitcon\_exit()

#### gitcon\_init()

## gitcon.c















## gitcon.h File Reference

### Detailed Description

### Macro Definition Documentation

#### GITCON\_LOG\_LEVEL

### Typedef Documentation

#### gitcon\_handle\_t

### Function Documentation

#### gitcon\_exit()

#### gitcon\_init()

## gitcon.h



## main.c File Reference

## main.c







## test\_fft.c File Reference

## test\_fft.c

## test\_midi.c File Reference

## test\_midi.c

# Index