libDaisy

Generated by Doxygen 1.8.18

Chapter 1

libdaisy

1.1 ↩ Documentation available on our wiki!

Multi-layer hardware abstraction library for Daisy Product family

On STM32H7 MCUs

Lower-levels use STM32 HAL (local copy w/ modifications in Drivers/)

Prefixes and their meanings:

- sys - System level configuration (clocks, dma, etc.)
- per - Peripheral level, internal to MCU (i2c, spi, etc.)
- dev - External device support (external flash chips, DACs, codecs, etc.)
- hid - User level interface elements (encoders, switches, audio, etc.)
- util - library level elements used within the library (not included via daisy.h)
- daisy - core API files (specific boards, platforms have extended user APIs that configure libdaisy more below).

Also included is a core/ folder containing:

- a generic Makefile that can be included in a project Makefile to simplify getting started
- a linker script for defining the sections of memory used by the firmware
- core files for starting the hardware (system_stm32h7xx.c, startup_stm32h750xx.s, etc.)

1.2 Using libdaisy

Due to the amount of hardware configuration and flexibility of the daisy platform, (in the present, and the future), a user can use libdaisy to define their own custom hardware, or include one of our supported board files to jumpstart the creativity, and hack on an existing piece of hardware.
If you are getting started, and have one of the Daisy Family Products, you can skip ahead to that section below.

1.2.1 daisy.h

The base-level include file. This is all you need to include to create your own custom hardware that uses libdaisy.
daisy_seed.h is an example of a board level file that utilizes libdaisy to define some hardware, and provide flexible access.

1.2.2 daisy_seed.h

The SOM-level include file. This can be used with any boards that use the Daisy Seed hardware.
Additional configuration files, with more specific hardware access are provided below for our supported hardware platforms.

1.2.3 daisy_platform.h

Several other pairs of files exist in the repo for each of the supported hardware platforms that work with Daisy Seed. These are:

- daisy_field
- daisy_patch
- daisy_petal
- daisy_pod

With these files a number of additional initialization, and configuration is done by the library.
This allows a user to jump right into their new product with a simple api to do things without having a full understanding of what's going on under the hood.
With this flexible approach to the hardware configuration, we hope to promote a lot of fantastic hardware along with code to go with it.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Namespace Index

3.1 Namespace List

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

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

Chapter 6

Module Documentation

6.1 LIBDAISY

The daisy library.

Modules

- HUMAN_INTERFACE

  Interface with the world.
- PERIPHERAL

  Peripheral devices, not meant for human interaction.
- SYSTEM

  Deals with system. DMA, clocks, etc.
- DEVICE

  Low level devices. Led drivers, codecs, etc.
- BOARDS

  Daisy devices. Pod, seed, etc.
- UTILITY

  General utilities. Ringbuffers, LED colors, OLED stuff, etc.

6.1.1 Detailed Description

The daisy library.

6.2 HUMAN_INTERFACE

Interface with the world.

Modules

- AUDIO

  Embedded Audio Engine.
- CONTROLS

  Hardware Controls.
- FEEDBACK

  Screens, leds, etc.
- EXTERNAL

  External interface devices.

6.2.1 Detailed Description

Interface with the world.

6.3 AUDIO

Embedded Audio Engine.

- enum { DSY_AUDIO_INTERNAL, DSY_AUDIO_EXTERNAL, DSY_AUDIO_LAST }
- typedef void(∗ dsy_audio_callback) (float ∗, float ∗, size_t)
- typedef void(∗ dsy_audio_mc_callback) (float ∗∗, float ∗∗, size_t)
- void dsy_audio_init (dsy_audio_handle ∗handle)
- void dsy_audio_set_callback (uint8_t intext, dsy_audio_callback cb)
- void dsy_audio_set_mc_callback (dsy_audio_mc_callback cb)
- void dsy_audio_set_blocksize (uint8_t intext, size_t blocksize)
- void dsy_audio_start (uint8_t intext)
- void dsy_audio_stop (uint8_t intext)
- void dsy_audio_enter_bypass (uint8_t intext)
- void dsy_audio_exit_bypass (uint8_t intext)
- void dsy_audio_passthru (float ∗in, float ∗out, size_t size)
- void dsy_audio_silence (float ∗in, float ∗out, size_t size)

6.3.1 Detailed Description

Embedded Audio Engine.

6.3.2 Typedef Documentation

6.3.2.1 dsy_audio_callback

```
typedef void(* dsy_audio_callback) (float *, float *, size_t)
```

These are user-defineable callbacks that are called when audio data is ready to be received/transmitted. Function to define for using a single Stereo device for I/O audio is packed as: { LEFT | RIGHT | LEFT | RIGHT }

6.3.2.2 dsy_audio_mc_callback

```
typedef void(* dsy_audio_mc_callback) (float **, float **, size_t)
```

Defaults to 4 channels, and is fixed for now.
(still works for stereo, but will still fill buffers)
audio is packed as:
{ LEFT | LEFT + 1 | . . . | LEFT + SIZE | RIGHT | RIGHT + 1 | . . . | RIGHT + SIZE }

6.3.3 Enumeration Type Documentation

6.3.3.1 anonymous enum

```
anonymous enum
```

Internally, there are two separate 'audio blocks' that can be configured together or separately

Enumerator

DSY_AUDIO_INTERNAL	&
DSY_AUDIO_EXTERNAL	&
DSY_AUDIO_LAST	&

6.3.4 Function Documentation

6.3.4.1 dsy_audio_enter_bypass()

```
void dsy_audio_enter_bypass (
            uint8_t intext )
```

If the device supports hardware bypass, enter that mode.

6.3.4.2 dsy_audio_exit_bypass()

```
void dsy_audio_exit_bypass (
            uint8_t intext )
```

If the device supports hardware bypass, exit that mode.

6.3.4.3 dsy_audio_init()

```
void dsy_audio_init (
            dsy_audio_handle * handle )
```

Initializes the Audio Engine using configurations
set to the sai_handle
i2c_handles can be set to NULL if not needed.

6.3.4.4 dsy_audio_passthru()

```
void dsy_audio_passthru (
            float * in,
            float * out,
            size_t size )
```

A few useful stereo-interleaved callbacks
Passes the input to the output

6.3.4.5 dsy_audio_set_blocksize()

```
void dsy_audio_set_blocksize (
            uint8_t intext,
            size_t blocksize )
```

Sets the number of samples (per-channel) to be handled in a single audio frame.

6.3.4.6 dsy_audio_set_callback()

```
void dsy_audio_set_callback (
            uint8_t intext,
            dsy_audio_callback cb )
```

Sets the user defined, interleaving callback to be called when audio data is ready.
intext is a specifier for DSY_AUDIO_INT/EXT (which audio peripheral to use).
When using this, each 'audio block' can have completely independent callbacks.

6.3.4.7 dsy_audio_set_mc_callback()

```
void dsy_audio_set_mc_callback (
            dsy_audio_mc_callback cb )
```

Sets the user defined, non-interleaving callback to be called when audio data is ready.
This will always use both DSY_AUDIO_INT and DSY_AUDIO_EXT blocks together.
To ensure clean audio you'll want to make sure the two SAIs are set to the same samplerate

6.3.4.8 dsy_audio_silence()

```
void dsy_audio_silence (
            float * in,
            float * out,
            size_t size )
```

sets outputs to 0 without stopping the Audio Engine

6.3.4.9 dsy_audio_start()

```
void dsy_audio_start (
            uint8_t intext )
```

Starts Audio Engine, callbacks will begin getting called.
When using with dsy_audio_mc_callback (for 4 channels), this function should be called for both audio blocks

6.3.4.10 dsy_audio_stop()

```
void dsy_audio_stop (
            uint8_t intext )
```

Stops transmitting/receiving audio on the specified audio block.

6.4 CONTROLS

Hardware Controls.

Classes

- class daisy::AnalogControl

  Hardware Interface for control inputs
  Primarily designed for ADC input controls such as
  potentiometers, and control voltage.
  .

- class daisy::Encoder

  Generic Class for handling Quadrature Encoders
  Inspired/influenced by Mutable Instruments (pichenettes) Encoder classes.

- class daisy::GateIn

  Generic Class for handling gate inputs through GPIO.

- class daisy::Parameter
- class daisy::Switch

6.4.1 Detailed Description

Hardware Controls.

6.5 FEEDBACK

Screens, leds, etc.

Classes

- class daisy::Led

  LED Class providing simple Software PWM ability, etc
  Eventually this will work with hardware PWM, and external LED Driver devices as well.
- class daisy::OledDisplay
- class daisy::RgbLed

6.5.1 Detailed Description

Screens, leds, etc.

6.6 EXTERNAL

External interface devices.

Classes

- struct daisy::NoteOnEvent
- struct daisy::ControlChangeEvent
- struct daisy::MidiEvent
- class daisy::MidiHandler

  Simple MIDI Handler
  Parses bytes from an input into valid MidiEvents.
  The MidiEvents fill a FIFO queue that the user can pop messages from.

Enumerations

- enum daisy::MidiMessageType {
  daisy::NoteOff, daisy::NoteOn, daisy::PolyphonicKeyPressure, daisy::ControlChange,
  daisy::ProgramChange, daisy::ChannelPressure, daisy::PitchBend, daisy::MessageLast }

6.6.1 Detailed Description

External interface devices.

6.6.2 Enumeration Type Documentation

6.6.2.1 MidiMessageType

`enum daisy::MidiMessageType`

Parsed from the Status Byte, these are the common Midi Messages that can be handled.
At this time only 3-byte messages are correctly parsed into MidiEvents.

Enumerator

NoteOff	&
NoteOn	&
PolyphonicKeyPressure	&
ControlChange	&
ProgramChange	&
ChannelPressure	&
PitchBend	&
MessageLast	&

6.7 PERIPHERAL

Peripheral devices, not meant for human interaction.

Modules

- SERIAL

  Serial Communications.
- ANALOG_DIGITAL_CONVERSION

  Convert from digital to analog, or vice-versa.
- OTHER

  GPIO, timers, and SDMMC.

6.7.1 Detailed Description

Peripheral devices, not meant for human interaction.

6.8 SERIAL

Serial Communications.

Classes

- struct dsy_i2c_handle
- struct dsy_qspi_handle
- struct dsy_sai_handle
- class daisy::SpiHandle
- class daisy::UartHandler

Enumerations

- enum dsy_i2c_periph { DSY_I2C_PERIPH_1, DSY_I2C_PERIPH_2, DSY_I2C_PERIPH_3, DSY_I2C_PERIPH_4 }
- enum dsy_i2c_pin { DSY_I2C_PIN_SCL, DSY_I2C_PIN_SDA, DSY_I2C_PIN_LAST }
- enum dsy_i2c_speed { DSY_I2C_SPEED_100KHZ, DSY_I2C_SPEED_400KHZ, DSY_I2C_SPEED_1MHZ, DSY_I2C_SPEED_LAST }
- enum dsy_qspi_pin { DSY_QSPI_PIN_IO0, DSY_QSPI_PIN_IO1, DSY_QSPI_PIN_IO2, DSY_QSPI_PIN_IO3, DSY_QSPI_PIN_CLK, DSY_QSPI_PIN_NCS, DSY_QSPI_PIN_LAST }
- enum dsy_qspi_mode { DSY_QSPI_MODE_DSY_MEMORY_MAPPED, DSY_QSPI_MODE_INDIRECT_POLLING, DSY_QSPI_MODE_LAST }
- enum dsy_qspi_device { DSY_QSPI_DEVICE_IS25LP080D, DSY_QSPI_DEVICE_IS25LP064A, DSY_QSPI_DEVICE_LAST }
- enum dsy_audio_sai { DSY_AUDIO_INIT_SAI1, DSY_AUDIO_INIT_SAI2, DSY_AUDIO_INIT_BOTH, DSY_AUDIO_INIT_NONE, DSY_AUDIO_INIT_LAST }
- enum dsy_audio_samplerate { DSY_AUDIO_SAMPLERATE_32K, DSY_AUDIO_SAMPLERATE_48K, DSY_AUDIO_SAMPLERATE_96K, DSY_AUDIO_SAMPLERATE_LAST }
- enum dsy_audio_bitdepth { DSY_AUDIO_BITDEPTH_16, DSY_AUDIO_BITDEPTH_24, DSY_AUDIO_BITDEPTH_LAST }
- enum dsy_audio_sync { DSY_AUDIO_SYNC_MASTER, DSY_AUDIO_SYNC_SLAVE, DSY_AUDIO_SYNC_LAST }
- enum dsy_audio_dir { DSY_AUDIO_RX, DSY_AUDIO_TX }
- enum dsy_sai_pin { DSY_SAI_PIN_MCLK, DSY_SAI_PIN_FS, DSY_SAI_PIN_SCK, DSY_SAI_PIN_SIN, DSY_SAI_PIN_SOUT, DSY_SAI_PIN_LAST }
- enum dsy_audio_device { DSY_AUDIO_NONE, DSY_AUDIO_DEVICE_PCM3060, DSY_AUDIO_DEVICE_WM8731, DSY_AUDIO_DEVICE_AK4556, DSY_AUDIO_DEVICE_LAST }
- enum { DSY_SAI_1, DSY_SAI_2, DSY_SAI_LAST }
- enum daisy::SpiPeriph { daisy::SPI_PERIPH_1, daisy::SPI_PERIPH_3, daisy::SPI_PERIPH_6 }
- enum daisy::SpiPin { daisy::SPI_PIN_CS, daisy::SPI_PIN_SCK, daisy::SPI_PIN_MOSI, daisy::SPI_PIN_MISO }

Functions

- void dsy_i2c_init (dsy_i2c_handle *dsy_hi2c)
- int dsy_qspi_init (dsy_qspi_handle *hqspi)
- int dsy_qspi_deinit ()
- int dsy_qspi_writepage (uint32_t adr, uint32_t sz, uint8_t *buf)
- int dsy_qspi_write (uint32_t address, uint32_t size, uint8_t *buffer)
- int dsy_qspi_erase (uint32_t start_adr, uint32_t end_adr)
- int dsy_qspi_erasesector (uint32_t addr)

- void dsy_sai_init (dsy_audio_sai init, dsy_audio_samplerate sr[2], dsy_audio_bitdepth bitdepth[2], dsy_audio_sync sync_config[2], dsy_gpio_pin *sai1_pin_list, dsy_gpio_pin *sai2_pin_list)
- void dsy_sai_init_from_handle (dsy_sai_handle *hsai)

Variables

- const size_t daisy::kUartMaxBufferSize = 32

6.8.1 Detailed Description

Serial Communications.

6.8.2 Enumeration Type Documentation

6.8.2.1 anonymous enum

anonymous enum

Index for the several arrays in the sai_handle struct below.

Enumerator

DSY_SAI_1	&
DSY_SAI_2	&
DSY_SAI_LAST	&

6.8.2.2 dsy_audio_bitdepth

enum dsy_audio_bitdepth

Specifies the bitdepth of the hardware connected to the SAI peripheral

Enumerator

DSY_AUDIO_BITDEPTH_16	&
DSY_AUDIO_BITDEPTH_24	&
DSY_AUDIO_BITDEPTH_LAST	&

6.8.2.3 dsy_audio_device

enum dsy_audio_device

List of devices with built in support. Devices not listed here, will need to have initialization done externally.

Enumerator

DSY_AUDIO_NONE	For unsupported, or custom devices.
DSY_AUDIO_DEVICE_PCM3060	&
DSY_AUDIO_DEVICE_WM8731	&
DSY_AUDIO_DEVICE_AK4556	&
DSY_AUDIO_DEVICE_LAST	&

6.8.2.4 dsy_audio_dir

enum dsy_audio_dir

Each SAI has two datalines, they can independently be configured as inputs or outputs.

Enumerator

DSY_AUDIO_RX	&
DSY_AUDIO_TX	&

6.8.2.5 dsy_audio_sai

enum dsy_audio_sai

Driver for the SAI peripheral Supports SAI1 and SAI2 with several configuration options selects which SAI (or both/none) to initialize

Enumerator

DSY_AUDIO_INIT_SAI1	&
DSY_AUDIO_INIT_SAI2	&
DSY_AUDIO_INIT_BOTH	&
DSY_AUDIO_INIT_NONE	&
DSY_AUDIO_INIT_LAST	&

6.8.2.6 dsy_audio_samplerate

enum dsy_audio_samplerate

Currently Sample Rates are not correctly supported. All audio is currently run at 48kHz

Enumerator

DSY_AUDIO_SAMPLERATE_32K	&
DSY_AUDIO_SAMPLERATE_48K	&
DSY_AUDIO_SAMPLERATE_96K	&
DSY_AUDIO_SAMPLERATE_LAST	&

6.8.2.7 dsy_audio_sync

enum dsy_audio_sync

Setting for each SAI that sets whether the processor is generating the MCLK signal or not.

Enumerator

DSY_AUDIO_SYNC_MASTER	No Crystal
DSY_AUDIO_SYNC_SLAVE	Crystal
DSY_AUDIO_SYNC_LAST	&

6.8.2.8 dsy_i2c_periph

enum dsy_i2c_periph
Driver for controlling I2C devices Specifices the internal peripheral to use (these are mapped to different pins on the hardware).

Enumerator

DSY_I2C_PERIPH↩ _1	&
DSY_I2C_PERIPH↩ _2	&
DSY_I2C_PERIPH↩ _3	&
DSY_I2C_PERIPH↩ _4	&

6.8.2.9 dsy_i2c_pin

enum dsy_i2c_pin
List of pins associated with the peripheral. These must be set in the handle's pin_config.

Enumerator

DSY_I2C_PIN_SCL	&
DSY_I2C_PIN_SDA	&
DSY_I2C_PIN_LAST	&

6.8.2.10 dsy_i2c_speed

enum dsy_i2c_speed
Rate at which the clock/data will be sent/received. The device being used will have maximum speeds. 1MHZ Mode is currently 886kHz**

Enumerator

DSY_I2C_SPEED_100KHZ	&
DSY_I2C_SPEED_400KHZ	&
DSY_I2C_SPEED_1MHZ	&
DSY_I2C_SPEED_LAST	&

6.8.2.11 dsy_qspi_device

enum dsy_qspi_device
Flash Devices supported. (Both of these are more-or-less the same, just different sizes).

Enumerator

DSY_QSPI_DEVICE_IS25LP080D	&
DSY_QSPI_DEVICE_IS25LP064A	&
DSY_QSPI_DEVICE_LAST	&

6.8.2.12 dsy_qspi_mode

enum dsy_qspi_mode
Modes of operation. Memory Mapped mode: QSPI configured so that the QSPI can be read from starting address 0x90000000. Writing is not possible in this mode.
Indirect Polling mode: Device driver enabled.

Read/Write possible via dsy_qspi_* functions

Enumerator

DSY_QSPI_MODE_DSY_MEMORY_MAPPED	&
DSY_QSPI_MODE_INDIRECT_POLLING	&
DSY_QSPI_MODE_LAST	&

6.8.2.13 dsy_qspi_pin

enum dsy_qspi_pin
Driver for QSPI peripheral to interface with external flash memory.
Currently supported QSPI Devices:
IS25LP080D List of Pins used in QSPI (passed in during Init)

Enumerator

DSY_QSPI_PIN_IO0	&
DSY_QSPI_PIN_IO1	&
DSY_QSPI_PIN_IO2	&
DSY_QSPI_PIN_IO3	&
DSY_QSPI_PIN_CLK	&
DSY_QSPI_PIN_NCS	&
DSY_QSPI_PIN_LAST	&

6.8.2.14 dsy_sai_pin

enum dsy_sai_pin
List of the pins that need to be initialized SIN/SOUT is a bit misleading, and should be turned into A/B since it is possible to configure two inputs or two outputs on a single SAI.

Enumerator

DSY_SAI_PIN_MCLK	&
DSY_SAI_PIN_FS	&
DSY_SAI_PIN_SCK	&
DSY_SAI_PIN_SIN	&
DSY_SAI_PIN_SOUT	&
DSY_SAI_PIN_LAST	&

6.8.2.15 SpiPeriph

`enum daisy::SpiPeriph`
SPI peripheral enum

Enumerator

SPI_PERIPH↩_1	SPI peripheral 1
SPI_PERIPH↩_3	SPI peripheral 3
SPI_PERIPH↩_6	SPI peripheral 3

6.8.2.16 SpiPin

`enum daisy::SpiPin`
SPI pins

Enumerator

SPI_PIN_CS	CS pin
SPI_PIN_SCK	SCK pin
SPI_PIN_MOSI	MOSI pin
SPI_PIN_MISO	MISO pin

6.8.3 Function Documentation

6.8.3.1 dsy_i2c_init()

```
void dsy_i2c_init (
            dsy_i2c_handle * dsy_hi2c )
```
Initializes an I2C peripheral with the data given from the handle.

Parameters

∗dsy_hi2c	Required to initialize.

6.8.3.2 dsy_qspi_deinit()

```
int dsy_qspi_deinit ( )
```
Deinitializes the peripheral This should be called before reinitializing QSPI in a different mode.

Returns

DSY_MEMORY_OK or DSY_MEMORY_ERROR

6.8.3.3 dsy_qspi_erase()

```
int dsy_qspi_erase (
```

```
            uint32_t start_adr,
            uint32_t end_adr )
```

Erases the area specified on the chip. Erasures will happen by 4K, 32K or 64K increments. Smallest erase possible is 4kB at a time. (on IS25LP∗)

Parameters

start_adr	Address to begin erasing from
end_adr	Address to stop erasing at

Returns

DSY_MEMORY_OK or DSY_MEMORY_ERROR

6.8.3.4 dsy_qspi_erasesector()

```
int dsy_qspi_erasesector (
            uint32_t addr )
```

Erases a single sector of the chip.
TODO: Document the size of this function.

Parameters

addr	Address of sector to erase

Returns

DSY_MEMORY_OK or DSY_MEMORY_ERROR

6.8.3.5 dsy_qspi_init()

```
int dsy_qspi_init (
            dsy_qspi_handle * hqspi )
```

Initializes QSPI peripheral, and Resets, and prepares memory for access.

Parameters

hqspi	should be populated with the mode, device and pin_config before calling this function.

Returns

DSY_MEMORY_OK or DSY_MEMORY_ERROR

6.8.3.6 dsy_qspi_write()

```
int dsy_qspi_write (
            uint32_t address,
            uint32_t size,
            uint8_t * buffer )
```

Writes data in buffer to to the QSPI. Starting at address to address+size

Parameters

address	Address to write to
size	Buffer size
buffer	Buffer to write

Returns

DSY_MEMORY_OK or DSY_MEMORY_ERROR

6.8.3.7 dsy_qspi_writepage()

```
int dsy_qspi_writepage (
            uint32_t adr,
            uint32_t sz,
            uint8_t * buf )
```

Writes a single page to to the specified address on the QSPI chip. For IS25LP∗ page size is 256 bytes.

Parameters

adr	Address to write to
sz	Buff size
buf	Buffer to write

Returns

DSY_MEMORY_OK or DSY_MEMORY_ERROR

6.8.3.8 dsy_sai_init()

```
void dsy_sai_init (
            dsy_audio_sai init,
            dsy_audio_samplerate sr[2],
            dsy_audio_bitdepth bitdepth[2],
            dsy_audio_sync sync_config[2],
            dsy_gpio_pin * sail_pin_list,
            dsy_gpio_pin * sai2_pin_list )
```

Intializes the SAI peripheral(s) with the specified settings. Pinlists should be arrays of DSY_SAI_PIN_LAST elements

Parameters

init	&
sr[ ]	Sample rate per chan: 0, 1
bitdepth[ ]	Bitdepth per chan: 0, 1
sync_config[ ]	& sync config per chan: 0, 1
∗*sai1_pin_list*	&
∗*sai2_pin_list*	&

6.8.3.9 dsy_sai_init_from_handle()

```
void dsy_sai_init_from_handle (
            dsy_sai_handle * hsai )
```

Uses the data within *hsai to initialize the peripheral(s)

Parameters

hsai	&

6.8.4 Variable Documentation

6.8.4.1 kUartMaxBufferSize

```
const size_t daisy::kUartMaxBufferSize = 32
```

Maximum Queue buffer size

6.9 ANALOG_DIGITAL_CONVERSION

Convert from digital to analog, or vice-versa.

Classes

- struct daisy::AdcChannelConfig
- class daisy::AdcHandle
- struct dsy_dac_handle

Enumerations

- enum dsy_dac_mode { DSY_DAC_MODE_POLLING, DSY_DAC_MODE_LAST }
- enum dsy_dac_bitdepth { DSY_DAC_BITS_8, DSY_DAC_BITS_12, DSY_DAC_BITS_LAST }
- enum dsy_dac_channel { DSY_DAC_CHN1, DSY_DAC_CHN2, DSY_DAC_CHN_LAST, DSY_DAC_CHN_BOTH }

Functions

- void dsy_dac_init (dsy_dac_handle *dsy_hdac, dsy_dac_channel channel)
- void dsy_dac_start (dsy_dac_channel channel)
- void dsy_dac_write (dsy_dac_channel channel, uint16_t val)

6.9.1 Detailed Description

Convert from digital to analog, or vice-versa.

6.9.2 Enumeration Type Documentation

6.9.2.1 dsy_dac_bitdepth

`enum dsy_dac_bitdepth`

Sets the bit depth of the DAC output This can be set independently for each channel.

Enumerator

DSY_DAC_BITS_8	&
DSY_DAC_BITS_12	&
DSY_DAC_BITS_LAST	&

6.9.2.2 dsy_dac_channel

`enum dsy_dac_channel`

Sets which channel(s) are initialized with the settings chosen.

Enumerator

DSY_DAC_CHN1	&
DSY_DAC_CHN2	&
DSY_DAC_CHN_LAST	&
DSY_DAC_CHN_BOTH	&

6.9.2.3 dsy_dac_mode

```
enum dsy_dac_mode
```

Driver for the built in DAC on the STM32 The STM32 has 2 Channels of independently configurable DACs, with up to 12-bit resolution. Currently only Polling is supported.

Enumerator

DSY_DAC_MODE_POLLING	Polling mode
DSY_DAC_MODE_LAST	3

6.9.3 Function Documentation

6.9.3.1 dsy_dac_init()

```
void dsy_dac_init (
            dsy_dac_handle * dsy_hdac,
            dsy_dac_channel channel )
```

Initializes the specified channel(s) of the DAC

Parameters

∗dsy_hdac	Dac to initialize
channel	Channels to init

6.9.3.2 dsy_dac_start()

```
void dsy_dac_start (
            dsy_dac_channel channel )
```

Turns on the DAC and turns on any internal timer if necessary.

Parameters

channel	Channel to start

6.9.3.3 dsy_dac_write()

```
void dsy_dac_write (
            dsy_dac_channel channel,
            uint16_t val )
```

Sets the specified channel of the dac to the value (within bitdepth) resolution. When set to 8-bit, val should be 0-255 When set to 12-bit, val should be 0-4095

Parameters

channel	Channel to write to
val	Value to write

6.10 OTHER

GPIO, timers, and SDMMC.

Classes

- struct dsy_gpio
- struct daisy::SdmmcHandlerInit
- class daisy::SdmmcHandler

Enumerations

- enum dsy_gpio_mode { DSY_GPIO_MODE_INPUT, DSY_GPIO_MODE_OUTPUT_PP, DSY_GPIO_MODE_OUTPUT_OD, DSY_GPIO_MODE_ANALOG, DSY_GPIO_MODE_LAST }
- enum dsy_gpio_pull { DSY_GPIO_NOPULL, DSY_GPIO_PULLUP, DSY_GPIO_PULLDOWN }
- enum daisy::SdmmcMode { daisy::SDMMC_MODE_FATFS }
- enum daisy::SdmmcBitWidth { daisy::SDMMC_BITS_1, daisy::SDMMC_BITS_4 }
- enum daisy::SdmmcSpeed { daisy::SDMMC_SPEED_400KHZ, daisy::SDMMC_SPEED_12MHZ }

Functions

- void dsy_gpio_init (dsy_gpio *p)
- void dsy_gpio_deinit (dsy_gpio *p)
- uint8_t dsy_gpio_read (dsy_gpio *p)
- void dsy_gpio_write (dsy_gpio *p, uint8_t state)
- void dsy_gpio_toggle (dsy_gpio *p)
- void dsy_tim_init ()
- void dsy_tim_start ()
- uint32_t dsy_tim_get_tick ()
- void dsy_tim_delay_tick (uint32_t cnt)
- uint32_t dsy_tim_get_ms ()
- void dsy_tim_delay_ms (uint32_t cnt)
- uint32_t dsy_tim_get_us ()
- void dsy_tim_delay_us (uint32_t cnt)

6.10.1 Detailed Description

GPIO, timers, and SDMMC.
General Purpose IO driver

6.10.2 Enumeration Type Documentation

6.10.2.1 dsy_gpio_mode

`enum dsy_gpio_mode`
Sets the mode of the GPIO

Enumerator

DSY_GPIO_MODE_INPUT	&
DSY_GPIO_MODE_OUTPUT_PP	Push-Pull
DSY_GPIO_MODE_OUTPUT_OD	Open-Drain
DSY_GPIO_MODE_ANALOG	&
DSY_GPIO_MODE_LAST	&

6.10.2.2 dsy_gpio_pull

enum dsy_gpio_pull
Configures whether an internal Pull up or Pull down resistor is used

Enumerator

DSY_GPIO_NOPULL	&
DSY_GPIO_PULLUP	&
DSY_GPIO_PULLDOWN	&

6.10.2.3 SdmmcBitWidth

enum daisy::SdmmcBitWidth
Sets whether 4-bit mode or 1-bit mode is used for the SDMMC

Enumerator

SDMMC_BITS↩ _1	&
SDMMC_BITS↩ _4	&

6.10.2.4 SdmmcMode

enum daisy::SdmmcMode
Operating Mode. Currently only FatFS is supported.

Enumerator

SDMMC_MODE_FATFS	&

6.10.2.5 SdmmcSpeed

enum daisy::SdmmcSpeed
Sets the desired clock speed of the SD card bus.
Initialization is always done at or below 400kHz, and then the user speed is set.

Enumerator

SDMMC_SPEED_400KHZ	&
SDMMC_SPEED_12MHZ	&

6.10.3 Function Documentation

6.10.3.1 dsy_gpio_deinit()

```
void dsy_gpio_deinit (
            dsy_gpio * p )
```

Deinitializes the gpio pin

Parameters

*p	Pin pointer

6.10.3.2 dsy_gpio_init()

```
void dsy_gpio_init (
            dsy_gpio * p )
```

Initializes the gpio with the settings configured.

Parameters

*p	Pin pointer

6.10.3.3 dsy_gpio_read()

```
uint8_t dsy_gpio_read (
            dsy_gpio * p )
```

Reads the state of the gpio pin

Parameters

*p	Pin pointer

Returns

1 if the pin is HIGH, and 0 if the pin is LOW

6.10.3.4 dsy_gpio_toggle()

```
void dsy_gpio_toggle (
            dsy_gpio * p )
```

Toggles the state of the pin so that it is not at the same state as it was previously.

Parameters

*p	Pin pointer

6.10.3.5 dsy_gpio_write()

```
void dsy_gpio_write (
            dsy_gpio * p,
            uint8_t state )
```

Writes the state to the gpio pin Pin will be set to 3v3 when state is 1, and 0V when state is 0

Parameters

*p	Pin pointer
state	State to write

6.10.3.6 dsy_tim_delay_ms()

```
void dsy_tim_delay_ms (
            uint32_t cnt )
```

blocking delay of cnt milliseconds.

Parameters

cnt	Delay time in ms

6.10.3.7 dsy_tim_delay_tick()

```
void dsy_tim_delay_tick (
            uint32_t cnt )
```

blocking delay of cnt timer ticks.

Parameters

cnt	Number of ticks

6.10.3.8 dsy_tim_delay_us()

```
void dsy_tim_delay_us (
            uint32_t cnt )
```

blocking delay of cnt microseconds.

Parameters

cnt	Delay time in us

6.10.3.9 dsy_tim_get_ms()

```
uint32_t dsy_tim_get_ms ( )
```

These functions are converted to use milliseconds as their time base.

Returns

the number of milliseconds through the timer period.

6.10.3.10 dsy_tim_get_tick()

```
uint32_t dsy_tim_get_tick ( )
```

These functions are specific to the actual clock ticks at the timer frequency which is currently fixed at 200MHz

Returns

a number 0x00000000-0xffffffff of the current tick

6.10.3.11 dsy_tim_get_us()

`uint32_t dsy_tim_get_us ( )`

These functions are converted to use microseconds as their time base.

Returns

the number of microseconds through the timer period.

6.10.3.12 dsy_tim_init()

`void dsy_tim_init ( )`

General purpose timer for delays and general timing. initializes the TIM2 peripheral with maximum counter autoreload, and no prescalers.

6.10.3.13 dsy_tim_start()

`void dsy_tim_start ( )`

Starts the timer ticking.

6.11 SYSTEM

Deals with system. DMA, clocks, etc.

Functions

- void dsy_dma_init (void)
- void dsy_system_init ()
- void dsy_system_jumpto (uint32_t addr)
- void dsy_system_jumptoqspi ()
- uint32_t dsy_system_getnow ()
- void dsy_system_delay (uint32_t delay_ms)

6.11.1 Detailed Description

Deals with system. DMA, clocks, etc.
Low level System Configuration

6.11.2 Function Documentation

6.11.2.1 dsy_dma_init()

```
void dsy_dma_init (
            void  )
```

Initializes the Direct Memory Access Peripheral used by many internal elements of libdaisy. Initializes the DMA (specifically for the modules used within the library)

6.11.2.2 dsy_system_delay()

```
void dsy_system_delay (
            uint32_t delay_ms )
```

Blocking Delay that uses the SysTick (1ms callback) to wait.

Parameters

delay_ms	Time to delay in ms

6.11.2.3 dsy_system_getnow()

```
uint32_t dsy_system_getnow ( )
```

Returns

a uint32_t value of milliseconds since the SysTick started
Note! This is a HAL_GetTick()

6.11.2.4 dsy_system_init()

```
void dsy_system_init ( )
```

Initializes Clock tree, MPU, and internal memories voltage regulators.
This function *must* be called at the beginning of any program using libdaisy Higher level daisy_ files call this through the DaisySeed object.

6.11.2.5 dsy_system_jumpto()

```
void dsy_system_jumpto (
            uint32_t addr )
```

Jump to an address within the internal memory
This may not work correctly, and may not be very useful with the single sector of memory on the stm32h750**

Parameters

addr	Address to jump to

6.11.2.6 dsy_system_jumptoqspi()

```
void dsy_system_jumptoqspi ( )
```

Jumps to the first address of the external flash chip (0x90000000) If there is no code there, the chip will likely fall through to the while() loop TODO: Documentation/Loader for using external flash coming soon.

6.12 DEVICE

Low level devices. Led drivers, codecs, etc.

Modules

- SHIFTREGISTER

  Digital shift registers.
- FLASH

  Flash memory.
- CODEC

  Audio codecs.
- LED

  LED driver devices.
- SDRAM

  SDRAM devices.

6.12.1 Detailed Description

Low level devices. Led drivers, codecs, etc.

6.13 SHIFTREGISTER

Digital shift registers.

Classes

- struct dsy_sr_4021_handle
- class ShiftRegister595

  Device Driver for 8-bit shift register.
  CD74HC595 - 8-bit serial to parallel output shift.

Enumerations

- enum {
  DSY_SR_4021_PIN_CS, DSY_SR_4021_PIN_CLK, DSY_SR_4021_PIN_DATA, DSY_SR_4021_PIN_DATA2,
  DSY_SR_4021_PIN_LAST }

Functions

- void dsy_sr_4021_init (dsy_sr_4021_handle *sr)
- void dsy_sr_4021_update (dsy_sr_4021_handle *sr)
- uint8_t dsy_sr_4021_state (dsy_sr_4021_handle *sr, uint8_t idx)

6.13.1 Detailed Description

Digital shift registers.
Device driver for the CD4021. Bit-banged serial shift input.

6.13.2 Enumeration Type Documentation

6.13.2.1 anonymous enum

```
anonymous enum
```

Pins that need to be configured to use. DATA2 only needs to be set if num_parallel is > 1

Enumerator

DSY_SR_4021_PIN_CS	CS Pin
DSY_SR_4021_PIN_CLK	CLK Pin
DSY_SR_4021_PIN_DATA	DATA pin
DSY_SR_4021_PIN_DATA2	DATA2 Pin, optional
DSY_SR_4021_PIN_LAST	Enum Last

6.13.3 Function Documentation

6.13.3.1 dsy_sr_4021_init()

```
void dsy_sr_4021_init (
            dsy_sr_4021_handle * sr )
```

Initialize CD4021 with settings from sr_4021_handle

Parameters

sr	handle to initialize

6.13.3.2 dsy_sr_4021_state()

```
uint8_t dsy_sr_4021_state (
            dsy_sr_4021_handle * sr,
            uint8_t idx )
```

Returns the state of a pin at a given index.

Parameters

∗sr	Handle containing desired pin
idx	Pin index

6.13.3.3 dsy_sr_4021_update()

```
void dsy_sr_4021_update (
            dsy_sr_4021_handle * sr )
```

Fills internal states with CD4021 data states.

Parameters

∗sr	Handle to update

6.14 FLASH

Flash memory.

Macros

- #define RESET_ENABLE_CMD 0x66
- #define RESET_MEMORY_CMD 0x99
- #define READ_ID_CMD 0x9E
- #define READ_ID_CMD2 0x9F
- #define MULTIPLE_IO_READ_ID_CMD 0xAF
- #define READ_SERIAL_FLASH_DISCO_PARAM_CMD 0x5A
- #define READ_CMD 0x03
- #define READ_4_BYTE_ADDR_CMD 0x13
- #define FAST_READ_CMD 0x0B
- #define FAST_READ_DTR_CMD 0x0D
- #define FAST_READ_4_BYTE_ADDR_CMD 0x0C
- #define DUAL_OUT_FAST_READ_CMD 0x3B
- #define DUAL_OUT_FAST_READ_DTR_CMD 0x3D
- #define DUAL_OUT_FAST_READ_4_BYTE_ADDR_CMD 0x3C
- #define DUAL_INOUT_FAST_READ_CMD 0xBB
- #define DUAL_INOUT_FAST_READ_DTR_CMD 0xBD
- #define DUAL_INOUT_FAST_READ_4_BYTE_ADDR_CMD 0xBC
- #define QUAD_OUT_FAST_READ_CMD 0x6B
- #define QUAD_OUT_FAST_READ_DTR_CMD 0x0D
- #define QUAD_OUT_FAST_READ_4_BYTE_ADDR_CMD 0x6C
- #define QUAD_INOUT_FAST_READ_CMD 0xEB
- #define QUAD_INOUT_FAST_READ_DTR_CMD 0xED
- #define QUAD_INOUT_FAST_READ_4_BYTE_ADDR_CMD 0xEC
- #define WRITE_ENABLE_CMD 0x06
- #define WRITE_DISABLE_CMD 0x04
- #define READ_STATUS_REG_CMD 0x05
- #define WRITE_STATUS_REG_CMD 0x01
- #define READ_LOCK_REG_CMD 0xE8
- #define WRITE_LOCK_REG_CMD 0xE5
- #define READ_FLAG_STATUS_REG_CMD 0x70
- #define CLEAR_FLAG_STATUS_REG_CMD 0x50
- #define READ_NONVOL_CFG_REG_CMD 0xB5
- #define WRITE_NONVOL_CFG_REG_CMD 0xB1
- #define READ_READ_PARAM_REG_CMD 0x61
- #define WRITE_READ_PARAM_REG_CMD 0xC0
- #define READ_ENHANCED_VOL_CFG_REG_CMD 0x81
- #define WRITE_ENHANCED_VOL_CFG_REG_CMD 0x85
- #define READ_EXT_ADDR_REG_CMD 0xC8
- #define WRITE_EXT_ADDR_REG_CMD 0xC5
- #define PAGE_PROG_CMD 0x02
- #define PAGE_PROG_4_BYTE_ADDR_CMD 0x12
- #define DUAL_IN_FAST_PROG_CMD 0xA2
- #define EXT_DUAL_IN_FAST_PROG_CMD 0xD2
- #define QUAD_IN_FAST_PROG_CMD 0x32
- #define EXT_QUAD_IN_FAST_PROG_CMD 0x38
- #define QUAD_IN_FAST_PROG_4_BYTE_ADDR_CMD 0x34
- #define SUBSECTOR_ERASE_CMD 0xd7
- #define SUBSECTOR_ERASE_QPI_CMD 0x20
- #define SUBSECTOR_ERASE_4_BYTE_ADDR_CMD 0x21

- #define SECTOR_ERASE_CMD 0xD8
- #define SECTOR_ERASE_4_BYTE_ADDR_CMD 0xDC
- #define BLOCK_ERASE_32K_CMD 0x52
- #define DIE_ERASE_CMD 0xC4
- #define PROG_ERASE_RESUME_CMD 0x7A
- #define PROG_ERASE_SUSPEND_CMD 0x75
- #define READ_OTP_ARRAY_CMD 0x4B
- #define PROG_OTP_ARRAY_CMD 0x42
- #define ENTER_4_BYTE_ADDR_MODE_CMD 0xB7
- #define EXIT_4_BYTE_ADDR_MODE_CMD 0xE9
- #define ENTER_QUAD_CMD 0x35
- #define EXIT_QUAD_CMD 0xF5
- #define IS25LP064A_SR_WIP ((uint8_t)0x01)

IS25LP08D Registers

- #define IS25LP064A_SR_WREN ((uint8_t)0x02)
- #define IS25LP064A_SR_SRWREN ((uint8_t)0x80)
- #define IS25LP064A_SR_QE ((uint8_t)0x40)
- #define IS25LP064A_NVCR_NBADDR ((uint16_t)0x0001)
- #define IS25LP064A_NVCR_SEGMENT ((uint16_t)0x0002)
- #define IS25LP064A_NVCR_DUAL ((uint16_t)0x0004)
- #define IS25LP064A_NVCR_QUAB ((uint16_t)0x0008)
- #define IS25LP064A_NVCR_RH ((uint16_t)0x0010)
- #define IS25LP064A_NVCR_DTRP ((uint16_t)0x0020)
- #define IS25LP064A_NVCR_ODS ((uint16_t)0x01C0)
- #define IS25LP064A_NVCR_XIP ((uint16_t)0x0E00)
- #define IS25LP064A_NVCR_NB_DUMMY ((uint16_t)0xF000)
- #define IS25LP064A_VCR_WRAP ((uint8_t)0x03)
- #define IS25LP064A_VCR_XIP ((uint8_t)0x08)
- #define IS25LP064A_VCR_NB_DUMMY ((uint8_t)0xF0)
- #define IS25LP064A_EAR_HIGHEST_SE ((uint8_t)0x03)
- #define IS25LP064A_EAR_THIRD_SEG ((uint8_t)0x02)
- #define IS25LP064A_EAR_SECOND_SEG ((uint8_t)0x01)
- #define IS25LP064A_EAR_LOWEST_SEG ((uint8_t)0x00)
- #define IS25LP064A_EVCR_ODS ((uint8_t)0x07)
- #define IS25LP064A_EVCR_RH ((uint8_t)0x10)
- #define IS25LP064A_EVCR_DTRP ((uint8_t)0x20)
- #define IS25LP064A_EVCR_DUAL ((uint8_t)0x40)
- #define IS25LP064A_EVCR_QUAD ((uint8_t)0x80)
- #define IS25LP064A_FSR_NBADDR ((uint8_t)0x01)
- #define IS25LP064A_FSR_PRERR ((uint8_t)0x02)
- #define IS25LP064A_FSR_PGSUS ((uint8_t)0x04)
- #define IS25LP064A_FSR_PGERR ((uint8_t)0x10)
- #define IS25LP064A_FSR_ERERR ((uint8_t)0x20)
- #define IS25LP064A_FSR_ERSUS ((uint8_t)0x40)
- #define IS25LP064A_FSR_READY ((uint8_t)0x80)
- #define RESET_ENABLE_CMD 0x66
- #define RESET_MEMORY_CMD 0x99
- #define READ_ID_CMD 0x9E
- #define READ_ID_CMD2 0x9F
- #define MULTIPLE_IO_READ_ID_CMD 0xAF
- #define READ_SERIAL_FLASH_DISCO_PARAM_CMD 0x5A
- #define READ_CMD 0x03
- #define READ_4_BYTE_ADDR_CMD 0x13

- #define FAST_READ_CMD 0x0B
- #define FAST_READ_DTR_CMD 0x0D
- #define FAST_READ_4_BYTE_ADDR_CMD 0x0C
- #define DUAL_OUT_FAST_READ_CMD 0x3B
- #define DUAL_OUT_FAST_READ_DTR_CMD 0x3D
- #define DUAL_OUT_FAST_READ_4_BYTE_ADDR_CMD 0x3C
- #define DUAL_INOUT_FAST_READ_CMD 0xBB
- #define DUAL_INOUT_FAST_READ_DTR_CMD 0xBD
- #define DUAL_INOUT_FAST_READ_4_BYTE_ADDR_CMD 0xBC
- #define QUAD_OUT_FAST_READ_CMD 0x6B
- #define QUAD_OUT_FAST_READ_DTR_CMD 0x0D
- #define QUAD_OUT_FAST_READ_4_BYTE_ADDR_CMD 0x6C
- #define QUAD_INOUT_FAST_READ_CMD 0xEB
- #define QUAD_INOUT_FAST_READ_DTR_CMD 0xED
- #define QUAD_INOUT_FAST_READ_4_BYTE_ADDR_CMD 0xEC
- #define WRITE_ENABLE_CMD 0x06
- #define WRITE_DISABLE_CMD 0x04
- #define READ_STATUS_REG_CMD 0x05
- #define WRITE_STATUS_REG_CMD 0x01
- #define READ_LOCK_REG_CMD 0xE8
- #define WRITE_LOCK_REG_CMD 0xE5
- #define READ_FLAG_STATUS_REG_CMD 0x70
- #define CLEAR_FLAG_STATUS_REG_CMD 0x50
- #define READ_NONVOL_CFG_REG_CMD 0xB5
- #define WRITE_NONVOL_CFG_REG_CMD 0xB1
- #define READ_READ_PARAM_REG_CMD 0x61
- #define WRITE_READ_PARAM_REG_CMD 0xC0
- #define READ_ENHANCED_VOL_CFG_REG_CMD 0x81
- #define WRITE_ENHANCED_VOL_CFG_REG_CMD 0x85
- #define READ_EXT_ADDR_REG_CMD 0xC8
- #define WRITE_EXT_ADDR_REG_CMD 0xC5
- #define PAGE_PROG_CMD 0x02
- #define PAGE_PROG_4_BYTE_ADDR_CMD 0x12
- #define DUAL_IN_FAST_PROG_CMD 0xA2
- #define EXT_DUAL_IN_FAST_PROG_CMD 0xD2
- #define QUAD_IN_FAST_PROG_CMD 0x32
- #define EXT_QUAD_IN_FAST_PROG_CMD 0x38
- #define QUAD_IN_FAST_PROG_4_BYTE_ADDR_CMD 0x34
- #define SUBSECTOR_ERASE_CMD 0xd7
- #define SUBSECTOR_ERASE_QPI_CMD 0x20
- #define SUBSECTOR_ERASE_4_BYTE_ADDR_CMD 0x21
- #define SECTOR_ERASE_CMD 0xD8
- #define SECTOR_ERASE_4_BYTE_ADDR_CMD 0xDC
- #define BLOCK_ERASE_32K_CMD 0x52
- #define DIE_ERASE_CMD 0xC4
- #define PROG_ERASE_RESUME_CMD 0x7A
- #define PROG_ERASE_SUSPEND_CMD 0x75
- #define READ_OTP_ARRAY_CMD 0x4B
- #define PROG_OTP_ARRAY_CMD 0x42
- #define ENTER_4_BYTE_ADDR_MODE_CMD 0xB7
- #define EXIT_4_BYTE_ADDR_MODE_CMD 0xE9
- #define ENTER_QUAD_CMD 0x35
- #define EXIT_QUAD_CMD 0xF5
- #define IS25LP080D_SR_WIP ((uint8_t)0x01)

IS25LP08D Registers

- #define IS25LP080D_SR_WREN ((uint8_t)0x02)
- #define IS25LP080D_SR_SRWREN ((uint8_t)0x80)
- #define IS25LP080D_SR_QE ((uint8_t)0x40)
- #define IS25LP080D_NVCR_NBADDR ((uint16_t)0x0001)
- #define IS25LP080D_NVCR_SEGMENT ((uint16_t)0x0002)
- #define IS25LP080D_NVCR_DUAL ((uint16_t)0x0004)
- #define IS25LP080D_NVCR_QUAB ((uint16_t)0x0008)
- #define IS25LP080D_NVCR_RH ((uint16_t)0x0010)
- #define IS25LP080D_NVCR_DTRP ((uint16_t)0x0020)
- #define IS25LP080D_NVCR_ODS ((uint16_t)0x01C0)
- #define IS25LP080D_NVCR_XIP ((uint16_t)0x0E00)
- #define IS25LP080D_NVCR_NB_DUMMY ((uint16_t)0xF000)
- #define IS25LP080D_VCR_WRAP ((uint8_t)0x03)
- #define IS25LP080D_VCR_XIP ((uint8_t)0x08)
- #define IS25LP080D_VCR_NB_DUMMY ((uint8_t)0xF0)
- #define IS25LP080D_EAR_HIGHEST_SE ((uint8_t)0x03)
- #define IS25LP080D_EAR_THIRD_SEG ((uint8_t)0x02)
- #define IS25LP080D_EAR_SECOND_SEG ((uint8_t)0x01)
- #define IS25LP080D_EAR_LOWEST_SEG ((uint8_t)0x00)
- #define IS25LP080D_EVCR_ODS ((uint8_t)0x07)
- #define IS25LP080D_EVCR_RH ((uint8_t)0x10)
- #define IS25LP080D_EVCR_DTRP ((uint8_t)0x20)
- #define IS25LP080D_EVCR_DUAL ((uint8_t)0x40)
- #define IS25LP080D_EVCR_QUAD ((uint8_t)0x80)
- #define IS25LP080D_FSR_NBADDR ((uint8_t)0x01)
- #define IS25LP080D_FSR_PRERR ((uint8_t)0x02)
- #define IS25LP080D_FSR_PGSUS ((uint8_t)0x04)
- #define IS25LP080D_FSR_PGERR ((uint8_t)0x10)
- #define IS25LP080D_FSR_ERERR ((uint8_t)0x20)
- #define IS25LP080D_FSR_ERSUS ((uint8_t)0x40)
- #define IS25LP080D_FSR_READY ((uint8_t)0x80)

6.14.1 Detailed Description

Flash memory.
IS25LP08D Commands

6.14.2 Macro Definition Documentation

6.14.2.1 BLOCK_ERASE_32K_CMD [1/2]

```
#define BLOCK_ERASE_32K_CMD 0x52
```
&

6.14.2.2 BLOCK_ERASE_32K_CMD [2/2]

```
#define BLOCK_ERASE_32K_CMD 0x52
```
&

6.14.2.3 CLEAR_FLAG_STATUS_REG_CMD [1/2]

```
#define CLEAR_FLAG_STATUS_REG_CMD 0x50
```
&

6.14.2.4 CLEAR_FLAG_STATUS_REG_CMD [2/2]

```
#define CLEAR_FLAG_STATUS_REG_CMD 0x50
&
```

6.14.2.5 DIE_ERASE_CMD [1/2]

```
#define DIE_ERASE_CMD 0xC4
&
```

6.14.2.6 DIE_ERASE_CMD [2/2]

```
#define DIE_ERASE_CMD 0xC4
&
```

6.14.2.7 DUAL_IN_FAST_PROG_CMD [1/2]

```
#define DUAL_IN_FAST_PROG_CMD 0xA2
&
```

6.14.2.8 DUAL_IN_FAST_PROG_CMD [2/2]

```
#define DUAL_IN_FAST_PROG_CMD 0xA2
&
```

6.14.2.9 DUAL_INOUT_FAST_READ_4_BYTE_ADDR_CMD [1/2]

```
#define DUAL_INOUT_FAST_READ_4_BYTE_ADDR_CMD 0xBC
&
```

6.14.2.10 DUAL_INOUT_FAST_READ_4_BYTE_ADDR_CMD [2/2]

```
#define DUAL_INOUT_FAST_READ_4_BYTE_ADDR_CMD 0xBC
&
```

6.14.2.11 DUAL_INOUT_FAST_READ_CMD [1/2]

```
#define DUAL_INOUT_FAST_READ_CMD 0xBB
&
```

6.14.2.12 DUAL_INOUT_FAST_READ_CMD [2/2]

```
#define DUAL_INOUT_FAST_READ_CMD 0xBB
&
```

6.14.2.13 DUAL_INOUT_FAST_READ_DTR_CMD [1/2]

```
#define DUAL_INOUT_FAST_READ_DTR_CMD 0xBD
&
```

6.14.2.14 DUAL_INOUT_FAST_READ_DTR_CMD [2/2]

```
#define DUAL_INOUT_FAST_READ_DTR_CMD 0xBD
&
```

6.14.2.15 DUAL_OUT_FAST_READ_4_BYTE_ADDR_CMD [1/2]

```
#define DUAL_OUT_FAST_READ_4_BYTE_ADDR_CMD 0x3C
&
```

6.14.2.16 DUAL_OUT_FAST_READ_4_BYTE_ADDR_CMD [2/2]

```
#define DUAL_OUT_FAST_READ_4_BYTE_ADDR_CMD 0x3C
```
&

6.14.2.17 DUAL_OUT_FAST_READ_CMD [1/2]

```
#define DUAL_OUT_FAST_READ_CMD 0x3B
```
&

6.14.2.18 DUAL_OUT_FAST_READ_CMD [2/2]

```
#define DUAL_OUT_FAST_READ_CMD 0x3B
```
&

6.14.2.19 DUAL_OUT_FAST_READ_DTR_CMD [1/2]

```
#define DUAL_OUT_FAST_READ_DTR_CMD 0x3D
```
&

6.14.2.20 DUAL_OUT_FAST_READ_DTR_CMD [2/2]

```
#define DUAL_OUT_FAST_READ_DTR_CMD 0x3D
```
&

6.14.2.21 ENTER_4_BYTE_ADDR_MODE_CMD [1/2]

```
#define ENTER_4_BYTE_ADDR_MODE_CMD 0xB7
```
4-byte Address Mode Operations

6.14.2.22 ENTER_4_BYTE_ADDR_MODE_CMD [2/2]

```
#define ENTER_4_BYTE_ADDR_MODE_CMD 0xB7
```
4-byte Address Mode Operations

6.14.2.23 ENTER_QUAD_CMD [1/2]

```
#define ENTER_QUAD_CMD 0x35
```
Quad Operations

6.14.2.24 ENTER_QUAD_CMD [2/2]

```
#define ENTER_QUAD_CMD 0x35
```
Quad Operations

6.14.2.25 EXIT_4_BYTE_ADDR_MODE_CMD [1/2]

```
#define EXIT_4_BYTE_ADDR_MODE_CMD 0xE9
```
&

6.14.2.26 EXIT_4_BYTE_ADDR_MODE_CMD [2/2]

```
#define EXIT_4_BYTE_ADDR_MODE_CMD 0xE9
```
&

6.14.2.27 EXIT_QUAD_CMD [1/2]

```
#define EXIT_QUAD_CMD 0xF5
```
&

6.14.2.28 EXIT_QUAD_CMD [2/2]

```
#define EXIT_QUAD_CMD 0xF5
```
&

6.14.2.29 EXT_DUAL_IN_FAST_PROG_CMD [1/2]

```
#define EXT_DUAL_IN_FAST_PROG_CMD 0xD2
```
&

6.14.2.30 EXT_DUAL_IN_FAST_PROG_CMD [2/2]

```
#define EXT_DUAL_IN_FAST_PROG_CMD 0xD2
```
&

6.14.2.31 EXT_QUAD_IN_FAST_PROG_CMD [1/2]

```
#define EXT_QUAD_IN_FAST_PROG_CMD 0x38
```
&

6.14.2.32 EXT_QUAD_IN_FAST_PROG_CMD [2/2]

```
#define EXT_QUAD_IN_FAST_PROG_CMD 0x38
```
&

6.14.2.33 FAST_READ_4_BYTE_ADDR_CMD [1/2]

```
#define FAST_READ_4_BYTE_ADDR_CMD 0x0C
```
&

6.14.2.34 FAST_READ_4_BYTE_ADDR_CMD [2/2]

```
#define FAST_READ_4_BYTE_ADDR_CMD 0x0C
```
&

6.14.2.35 FAST_READ_CMD [1/2]

```
#define FAST_READ_CMD 0x0B
```
&

6.14.2.36 FAST_READ_CMD [2/2]

```
#define FAST_READ_CMD 0x0B
```
&

6.14.2.37 FAST_READ_DTR_CMD [1/2]

```
#define FAST_READ_DTR_CMD 0x0D
```
&

6.14.2.38 FAST_READ_DTR_CMD [2/2]

```
#define FAST_READ_DTR_CMD 0x0D
```
&

6.14.2.39 IS25LP064A_EAR_HIGHEST_SE

```
#define IS25LP064A_EAR_HIGHEST_SE ((uint8_t)0x03)
```
Select the Highest 128Mb segment

6.14.2.40 IS25LP064A_EAR_LOWEST_SEG

```
#define IS25LP064A_EAR_LOWEST_SEG ((uint8_t)0x00)
```
Select the Lowest 128Mb segment (default)

6.14.2.41 IS25LP064A_EAR_SECOND_SEG

```
#define IS25LP064A_EAR_SECOND_SEG ((uint8_t)0x01)
```
Select the Second 128Mb segment

6.14.2.42 IS25LP064A_EAR_THIRD_SEG

```
#define IS25LP064A_EAR_THIRD_SEG ((uint8_t)0x02)
```
Select the Third 128Mb segment

6.14.2.43 IS25LP064A_EVCR_DTRP

```
#define IS25LP064A_EVCR_DTRP ((uint8_t)0x20)
```
Double transfer rate protocol

6.14.2.44 IS25LP064A_EVCR_DUAL

```
#define IS25LP064A_EVCR_DUAL ((uint8_t)0x40)
```
Dual I/O protocol

6.14.2.45 IS25LP064A_EVCR_ODS

```
#define IS25LP064A_EVCR_ODS ((uint8_t)0x07)
```
Output driver strength

6.14.2.46 IS25LP064A_EVCR_QUAD

```
#define IS25LP064A_EVCR_QUAD ((uint8_t)0x80)
```
Quad I/O protocol

6.14.2.47 IS25LP064A_EVCR_RH

```
#define IS25LP064A_EVCR_RH ((uint8_t)0x10)
```
Reset/hold

6.14.2.48 IS25LP064A_FSR_ERERR

```
#define IS25LP064A_FSR_ERERR ((uint8_t)0x20)
```
Erase error

6.14.2.49 IS25LP064A_FSR_ERSUS

```
#define IS25LP064A_FSR_ERSUS ((uint8_t)0x40)
```
Erase operation suspended

6.14.2.50 IS25LP064A_FSR_NBADDR

```
#define IS25LP064A_FSR_NBADDR ((uint8_t)0x01)
```
3-bytes or 4-bytes addressing

6.14.2.51 IS25LP064A_FSR_PGERR

```
#define IS25LP064A_FSR_PGERR ((uint8_t)0x10)
```
Program error

6.14.2.52 IS25LP064A_FSR_PGSUS

```
#define IS25LP064A_FSR_PGSUS ((uint8_t)0x04)
```
Program operation suspended

6.14.2.53 IS25LP064A_FSR_PRERR

```
#define IS25LP064A_FSR_PRERR ((uint8_t)0x02)
```
Protection error

6.14.2.54 IS25LP064A_FSR_READY

```
#define IS25LP064A_FSR_READY ((uint8_t)0x80)
```
Ready or command in progress

6.14.2.55 IS25LP064A_NVCR_DTRP

```
#define IS25LP064A_NVCR_DTRP ((uint16_t)0x0020)
```
Double transfer rate protocol

6.14.2.56 IS25LP064A_NVCR_DUAL

```
#define IS25LP064A_NVCR_DUAL ((uint16_t)0x0004)
```
Dual I/O protocol

6.14.2.57 IS25LP064A_NVCR_NB_DUMMY

```
#define IS25LP064A_NVCR_NB_DUMMY ((uint16_t)0xF000)
```
Number of dummy clock cycles

6.14.2.58 IS25LP064A_NVCR_NBADDR

```
#define IS25LP064A_NVCR_NBADDR ((uint16_t)0x0001)
```
3-bytes or 4-bytes addressing

6.14.2.59 IS25LP064A_NVCR_ODS

```
#define IS25LP064A_NVCR_ODS ((uint16_t)0x01C0)
```
Output driver strength

6.14.2.60 IS25LP064A_NVCR_QUAB

```
#define IS25LP064A_NVCR_QUAB ((uint16_t)0x0008)
```
Quad I/O protocol

6.14.2.61 IS25LP064A_NVCR_RH

```
#define IS25LP064A_NVCR_RH ((uint16_t)0x0010)
```
Reset/hold

6.14.2.62 IS25LP064A_NVCR_SEGMENT

```
#define IS25LP064A_NVCR_SEGMENT ((uint16_t)0x0002)
```
Upper or lower 128Mb segment selected by default

6.14.2.63 IS25LP064A_NVCR_XIP

```
#define IS25LP064A_NVCR_XIP ((uint16_t)0x0E00)
```
XIP mode at power-on reset

6.14.2.64 IS25LP064A_SR_QE

```
#define IS25LP064A_SR_QE ((uint8_t)0x40)
```
&

6.14.2.65 IS25LP064A_SR_SRWREN

```
#define IS25LP064A_SR_SRWREN ((uint8_t)0x80)
```
Status register write enable/disable

6.14.2.66 IS25LP064A_SR_WIP

```
#define IS25LP064A_SR_WIP ((uint8_t)0x01)
```
IS25LP08D Registers

Write in progress

6.14.2.67 IS25LP064A_SR_WREN

```
#define IS25LP064A_SR_WREN ((uint8_t)0x02)
```
Write enable latch

6.14.2.68 IS25LP064A_VCR_NB_DUMMY

```
#define IS25LP064A_VCR_NB_DUMMY ((uint8_t)0xF0)
```
Number of dummy clock cycles

6.14.2.69 IS25LP064A_VCR_WRAP

```
#define IS25LP064A_VCR_WRAP ((uint8_t)0x03)
```
Wrap

6.14.2.70 IS25LP064A_VCR_XIP

```
#define IS25LP064A_VCR_XIP ((uint8_t)0x08)
```
XIP

6.14.2.71 IS25LP080D_EAR_HIGHEST_SE

```
#define IS25LP080D_EAR_HIGHEST_SE ((uint8_t)0x03)
```
Select the Highest 128Mb segment

6.14.2.72 IS25LP080D_EAR_LOWEST_SEG

```
#define IS25LP080D_EAR_LOWEST_SEG ((uint8_t)0x00)
```
Select the Lowest 128Mb segment (default)

6.14.2.73 IS25LP080D_EAR_SECOND_SEG

```
#define IS25LP080D_EAR_SECOND_SEG ((uint8_t)0x01)
```
Select the Second 128Mb segment

6.14.2.74 IS25LP080D_EAR_THIRD_SEG

```
#define IS25LP080D_EAR_THIRD_SEG ((uint8_t)0x02)
```
Select the Third 128Mb segment

6.14.2.75 IS25LP080D_EVCR_DTRP

```
#define IS25LP080D_EVCR_DTRP ((uint8_t)0x20)
```
Double transfer rate protocol

6.14.2.76 IS25LP080D_EVCR_DUAL

```
#define IS25LP080D_EVCR_DUAL ((uint8_t)0x40)
```
Dual I/O protocol

6.14.2.77 IS25LP080D_EVCR_ODS

```
#define IS25LP080D_EVCR_ODS ((uint8_t)0x07)
```
Output driver strength

6.14.2.78 IS25LP080D_EVCR_QUAD

```
#define IS25LP080D_EVCR_QUAD ((uint8_t)0x80)
```
Quad I/O protocol

6.14.2.79 IS25LP080D_EVCR_RH

```
#define IS25LP080D_EVCR_RH ((uint8_t)0x10)
```
Reset/hold

6.14.2.80 IS25LP080D_FSR_ERERR

```
#define IS25LP080D_FSR_ERERR ((uint8_t)0x20)
```
Erase error

6.14.2.81 IS25LP080D_FSR_ERSUS

```
#define IS25LP080D_FSR_ERSUS ((uint8_t)0x40)
```
Erase operation suspended

6.14.2.82 IS25LP080D_FSR_NBADDR

```
#define IS25LP080D_FSR_NBADDR ((uint8_t)0x01)
```
3-bytes or 4-bytes addressing

6.14.2.83 IS25LP080D_FSR_PGERR

```
#define IS25LP080D_FSR_PGERR ((uint8_t)0x10)
```
Program error

6.14.2.84 IS25LP080D_FSR_PGSUS

```
#define IS25LP080D_FSR_PGSUS ((uint8_t)0x04)
```
Program operation suspended

6.14.2.85 IS25LP080D_FSR_PRERR

```
#define IS25LP080D_FSR_PRERR ((uint8_t)0x02)
```
Protection error

6.14.2.86 IS25LP080D_FSR_READY

```
#define IS25LP080D_FSR_READY ((uint8_t)0x80)
```
Ready or command in progress

6.14.2.87 IS25LP080D_NVCR_DTRP

```
#define IS25LP080D_NVCR_DTRP ((uint16_t)0x0020)
```
Double transfer rate protocol

6.14.2.88 IS25LP080D_NVCR_DUAL

```
#define IS25LP080D_NVCR_DUAL ((uint16_t)0x0004)
```
Dual I/O protocol

6.14.2.89 IS25LP080D_NVCR_NB_DUMMY

```
#define IS25LP080D_NVCR_NB_DUMMY ((uint16_t)0xF000)
```
Number of dummy clock cycles

6.14.2.90 IS25LP080D_NVCR_NBADDR

```
#define IS25LP080D_NVCR_NBADDR ((uint16_t)0x0001)
```
3-bytes or 4-bytes addressing

6.14.2.91 IS25LP080D_NVCR_ODS

```
#define IS25LP080D_NVCR_ODS ((uint16_t)0x01C0)
```
Output driver strength

6.14.2.92 IS25LP080D_NVCR_QUAB

```
#define IS25LP080D_NVCR_QUAB ((uint16_t)0x0008)
```
Quad I/O protocol

6.14.2.93 IS25LP080D_NVCR_RH

```
#define IS25LP080D_NVCR_RH ((uint16_t)0x0010)
```
Reset/hold

6.14.2.94 IS25LP080D_NVCR_SEGMENT

```
#define IS25LP080D_NVCR_SEGMENT ((uint16_t)0x0002)
```
Upper or lower 128Mb segment selected by default

6.14.2.95 IS25LP080D_NVCR_XIP

```
#define IS25LP080D_NVCR_XIP ((uint16_t)0x0E00)
```
XIP mode at power-on reset

6.14.2.96 IS25LP080D_SR_QE

```
#define IS25LP080D_SR_QE ((uint8_t)0x40)
```
&

6.14.2.97 IS25LP080D_SR_SRWREN

```
#define IS25LP080D_SR_SRWREN ((uint8_t)0x80)
```
Status register write enable/disable

6.14.2.98 IS25LP080D_SR_WIP

```
#define IS25LP080D_SR_WIP ((uint8_t)0x01)
```
IS25LP08D Registers

Status Register Write in progress

6.14.2.99 IS25LP080D_SR_WREN

```
#define IS25LP080D_SR_WREN ((uint8_t)0x02)
```
Write enable latch

6.14.2.100 IS25LP080D_VCR_NB_DUMMY

```
#define IS25LP080D_VCR_NB_DUMMY ((uint8_t)0xF0)
```
Number of dummy clock cycles

6.14.2.101 IS25LP080D_VCR_WRAP

```
#define IS25LP080D_VCR_WRAP ((uint8_t)0x03)
```
Wrap

6.14.2.102 IS25LP080D_VCR_XIP

```
#define IS25LP080D_VCR_XIP ((uint8_t)0x08)
```
XIP

6.14.2.103 MULTIPLE_IO_READ_ID_CMD [1/2]

```
#define MULTIPLE_IO_READ_ID_CMD 0xAF
```
&

6.14.2.104 MULTIPLE_IO_READ_ID_CMD [2/2]

```
#define MULTIPLE_IO_READ_ID_CMD 0xAF
```
&

6.14.2.105 PAGE_PROG_4_BYTE_ADDR_CMD [1/2]

```
#define PAGE_PROG_4_BYTE_ADDR_CMD 0x12
```
&

6.14.2.106 PAGE_PROG_4_BYTE_ADDR_CMD [2/2]

```
#define PAGE_PROG_4_BYTE_ADDR_CMD 0x12
```
&

6.14.2.107 PAGE_PROG_CMD [1/2]

```
#define PAGE_PROG_CMD 0x02
```
Program Operations

6.14.2.108 PAGE_PROG_CMD [2/2]

```
#define PAGE_PROG_CMD 0x02
```
Program Operations

6.14.2.109 PROG_ERASE_RESUME_CMD [1/2]

```
#define PROG_ERASE_RESUME_CMD 0x7A
```
&

6.14.2.110 PROG_ERASE_RESUME_CMD [2/2]

```
#define PROG_ERASE_RESUME_CMD 0x7A
```
&

6.14.2.111 PROG_ERASE_SUSPEND_CMD [1/2]

```
#define PROG_ERASE_SUSPEND_CMD 0x75
```
&

6.14.2.112 PROG_ERASE_SUSPEND_CMD [2/2]

```
#define PROG_ERASE_SUSPEND_CMD 0x75
```
&

6.14.2.113 PROG_OTP_ARRAY_CMD [1/2]

```
#define PROG_OTP_ARRAY_CMD 0x42
```
&

6.14.2.114 PROG_OTP_ARRAY_CMD [2/2]

```
#define PROG_OTP_ARRAY_CMD 0x42
```
&

6.14.2.115 QUAD_IN_FAST_PROG_4_BYTE_ADDR_CMD [1/2]

```
#define QUAD_IN_FAST_PROG_4_BYTE_ADDR_CMD 0x34
```
&

6.14.2.116 QUAD_IN_FAST_PROG_4_BYTE_ADDR_CMD [2/2]

```
#define QUAD_IN_FAST_PROG_4_BYTE_ADDR_CMD 0x34
```
&

6.14.2.117 QUAD_IN_FAST_PROG_CMD [1/2]

```
#define QUAD_IN_FAST_PROG_CMD 0x32
```
&

6.14.2.118 QUAD_IN_FAST_PROG_CMD [2/2]

```
#define QUAD_IN_FAST_PROG_CMD 0x32
```
&

6.14.2.119 QUAD_INOUT_FAST_READ_4_BYTE_ADDR_CMD [1/2]

```
#define QUAD_INOUT_FAST_READ_4_BYTE_ADDR_CMD 0xEC
```
&

6.14.2.120 QUAD_INOUT_FAST_READ_4_BYTE_ADDR_CMD [2/2]

```
#define QUAD_INOUT_FAST_READ_4_BYTE_ADDR_CMD 0xEC
```
&

6.14.2.121 QUAD_INOUT_FAST_READ_CMD [1/2]

```
#define QUAD_INOUT_FAST_READ_CMD 0xEB
```
&

6.14.2.122 QUAD_INOUT_FAST_READ_CMD [2/2]

```
#define QUAD_INOUT_FAST_READ_CMD 0xEB
```
&

6.14.2.123 QUAD_INOUT_FAST_READ_DTR_CMD [1/2]

```
#define QUAD_INOUT_FAST_READ_DTR_CMD 0xED
```
&

6.14.2.124 QUAD_INOUT_FAST_READ_DTR_CMD [2/2]

```
#define QUAD_INOUT_FAST_READ_DTR_CMD 0xED
```
&

6.14.2.125 QUAD_OUT_FAST_READ_4_BYTE_ADDR_CMD [1/2]

```
#define QUAD_OUT_FAST_READ_4_BYTE_ADDR_CMD 0x6C
```
&

6.14.2.126 QUAD_OUT_FAST_READ_4_BYTE_ADDR_CMD [2/2]

```
#define QUAD_OUT_FAST_READ_4_BYTE_ADDR_CMD 0x6C
```
&

6.14.2.127 QUAD_OUT_FAST_READ_CMD [1/2]

```
#define QUAD_OUT_FAST_READ_CMD 0x6B
```
&

6.14.2.128 QUAD_OUT_FAST_READ_CMD [2/2]

```
#define QUAD_OUT_FAST_READ_CMD 0x6B
```
&

6.14.2.129 QUAD_OUT_FAST_READ_DTR_CMD [1/2]

```
#define QUAD_OUT_FAST_READ_DTR_CMD 0x0D
```
&

6.14.2.130 QUAD_OUT_FAST_READ_DTR_CMD [2/2]

```
#define QUAD_OUT_FAST_READ_DTR_CMD 0x0D
```
&

6.14.2.131 READ_4_BYTE_ADDR_CMD [1/2]

```
#define READ_4_BYTE_ADDR_CMD 0x13
```
&

6.14.2.132 READ_4_BYTE_ADDR_CMD [2/2]

```
#define READ_4_BYTE_ADDR_CMD 0x13
```
&

6.14.2.133 READ_CMD [1/2]

```
#define READ_CMD 0x03
```
Read Operations

6.14.2.134 READ_CMD [2/2]

```
#define READ_CMD 0x03
```
Read Operations

6.14.2.135 READ_ENHANCED_VOL_CFG_REG_CMD [1/2]

```
#define READ_ENHANCED_VOL_CFG_REG_CMD 0x81
```
&

6.14.2.136 READ_ENHANCED_VOL_CFG_REG_CMD [2/2]

```
#define READ_ENHANCED_VOL_CFG_REG_CMD 0x81
```
&

6.14.2.137 READ_EXT_ADDR_REG_CMD [1/2]

```
#define READ_EXT_ADDR_REG_CMD 0xC8
```
&

6.14.2.138 READ_EXT_ADDR_REG_CMD [2/2]

```
#define READ_EXT_ADDR_REG_CMD 0xC8
```
&

6.14.2.139 READ_FLAG_STATUS_REG_CMD [1/2]

```
#define READ_FLAG_STATUS_REG_CMD 0x70
```
&

6.14.2.140 READ_FLAG_STATUS_REG_CMD [2/2]

```
#define READ_FLAG_STATUS_REG_CMD 0x70
```
&

6.14.2.141 READ_ID_CMD [1/2]

```
#define READ_ID_CMD 0x9E
```
Identification Operations

6.14.2.142 READ_ID_CMD [2/2]

```
#define READ_ID_CMD 0x9E
```
Identification Operations

6.14.2.143 READ_ID_CMD2 [1/2]

```
#define READ_ID_CMD2 0x9F
```
&

6.14.2.144 READ_ID_CMD2 [2/2]

```
#define READ_ID_CMD2 0x9F
```
&

6.14.2.145 READ_LOCK_REG_CMD [1/2]

```
#define READ_LOCK_REG_CMD 0xE8
```
&

6.14.2.146 READ_LOCK_REG_CMD [2/2]

```
#define READ_LOCK_REG_CMD 0xE8
```
&

6.14.2.147 READ_NONVOL_CFG_REG_CMD [1/2]

```
#define READ_NONVOL_CFG_REG_CMD 0xB5
```
&

6.14.2.148 READ_NONVOL_CFG_REG_CMD [2/2]

```
#define READ_NONVOL_CFG_REG_CMD 0xB5
```
&

6.14.2.149 READ_OTP_ARRAY_CMD [1/2]

```
#define READ_OTP_ARRAY_CMD 0x4B
```
One-Time Programmable Operations

6.14.2.150 READ_OTP_ARRAY_CMD [2/2]

```
#define READ_OTP_ARRAY_CMD 0x4B
```
One-Time Programmable Operations

6.14.2.151 READ_READ_PARAM_REG_CMD [1/2]

```
#define READ_READ_PARAM_REG_CMD 0x61
```
&

6.14.2.152 READ_READ_PARAM_REG_CMD [2/2]

```
#define READ_READ_PARAM_REG_CMD 0x61
```
&

6.14.2.153 READ_SERIAL_FLASH_DISCO_PARAM_CMD [1/2]

```
#define READ_SERIAL_FLASH_DISCO_PARAM_CMD 0x5A
```
&

6.14.2.154 READ_SERIAL_FLASH_DISCO_PARAM_CMD [2/2]

```
#define READ_SERIAL_FLASH_DISCO_PARAM_CMD 0x5A
```
&

6.14.2.155 READ_STATUS_REG_CMD [1/2]

```
#define READ_STATUS_REG_CMD 0x05
```
Register Operations

6.14.2.156 READ_STATUS_REG_CMD [2/2]

```
#define READ_STATUS_REG_CMD 0x05
```
Register Operations

6.14.2.157 RESET_ENABLE_CMD [1/2]

```
#define RESET_ENABLE_CMD 0x66
```
Reset Operations

6.14.2.158 RESET_ENABLE_CMD [2/2]

```
#define RESET_ENABLE_CMD 0x66
```
Reset Operations

6.14.2.159 RESET_MEMORY_CMD [1/2]

```
#define RESET_MEMORY_CMD 0x99
```
&

6.14.2.160 RESET_MEMORY_CMD [2/2]

```
#define RESET_MEMORY_CMD 0x99
```
&

6.14.2.161 SECTOR_ERASE_4_BYTE_ADDR_CMD [1/2]

```
#define SECTOR_ERASE_4_BYTE_ADDR_CMD 0xDC
```
&

6.14.2.162 SECTOR_ERASE_4_BYTE_ADDR_CMD [2/2]

```
#define SECTOR_ERASE_4_BYTE_ADDR_CMD 0xDC
```
&

6.14.2.163 SECTOR_ERASE_CMD [1/2]

```
#define SECTOR_ERASE_CMD 0xD8
```
&

6.14.2.164 SECTOR_ERASE_CMD [2/2]

```
#define SECTOR_ERASE_CMD 0xD8
```
&

6.14.2.165 SUBSECTOR_ERASE_4_BYTE_ADDR_CMD [1/2]

```
#define SUBSECTOR_ERASE_4_BYTE_ADDR_CMD 0x21
```
&

6.14.2.166 SUBSECTOR_ERASE_4_BYTE_ADDR_CMD [2/2]

```
#define SUBSECTOR_ERASE_4_BYTE_ADDR_CMD 0x21
```
&

6.14.2.167 SUBSECTOR_ERASE_CMD [1/2]

```
#define SUBSECTOR_ERASE_CMD 0xd7
```
Erase Operations

6.14.2.168 SUBSECTOR_ERASE_CMD [2/2]

```
#define SUBSECTOR_ERASE_CMD 0xd7
```
Erase Operations

6.14.2.169 SUBSECTOR_ERASE_QPI_CMD [1/2]

```
#define SUBSECTOR_ERASE_QPI_CMD 0x20
```
&

6.14.2.170 SUBSECTOR_ERASE_QPI_CMD [2/2]

```
#define SUBSECTOR_ERASE_QPI_CMD 0x20
```
&

6.14.2.171 WRITE_DISABLE_CMD [1/2]

```
#define WRITE_DISABLE_CMD 0x04
```
&

6.14.2.172 WRITE_DISABLE_CMD [2/2]

```
#define WRITE_DISABLE_CMD 0x04
```
&

6.14.2.173 WRITE_ENABLE_CMD [1/2]

```
#define WRITE_ENABLE_CMD 0x06
```
Write Operations

6.14.2.174 WRITE_ENABLE_CMD [2/2]

```
#define WRITE_ENABLE_CMD 0x06
```
Write Operations

6.14.2.175 WRITE_ENHANCED_VOL_CFG_REG_CMD [1/2]

```
#define WRITE_ENHANCED_VOL_CFG_REG_CMD 0x85
```
&

6.14.2.176 WRITE_ENHANCED_VOL_CFG_REG_CMD [2/2]

```
#define WRITE_ENHANCED_VOL_CFG_REG_CMD 0x85
```
&

6.14.2.177 WRITE_EXT_ADDR_REG_CMD [1/2]

```
#define WRITE_EXT_ADDR_REG_CMD 0xC5
```
&

6.14.2.178 WRITE_EXT_ADDR_REG_CMD [2/2]

```
#define WRITE_EXT_ADDR_REG_CMD 0xC5
```
&

6.14.2.179 WRITE_LOCK_REG_CMD [1/2]

```
#define WRITE_LOCK_REG_CMD 0xE5
```
&

6.14.2.180 WRITE_LOCK_REG_CMD [2/2]

```
#define WRITE_LOCK_REG_CMD 0xE5
```
&

6.14.2.181 WRITE_NONVOL_CFG_REG_CMD [1/2]

```
#define WRITE_NONVOL_CFG_REG_CMD 0xB1
```
&

6.14.2.182 WRITE_NONVOL_CFG_REG_CMD [2/2]

```
#define WRITE_NONVOL_CFG_REG_CMD 0xB1
```
&

6.14.2.183 WRITE_READ_PARAM_REG_CMD [1/2]

```
#define WRITE_READ_PARAM_REG_CMD 0xC0
```
&

6.14.2.184 WRITE_READ_PARAM_REG_CMD [2/2]

```
#define WRITE_READ_PARAM_REG_CMD 0xC0
```
&

6.14.2.185 WRITE_STATUS_REG_CMD [1/2]

```
#define WRITE_STATUS_REG_CMD 0x01
```
&

6.14.2.186 WRITE_STATUS_REG_CMD [2/2]

```
#define WRITE_STATUS_REG_CMD 0x01
```
&

6.15 CODEC

Audio codecs.

Classes

- struct codec_frame_t

Typedefs

- typedef void(∗ sa_audio_callback) (codec_frame_t ∗, codec_frame_t ∗, size_t)

Functions

- void codec_ak4556_init (dsy_gpio_pin reset_pin)
- void codec_pcm3060_init (dsy_i2c_handle ∗hi2c)
- uint8_t codec_wm8731_init (dsy_i2c_handle ∗hi2c, uint8_t mcu_is_master, int32_t sample_rate, uint8_↩ t bitdepth)
- uint8_t codec_wm8731_enter_bypass (dsy_i2c_handle ∗hi2c)
- uint8_t codec_wm8731_exit_bypass (dsy_i2c_handle ∗hi2c)

6.15.1 Detailed Description

Audio codecs.
WM8731 Codec framework.
Driver for the WM8731 Codec.
Driver for the PCM3060 Codec.
Driver for the AK4556 Stereo Codec.

6.15.2 Typedef Documentation

6.15.2.1 sa_audio_callback

```
typedef void(* sa_audio_callback) (codec_frame_t *, codec_frame_t *, size_t)
```
&

6.15.3 Function Documentation

6.15.3.1 codec_ak4556_init()

```
void codec_ak4556_init (
            dsy_gpio_pin reset_pin )
```
Resets the AK4556

Parameters

reset_pin	should be a dsy_gpio_pin that is connected to the RST pin of the AK4556

6.15.3.2 codec_pcm3060_init()

```
void codec_pcm3060_init (
            dsy_i2c_handle * hi2c )
```

Resets the PCM060

Parameters

∗hi2c	array of pins handling i2c?

6.15.3.3 codec_wm8731_enter_bypass()

```
uint8_t codec_wm8731_enter_bypass (
            dsy_i2c_handle * hi2c )
```

Put codec into bypass mode

Parameters

∗hi2c	pins handling i2c

6.15.3.4 codec_wm8731_exit_bypass()

```
uint8_t codec_wm8731_exit_bypass (
            dsy_i2c_handle * hi2c )
```

Take codec out of bypass mode

Parameters

∗hi2c	pins handling i2c

6.15.3.5 codec_wm8731_init()

```
uint8_t codec_wm8731_init (
            dsy_i2c_handle * hi2c,
            uint8_t mcu_is_master,
            int32_t sample_rate,
            uint8_t bitdepth )
```

Resets the WM8731

Parameters

∗hi2c	array of pins handling i2c?
mcu_is_master	&
sample_rate	Sample rate to run codec at
bitdepth	Bit depth to run codec at

6.16 LED

LED driver devices.

Classes

- struct color

Enumerations

- enum {
  LED_COLOR_RED, LED_COLOR_GREEN, LED_COLOR_BLUE, LED_COLOR_WHITE,
  LED_COLOR_PURPLE, LED_COLOR_CYAN, LED_COLOR_GOLD, LED_COLOR_OFF,
  LED_COLOR_LAST }

Functions

- void dsy_led_driver_init (dsy_i2c_handle *dsy_i2c, uint8_t *addr, uint8_t addr_cnt)
- void dsy_led_driver_update ()
- void dsy_led_driver_set_led (uint8_t idx, float bright)
- color * dsy_led_driver_color_by_name (uint8_t name)

6.16.1 Detailed Description

LED driver devices.
Device driver for PCA9685 16-channel 12-bit PWM generator.

6.16.2 Enumeration Type Documentation

6.16.2.1 anonymous enum

```
anonymous enum
```

Different Led colors

Enumerator

LED_COLOR_RED	&
LED_COLOR_GREEN	&
LED_COLOR_BLUE	&
LED_COLOR_WHITE	&
LED_COLOR_PURPLE	&
LED_COLOR_CYAN	&
LED_COLOR_GOLD	&
LED_COLOR_OFF	&
LED_COLOR_LAST	&

6.16.3 Function Documentation

6.16.3.1 dsy_led_driver_color_by_name()

```
color* dsy_led_driver_color_by_name (
            uint8_t name )
```

Passing in one of the preset colors will return a pointer to a color struct

Parameters

name	Preset color

6.16.3.2 dsy_led_driver_init()

```
void dsy_led_driver_init (
            dsy_i2c_handle * dsy_i2c,
            uint8_t * addr,
            uint8_t addr_cnt )
```

Initializes the LED Driver(s) on the specified I2C Bus

Parameters

∗dsy_i2c	should be any dsy_i2c_handle with pins and speed configured.
addr	is either a pointer to 1 device address, or an array of addresses for multiple devices
addr_cnt	is the number of addresses passed in (use '1' for a single device)

6.16.3.3 dsy_led_driver_set_led()

```
void dsy_led_driver_set_led (
            uint8_t idx,
            float bright )
```

sets the LED at the index to the specified brightness (0-1) Index is sequential so device 0 will have idx 0-15, while device 1 will have idx 16-31, etc.

Parameters

idx	Index
bright	Brightness

6.16.3.4 dsy_led_driver_update()

```
void dsy_led_driver_update ( )
```

Updates the LED Driver with the values set using the set function
Currently only updates one driver at a time due to the time it takes to update all of the devices. This can likely be set up to use DMA so that the function doesn't block for so long.

6.17 SDRAM

SDRAM devices.

Classes

- struct dsy_sdram_handle

Macros

- #define DSY_SDRAM_DATA __attribute__((section(".sdram_data")))
- #define DSY_SDRAM_BSS __attribute__((section(".sdram_bss")))

Enumerations

- enum { DSY_SDRAM_OK, DSY_SDRAM_ERR }
- enum dsy_sdram_state { DSY_SDRAM_STATE_ENABLE, DSY_SDRAM_STATE_DISABLE, DSY_SDRAM_STATE_LAST }
- enum dsy_sdram_pin { DSY_SDRAM_PIN_SDNWE, DSY_SDRAM_PIN_LAST }

Functions

- uint8_t dsy_sdram_init (dsy_sdram_handle *dsy_hsdram)

6.17.1 Detailed Description

SDRAM devices.

6.17.2 Macro Definition Documentation

6.17.2.1 DSY_SDRAM_BSS

```
#define DSY_SDRAM_BSS __attribute__((section(".sdram_bss")))
```

Variables placed here will not be initialized.
Usage
E.g. int DSY_SDRAM_BSS uninitialized_var;

6.17.2.2 DSY_SDRAM_DATA

```
#define DSY_SDRAM_DATA __attribute__((section(".sdram_data")))
```

Usage:
E.g. int DSY_SDRAM_DATA initialized_var = 1;

6.17.3 Enumeration Type Documentation

6.17.3.1 anonymous enum

```
anonymous enum
```

Enumerator

DSY_SDRAM_OK	&
DSY_SDRAM_ERR	&

6.17.3.2 dsy_sdram_pin

`enum dsy_sdram_pin`
This is PH5 on Daisy

Enumerator

DSY_SDRAM_PIN_SDNWE	&
DSY_SDRAM_PIN_LAST	&

6.17.3.3 dsy_sdram_state

`enum dsy_sdram_state`
Determines whether chip is initialized, and activated.

Enumerator

DSY_SDRAM_STATE_ENABLE	&
DSY_SDRAM_STATE_DISABLE	&
DSY_SDRAM_STATE_LAST	&

6.17.4 Function Documentation

6.17.4.1 dsy_sdram_init()

```
uint8_t dsy_sdram_init (
            dsy_sdram_handle * dsy_hsdram )
```
Initializes the SDRAM peripheral

6.18 BOARDS

Daisy devices. Pod, seed, etc.

Classes

- struct daisy::daisy_field
- class daisy::DaisyPatch

  Class that handles initializing all of the hardware specific to the Daisy Patch Board.
  Helper funtions are also in place to provide easy access to built-in controls and peripherals.
- class daisy::DaisyPetal

  Helpers and hardware definitions for daisy petal.
- class daisy::DaisyPod

  Class that handles initializing all of the hardware specific to the Daisy Patch Board.
  Helper funtions are also in place to provide easy access to built-in controls and peripherals.
- class daisy::DaisySeed

  This is the higher-level interface for the Daisy board.
  All basic peripheral configuration/initialization is setup here.

Enumerations

- enum { daisy::SW_2, daisy::SW_1, daisy::SW_3, daisy::SW_LAST }
- enum {
  daisy::KNOB_1, daisy::KNOB_3, daisy::KNOB_5, daisy::KNOB_2,
  daisy::KNOB_4, daisy::KNOB_6, daisy::KNOB_7, daisy::KNOB_8,
  daisy::KNOB_LAST }
- enum {
  CV_1, daisy::CV_2, daisy::CV_3, daisy::CV_4,
  daisy::CV_LAST }
- enum {
  daisy::LED_KEY_A8, daisy::LED_KEY_A7, daisy::LED_KEY_A6, daisy::LED_KEY_A5,
  daisy::LED_KEY_A4, daisy::LED_KEY_A3, daisy::LED_KEY_A2, daisy::LED_KEY_A1,
  daisy::LED_KEY_B1, daisy::LED_KEY_B2, daisy::LED_KEY_B3, daisy::LED_KEY_B4,
  daisy::LED_KEY_B5, daisy::LED_KEY_B6, daisy::LED_KEY_B7, daisy::LED_KEY_B8,
  daisy::LED_KNOB_1, daisy::LED_KNOB_2, daisy::LED_KNOB_3, daisy::LED_KNOB_4,
  daisy::LED_KNOB_5, daisy::LED_KNOB_6, daisy::LED_KNOB_7, daisy::LED_KNOB_8,
  daisy::LED_SW_1, daisy::LED_SW_2, daisy::LED_LAST }

Functions

- FORCE_INLINE float s162f (int16_t x)
- FORCE_INLINE int16_t f2s16 (float x)
- FORCE_INLINE float s242f (int32_t x)
- FORCE_INLINE int32_t f2s24 (float x)
- FORCE_INLINE void daisy::daisy_field_init (daisy_field *p)

6.18.1 Detailed Description

Daisy devices. Pod, seed, etc.

6.18.2 Enumeration Type Documentation

6.18.2.1 anonymous enum

`anonymous enum`

enums for controls, etc.

Enumerator

SW_2	tactile switch
SW_1	tactile switch
SW_3	toggle
SW_LAST	&

6.18.2.2 anonymous enum

`anonymous enum`

All knobs connect to ADC1_INP10 via CD4051 mux

Enumerator

KNOB_1	&
KNOB_3	&
KNOB_5	&
KNOB_2	&
KNOB_4	&
KNOB_6	&
KNOB_7	&
KNOB_8	&
KNOB_LAST	&

6.18.2.3 anonymous enum

`anonymous enum`

Enumerator

CV_2	Connected to ADC1_INP17
CV_3	Connected to ADC1_INP15
CV_4	Connected to ADC1_INP4
CV_LAST	Connected to ADC1_INP11 &

6.18.2.4 anonymous enum

`anonymous enum`

Enumerator

LED_KEY_A8	&
LED_KEY_A7	&
LED_KEY_A6	&
LED_KEY_A5	&
LED_KEY_A4	&
LED_KEY_A3	&
LED_KEY_A2	&
LED_KEY_A1	&

Enumerator

LED_KEY_B1	&
LED_KEY_B2	&
LED_KEY_B3	&
LED_KEY_B4	&
LED_KEY_B5	&
LED_KEY_B6	&
LED_KEY_B7	&
LED_KEY_B8	&
LED_KNOB↩_1	&
LED_KNOB↩_2	&
LED_KNOB↩_3	&
LED_KNOB↩_4	&
LED_KNOB↩_5	&
LED_KNOB↩_6	&
LED_KNOB↩_7	&
LED_KNOB↩_8	&
LED_SW_1	&
LED_SW_2	&
LED_LAST	&

6.18.3 Function Documentation

6.18.3.1 daisy_field_init()

```
FORCE_INLINE void daisy::daisy_field_init (
            daisy_field * p )
```

Initializes daisy field

Parameters

p	daisy_field struct to initialize

< &
< &
< &
< &
< &
< &
< &
< &
< &
< &

< &
< &
< &
< &
< &
< &
< &
< &

6.18.3.2 f2s16()

```
FORCE_INLINE int16_t f2s16 (
            float x )
```

& < close to 1.0f-LSB at 16 bit
< - (1 - LSB)
< close to 1.0f-LSB at 16 bit
< - (1 - LSB)
< close to 1.0f-LSB at 16 bit
< close to 1.0f-LSB at 16 bit
< (2 ** 15) - 1

6.18.3.3 f2s24()

```
FORCE_INLINE int32_t f2s24 (
            float x )
```

& < close to 1.0f-LSB at 16 bit
< - (1 - LSB)
< close to 1.0f-LSB at 16 bit
< - (1 - LSB)
< close to 1.0f-LSB at 16 bit
< close to 1.0f-LSB at 16 bit
< 2 ** 23

6.18.3.4 s162f()

```
FORCE_INLINE float s162f (
            int16_t x )
```

Scales float by 1/(2 ^ 15)

Parameters

x	Number to be scaled.

Returns

Scaled number.

< 1 / (2** 15)

6.18.3.5 s242f()

```
FORCE_INLINE float s242f (
            int32_t x )
```

< 2 ** 23
< 2 ** 23
< 1 / (2 ** 23)

6.19 UTILITY

General utilities. Ringbuffers, LED colors, OLED stuff, etc.

Classes

- struct dsy_gpio_pin
- struct DSY_SD_CardInfoTypeDef
- class daisy::Color
- struct FontDef
- class daisy::RingBuffer< T, size >
- class daisy::RingBuffer< T, 0 >
- struct WAV_FormatTypeDef

Macros

- #define DMA_BUFFER_MEM_SECTION __attribute__((section(".sram1_bss")))
- #define DTCM_MEM_SECTION __attribute__((section(".dtcmram_bss")))
- #define BSP_SD_CardInfo DSY_SD_CardInfoTypeDef
- #define MSD_OK ((uint8_t)0x00)
- #define MSD_ERROR ((uint8_t)0x01)
- #define MSD_ERROR_SD_NOT_PRESENT ((uint8_t)0x02)
- #define SD_TRANSFER_OK ((uint8_t)0x00)
- #define SD_TRANSFER_BUSY ((uint8_t)0x01)
- #define SD_PRESENT ((uint8_t)0x01)
- #define SD_NOT_PRESENT ((uint8_t)0x00)
- #define SD_DATATIMEOUT ((uint32_t)100000000)

Enumerations

- enum dsy_gpio_port {
  DSY_GPIOA, DSY_GPIOB, DSY_GPIOC, DSY_GPIOD,
  DSY_GPIOE, DSY_GPIOF, DSY_GPIOG, DSY_GPIOH,
  DSY_GPIOI, DSY_GPIOJ, DSY_GPIOK, **DSY_GPIOX**,
  DSY_GPIO_LAST }

Functions

- FORCE_INLINE float cube (float x)
- FORCE_INLINE dsy_gpio_pin dsy_pin (dsy_gpio_port port, uint8_t pin)
- FORCE_INLINE uint8_t dsy_pin_cmp (dsy_gpio_pin *a, dsy_gpio_pin *b)
- uint8_t BSP_SD_Init (void)
- uint8_t BSP_SD_ITConfig (void)
- uint8_t BSP_SD_ReadBlocks (uint32_t *pData, uint32_t ReadAddr, uint32_t NumOfBlocks, uint32_t Timeout)
- uint8_t BSP_SD_WriteBlocks (uint32_t *pData, uint32_t WriteAddr, uint32_t NumOfBlocks, uint32_t Timeout)
- uint8_t BSP_SD_ReadBlocks_DMA (uint32_t *pData, uint32_t ReadAddr, uint32_t NumOfBlocks)
- uint8_t BSP_SD_WriteBlocks_DMA (uint32_t *pData, uint32_t WriteAddr, uint32_t NumOfBlocks)
- uint8_t BSP_SD_Erase (uint32_t StartAddr, uint32_t EndAddr)
- uint8_t BSP_SD_GetCardState (void)
- void BSP_SD_GetCardInfo (DSY_SD_CardInfoTypeDef *CardInfo)
- uint8_t BSP_SD_IsDetected (void)
- void BSP_SD_AbortCallback (void)
- void BSP_SD_WriteCpltCallback (void)
- void BSP_SD_ReadCpltCallback (void)
- GPIO_TypeDef * dsy_hal_map_get_port (dsy_gpio_pin *p)
- uint16_t dsy_hal_map_get_pin (dsy_gpio_pin *p)
- I2C_HandleTypeDef * dsy_hal_map_get_i2c (dsy_i2c_handle *p)
- void dsy_get_unique_id (uint32_t *w0, uint32_t *w1, uint32_t *w2)

Variables

- I2C_HandleTypeDef hi2c1
- I2C_HandleTypeDef hi2c2
- I2C_HandleTypeDef hi2c3
- I2C_HandleTypeDef hi2c4
- FontDef Font_6x8
- FontDef Font_7x10
- FontDef Font_11x18
- FontDef Font_16x26

6.19.1 Detailed Description

General utilities. Ringbuffers, LED colors, OLED stuff, etc.

6.19.2 Macro Definition Documentation

6.19.2.1 BSP_SD_CardInfo

```
#define BSP_SD_CardInfo DSY_SD_CardInfoTypeDef
```
&

6.19.2.2 DMA_BUFFER_MEM_SECTION

```
#define DMA_BUFFER_MEM_SECTION __attribute__((section(".sram1_bss")))
```
Macro for area of memory that is configured as cacheless This should be used primarily for DMA buffers, and the like.

6.19.2.3 DTCM_MEM_SECTION

```
#define DTCM_MEM_SECTION __attribute__((section(".dtcmram_bss")))
```
THE DTCM RAM section is also non-cached. However, is not suitable for DMA transfers. Performance is on par with internal SRAM w/ cache enabled.

6.19.2.4 MSD_ERROR

```
#define MSD_ERROR ((uint8_t)0x01)
```
&

6.19.2.5 MSD_ERROR_SD_NOT_PRESENT

```
#define MSD_ERROR_SD_NOT_PRESENT ((uint8_t)0x02)
```
&

6.19.2.6 MSD_OK

```
#define MSD_OK ((uint8_t)0x00)
```
&

6.19.2.7 SD_DATATIMEOUT

```
#define SD_DATATIMEOUT ((uint32_t)100000000)
```
&

6.19.2.8 SD_NOT_PRESENT

```
#define SD_NOT_PRESENT ((uint8_t)0x00)
```
&

6.19.2.9 SD_PRESENT

```
#define SD_PRESENT ((uint8_t)0x01)
```
&

6.19.2.10 SD_TRANSFER_BUSY

```
#define SD_TRANSFER_BUSY ((uint8_t)0x01)
```
&

6.19.2.11 SD_TRANSFER_OK

```
#define SD_TRANSFER_OK ((uint8_t)0x00)
```
&

6.19.3 Enumeration Type Documentation

6.19.3.1 dsy_gpio_port

```
enum dsy_gpio_port
```
Enums and a simple struct for defining a hardware pin on the MCU These correlate with the stm32 datasheet, and are used to configure the hardware.

Enumerator

DSY_GPIOA	&
DSY_GPIOB	&
DSY_GPIOC	&
DSY_GPIOD	&
DSY_GPIOE	&
DSY_GPIOF	&
DSY_GPIOG	&
DSY_GPIOH	&
DSY_GPIOI	&
DSY_GPIOJ	&
DSY_GPIOK	&
DSY_GPIO_LAST	This is a non-existant port for unsupported bits of hardware.

6.19.4 Function Documentation

6.19.4.1 BSP_SD_AbortCallback()

```
void BSP_SD_AbortCallback (
            void  )
```
These functions can be modified in case the current settings (e.g. DMA stream) need to be changed for specific application needs /n
Abort the callback

6.19.4.2 BSP_SD_Erase()

```
uint8_t BSP_SD_Erase (
            uint32_t StartAddr,
            uint32_t EndAddr )
```

Erase a section of memory

Parameters

StartAddr	Address to start erasing at
EndAddr	Address to stop erasing at

Returns

card state, ERROR, etc.

6.19.4.3 BSP_SD_GetCardInfo()

```
void BSP_SD_GetCardInfo (
            DSY_SD_CardInfoTypeDef * CardInfo )
```

Parameters

∗CardInfo	Pointer to write card info to

Parameters

CardInfo	&

6.19.4.4 BSP_SD_GetCardState()

```
uint8_t BSP_SD_GetCardState (
            void  )
```

Returns

card state, ERROR, etc.

6.19.4.5 BSP_SD_Init()

```
uint8_t BSP_SD_Init (
            void  )
```

Returns

card state, ERROR, etc.

6.19.4.6 BSP_SD_IsDetected()

```
uint8_t BSP_SD_IsDetected (
            void  )
```

Returns

Is card detected

6.19.4.7 BSP_SD_ITConfig()

```
uint8_t BSP_SD_ITConfig (
            void  )
```

Returns

card state, ERROR, etc.

6.19.4.8 BSP_SD_ReadBlocks()

```
uint8_t BSP_SD_ReadBlocks (
            uint32_t * pData,
            uint32_t ReadAddr,
            uint32_t NumOfBlocks,
            uint32_t Timeout )
```

Parameters

∗pData	&
ReadAddr	Address to read from
NumOfBlocks	Number of blocks to be read
Timeout	Timeout len in ms

Returns

OK ERROR, etc.

6.19.4.9 BSP_SD_ReadBlocks_DMA()

```
uint8_t BSP_SD_ReadBlocks_DMA (
            uint32_t * pData,
            uint32_t ReadAddr,
            uint32_t NumOfBlocks )
```

No timeout

Parameters

∗pData	&
ReadAddr	Address to read from
NumOfBlocks	Number of blocks to be read

Returns

card state, ERROR, etc.

6.19.4.10 BSP_SD_ReadCpltCallback()

```
void BSP_SD_ReadCpltCallback (
            void  )
```

Write complete callback

6.19.4.11 BSP_SD_WriteBlocks()

```
uint8_t BSP_SD_WriteBlocks (
```

```
            uint32_t * pData,
            uint32_t WriteAddr,
            uint32_t NumOfBlocks,
            uint32_t Timeout )
```

Parameters

∗pData	&
WriteAddr	Address to write to
NumOfBlocks	Number of blocks to be written
Timeout	Timeout len in ms

Returns

card state, ERROR, etc.

6.19.4.12 BSP_SD_WriteBlocks_DMA()

```
uint8_t BSP_SD_WriteBlocks_DMA (
            uint32_t * pData,
            uint32_t WriteAddr,
            uint32_t NumOfBlocks )
```

No timeout

Parameters

∗pData	&
WriteAddr	Address to write to
NumOfBlocks	Number of blocks to be read

Returns

card state, ERROR, etc.

6.19.4.13 BSP_SD_WriteCpltCallback()

```
void BSP_SD_WriteCpltCallback (
            void  )
```

Read complete callback

6.19.4.14 cube()

```
FORCE_INLINE float cube (
            float x )
```

Computes cube.

Parameters

x	Number to be cubed

Returns

x ^ 3

6.19.4.15 dsy_get_unique_id()

```
void dsy_get_unique_id (
            uint32_t * w0,
            uint32_t * w1,
            uint32_t * w2 )
```

Returns 96-bit Unique ID of the MCU

Author

shensley

Date

May 2020 fills the three pointer arguments with the unique ID of the MCU.

Parameters

*w0	First pointer
*w1	Second pointer
*w2	Third pointer

6.19.4.16 dsy_hal_map_get_i2c()

```
I2C_HandleTypeDef* dsy_hal_map_get_i2c (
            dsy_i2c_handle * p )
```

Parameters

*p	dsy_i2c_handle to get

Returns

The I2C_HandleTypeDef for the given *p

6.19.4.17 dsy_hal_map_get_pin()

```
uint16_t dsy_hal_map_get_pin (
            dsy_gpio_pin * p )
```

Parameters

*p	Pin pin to get

Returns

HAL GPIO Pin as used in the HAL from a dsy_gpio_pin input.

6.19.4.18 dsy_hal_map_get_port()

```
GPIO_TypeDef* dsy_hal_map_get_port (
            dsy_gpio_pin * p )
```

Parameters

*p	Pin pin to get

Returns

HAL GPIO_TypeDef as used in the HAL from a dsy_gpio_pin input.

6.19.4.19 dsy_pin()

```
FORCE_INLINE dsy_gpio_pin dsy_pin (
            dsy_gpio_port port,
            uint8_t pin )
```

Helper for creating pins from port/pin combos easily

6.19.4.20 dsy_pin_cmp()

```
FORCE_INLINE uint8_t dsy_pin_cmp (
            dsy_gpio_pin * a,
            dsy_gpio_pin * b )
```

Helper for testing sameness of two dsy_gpio_pins

Returns

1 if same, 0 if different

6.19.5 Variable Documentation

6.19.5.1 Font_11x18

```
FontDef Font_11x18
```

&

6.19.5.2 Font_16x26

```
FontDef Font_16x26
```

&

6.19.5.3 Font_6x8

```
FontDef Font_6x8
```

These are the different sizes of fonts (width x height in pixels per character)

6.19.5.4 Font_7x10

```
FontDef Font_7x10
```

&

6.19.5.5 hi2c1

```
I2C_HandleTypeDef hi2c1
```

global structs, and helper functions for interfacing with the stm32 HAL library while it remains a dependancy. This file should only be included from source files (c/cpp) Including it from a header within libdaisy would expose the entire HAL to the users. This should be an option for users, but should not be required. externs of HAL handles...

6.19.5.6 hi2c2

`I2C_HandleTypeDef hi2c2`
externs of HAL handles...

6.19.5.7 hi2c3

`I2C_HandleTypeDef hi2c3`
externs of HAL handles...

6.19.5.8 hi2c4

`I2C_HandleTypeDef hi2c4`
externs of HAL handles...

6.20 USBD_CDC_IF

Usb VCP device module.

Modules

- USBD_CDC_IF_Exported_Defines

  Defines.
- USBD_CDC_IF_Exported_Types

  Types.
- USBD_CDC_IF_Exported_Macros

  Aliases.
- USBD_CDC_IF_Exported_Variables

  Public variables.
- USBD_CDC_IF_Exported_FunctionsPrototype

  Public functions declaration.

6.20.1 Detailed Description

Usb VCP device module.

6.21 USBD_CDC_IF_Exported_Defines

Defines.
Defines.

6.22 USBD_CDC_IF_Exported_Types

Types.

Typedefs

- typedef void(∗ CDC_ReceiveCallback) (uint8_t ∗buf, uint32_t ∗size)

6.22.1 Detailed Description

Types.

6.22.2 Typedef Documentation

6.22.2.1 CDC_ReceiveCallback

```
typedef void(* CDC_ReceiveCallback) (uint8_t *buf, uint32_t *size)
```

Parameters

buf	buffer
size	buffer size

6.23 USBD_CDC_IF_Exported_Macros

Aliases.
Aliases.

6.24 USBD_CDC_IF_Exported_Variables

Public variables.

Variables

- USBD_CDC_ItfTypeDef USBD_Interface_fops_FS
- USBD_CDC_ItfTypeDef USBD_Interface_fops_HS

6.24.1 Detailed Description

Public variables.

6.24.2 Variable Documentation

6.24.2.1 USBD_Interface_fops_FS

```
USBD_CDC_ItfTypeDef USBD_Interface_fops_FS
```

CDC Interface callback.

6.24.2.2 USBD_Interface_fops_HS

```
USBD_CDC_ItfTypeDef USBD_Interface_fops_HS
```

CDC Interface callback.

6.25 USBD_CDC_IF_Exported_FunctionsPrototype

Public functions declaration.

Functions

- void CDC_Set_Rx_Callback_FS (CDC_ReceiveCallback cb)
- uint8_t CDC_Transmit_FS (uint8_t ∗Buf, uint16_t Len)
- uint8_t CDC_Transmit_HS (uint8_t ∗Buf, uint16_t Len)

6.25.1 Detailed Description

Public functions declaration.

6.25.2 Function Documentation

6.25.2.1 CDC_Set_Rx_Callback_FS()

```
void CDC_Set_Rx_Callback_FS (
            CDC_ReceiveCallback cb )
```

&

6.25.2.2 CDC_Transmit_FS()

```
uint8_t CDC_Transmit_FS (
            uint8_t * Buf,
            uint16_t Len )
```

&

6.25.2.3 CDC_Transmit_HS()

```
uint8_t CDC_Transmit_HS (
            uint8_t * Buf,
            uint16_t Len )
```

&

6.26 USBD_CONF

Configuration file for Usb otg low level driver.

Modules

- USBD_CONF_Exported_Variables

  Public variables.
- USBD_CONF_Exported_Defines

  Defines for configuration of the Usb device.
- USBD_CONF_Exported_Macros

  Aliases.
- USBD_CONF_Exported_Types

  Types.
- USBD_CONF_Exported_FunctionsPrototype

  Declaration of public functions for Usb device.

6.26.1 Detailed Description

Configuration file for Usb otg low level driver.

6.27 USBD_CONF_Exported_Variables

Public variables.
Public variables.

6.28 USBD_CONF_Exported_Defines

Defines for configuration of the Usb device.

Macros

- #define USBD_MAX_NUM_INTERFACES 1U
- #define USBD_MAX_NUM_CONFIGURATION 1U
- #define USBD_MAX_STR_DESC_SIZ 512U
- #define USBD_SUPPORT_USER_STRING 0U
- #define USBD_DEBUG_LEVEL 3U
- #define USBD_LPM_ENABLED 0U
- #define USBD_SELF_POWERED 1U
- #define DEVICE_FS 0
- #define DEVICE_HS 1

6.28.1 Detailed Description

Defines for configuration of the Usb device.

6.28.2 Macro Definition Documentation

6.28.2.1 DEVICE_FS

```
#define DEVICE_FS 0
```

FS and HS identification

6.28.2.2 DEVICE_HS

```
#define DEVICE_HS 1
```

&

6.28.2.3 USBD_DEBUG_LEVEL

```
#define USBD_DEBUG_LEVEL 3U
```

&

6.28.2.4 USBD_LPM_ENABLED

```
#define USBD_LPM_ENABLED 0U
```

&

6.28.2.5 USBD_MAX_NUM_CONFIGURATION

```
#define USBD_MAX_NUM_CONFIGURATION 1U
```

&

6.28.2.6 USBD_MAX_NUM_INTERFACES

```
#define USBD_MAX_NUM_INTERFACES 1U
```

&

6.28.2.7 USBD_MAX_STR_DESC_SIZ

```
#define USBD_MAX_STR_DESC_SIZ 512U
```

&

6.28.2.8 USBD_SELF_POWERED

```
#define USBD_SELF_POWERED 1U
```

&

6.28.2.9 USBD_SUPPORT_USER_STRING

```
#define USBD_SUPPORT_USER_STRING 0U
```

&

6.29 USBD_CONF_Exported_Macros

Aliases.

Macros

- #define USBD_malloc malloc
- #define USBD_free free
- #define USBD_memset memset
- #define USBD_memcpy memcpy
- #define USBD_Delay HAL_Delay
- #define USBD_UsrLog(...)
- #define USBD_ErrLog(...)
- #define USBD_DbgLog(...)

6.29.1 Detailed Description

Aliases.

6.29.2 Macro Definition Documentation

6.29.2.1 USBD_DbgLog

```
#define USBD_DbgLog(
               ...  )
```

Value:

```
    printf("DEBUG : ");  \
    printf(__VA_ARGS__); \
    printf("\n");
```

&

6.29.2.2 USBD_Delay

```
#define USBD_Delay HAL_Delay
```

Alias for delay.

6.29.2.3 USBD_ErrLog

```
#define USBD_ErrLog(
               ...  )
```

Value:

```
    printf("ERROR: ");   \
    printf(__VA_ARGS__); \
    printf("\n");
```

&

6.29.2.4 USBD_free

```
#define USBD_free free
```

Alias for memory release.

6.29.2.5 USBD_malloc

```
#define USBD_malloc malloc
```

Alias for memory allocation.

6.29.2.6 USBD_memcpy

```
#define USBD_memcpy memcpy
```

Alias for memory copy.

6.29.2.7 USBD_memset

```
#define USBD_memset memset
```

Alias for memory set.

6.29.2.8 USBD_UsrLog

```
#define USBD_UsrLog(
               ...  )
```

Value:

```
    printf(__VA_ARGS__); \
    printf("\n");
```

&

6.30 USBD_CONF_Exported_Types

Types.
Types.

6.31 USBD_CONF_Exported_FunctionsPrototype

Declaration of public functions for Usb device.
Declaration of public functions for Usb device.

6.32 USBD_DESC

Usb device descriptors module.

Modules

- USBD_DESC_Exported_Constants

  Constants.
- USBD_DESC_Exported_Defines

  Defines.
- USBD_DESC_Exported_TypesDefinitions

  Types.
- USBD_DESC_Exported_Macros

  Aliases.
- USBD_DESC_Exported_Variables

  Public variables.
- USBD_DESC_Exported_FunctionsPrototype

  Public functions declaration.

6.32.1 Detailed Description

Usb device descriptors module.

6.33 USBD_DESC_Exported_Constants

Constants.

Macros

- #define DEVICE_ID1 (UID_BASE)
- #define DEVICE_ID2 (UID_BASE + 0x4)
- #define DEVICE_ID3 (UID_BASE + 0x8)
- #define USB_SIZ_STRING_SERIAL 0x1A

6.33.1 Detailed Description

Constants.

6.33.2 Macro Definition Documentation

6.33.2.1 DEVICE_ID1

```
#define DEVICE_ID1 (UID_BASE)
```
&

6.33.2.2 DEVICE_ID2

```
#define DEVICE_ID2 (UID_BASE + 0x4)
```
&

6.33.2.3 DEVICE_ID3

```
#define DEVICE_ID3 (UID_BASE + 0x8)
```
&

6.33.2.4 USB_SIZ_STRING_SERIAL

```
#define USB_SIZ_STRING_SERIAL 0x1A
```
&

6.34 USBD_DESC_Exported_Defines

Defines.
Defines.

6.35 USBD_DESC_Exported_TypesDefinitions

Types.
Types.

6.36 USBD_DESC_Exported_Macros

Aliases.
Aliases.

6.37 USBD_DESC_Exported_Variables

Public variables.

Variables

- USBD_DescriptorsTypeDef HS_Desc
- USBD_DescriptorsTypeDef FS_Desc

6.37.1 Detailed Description

Public variables.

6.37.2 Variable Documentation

6.37.2.1 FS_Desc

`USBD_DescriptorsTypeDef FS_Desc`
Descriptor for the Usb device.

6.37.2.2 HS_Desc

`USBD_DescriptorsTypeDef HS_Desc`
Descriptor for the Usb device.

6.38 USBD_DESC_Exported_FunctionsPrototype

Public functions declaration.
Public functions declaration.

6.39 Externals

6.40 STM32_USB_OTG_DEVICE_LIBRARY

For Usb device.

Modules

- USBD_CDC_IF

  Usb VCP device module.
- USBD_DESC

  Usb device descriptors module.

6.40.1 Detailed Description

For Usb device.
< Define to prevent recursive inclusion -------------------------------------

6.41 USBD_OTG_DRIVER

Modules

- USBD_CONF

  Configuration file for Usb otg low level driver.

6.41.1 Detailed Description

Chapter 7

Namespace Documentation

7.1 daisy Namespace Reference

Hardware defines and helpers for daisy field platform.

Classes

- struct AdcChannelConfig
- class AdcHandle
- class AnalogControl

  Hardware Interface for control inputs
  Primarily designed for ADC input controls such as
  potentiometers, and control voltage.
  .
- class Color
- struct ControlChangeEvent
- struct daisy_field
- class DaisyPatch

  Class that handles initializing all of the hardware specific to the Daisy Patch Board.
  Helper funtions are also in place to provide easy access to built-in controls and peripherals.
- class DaisyPetal

  Helpers and hardware definitions for daisy petal.
- class DaisyPod

  Class that handles initializing all of the hardware specific to the Daisy Patch Board.
  Helper funtions are also in place to provide easy access to built-in controls and peripherals.
- class DaisySeed

  This is the higher-level interface for the Daisy board.
  All basic peripheral configuration/initialization is setup here.
- class Encoder

  Generic Class for handling Quadrature Encoders
  Inspired/influenced by Mutable Instruments (pichenettes) Encoder classes.
- class GateIn

  Generic Class for handling gate inputs through GPIO.
- class Led

  LED Class providing simple Software PWM ability, etc
  Eventually this will work with hardware PWM, and external LED Driver devices as well.
- struct MidiEvent
- class MidiHandler

  Simple MIDI Handler
  Parses bytes from an input into valid MidiEvents.
  The MidiEvents fill a FIFO queue that the user can pop messages from.
- struct NoteOnEvent

- class OledDisplay
- class Parameter
- class RgbLed
- class RingBuffer
- class RingBuffer< T, 0 >
- class SdmmcHandler
- struct SdmmcHandlerInit
- class SpiHandle
- class Switch
- class UartHandler
- struct WavFileInfo
- class WavPlayer

Enumerations

- enum { SW_2, SW_1, SW_3, SW_LAST }
- enum {
  KNOB_1, KNOB_3, KNOB_5, KNOB_2,
  KNOB_4, KNOB_6, KNOB_7, KNOB_8,
  KNOB_LAST }
- enum {
  CV_1, CV_2, CV_3, CV_4,
  CV_LAST }
- enum {
  LED_KEY_A8, LED_KEY_A7, LED_KEY_A6, LED_KEY_A5,
  LED_KEY_A4, LED_KEY_A3, LED_KEY_A2, LED_KEY_A1,
  LED_KEY_B1, LED_KEY_B2, LED_KEY_B3, LED_KEY_B4,
  LED_KEY_B5, LED_KEY_B6, LED_KEY_B7, LED_KEY_B8,
  LED_KNOB_1, LED_KNOB_2, LED_KNOB_3, LED_KNOB_4,
  LED_KNOB_5, LED_KNOB_6, LED_KNOB_7, LED_KNOB_8,
  LED_SW_1, LED_SW_2, LED_LAST }
- enum MidiMessageType {
  NoteOff, NoteOn, PolyphonicKeyPressure, ControlChange,
  ProgramChange, ChannelPressure, PitchBend, MessageLast }
- enum SdmmcMode { SDMMC_MODE_FATFS }
- enum SdmmcBitWidth { SDMMC_BITS_1, SDMMC_BITS_4 }
- enum SdmmcSpeed { SDMMC_SPEED_400KHZ, SDMMC_SPEED_12MHZ }
- enum SpiPeriph { SPI_PERIPH_1, SPI_PERIPH_3, SPI_PERIPH_6 }
- enum SpiPin { SPI_PIN_CS, SPI_PIN_SCK, SPI_PIN_MOSI, SPI_PIN_MISO }

Functions

- FORCE_INLINE void daisy_field_init (daisy_field *p)

Variables

- const size_t kUartMaxBufferSize = 32

7.1.1 Detailed Description

Hardware defines and helpers for daisy field platform.

Chapter 8

Class Documentation

8.1 daisy::AdcChannelConfig Struct Reference

`#include <per_adc.h>`

Public Types

- enum MuxPin { MUX_SEL_0, MUX_SEL_1, MUX_SEL_2, MUX_SEL_LAST }

Public Member Functions

- void InitSingle (dsy_gpio_pin pin)
- void InitMux (dsy_gpio_pin adc_pin, dsy_gpio_pin mux_0, dsy_gpio_pin mux_1, dsy_gpio_pin mux_2, size_t channels)

Public Attributes

- dsy_gpio pin_
- dsy_gpio mux_pin_ [MUX_SEL_LAST]
- uint8_t mux_channels_

8.1.1 Detailed Description

Configuration Structure for a given channel

8.1.2 Member Enumeration Documentation

8.1.2.1 MuxPin

`enum daisy::AdcChannelConfig::MuxPin`

Which pin to use for multiplexing

Enumerator

MUX_SEL_0	&
MUX_SEL_1	&
MUX_SEL_2	&
MUX_SEL_LAST	&

8.1.3 Member Function Documentation

8.1.3.1 InitMux()

```
void daisy::AdcChannelConfig::InitMux (
            dsy_gpio_pin adc_pin,
            dsy_gpio_pin mux_0,
            dsy_gpio_pin mux_1,
            dsy_gpio_pin mux_2,
            size_t channels )
```

Initializes a single ADC pin as a Multiplexed ADC. Requires a CD4051 Multiplexor connected to the pin Internal Callbacks handle the pin addressing.

Parameters

channels	must be 1-8
mux_0	First mux pin
mux_1	Second mux pin
mux_2	Third mux pin
adc_pin	&

8.1.3.2 InitSingle()

```
void daisy::AdcChannelConfig::InitSingle (
            dsy_gpio_pin pin )
```

Initializes a single ADC pin as an ADC.

Parameters

pin	Pin to init.

8.1.4 Member Data Documentation

8.1.4.1 mux_channels_

```
uint8_t daisy::AdcChannelConfig::mux_channels_
```

&

8.1.4.2 mux_pin_

```
dsy_gpio daisy::AdcChannelConfig::mux_pin_[MUX_SEL_LAST]
```

&

8.1.4.3 pin_

```
dsy_gpio daisy::AdcChannelConfig::pin_
```

&

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

- src/per_adc.h

8.2 daisy::AdcHandle Class Reference

`#include <per_adc.h>`

Public Types

- enum OverSampling {
  OVS_NONE, OVS_4, OVS_8, OVS_16,
  OVS_32, OVS_64, OVS_128, OVS_256,
  OVS_512, OVS_1024, OVS_LAST }

Public Member Functions

- void Init (AdcChannelConfig *cfg, size_t num_channels, OverSampling ovs=OVS_32)
- void Start ()
- void Stop ()
- uint16_t Get (uint8_t chn)
- uint16_t * GetPtr (uint8_t chn)
- float GetFloat (uint8_t chn)
- uint16_t GetMux (uint8_t chn, uint8_t idx)
- uint16_t * GetMuxPtr (uint8_t chn, uint8_t idx)
- float GetMuxFloat (uint8_t chn, uint8_t idx)

8.2.1 Detailed Description

Handler for analog to digital conversion

8.2.2 Member Enumeration Documentation

8.2.2.1 OverSampling

`enum daisy::AdcHandle::OverSampling`

Supported oversampling amounts

Enumerator

OVS_NONE	&
OVS_4	&
OVS_8	&
OVS_16	&
OVS_32	&
OVS_64	&
OVS_128	&
OVS_256	&
OVS_512	&
OVS_1024	&
OVS_LAST	&

8.2.3 Member Function Documentation

8.2.3.1 Get()

```
uint16_t daisy::AdcHandle::Get (
            uint8_t chn )
```

Single channel getter

Parameters

chn	channel to get

Returns

Converted value

8.2.3.2 GetFloat()

```
float daisy::AdcHandle::GetFloat (
            uint8_t chn )
```

Get floating point from single channel

Parameters

chn	Channel to get from

Returns

Floating point converted value

8.2.3.3 GetMux()

```
uint16_t daisy::AdcHandle::GetMux (
            uint8_t chn,
            uint8_t idx )
```

Getters for multiplexed inputs on a single channel (up to 8 per ADC input).

Parameters

chn	Channel to get from
idx	&

Returns

data

8.2.3.4 GetMuxFloat()

```
float daisy::AdcHandle::GetMuxFloat (
            uint8_t chn,
            uint8_t idx )
```

Getters for multiplexed inputs on a single channel (up to 8 per ADC input).

Parameters

chn	Channel to get from
idx	&

Returns

Floating point data

8.2.3.5 GetMuxPtr()

```
uint16_t* daisy::AdcHandle::GetMuxPtr (
            uint8_t chn,
            uint8_t idx )
```

Getters for multiplexed inputs on a single channel. (Max 8 per chan)

Parameters

chn	Channel to get from
idx	&

Returns

Pointer to data

8.2.3.6 GetPtr()

```
uint16_t* daisy::AdcHandle::GetPtr (
            uint8_t chn )
```

Get pointer to a value from a single channel

Parameters

chn	

Returns

Pointer to converted value

8.2.3.7 Init()

```
void daisy::AdcHandle::Init (
            AdcChannelConfig * cfg,
            size_t num_channels,
            OverSampling ovs = OVS_32 )
```

Initializes the ADC with the pins passed in.

Parameters

∗cfg	an array of AdcChannelConfig of the desired channel
num_channels	number of ADC channels to initialize
ovs	Oversampling amount - Defaults to OVS_32

8.2.3.8 Start()

`void daisy::AdcHandle::Start ( )`
Starts reading from the ADC

8.2.3.9 Stop()

`void daisy::AdcHandle::Stop ( )`
Stops reading from the ADC
The documentation for this class was generated from the following file:

- src/per_adc.h

8.3 daisy::AnalogControl Class Reference

Hardware Interface for control inputs
Primarily designed for ADC input controls such as
potentiometers, and control voltage.
.
`#include <hid_ctrl.h>`

Public Member Functions

- AnalogControl ()
- ~AnalogControl ()
- void Init (uint16_t *adcptr, float sr, bool flip=false, bool invert=false, float slew_seconds=0.002f)
- void InitBipolarCv (uint16_t *adcptr, float sr)
- float Process ()
- float Value () const

8.3.1 Detailed Description

Hardware Interface for control inputs
Primarily designed for ADC input controls such as
potentiometers, and control voltage.
.

Author

Stephen Hensley

Date

November 2019

8.3.2 Constructor & Destructor Documentation

8.3.2.1 AnalogControl()

`daisy::AnalogControl::AnalogControl ( ) [inline]`
Constructor

8.3.2.2 ~AnalogControl()

`daisy::AnalogControl::~AnalogControl ( ) [inline]`
destructor

8.3.3 Member Function Documentation

8.3.3.1 Init()

```
void daisy::AnalogControl::Init (
            uint16_t * adcptr,
            float sr,
            bool flip = false,
            bool invert = false,
            float slew_seconds = 0.002f )
```

Initializes the control

Parameters

∗adcptr	is a pointer to the raw adc read value – This can be acquired with dsy_adc_get_rawptr(), or dsy_adc_get_mux_rawptr()
sr	is the samplerate in Hz that the Process function will be called at.
flip	determines whether the input is flipped (i.e. 1.f - input) or not before being processed.1
invert	determines whether the input is inverted (i.e. -1.f ∗ input) or note before being processed.
slew_seconds	is the slew time in seconds that it takes for the control to change to a new value.

8.3.3.2 InitBipolarCv()

```
void daisy::AnalogControl::InitBipolarCv (
            uint16_t * adcptr,
            float sr )
```

This Initializes the AnalogControl for a -5V to 5V inverted input All of the Init details are the same otherwise

Parameters

∗adcptr	Pointer to analog digital converter
sr	Audio engine sample rate

8.3.3.3 Process()

```
float daisy::AnalogControl::Process ( )
```

Filters, and transforms a raw ADC read into a normalized range. this should be called at the rate of specified by samplerate at Init time.
Default Initializations will return 0.0 -> 1.0 Bi-polar CV inputs will return -1.0 -> 1.0

8.3.3.4 Value()

```
float daisy::AnalogControl::Value ( ) const  [inline]
```

Returns the current stored value, without reprocessing
The documentation for this class was generated from the following file:

- src/hid_ctrl.h

8.4 codec_frame_t Struct Reference

```
#include <dev_codec_wm8731_frame.h>
```

Public Attributes

- short l
- short r

8.4.1 Detailed Description

&

8.4.2 Member Data Documentation

8.4.2.1 l

```
short codec_frame_t::l
```
&

8.4.2.2 r

```
short codec_frame_t::r
```
&
The documentation for this struct was generated from the following file:

- src/dev_codec_wm8731_frame.h

8.5 color Struct Reference

```
#include <dev_leddriver.h>
```

Public Attributes

- uint16_t red
- uint16_t green
- uint16_t blue

8.5.1 Detailed Description

Simple color struct Different from util_color only in type (0-4095 vs 0-1)
This could easily be migrated to work with those instead.

8.5.2 Member Data Documentation

8.5.2.1 blue

```
uint16_t color::blue
```
&

8.5.2.2 green

```
uint16_t color::green
```
&

8.5.2.3 red

`uint16_t color::red`
&
The documentation for this struct was generated from the following file:

- src/dev_leddriver.h

8.6 daisy::Color Class Reference

`#include <util_color.h>`

Public Types

- enum PresetColor {
  RED, GREEN, BLUE, WHITE,
  PURPLE, CYAN, GOLD, OFF,
  LAST }

Public Member Functions

- void Init (PresetColor c)
- void Init (float red, float green, float blue)
- float Red () const
- float Green () const
- float Blue () const

8.6.1 Detailed Description

Class for handling simple colors

8.6.2 Member Enumeration Documentation

8.6.2.1 PresetColor

`enum daisy::Color::PresetColor`
List of colors that have a preset RGB value

Enumerator

RED	&
GREEN	&
BLUE	&
WHITE	&
PURPLE	&
CYAN	&
GOLD	&
OFF	&
LAST	&

8.6.3 Member Function Documentation

8.6.3.1 Blue()

```
float daisy::Color::Blue ( ) const  [inline]
```

Returns the 0-1 value for Blue

8.6.3.2 Green()

```
float daisy::Color::Green ( ) const  [inline]
```

Returns the 0-1 value for Green

8.6.3.3 Init() [1/2]

```
void daisy::Color::Init (
            float red,
            float green,
            float blue )
```

Initializes the Color with a specific RGB value red, green, and blue should be floats between 0 and 1

Parameters

red	Red value
green	Green value
blue	Blue value

8.6.3.4 Init() [2/2]

```
void daisy::Color::Init (
            PresetColor c )
```

Initializes the Color with a given preset.

Parameters

c	Color to init to

8.6.3.5 Red()

```
float daisy::Color::Red ( ) const  [inline]
```

Returns the 0-1 value for Red

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

- src/util_color.h

8.7 daisy::ControlChangeEvent Struct Reference

```
#include <hid_midi.h>
```

Public Attributes

- int channel
- uint8_t control_number
- uint8_t value

8.7.1 Detailed Description

Struct containing control number, and value for a given channel. Can be made from MidiEvent

8.7.2 Member Data Documentation

8.7.2.1 channel

```
int daisy::ControlChangeEvent::channel
```
&

8.7.2.2 control_number

```
uint8_t daisy::ControlChangeEvent::control_number
```
&

8.7.2.3 value

```
uint8_t daisy::ControlChangeEvent::value
```
&
The documentation for this struct was generated from the following file:

- src/hid_midi.h

8.8 daisy::daisy_field Struct Reference

```
#include <daisy_field.h>
```

Public Attributes

- daisy::DaisySeed seed
- daisy::Switch switches [SW_LAST]
- dsy_gpio gate_in
- dsy_gpio gate_out
- dsy_sr_4021_handle keyboard_sr
- AnalogControl knobs [KNOB_LAST]
- AnalogControl cvs [CV_LAST]

8.8.1 Detailed Description

Struct containing hardware defines and daisy seed

8.8.2 Member Data Documentation

8.8.2.1 cvs

```
AnalogControl daisy::daisy_field::cvs[CV_LAST]
```
Array of cv ins

8.8.2.2 gate_in

```
dsy_gpio daisy::daisy_field::gate_in
```
Gate input.

8.8.2.3 gate_out

```
dsy_gpio daisy::daisy_field::gate_out
```
Gate output

8.8.2.4 keyboard_sr

`dsy_sr_4021_handle daisy::daisy_field::keyboard_sr`
Keyboard shift register

8.8.2.5 knobs

`AnalogControl daisy::daisy_field::knobs[KNOB_LAST]`
Array of hardware knobs

8.8.2.6 seed

`daisy::DaisySeed daisy::daisy_field::seed`
Daisy seed

8.8.2.7 switches

`daisy::Switch daisy::daisy_field::switches[SW_LAST]`
Array of hardware switches
The documentation for this struct was generated from the following file:

- src/daisy_field.h

8.9 daisy::DaisyPatch Class Reference

Class that handles initializing all of the hardware specific to the Daisy Patch Board.
Helper funtions are also in place to provide easy access to built-in controls and peripherals.
`#include <daisy_patch.h>`

Public Types

- enum Ctrl {
  CTRL_1, **CTRL_2**, **CTRL_3**, **CTRL_4**,
  CTRL_LAST }
- enum GateInput { **GATE_IN_1**, **GATE_IN_2**, GATE_IN_LAST }

Public Member Functions

- DaisyPatch ()
- ~DaisyPatch ()
- void Init ()
- void DelayMs (size_t del)
- void SetAudioBlockSize (size_t size)
- void StartAudio (dsy_audio_mc_callback cb)
- void ChangeAudioCallback (dsy_audio_callback cb)
- void StartAdc ()
- float AudioSampleRate ()
- size_t AudioBlockSize ()
- float AudioCallbackRate ()
- void UpdateAnalogControls ()
- float GetCtrlValue (Ctrl k)
- void DebounceControls ()
- void DisplayControls (bool invert=true)

Public Attributes

- DaisySeed seed
- Encoder encoder
- AnalogControl controls [CTRL_LAST]
- GateIn gate_input [GATE_IN_LAST]
- MidiHandler midi
- OledDisplay display
- dsy_gpio gate_output

8.9.1 Detailed Description

Class that handles initializing all of the hardware specific to the Daisy Patch Board.
Helper funtions are also in place to provide easy access to built-in controls and peripherals.

Author

Stephen Hensley

Date

November 2019

8.9.2 Member Enumeration Documentation

8.9.2.1 Ctrl

`enum daisy::DaisyPatch::Ctrl`
Enum of Ctrls to represent the four CV/Knob combos on the Patch

8.9.2.2 GateInput

`enum daisy::DaisyPatch::GateInput`
Daisy patch gate inputs

Enumerator

GATE_IN_LAST	<

8.9.3 Constructor & Destructor Documentation

8.9.3.1 DaisyPatch()

`daisy::DaisyPatch::DaisyPatch ( ) [inline]`
Constructor

8.9.3.2 ~DaisyPatch()

`daisy::DaisyPatch::~DaisyPatch ( ) [inline]`
Destructor

8.9.4 Member Function Documentation

8.9.4.1 AudioBlockSize()

```
size_t daisy::DaisyPatch::AudioBlockSize ( )
```

Get block size

8.9.4.2 AudioCallbackRate()

```
float daisy::DaisyPatch::AudioCallbackRate ( )
```

Get callback rate

8.9.4.3 AudioSampleRate()

```
float daisy::DaisyPatch::AudioSampleRate ( )
```

Get sample rate

8.9.4.4 ChangeAudioCallback()

```
void daisy::DaisyPatch::ChangeAudioCallback (
            dsy_audio_callback cb )
```

Change to a different callback function.

Parameters

cb	New callback function.

8.9.4.5 DebounceControls()

```
void daisy::DaisyPatch::DebounceControls ( )
```

Debounce analog controls. Call at same rate as reading controls.

8.9.4.6 DelayMs()

```
void daisy::DaisyPatch::DelayMs (
            size_t del )
```

Wait some ms before going on.

Parameters

del	Delay time in ms.

8.9.4.7 DisplayControls()

```
void daisy::DaisyPatch::DisplayControls (
            bool invert = true )
```

Control the display

8.9.4.8 GetCtrlValue()

```
float daisy::DaisyPatch::GetCtrlValue (
            Ctrl k )
```

Get value for a partiular control

Parameters

k	Which control to get

8.9.4.9 Init()

```
void daisy::DaisyPatch::Init ( )
```

Initializes the daisy seed, and patch hardware.

8.9.4.10 SetAudioBlockSize()

```
void daisy::DaisyPatch::SetAudioBlockSize (
            size_t size )
```

Audio Block size defaults to 48. Change it using this function before StartingAudio

Parameters

size	Audio block size.

8.9.4.11 StartAdc()

```
void daisy::DaisyPatch::StartAdc ( )
```

Start analog to digital conversion.

8.9.4.12 StartAudio()

```
void daisy::DaisyPatch::StartAudio (
            dsy_audio_mc_callback cb )
```

Start audio output.

Parameters

cb	Audio callback function

8.9.4.13 UpdateAnalogControls()

```
void daisy::DaisyPatch::UpdateAnalogControls ( )
```

Call at same rate as reading controls for good reads.

8.9.5 Member Data Documentation

8.9.5.1 controls

```
AnalogControl daisy::DaisyPatch::controls[CTRL_LAST]
```

Array of controls

8.9.5.2 display

```
OledDisplay daisy::DaisyPatch::display
```

&

8.9.5.3 encoder

```
Encoder daisy::DaisyPatch::encoder
```

Encoder object

8.9.5.4 gate_input

`GateIn daisy::DaisyPatch::gate_input[GATE_IN_LAST]`
Gate inputs

8.9.5.5 gate_output

`dsy_gpio daisy::DaisyPatch::gate_output`
&

8.9.5.6 midi

`MidiHandler daisy::DaisyPatch::midi`
Handles midi

8.9.5.7 seed

`DaisySeed daisy::DaisyPatch::seed`
Seed object
The documentation for this class was generated from the following file:

- src/daisy_patch.h

8.10 daisy::DaisyPetal Class Reference

Helpers and hardware definitions for daisy petal.
`#include <daisy_petal.h>`

Public Types

- enum Sw {
  SW_1, SW_2, SW_3, SW_4,
  SW_5, SW_6, SW_7, SW_LAST }
- enum Knob {
  KNOB_1, KNOB_2, KNOB_3, KNOB_4,
  KNOB_5, KNOB_6, KNOB_LAST }
- enum RingLed {
  RING_LED_1, RING_LED_2, RING_LED_3, RING_LED_4,
  RING_LED_5, RING_LED_6, RING_LED_7, RING_LED_8,
  RING_LED_LAST }
- enum FootswitchLed {
  FOOTSWITCH_LED_1, FOOTSWITCH_LED_2, FOOTSWITCH_LED_3, FOOTSWITCH_LED_4,
  FOOTSWITCH_LED_LAST }

Public Member Functions

- DaisyPetal ()
- ~DaisyPetal ()
- void Init ()
- void DelayMs (size_t del)
- void SetAudioBlockSize (size_t size)
- void StartAudio (dsy_audio_callback cb)
- void ChangeAudioCallback (dsy_audio_callback cb)
- void StartAdc ()
- float AudioSampleRate ()
- size_t AudioBlockSize ()

- float AudioCallbackRate ()
- void UpdateAnalogControls ()
- float GetKnobValue (Knob k)
- float GetExpression ()
- void DebounceControls ()
- void ClearLeds ()
- void UpdateLeds ()
- void SetRingLed (RingLed idx, float r, float g, float b)
- void SetFootswitchLed (FootswitchLed idx, float bright)

Public Attributes

- DaisySeed seed
- Encoder encoder
- AnalogControl knob [KNOB_LAST]
- AnalogControl expression
- Switch switches [SW_LAST]
- RgbLed ring_led [8]
- Led footswitch_led [4]

8.10.1 Detailed Description

Helpers and hardware definitions for daisy petal.

8.10.2 Member Enumeration Documentation

8.10.2.1 FootswitchLed

`enum daisy::DaisyPetal::FootswitchLed`
footswitch leds

Enumerator

FOOTSWITCH_LED_1	&
FOOTSWITCH_LED_2	&
FOOTSWITCH_LED_3	&
FOOTSWITCH_LED_4	&
FOOTSWITCH_LED_LAST	&

8.10.2.2 Knob

`enum daisy::DaisyPetal::Knob`
Knobs

Enumerator

KNOB_1	&
KNOB_2	&
KNOB_3	&
KNOB_4	&
KNOB_5	&
KNOB_6	&
KNOB_LAST	&

8.10.2.3 RingLed

`enum daisy::DaisyPetal::RingLed`
Leds in ringled

Enumerator

RING_LED_1	&
RING_LED_2	&
RING_LED_3	&
RING_LED_4	&
RING_LED_5	&
RING_LED_6	&
RING_LED_7	&
RING_LED_8	&
RING_LED_LAST	&

8.10.2.4 Sw

`enum daisy::DaisyPetal::Sw`
Switches

Enumerator

SW_1	Footswitch
SW_2	Footswitch
SW_3	Footswitch
SW_4	Footswitch
SW_5	Toggle
SW_6	Toggle
SW_7	Toggle
SW_LAST	Last enum item

8.10.3 Constructor & Destructor Documentation

8.10.3.1 DaisyPetal()

`daisy::DaisyPetal::DaisyPetal ( ) [inline]`
Constructor

8.10.3.2 ~DaisyPetal()

`daisy::DaisyPetal::~DaisyPetal ( ) [inline]`
Destructor

8.10.4 Member Function Documentation

8.10.4.1 AudioBlockSize()

```
size_t daisy::DaisyPetal::AudioBlockSize ( )
```

Get audio block size

8.10.4.2 AudioCallbackRate()

```
float daisy::DaisyPetal::AudioCallbackRate ( )
```

Get callback rate

8.10.4.3 AudioSampleRate()

```
float daisy::DaisyPetal::AudioSampleRate ( )
```

Device audio sample rate.

8.10.4.4 ChangeAudioCallback()

```
void daisy::DaisyPetal::ChangeAudioCallback (
            dsy_audio_callback cb )
```

Change callback function

Parameters

cb	New callback function.

8.10.4.5 ClearLeds()

```
void daisy::DaisyPetal::ClearLeds ( )
```

Turn all leds off

8.10.4.6 DebounceControls()

```
void daisy::DaisyPetal::DebounceControls ( )
```

Debounce inputs.

8.10.4.7 DelayMs()

```
void daisy::DaisyPetal::DelayMs (
            size_t del )
```

Wait before moving on.

Parameters

del	Delay time in ms.

8.10.4.8 GetExpression()

```
float daisy::DaisyPetal::GetExpression ( )
```

&

8.10.4.9 GetKnobValue()

```
float daisy::DaisyPetal::GetKnobValue (
            Knob k )
```

Get value per knob.

Parameters

k	Which knob to get

Returns

Floating point knob position.

8.10.4.10 Init()

```
void daisy::DaisyPetal::Init ( )
```

Initialize daisy petal

8.10.4.11 SetAudioBlockSize()

```
void daisy::DaisyPetal::SetAudioBlockSize (
            size_t size )
```

Set size of audio blocks.

Parameters

size	Audio block size

8.10.4.12 SetFootswitchLed()

```
void daisy::DaisyPetal::SetFootswitchLed (
            FootswitchLed idx,
            float bright )
```

Set footswitch LED

Parameters

idx	Led Index
bright	Brightness

8.10.4.13 SetRingLed()

```
void daisy::DaisyPetal::SetRingLed (
            RingLed idx,
            float r,
            float g,
            float b )
```

Set ring LED colors

Parameters

idx	Index to set
r	Red value
g	Green value
b	Blue value

8.10.4.14 StartAdc()

```
void daisy::DaisyPetal::StartAdc ( )
```

Start analog to digital conversion.

8.10.4.15 StartAudio()

```
void daisy::DaisyPetal::StartAudio (
            dsy_audio_callback cb )
```

Start audio callback

Parameters

cb	Callback function.

8.10.4.16 UpdateAnalogControls()

```
void daisy::DaisyPetal::UpdateAnalogControls ( )
```

Call at the same frequency as controls are read for stable readings.

8.10.4.17 UpdateLeds()

```
void daisy::DaisyPetal::UpdateLeds ( )
```

Update Leds to values you had set.

8.10.5 Member Data Documentation

8.10.5.1 encoder

```
Encoder daisy::DaisyPetal::encoder
```

&

8.10.5.2 expression

```
AnalogControl daisy::DaisyPetal::expression
```

&

8.10.5.3 footswitch_led

```
Led daisy::DaisyPetal::footswitch_led[4]
```

&

8.10.5.4 knob

```
AnalogControl daisy::DaisyPetal::knob[KNOB_LAST]
```

&

8.10.5.5 ring_led

```
RgbLed daisy::DaisyPetal::ring_led[8]
```

&

8.10.5.6 seed

```
DaisySeed daisy::DaisyPetal::seed
```

&

8.10.5.7 switches

`Switch daisy::DaisyPetal::switches[SW_LAST]`
< &
The documentation for this class was generated from the following file:

- src/daisy_petal.h

8.11 daisy::DaisyPod Class Reference

Class that handles initializing all of the hardware specific to the Daisy Patch Board.
Helper funtions are also in place to provide easy access to built-in controls and peripherals.
`#include <daisy_pod.h>`

Public Types

- enum Sw { **BUTTON_1**, BUTTON_2, BUTTON_LAST }
- enum Knob { **KNOB_1**, KNOB_2, KNOB_LAST }

Public Member Functions

- void Init ()
- void DelayMs (size_t del)
- void SetAudioBlockSize (size_t size)
- void StartAudio (dsy_audio_callback cb)
- void ChangeAudioCallback (dsy_audio_callback cb)
- void StartAdc ()
- float AudioSampleRate ()
- size_t AudioBlockSize ()
- float AudioCallbackRate ()
- void UpdateAnalogControls ()
- float GetKnobValue (Knob k)
- void DebounceControls ()
- void ClearLeds ()
- void UpdateLeds ()

Public Attributes

- DaisySeed seed
- Encoder encoder
- AnalogControl knob1
- AnalogControl knob2
- AnalogControl * knobs [KNOB_LAST]
- Switch button1
- Switch button2
- Switch * buttons [BUTTON_LAST]
- RgbLed led1
- RgbLed led2

8.11.1 Detailed Description

Class that handles initializing all of the hardware specific to the Daisy Patch Board.
Helper funtions are also in place to provide easy access to built-in controls and peripherals.

Author

Stephen Hensley

Date

November 2019

8.11.2 Member Enumeration Documentation

8.11.2.1 Knob

`enum daisy::DaisyPod::Knob`

Knobs

Enumerator

KNOB_2	&
KNOB_LAST	&

8.11.2.2 Sw

`enum daisy::DaisyPod::Sw`

Switches

Enumerator

BUTTON_2	&
BUTTON_LAST	&

8.11.3 Member Function Documentation

8.11.3.1 AudioBlockSize()

```
size_t daisy::DaisyPod::AudioBlockSize ( )
```

Get block size

8.11.3.2 AudioCallbackRate()

```
float daisy::DaisyPod::AudioCallbackRate ( )
```

Get callback rate

8.11.3.3 AudioSampleRate()

```
float daisy::DaisyPod::AudioSampleRate ( )
```

Get sample rate

8.11.3.4 ChangeAudioCallback()

```
void daisy::DaisyPod::ChangeAudioCallback (
            dsy_audio_callback cb )
```

Switch callback functions

Parameters

cb	New callback function.

8.11.3.5 ClearLeds()

```
void daisy::DaisyPod::ClearLeds ( )
```

Reset Leds

8.11.3.6 DebounceControls()

```
void daisy::DaisyPod::DebounceControls ( )
```

&

8.11.3.7 DelayMs()

```
void daisy::DaisyPod::DelayMs (
            size_t del )
```

Wait for a bit

Parameters

del	Time to wait in ms.

8.11.3.8 GetKnobValue()

```
float daisy::DaisyPod::GetKnobValue (
            Knob k )
```

&

8.11.3.9 Init()

```
void daisy::DaisyPod::Init ( )
```

Init related stuff.

8.11.3.10 SetAudioBlockSize()

```
void daisy::DaisyPod::SetAudioBlockSize (
            size_t size )
```

Audio Block size defaults to 48. Change it using this function before StartingAudio.

Parameters

size	Block size to set.

8.11.3.11 StartAdc()

```
void daisy::DaisyPod::StartAdc ( )
```

Start analog to digital conversion.

8.11.3.12 StartAudio()

```
void daisy::DaisyPod::StartAudio (
            dsy_audio_callback cb )
```

Start audio callback

Parameters

cb	Callback function.

8.11.3.13 UpdateAnalogControls()

```
void daisy::DaisyPod::UpdateAnalogControls ( )
```

Call at same rate as analog reads for smooth reading.

8.11.3.14 UpdateLeds()

```
void daisy::DaisyPod::UpdateLeds ( )
```

Update Leds to set colors

8.11.4 Member Data Documentation

8.11.4.1 button1

```
Switch daisy::DaisyPod::button1
```

&

8.11.4.2 button2

```
Switch daisy::DaisyPod::button2
```

&

8.11.4.3 buttons

```
Switch * daisy::DaisyPod::buttons[BUTTON_LAST]
```

&

8.11.4.4 encoder

```
Encoder daisy::DaisyPod::encoder
```

&

8.11.4.5 knob1

```
AnalogControl daisy::DaisyPod::knob1
```

&

8.11.4.6 knob2

```
AnalogControl daisy::DaisyPod::knob2
```

&

8.11.4.7 knobs

```
AnalogControl * daisy::DaisyPod::knobs[KNOB_LAST]
```

&

8.11.4.8 led1

```
RgbLed daisy::DaisyPod::led1
```

&

8.11.4.9 led2

`RgbLed daisy::DaisyPod::led2`
&

8.11.4.10 seed

`DaisySeed daisy::DaisyPod::seed`
Public Members

8.11.5 autotoc_md8

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

- src/daisy_pod.h

8.12 daisy::DaisySeed Class Reference

This is the higher-level interface for the Daisy board.
All basic peripheral configuration/initialization is setup here.
`#include <daisy_seed.h>`

Public Member Functions

- void Configure ()
- void Init ()
- dsy_gpio_pin GetPin (uint8_t pin_idx)
- void StartAudio (dsy_audio_callback cb)
- void SetLed (bool state)
- void SetTestPoint (bool state)
- float AudioSampleRate ()
- void SetAudioBlockSize (size_t blocksize)

Public Attributes

- dsy_sdram_handle sdram_handle
- dsy_qspi_handle qspi_handle
- dsy_audio_handle audio_handle
- dsy_sai_handle sai_handle
- dsy_i2c_handle i2c1_handle
- dsy_i2c_handle i2c2_handle
- AdcHandle adc
- dsy_dac_handle dac_handle
- UsbHandle usb_handle

8.12.1 Detailed Description

This is the higher-level interface for the Daisy board.
All basic peripheral configuration/initialization is setup here.

8.12.2 Member Function Documentation

8.12.2.1 AudioSampleRate()

`float daisy::DaisySeed::AudioSampleRate ( )`
Returns the audio sample rate in Hz as a floating point number.

8.12.2.2 Configure()

```
void daisy::DaisySeed::Configure ( )
```

Configures the settings for all internal peripherals, but does not initialize them. This allows for modification of the configuration handles prior to initialization. &

8.12.2.3 GetPin()

```
dsy_gpio_pin daisy::DaisySeed::GetPin (
            uint8_t pin_idx )
```

Returns the gpio_pin corresponding to the index 0-31. For the given GPIO on the Daisy Seed (labeled 1-32 in docs).

8.12.2.4 Init()

```
void daisy::DaisySeed::Init ( )
```

Initializes the Daisy Seed and the following peripherals: SDRAM, QSPI, 24-bit 48kHz Audio via AK4556, Internal USB, as well as the built-in LED and Testpoint.
ADCs, DACs, and other special peripherals (such as I2C, SPI, etc.) can be initialized using their specific initializers within libdaisy for a specific application.

8.12.2.5 SetAudioBlockSize()

```
void daisy::DaisySeed::SetAudioBlockSize (
            size_t blocksize )
```

Sets the number of samples processed per channel by the audio callback.

8.12.2.6 SetLed()

```
void daisy::DaisySeed::SetLed (
            bool state )
```

Sets the state of the built in LED

8.12.2.7 SetTestPoint()

```
void daisy::DaisySeed::SetTestPoint (
            bool state )
```

Sets the state of the test point near pin 10

8.12.2.8 StartAudio()

```
void daisy::DaisySeed::StartAudio (
            dsy_audio_callback cb )
```

Begins the audio for the seeds builtin audio. the specified callback will get called whenever new data is ready to be prepared.

8.12.3 Member Data Documentation

8.12.3.1 adc

```
AdcHandle daisy::DaisySeed::adc
```

&

8.12.3.2 audio_handle

```
dsy_audio_handle daisy::DaisySeed::audio_handle
```

&

8.12.3.3 dac_handle

```
dsy_dac_handle daisy::DaisySeed::dac_handle
```
&

8.12.3.4 i2c1_handle

```
dsy_i2c_handle daisy::DaisySeed::i2c1_handle
```
&

8.12.3.5 i2c2_handle

```
dsy_i2c_handle daisy::DaisySeed::i2c2_handle
```
&

8.12.3.6 qspi_handle

```
dsy_qspi_handle daisy::DaisySeed::qspi_handle
```
&

8.12.3.7 sai_handle

```
dsy_sai_handle daisy::DaisySeed::sai_handle
```
&

8.12.3.8 sdram_handle

```
dsy_sdram_handle daisy::DaisySeed::sdram_handle
```
&

8.12.3.9 usb_handle

```
UsbHandle daisy::DaisySeed::usb_handle
```
&

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

- src/daisy_seed.h

8.13 dsy_audio_handle Struct Reference

```
#include <hid_audio.h>
```

Public Attributes

- size_t block_size
- dsy_sai_handle * sai
- dsy_i2c_handle * dev0_i2c
- dsy_i2c_handle * dev1_i2c

8.13.1 Detailed Description

Simple config struct that holds peripheral drivers.

8.13.2 Member Data Documentation

8.13.2.1 block_size

```
size_t dsy_audio_handle::block_size
```
&

8.13.2.2 dev0_i2c

```
dsy_i2c_handle* dsy_audio_handle::dev0_i2c
```
&

8.13.2.3 dev1_i2c

```
dsy_i2c_handle* dsy_audio_handle::dev1_i2c
```
&

8.13.2.4 sai

```
dsy_sai_handle* dsy_audio_handle::sai
```
&
The documentation for this struct was generated from the following file:

- src/hid_audio.h

8.14 dsy_dac_handle Struct Reference

```
#include <per_dac.h>
```

Public Attributes

- dsy_dac_mode mode
- dsy_dac_bitdepth bitdepth
- dsy_gpio_pin pin_config [DSY_DAC_CHN_LAST]

8.14.1 Detailed Description

Configuration structure for DAC initialization and settings. pin_config must be filled out. However, the DACs are pretty consistently on pins PA4, and PA5 across all STM32 MCUs that I've used.

8.14.2 Member Data Documentation

8.14.2.1 bitdepth

```
dsy_dac_bitdepth dsy_dac_handle::bitdepth
```
&

8.14.2.2 mode

```
dsy_dac_mode dsy_dac_handle::mode
```
&

8.14.2.3 pin_config

```
dsy_gpio_pin dsy_dac_handle::pin_config[DSY_DAC_CHN_LAST]
```
&
The documentation for this struct was generated from the following file:

- src/per_dac.h

8.15 dsy_gpio Struct Reference

`#include <per_gpio.h>`

Public Attributes

- dsy_gpio_pin pin
- dsy_gpio_mode mode
- dsy_gpio_pull pull

8.15.1 Detailed Description

Struct for holding the pin, and configuration

8.15.2 Member Data Documentation

8.15.2.1 mode

`dsy_gpio_mode dsy_gpio::mode`
&

8.15.2.2 pin

`dsy_gpio_pin dsy_gpio::pin`
&

8.15.2.3 pull

`dsy_gpio_pull dsy_gpio::pull`
&
The documentation for this struct was generated from the following file:

- src/per_gpio.h

8.16 dsy_gpio_pin Struct Reference

`#include <daisy_core.h>`

Public Attributes

- dsy_gpio_port port
- uint8_t pin

8.16.1 Detailed Description

Hardware define pins

8.16.2 Member Data Documentation

8.16.2.1 pin

`uint8_t dsy_gpio_pin::pin`
number 0-15

8.16.2.2 port

dsy_gpio_port dsy_gpio_pin::port
&
The documentation for this struct was generated from the following file:

- src/daisy_core.h

8.17 dsy_i2c_handle Struct Reference

#include <per_i2c.h>

Public Attributes

- dsy_i2c_periph periph
- dsy_gpio_pin pin_config [DSY_I2C_PIN_LAST]
- dsy_i2c_speed speed

8.17.1 Detailed Description

this object will be used to initialize the I2C interface, and can be passed to dev_ drivers that require I2C.

8.17.2 Member Data Documentation

8.17.2.1 periph

dsy_i2c_periph dsy_i2c_handle::periph
&

8.17.2.2 pin_config

dsy_gpio_pin dsy_i2c_handle::pin_config[DSY_I2C_PIN_LAST]
&

8.17.2.3 speed

dsy_i2c_speed dsy_i2c_handle::speed
&
The documentation for this struct was generated from the following file:

- src/per_i2c.h

8.18 dsy_qspi_handle Struct Reference

#include <per_qspi.h>

Public Attributes

- dsy_qspi_mode mode
- dsy_qspi_device device
- dsy_gpio_pin pin_config [DSY_QSPI_PIN_LAST]

8.18.1 Detailed Description

Configuration structure for interfacing with QSPI Driver

8.18.2 Member Data Documentation

8.18.2.1 device

```
dsy_qspi_device dsy_qspi_handle::device
```
&

8.18.2.2 mode

```
dsy_qspi_mode dsy_qspi_handle::mode
```
&

8.18.2.3 pin_config

```
dsy_gpio_pin dsy_qspi_handle::pin_config[DSY_QSPI_PIN_LAST]
```
&
The documentation for this struct was generated from the following file:

- src/per_qspi.h

8.19 dsy_sai_handle Struct Reference

```
#include <per_sai.h>
```

Public Attributes

- dsy_audio_sai init
- dsy_audio_samplerate samplerate [DSY_SAI_LAST]
- dsy_audio_bitdepth bitdepth [DSY_SAI_LAST]
- dsy_audio_dir a_direction [DSY_SAI_LAST]
- dsy_audio_dir b_direction [DSY_SAI_LAST]
- dsy_audio_sync sync_config [DSY_SAI_LAST]
- dsy_audio_device device [DSY_SAI_LAST]
- dsy_gpio_pin sai1_pin_config [DSY_SAI_PIN_LAST]
- dsy_gpio_pin sai2_pin_config [DSY_SAI_PIN_LAST]

8.19.1 Detailed Description

Configuration structure for SAI contains all above settings, and passes them to internal structure for hardware initialization.

8.19.2 Member Data Documentation

8.19.2.1 a_direction

```
dsy_audio_dir dsy_sai_handle::a_direction[DSY_SAI_LAST]
```
&

8.19.2.2 b_direction

```
dsy_audio_dir dsy_sai_handle::b_direction[DSY_SAI_LAST]
```
&

8.19.2.3 bitdepth

```
dsy_audio_bitdepth dsy_sai_handle::bitdepth[DSY_SAI_LAST]
```
&

8.19.2.4 device

```
dsy_audio_device dsy_sai_handle::device[DSY_SAI_LAST]
```
&

8.19.2.5 init

```
dsy_audio_sai dsy_sai_handle::init
```
&

8.19.2.6 sai1_pin_config

```
dsy_gpio_pin dsy_sai_handle::sai1_pin_config[DSY_SAI_PIN_LAST]
```
&

8.19.2.7 sai2_pin_config

```
dsy_gpio_pin dsy_sai_handle::sai2_pin_config[DSY_SAI_PIN_LAST]
```
&

8.19.2.8 samplerate

```
dsy_audio_samplerate dsy_sai_handle::samplerate[DSY_SAI_LAST]
```
&

8.19.2.9 sync_config

```
dsy_audio_sync dsy_sai_handle::sync_config[DSY_SAI_LAST]
```
&

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

- src/per_sai.h

8.20 DSY_SD_CardInfoTypeDef Struct Reference

`#include <util_bsp_sd_diskio.h>`

Public Attributes

- uint32_t CardType
- uint32_t CardVersion
- uint32_t Class
- uint32_t RelCardAdd
- uint32_t BlockNbr
- uint32_t BlockSize
- uint32_t LogBlockNbr
- uint32_t LogBlockSize
- uint32_t CardSpeed

8.20.1 Detailed Description

Functions for handling DiskIO via SDMMC These are usually configured through the FatFS driver/interface, and won't need to be accessed directly often.

8.20.2 Member Data Documentation

8.20.2.1 BlockNbr

`uint32_t DSY_SD_CardInfoTypeDef::BlockNbr`
Specifies the Card Capacity in blocks

8.20.2.2 BlockSize

`uint32_t DSY_SD_CardInfoTypeDef::BlockSize`
Specifies one block size in bytes

8.20.2.3 CardSpeed

`uint32_t DSY_SD_CardInfoTypeDef::CardSpeed`
Specifies the card Speed

8.20.2.4 CardType

`uint32_t DSY_SD_CardInfoTypeDef::CardType`
Specifies the card Type

8.20.2.5 CardVersion

`uint32_t DSY_SD_CardInfoTypeDef::CardVersion`
Specifies the card version

8.20.2.6 Class

`uint32_t DSY_SD_CardInfoTypeDef::Class`
Specifies the class of the card class

8.20.2.7 LogBlockNbr

`uint32_t DSY_SD_CardInfoTypeDef::LogBlockNbr`
Specifies the Card logical Capacity in blocks

8.20.2.8 LogBlockSize

`uint32_t DSY_SD_CardInfoTypeDef::LogBlockSize`
Specifies logical block size in bytes

8.20.2.9 RelCardAdd

`uint32_t DSY_SD_CardInfoTypeDef::RelCardAdd`
Specifies the Relative Card Address
The documentation for this struct was generated from the following file:

- src/util_bsp_sd_diskio.h

8.21 dsy_sdram_handle Struct Reference

`#include <dev_sdram.h>`

Public Attributes

- dsy_sdram_state state
- dsy_gpio_pin pin_config [DSY_SDRAM_PIN_LAST]

8.21.1 Detailed Description

Configuration struct for passing to initialization

8.21.2 Member Data Documentation

8.21.2.1 pin_config

```
dsy_gpio_pin dsy_sdram_handle::pin_config[DSY_SDRAM_PIN_LAST]
```
&

8.21.2.2 state

```
dsy_sdram_state dsy_sdram_handle::state
```
&
The documentation for this struct was generated from the following file:

- src/dev_sdram.h

8.22 dsy_sr_4021_handle Struct Reference

```
#include <dev_sr_4021.h>
```

Public Attributes

- dsy_gpio_pin pin_config [DSY_SR_4021_PIN_LAST]
- uint8_t num_parallel
- uint8_t num_daisychained
- dsy_gpio cs
- dsy_gpio clk
- dsy_gpio data [2]
- uint8_t states [8 ∗1 ∗2]

8.22.1 Detailed Description

configuration strucutre for 4021 pin config is used to initialize the dsy_gpio num_parallel is the number of devices connected that share the same clk/cs, etc. but have independent data num_daisychained is the number of devices in a daisy-chain configuration

8.22.2 Member Data Documentation

8.22.2.1 clk

```
dsy_gpio dsy_sr_4021_handle::clk
```
clk pin

8.22.2.2 cs

```
dsy_gpio dsy_sr_4021_handle::cs
```
cs pin

8.22.2.3 data

dsy_gpio dsy_sr_4021_handle::data[2]
array of data pins

8.22.2.4 num_daisychained

uint8_t dsy_sr_4021_handle::num_daisychained
Number of devices daisy chained

8.22.2.5 num_parallel

uint8_t dsy_sr_4021_handle::num_parallel
number of devices connected

8.22.2.6 pin_config

dsy_gpio_pin dsy_sr_4021_handle::pin_config[DSY_SR_4021_PIN_LAST]
used to initialize the dsy_gpio

8.22.2.7 states

uint8_t dsy_sr_4021_handle::states[8 * 1 * 2]
array of states
The documentation for this struct was generated from the following file:

- src/dev_sr_4021.h

8.23 daisy::Encoder Class Reference

Generic Class for handling Quadrature Encoders
Inspired/influenced by Mutable Instruments (pichenettes) Encoder classes.
`#include <hid_encoder.h>`

Public Member Functions

- void Init (dsy_gpio_pin a, dsy_gpio_pin b, dsy_gpio_pin click, float update_rate)
- void Debounce ()
- int32_t Increment () const
- bool RisingEdge () const
- bool FallingEdge () const
- bool Pressed () const
- float TimeHeldMs () const

8.23.1 Detailed Description

Generic Class for handling Quadrature Encoders
Inspired/influenced by Mutable Instruments (pichenettes) Encoder classes.

Author

Stephen Hensley

Date

December 2019

8.23.2 Member Function Documentation

8.23.2.1 Debounce()

```
void daisy::Encoder::Debounce ( )
```
Called at update_rate to debounce and handle timing for the switch. In order for events not to be missed, its important that the Edge/Pressed checks be made at the same rate as the debounce function is being called.

8.23.2.2 FallingEdge()

```
bool daisy::Encoder::FallingEdge ( ) const  [inline]
```
Returns true if the encoder was just released.

8.23.2.3 Increment()

```
int32_t daisy::Encoder::Increment ( ) const  [inline]
```
Returns +1 if the encoder was turned clockwise, -1 if it was turned counter-clockwise, or 0 if it was not just turned.

8.23.2.4 Init()

```
void daisy::Encoder::Init (
            dsy_gpio_pin a,
            dsy_gpio_pin b,
            dsy_gpio_pin click,
            float update_rate )
```
Initializes the encoder with the specified hardware pins. Update rate should be the rate at which Debounce() gets called in Hertz.

8.23.2.5 Pressed()

```
bool daisy::Encoder::Pressed ( ) const  [inline]
```
Returns true while the encoder is held down.

8.23.2.6 RisingEdge()

```
bool daisy::Encoder::RisingEdge ( ) const  [inline]
```
Returns true if the encoder was just pressed.

8.23.2.7 TimeHeldMs()

```
float daisy::Encoder::TimeHeldMs ( ) const  [inline]
```
Returns the time in milliseconds that the encoder has been held down.
The documentation for this class was generated from the following file:

- src/hid_encoder.h

8.24 FontDef Struct Reference

```
#include <util_oled_fonts.h>
```

Public Attributes

- const uint8_t FontWidth
- uint8_t FontHeight
- const uint16_t * data

8.24.1 Detailed Description

Utility for displaying fonts on OLED displays
Migrated to work with libdaisy from stm32-ssd1306

Author

afiskon on github. Font struct

8.24.2 Member Data Documentation

8.24.2.1 data

`const uint16_t* FontDef::data`
Pointer to data font data array

8.24.2.2 FontHeight

`uint8_t FontDef::FontHeight`
Font height in pixels

8.24.2.3 FontWidth

`const uint8_t FontDef::FontWidth`
Font width in pixels
The documentation for this struct was generated from the following file:

- src/util_oled_fonts.h

8.25 daisy::GateIn Class Reference

Generic Class for handling gate inputs through GPIO.
`#include <hid_gatein.h>`

Public Member Functions

- GateIn ()
- ~GateIn ()
- void Init (dsy_gpio_pin *pin_cfg)
- bool Trig ()
- bool State ()

8.25.1 Detailed Description

Generic Class for handling gate inputs through GPIO.

Author

Stephen Hensley

Date

March 2020

8.25.2 Constructor & Destructor Documentation

8.25.2.1 GateIn()

`daisy::GateIn::GateIn ( )  [inline]`
GateIn Constructor

8.25.2.2 ~GateIn()

```
daisy::GateIn::~GateIn ( )  [inline]
```
GateIn~ Destructor

8.25.3 Member Function Documentation

8.25.3.1 Init()

```
void daisy::GateIn::Init (
            dsy_gpio_pin * pin_cfg )
```
Init Initializes the gate input with specified hardware pin

8.25.3.2 State()

```
bool daisy::GateIn::State ( )  [inline]
```
State Checks current state of gate input (no state required)
read function is inverted because of suggested BJT input circuit

8.25.3.3 Trig()

```
bool daisy::GateIn::Trig ( )
```
Trig Checks current state of gate input.

Returns

True if the GPIO just transitioned.

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

- src/hid_gatein.h

8.26 daisy::Led Class Reference

LED Class providing simple Software PWM ability, etc
Eventually this will work with hardware PWM, and external LED Driver devices as well.
`#include <hid_led.h>`

Public Member Functions

- void Init (dsy_gpio_pin pin, bool invert, float samplerate=1000.0f)
- void Set (float val)
- void Update ()

8.26.1 Detailed Description

LED Class providing simple Software PWM ability, etc
Eventually this will work with hardware PWM, and external LED Driver devices as well.

Author

shensley

Date

March 2020

8.26.2 Member Function Documentation

8.26.2.1 Init()

```
void daisy::Led::Init (
            dsy_gpio_pin pin,
            bool invert,
            float samplerate = 1000.0f )
```

Initializes an LED using the specified hardware pin.

Parameters

pin	chooses LED pin
invert	will set whether to internally invert the brightness due to hardware config.
samplerate	sets the rate at which 'Update()' will be called (used for software PWM)

8.26.2.2 Set()

```
void daisy::Led::Set (
            float val )
```

Sets the brightness of the Led.

Parameters

val	will be cubed for gamma correction, and then quantized to 8-bit values for Software PWM 8-bit is for more flexible update rate options, as 12-bit or more would require faster update rates.

8.26.2.3 Update()

```
void daisy::Led::Update ( )
```

This processes the pwm of the LED sets the hardware accordingly.
The documentation for this class was generated from the following file:

- src/hid_led.h

8.27 daisy::MidiEvent Struct Reference

`#include <hid_midi.h>`

Public Member Functions

- NoteOnEvent AsNoteOn ()
- ControlChangeEvent AsControlChange ()

Public Attributes

- MidiMessageType type
- int channel
- uint8_t data [2]

8.27.1 Detailed Description

Simple MidiEvent with message type, channel, and data[2] members.

8.27.2 Member Function Documentation

8.27.2.1 AsControlChange()

`ControlChangeEvent daisy::MidiEvent::AsControlChange ( ) [inline]`
Returns the data within the MidiEvent as a NoteOnEvent struct.

8.27.2.2 AsNoteOn()

`NoteOnEvent daisy::MidiEvent::AsNoteOn ( ) [inline]`
Returns the data within the MidiEvent as a NoteOnEvent struct

8.27.3 Member Data Documentation

8.27.3.1 channel

`int daisy::MidiEvent::channel`
&

8.27.3.2 data

`uint8_t daisy::MidiEvent::data[2]`
&

8.27.3.3 type

`MidiMessageType daisy::MidiEvent::type`
&
The documentation for this struct was generated from the following file:

- src/hid_midi.h

8.28 daisy::MidiHandler Class Reference

Simple MIDI Handler
Parses bytes from an input into valid MidiEvents.
The MidiEvents fill a FIFO queue that the user can pop messages from.
`#include <hid_midi.h>`

Public Types

- enum MidiInputMode { INPUT_MODE_NONE = 0x00, INPUT_MODE_UART1 = 0x01, INPUT_MODE_USB_INT = 0x02, INPUT_MODE_USB_EXT = 0x04 }
- enum MidiOutputMode { OUTPUT_MODE_NONE = 0x00, OUTPUT_MODE_UART1 = 0x01, OUTPUT_MODE_USB_INT = 0x02, OUTPUT_MODE_USB_EXT = 0x04 }

Public Member Functions

- void Init (MidiInputMode in_mode, MidiOutputMode out_mode)
- void StartReceive ()
- void Listen ()
- void Parse (uint8_t byte)
- bool HasEvents () const
- MidiEvent PopEvent ()
- void SendMessage (uint8_t *bytes, size_t size)

8.28.1 Detailed Description

Simple MIDI Handler
Parses bytes from an input into valid MidiEvents.
The MidiEvents fill a FIFO queue that the user can pop messages from.

Author

shensley

Date

March 2020

8.28.2 Member Enumeration Documentation

8.28.2.1 MidiInputMode

`enum daisy::MidiHandler::MidiInputMode`
Input and Output can be configured separately Multiple Input modes can be selected by OR'ing the values.

Enumerator

INPUT_MODE_NONE	&
INPUT_MODE_UART1	&
INPUT_MODE_USB_INT	&
INPUT_MODE_USB_EXT	&

8.28.2.2 MidiOutputMode

`enum daisy::MidiHandler::MidiOutputMode`
Output mode

Enumerator

OUTPUT_MODE_NONE	&
OUTPUT_MODE_UART1	&
OUTPUT_MODE_USB_INT	&
OUTPUT_MODE_USB_EXT	&

8.28.3 Member Function Documentation

8.28.3.1 HasEvents()

`bool daisy::MidiHandler::HasEvents ( ) const [inline]`
Checks if there are unhandled messages in the queue

Returns

True if there are events to be handled, else false.

8.28.3.2 Init()

```
void daisy::MidiHandler::Init (
            MidiInputMode in_mode,
            MidiOutputMode out_mode )
```

Initializes the MidiHandler

Parameters

in_mode	Input mode
out_mode	Output mode

8.28.3.3 Listen()

```
void daisy::MidiHandler::Listen ( )
```

Start listening

8.28.3.4 Parse()

```
void daisy::MidiHandler::Parse (
            uint8_t byte )
```

Feed in bytes to state machine from a queue. Populates internal FIFO queue with MIDI Messages For example with uart: midi.Parse(uart.PopRx());

Parameters

byte	&

8.28.3.5 PopEvent()

```
MidiEvent daisy::MidiHandler::PopEvent ( )  [inline]
```

Pops the oldest unhandled MidiEvent from the internal queue

Returns

The event to be handled

8.28.3.6 SendMessage()

```
void daisy::MidiHandler::SendMessage (
            uint8_t * bytes,
            size_t size )
```

SendMessage Send raw bytes as message

8.28.3.7 StartReceive()

```
void daisy::MidiHandler::StartReceive ( )
```

Starts listening on the selected input mode(s). MidiEvent Queue will begin to fill, and can be checked with
The documentation for this class was generated from the following file:

- src/hid_midi.h

8.29 daisy::NoteOnEvent Struct Reference

```
#include <hid_midi.h>
```

Public Attributes

- int channel
- uint8_t note
- uint8_t velocity

8.29.1 Detailed Description

Struct containing note, and velocity data for a given channel. Can be made from MidiEvent

8.29.2 Member Data Documentation

8.29.2.1 channel

```
int daisy::NoteOnEvent::channel
```
&

8.29.2.2 note

```
uint8_t daisy::NoteOnEvent::note
```
&

8.29.2.3 velocity

```
uint8_t daisy::NoteOnEvent::velocity
```
&
The documentation for this struct was generated from the following file:

- src/hid_midi.h

8.30 daisy::OledDisplay Class Reference

```
#include <hid_oled_display.h>
```

Public Types

- enum Pins { DATA_COMMAND, RESET, NUM_PINS }

Public Member Functions

- void Init (dsy_gpio_pin *pin_cfg)
- void Fill (bool on)
- void DrawPixel (uint8_t x, uint8_t y, bool on)
- char WriteChar (char ch, FontDef font, bool on)
- char WriteString (char *str, FontDef font, bool on)
- void SetCursor (uint8_t x, uint8_t y)
- void Update ()

8.30.1 Detailed Description

Human Interface Driver for using an OLED Display (SSD1309) For all `bool on` arguments: true is on, false is off. Credit to Aleksander Alekseev (github.com/afiskon/stm32-ssd1306) on github for a great starting point. adapted for SSD1309 and H7 by shensley, 2020

8.30.2 Member Enumeration Documentation

8.30.2.1 Pins

enum daisy::OledDisplay::Pins
GPIO Pins that need to be used independent of peripheral used.

Enumerator

DATA_COMMAND	Data command pin.
RESET	Reset pin
NUM_PINS	Num pins

8.30.3 Member Function Documentation

8.30.3.1 DrawPixel()

```
void daisy::OledDisplay::DrawPixel (
            uint8_t x,
            uint8_t y,
            bool on )
```

Sets the pixel at the specified coordinate to be on/off.

Parameters

x	x Coordinate
y	y coordinate
on	on or off

8.30.3.2 Fill()

```
void daisy::OledDisplay::Fill (
            bool on )
```

Fills the entire display with either on/off.

Parameters

on	Sets on or off.

8.30.3.3 Init()

```
void daisy::OledDisplay::Init (
            dsy_gpio_pin * pin_cfg )
```

Takes an argument for the pin cfg

Parameters

pin_cfg	should be a pointer to an array of OledDisplay::NUM_PINS dsy_gpio_pins

8.30.3.4 SetCursor()

```
void daisy::OledDisplay::SetCursor (
            uint8_t x,
            uint8_t y )
```

Moves the 'Cursor' position used for WriteChar, and WriteStr to the specified coordinate.

Parameters

x	x pos
y	y pos

8.30.3.5 Update()

```
void daisy::OledDisplay::Update ( )
```

Writes the current display buffer to the OLED device using SPI or I2C depending on how the object was initialized.

8.30.3.6 WriteChar()

```
char daisy::OledDisplay::WriteChar (
            char ch,
            FontDef font,
            bool on )
```

Writes the character with the specific FontDef to the display buffer at the current Cursor position.

Parameters

ch	character to be written
font	font to be written in
on	on or off

Returns

&

8.30.3.7 WriteString()

```
char daisy::OledDisplay::WriteString (
            char * str,
            FontDef font,
            bool on )
```

Similar to WriteChar, except it will handle an entire String. Wrapping does not happen automatically, so the width of the string must be kept within the dimensions of the screen.

Parameters

str	string to be written
font	font to use
on	on or off

Returns

&

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

- src/hid_oled_display.h

8.31 daisy::Parameter Class Reference

`#include <hid_parameter.h>`

Public Types

- enum Curve {
  LINEAR, EXPONENTIAL, LOGARITHMIC, CUBE,
  LAST }

Public Member Functions

- Parameter ()
- ~Parameter ()
- void Init (AnalogControl input, float min, float max, Curve curve)
- float Process ()
- float Value ()

8.31.1 Detailed Description

Simple parameter mapping tool that takes a 0-1 input from an hid_ctrl.

8.31.2 Member Enumeration Documentation

8.31.2.1 Curve

`enum daisy::Parameter::Curve`
Curves are applied to the output signal

Enumerator

LINEAR	Linear curve
EXPONENTIAL	Exponential curve
LOGARITHMIC	Logarithmic curve
CUBE	Cubic curve
LAST	Final enum element.

8.31.3 Constructor & Destructor Documentation

8.31.3.1 Parameter()

`daisy::Parameter::Parameter ( ) [inline]`
Constructor

8.31.3.2 ~Parameter()

`daisy::Parameter::~Parameter ( ) [inline]`
Destructor

8.31.4 Member Function Documentation

8.31.4.1 Init()

```
void daisy::Parameter::Init (
            AnalogControl input,
            float min,
            float max,
            Curve curve )
```

initialize a parameter using an hid_ctrl object.

Parameters

input	- object containing the direct link to a hardware control source.
min	- bottom of range. (when input is 0.0)
max	- top of range (when input is 1.0)
curve	- the scaling curve for the input->output transformation.

8.31.4.2 Process()

```
float daisy::Parameter::Process ( )
```

processes the input signal, this should be called at the samplerate of the hid_ctrl passed in.

Returns

a float with the specified transformation applied.

8.31.4.3 Value()

```
float daisy::Parameter::Value ( )  [inline]
```

Returns

the current value from the parameter without processing another sample. this is useful if you need to use the value multiple times, and don't store

the output of process in a local variable.
The documentation for this class was generated from the following file:

- src/hid_parameter.h

8.32 daisy::RgbLed Class Reference

```
#include <hid_rgb_led.h>
```

Public Member Functions

- void Init (dsy_gpio_pin red, dsy_gpio_pin green, dsy_gpio_pin blue, bool invert)
- void Set (float r, float g, float b)
- void SetColor (Color c)
- void Update ()

8.32.1 Detailed Description

3x LEDs configured as an RGB for ease of use.

8.32.2 Member Function Documentation

8.32.2.1 Init()

```
void daisy::RgbLed::Init (
            dsy_gpio_pin red,
            dsy_gpio_pin green,
            dsy_gpio_pin blue,
            bool invert )
```

Initializes 3x GPIO Pins as red, green, and blue elements of an RGB LED

Parameters

red	Red element
green	Green element
blue	Blue element
invert	Flips led polarity

8.32.2.2 Set()

```
void daisy::RgbLed::Set (
            float r,
            float g,
            float b )
```

Sets each element of the LED with a floating point number 0-1

Parameters

r	Red element
g	Green element
b	Blue element

8.32.2.3 SetColor()

```
void daisy::RgbLed::SetColor (
            Color c )
```

Sets the RGB using a Color object.

Parameters

c	Color object to set.

8.32.2.4 Update()

```
void daisy::RgbLed::Update ( )
```

Updates the PWM of the LED based on the current values. Should be called at a regular interval. (i.e. 1kHz/1ms)
The documentation for this class was generated from the following file:

- src/hid_rgb_led.h

8.33 daisy::RingBuffer< T, size > Class Template Reference

`#include <util_ringbuffer.h>`

Public Member Functions

- void Init ()
- size_t capacity () const
- size_t writable () const
- size_t readable () const
- void Write (T v)
- void Overwrite (T v)
- T Read ()
- T ImmediateRead ()
- void Flush ()
- void Swallow (size_t n)
- void ImmediateRead (T *destination, size_t num_elements)
- void Overwrite (const T *source, size_t num_elements)

8.33.1 Detailed Description

template<typename T, size_t size>
class daisy::RingBuffer< T, size >

Utility Ring Buffer
imported from pichenettes/stmlib

8.33.2 Member Function Documentation

8.33.2.1 capacity()

```
template<typename T , size_t size>
size_t daisy::RingBuffer< T, size >::capacity ( ) const  [inline]
```

Returns

The total size of the ring buffer

8.33.2.2 Flush()

```
template<typename T , size_t size>
void daisy::RingBuffer< T, size >::Flush ( )  [inline]
```

Flushes unread elements from the ring buffer

8.33.2.3 ImmediateRead() [1/2]

```
template<typename T , size_t size>
T daisy::RingBuffer< T, size >::ImmediateRead ( )  [inline]
```

Reads next element from ring buffer immediately

Returns

read value

8.33.2.4 ImmediateRead() [2/2]

```
template<typename T , size_t size>
void daisy::RingBuffer< T, size >::ImmediateRead (
            T * destination,
            size_t num_elements )  [inline]
```

Reads a number of elements into a buffer immediately

Parameters

destination	buffer to write to
num_elements	number of elements in buffer

8.33.2.5 Init()

```
template<typename T , size_t size>
void daisy::RingBuffer< T, size >::Init ( )  [inline]
```

Initializes the Ring Buffer

8.33.2.6 Overwrite() [1/2]

```
template<typename T , size_t size>
void daisy::RingBuffer< T, size >::Overwrite (
            const T * source,
            size_t num_elements )  [inline]
```

Overwrites a number of elements using the source buffer as input.

Parameters

source	Input buffer
num_elements	Number of elements in source

8.33.2.7 Overwrite() [2/2]

```
template<typename T , size_t size>
void daisy::RingBuffer< T, size >::Overwrite (
            T v )  [inline]
```

Writes the new element to the ring buffer, overwriting unread data if necessary.

Parameters

v	Value to overwrite

8.33.2.8 Read()

```
template<typename T , size_t size>
T daisy::RingBuffer< T, size >::Read ( )  [inline]
```

Reads the first available element from the ring buffer

Returns

read value

8.33.2.9 readable()

```
template<typename T , size_t size>
size_t daisy::RingBuffer< T, size >::readable ( ) const  [inline]
```

Returns

number of unread elements in ring buffer

8.33.2.10 Swallow()

```
template<typename T , size_t size>
void daisy::RingBuffer< T, size >::Swallow (
            size_t n )  [inline]
```

Read enough samples to make it possible to read 1 sample.

Parameters

n	Size of T ?

8.33.2.11 writable()

```
template<typename T , size_t size>
size_t daisy::RingBuffer< T, size >::writable ( ) const  [inline]
```

Returns

the number of samples that can be written to ring buffer without overwriting unread data.

8.33.2.12 Write()

```
template<typename T , size_t size>
void daisy::RingBuffer< T, size >::Write (
            T v )  [inline]
```

Writes the value to the next available position in the ring buffer

Parameters

v	Value to write

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

- src/util_ringbuffer.h

8.34 daisy::RingBuffer< T, 0 > Class Template Reference

`#include <util_ringbuffer.h>`

Public Member Functions

- void Init ()
- size_t capacity () const
- size_t writable () const
- size_t readable () const
- void Write (T v)
- void Overwrite (T v)
- T Read ()
- T ImmediateRead ()
- void Flush ()
- void ImmediateRead (T *destination, size_t num_elements)
- void Overwrite (const T *source, size_t num_elements)

8.34.1 Detailed Description

template<typename T>
class daisy::RingBuffer< T, 0 >

Utility Ring Buffer imported from pichenettes/stmlib

8.34.2 Member Function Documentation

8.34.2.1 capacity()

```
template<typename T >
size_t daisy::RingBuffer< T, 0 >::capacity ( ) const  [inline]
```

Returns

0

8.34.2.2 Flush()

```
template<typename T >
void daisy::RingBuffer< T, 0 >::Flush ( )  [inline]
```

Flush the buffer

8.34.2.3 ImmediateRead() [1/2]

```
template<typename T >
T daisy::RingBuffer< T, 0 >::ImmediateRead ( )  [inline]
```

Returns

Read value

8.34.2.4 ImmediateRead() [2/2]

```
template<typename T >
void daisy::RingBuffer< T, 0 >::ImmediateRead (
            T * destination,
            size_t num_elements )  [inline]
```

Parameters

destination	&
num_elements	&

8.34.2.5 Init()

```
template<typename T >
void daisy::RingBuffer< T, 0 >::Init ( )  [inline]
```

Initialize ringbuffer

8.34.2.6 Overwrite() [1/2]

```
template<typename T >
void daisy::RingBuffer< T, 0 >::Overwrite (
            const T * source,
            size_t num_elements )  [inline]
```

Parameters

source	3
num_elements	&

8.34.2.7 Overwrite() [2/2]

```
template<typename T >
void daisy::RingBuffer< T, 0 >::Overwrite (
            T v )  [inline]
```

Parameters

v	Value to overwrite

8.34.2.8 Read()

```
template<typename T >
T daisy::RingBuffer< T, 0 >::Read ( )  [inline]
```

Returns

Read value

8.34.2.9 readable()

```
template<typename T >
size_t daisy::RingBuffer< T, 0 >::readable ( ) const  [inline]
```

Returns

0

8.34.2.10 writable()

```
template<typename T >
size_t daisy::RingBuffer< T, 0 >::writable ( ) const  [inline]
```

Returns

0

8.34.2.11 Write()

```
template<typename T >
void daisy::RingBuffer< T, 0 >::Write (
            T v )  [inline]
```

Parameters

v	Value to write

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

- src/util_ringbuffer.h

8.35 daisy::SdmmcHandler Class Reference

`#include <per_sdmmc.h>`

Public Member Functions

- void Init ()

8.35.1 Detailed Description

Configuration for interfacing with SD cards. Currently only supports operation using FatFS filesystem

8.35.2 Member Function Documentation

8.35.2.1 Init()

```
void daisy::SdmmcHandler::Init ( )
```

Initializes the SD Card Interface For now all settings are fixed (See todo at top of section)
The documentation for this class was generated from the following file:

- src/per_sdmmc.h

8.36 daisy::SdmmcHandlerInit Struct Reference

`#include <per_sdmmc.h>`

Public Attributes

- SdmmcBitWidth bitdepth
- SdmmcSpeed speed

8.36.1 Detailed Description

Structure for setting the options above. Used to intiailize SdmmcHandler

8.36.2 Member Data Documentation

8.36.2.1 bitdepth

`SdmmcBitWidth daisy::SdmmcHandlerInit::bitdepth`
&

8.36.2.2 speed

`SdmmcSpeed daisy::SdmmcHandlerInit::speed`
&
The documentation for this struct was generated from the following file:

- src/per_sdmmc.h

8.37 ShiftRegister595 Class Reference

Device Driver for 8-bit shift register.
CD74HC595 - 8-bit serial to parallel output shift.
`#include <dev_sr_595.h>`

Public Types

- enum Pins { **PIN_LATCH**, PIN_CLK, PIN_DATA, NUM_PINS }

Public Member Functions

- void Init (dsy_gpio_pin *pin_cfg, size_t num_daisy_chained=1)
- void Set (uint8_t idx, bool state)
- void Write ()

8.37.1 Detailed Description

Device Driver for 8-bit shift register.
CD74HC595 - 8-bit serial to parallel output shift.

Author

shensley

Date

May 2020

8.37.2 Member Enumeration Documentation

8.37.2.1 Pins

`enum ShiftRegister595::Pins`
The following pins correspond to the hardware connections to the 595.

Enumerator

PIN_CLK	LATCH corresonds to Pin 12 "RCLK"
PIN_DATA	CLK corresponds to Pin 11 "SRCLK"
NUM_PINS	DATA corresponds to Pin 14 "SER"

8.37.3 Member Function Documentation

8.37.3.1 Init()

```
void ShiftRegister595::Init (
            dsy_gpio_pin * pin_cfg,
            size_t num_daisy_chained = 1 )
```

Initializes the GPIO, and data for the ShiftRegister

Parameters

pin_cfg	is an array of dsy_gpio_pin corresponding the the Pins enum above.
num_daisy_chained	(default = 1) is the number of 595 devices daisy chained together.

8.37.3.2 Set()

```
void ShiftRegister595::Set (
            uint8_t idx,
            bool state )
```

Sets the state of the specified output.

Parameters

idx	The index starts with QA on the first device and ends with QH on the last device.
state	A true state will set the output HIGH, while a false state will set the output LOW.

8.37.3.3 Write()

```
void ShiftRegister595::Write ( )
```

Writes the states of shift register out to the connected devices.
The documentation for this class was generated from the following file:

- src/dev_sr_595.h

8.38 daisy::SpiHandle Class Reference

```
#include <per_spi.h>
```

Public Member Functions

- void Init ()
- void BlockingTransmit (uint8_t *buff, size_t size)

8.38.1 Detailed Description

Handler for serial peripheral interface

8.38.2 Member Function Documentation

8.38.2.1 BlockingTransmit()

```
void daisy::SpiHandle::BlockingTransmit (
            uint8_t * buff,
            size_t size )
```

Blocking transmit

Parameters

**buff*	input buffer
size	buffer size

8.38.2.2 Init()

```
void daisy::SpiHandle::Init ( )
```

Initializes handler

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

- src/per_spi.h

8.39 daisy::Switch Class Reference

```
#include <hid_switch.h>
```

Public Types

- enum Type { TYPE_TOGGLE, TYPE_MOMENTARY }
- enum Polarity { POLARITY_NORMAL, POLARITY_INVERTED }
- enum Pull { PULL_UP, PULL_DOWN, PULL_NONE }

Public Member Functions

- void Init (dsy_gpio_pin pin, float update_rate, Type t, Polarity pol, Pull pu)
- void Init (dsy_gpio_pin pin, float update_rate)
- void Debounce ()
- bool RisingEdge () const
- bool FallingEdge () const
- bool Pressed () const
- float TimeHeldMs () const

8.39.1 Detailed Description

Generic Class for handling momentary/latching switches
Inspired/influenced by Mutable Instruments (pichenettes) Switch classes

Author

Stephen Hensley

Date

December 2019

8.39.2 Member Enumeration Documentation

8.39.2.1 Polarity

`enum daisy::Switch::Polarity`

Specifies whether the pressed is HIGH or LOW.

Enumerator

POLARITY_NORMAL	&
POLARITY_INVERTED	&

8.39.2.2 Pull

`enum daisy::Switch::Pull`

Specifies whether to use built-in Pull Up/Down resistors to hold button at a given state when not engaged.

Enumerator

PULL_UP	&
PULL_DOWN	&
PULL_NONE	&

8.39.2.3 Type

`enum daisy::Switch::Type`

Specifies the expected behavior of the switch

Enumerator

TYPE_TOGGLE	&
TYPE_MOMENTARY	&

8.39.3 Member Function Documentation

8.39.3.1 Debounce()

`void daisy::Switch::Debounce ( )`

Called at update_rate to debounce and handle timing for the switch. In order for events not to be missed, its important that the Edge/Pressed checks be made at the same rate as the debounce function is being called.

8.39.3.2 FallingEdge()

`bool daisy::Switch::FallingEdge ( ) const [inline]`

Returns

true if the button was just released

8.39.3.3 Init() [1/2]

```
void daisy::Switch::Init (
            dsy_gpio_pin pin,
            float update_rate )
```

Simplified Init.

Parameters

pin	port/pin object to tell the switch which hardware pin to use.
update_rate	the rate at which the Debounce() function will be called. (used for timing).

8.39.3.4 Init() [2/2]

```
void daisy::Switch::Init (
            dsy_gpio_pin pin,
            float update_rate,
            Type t,
            Polarity pol,
            Pull pu )
```

Initializes the switch object with a given port/pin combo.

Parameters

pin	port/pin object to tell the switch which hardware pin to use.
update_rate	the rate at which the Debounce() function will be called. (used for timing).
t	switch type – Default: TYPE_MOMENTARY
pol	switch polarity – Default: POLARITY_INVERTED
pu	switch pull up/down – Default: PULL_UP

8.39.3.5 Pressed()

```
bool daisy::Switch::Pressed ( ) const  [inline]
```

Returns

true if the button is held down (or if the toggle is on)

8.39.3.6 RisingEdge()

```
bool daisy::Switch::RisingEdge ( ) const  [inline]
```

Returns

true if a button was just pressed.

8.39.3.7 TimeHeldMs()

```
float daisy::Switch::TimeHeldMs ( ) const  [inline]
```

Returns

the time in milliseconds that the button has been held (or toggle has been on)

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

- src/hid_switch.h

8.40 daisy::UartHandler Class Reference

```
#include <per_uart.h>
```

Public Member Functions

- void Init ()
- int PollReceive (uint8_t *buff, size_t size, uint32_t timeout)
- int StartRx (size_t size)
- bool RxActive ()
- int FlushRx ()
- int PollTx (uint8_t *buff, size_t size)
- uint8_t PopRx ()
- size_t Readable ()
- int CheckError ()

8.40.1 Detailed Description

Uart Peripheral

Author

shensley

Date

March 2020

8.40.2 Member Function Documentation

8.40.2.1 CheckError()

```
int daisy::UartHandler::CheckError ( )
```

Returns

the result of HAL_UART_GetError() to the user.

8.40.2.2 FlushRx()

```
int daisy::UartHandler::FlushRx ( )
```

Flushes the Receive Queue

Returns

OK or ERROR

8.40.2.3 Init()

```
void daisy::UartHandler::Init ( )
```

Initializes the UART Peripheral

8.40.2.4 PollReceive()

```
int daisy::UartHandler::PollReceive (
            uint8_t * buff,
            size_t size,
            uint32_t timeout )
```

Reads the amount of bytes in blocking mode with a 10ms timeout.

Parameters

∗buff	Buffer to read to
size	Buff size
timeout	How long to timeout for (10ms?)

Returns

Data received

8.40.2.5 PollTx()

```
int daisy::UartHandler::PollTx (
            uint8_t * buff,
            size_t size )
```

Sends an amount of data in blocking mode.

Parameters

∗buff	Buffer of data to send
size	Buffer size

Returns

OK or ERROR

8.40.2.6 PopRx()

```
uint8_t daisy::UartHandler::PopRx ( )
```

Pops the oldest byte from the FIFO.

Returns

Popped byte

8.40.2.7 Readable()

```
size_t daisy::UartHandler::Readable ( )
```

Checks if there are any unread bytes in the FIFO

Returns

1 or 0 ??

8.40.2.8 RxActive()

```
bool daisy::UartHandler::RxActive ( )
```

Returns

whether Rx DMA is listening or not.

8.40.2.9 StartRx()

```
int daisy::UartHandler::StartRx (
            size_t size )
```

Starts a DMA Receive callback to fill a buffer of specified size. Data is populated into a FIFO queue, and can be queried with the functions below. Maximum Buffer size is defined above. If a value outside of the maximum is specified, the size will be set to the maximum.

Parameters

size	Queue size

Returns

OK or ERROR

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

- src/per_uart.h

8.41 UsbHandle Class Reference

Interface for initializing and using the USB Peripherals on the daisy.
`#include <hid_usb.h>`

Public Types

- enum UsbPeriph {
  FS_INTERNAL, FS_EXTERNAL, FS_BOTH, FS_INTERNAL,
  FS_EXTERNAL, FS_BOTH }
- enum UsbPeriph {
  FS_INTERNAL, FS_EXTERNAL, FS_BOTH, FS_INTERNAL,
  FS_EXTERNAL, FS_BOTH }
- typedef void(∗ ReceiveCallback) (uint8_t ∗buff, uint32_t ∗len)
- typedef void(∗ ReceiveCallback) (uint8_t ∗buff, uint32_t ∗len)

Public Member Functions

- void Init (UsbPeriph dev)
- void TransmitInternal (uint8_t ∗buff, size_t size)
- void TransmitExternal (uint8_t ∗buff, size_t size)
- void SetReceiveCallback (ReceiveCallback cb)
- void Init (UsbPeriph dev)
- void TransmitInternal (uint8_t ∗buff, size_t size)
- void TransmitExternal (uint8_t ∗buff, size_t size)
- void SetReceiveCallback (ReceiveCallback cb)

8.41.1 Detailed Description

Interface for initializing and using the USB Peripherals on the daisy.

Author

Stephen Hensley

Date

December 2019

8.41.2 Member Typedef Documentation

8.41.2.1 ReceiveCallback [1/2]

```
typedef void(* UsbHandle::ReceiveCallback) (uint8_t *buff, uint32_t *len)
```

Function called upon reception of a buffer

8.41.2.2 ReceiveCallback [2/2]

```
typedef void(* UsbHandle::ReceiveCallback) (uint8_t *buff, uint32_t *len)
```

Function called upon reception of a buffer

8.41.3 Member Enumeration Documentation

8.41.3.1 UsbPeriph [1/2]

```
enum UsbHandle::UsbPeriph
```

Specified which of the two USB Peripherals to initialize.

Enumerator

FS_INTERNAL	Internal pin
FS_EXTERNAL	FS External D+ pin is Pin 38 (GPIO32). FS External D- pin is Pin 37 (GPIO31)
FS_BOTH	Both
FS_INTERNAL	Internal pin
FS_EXTERNAL	FS External D+ pin is Pin 38 (GPIO32). FS External D- pin is Pin 37 (GPIO31)
FS_BOTH	Both

8.41.3.2 UsbPeriph [2/2]

```
enum UsbHandle::UsbPeriph
```

Specified which of the two USB Peripherals to initialize.

Enumerator

FS_INTERNAL	Internal pin
FS_EXTERNAL	FS External D+ pin is Pin 38 (GPIO32). FS External D- pin is Pin 37 (GPIO31)
FS_BOTH	Both
FS_INTERNAL	Internal pin
FS_EXTERNAL	FS External D+ pin is Pin 38 (GPIO32). FS External D- pin is Pin 37 (GPIO31)
FS_BOTH	Both

8.41.4 Member Function Documentation

8.41.4.1 Init() [1/2]

```
void UsbHandle::Init (
            UsbPeriph dev )
```

Initializes the specified peripheral(s) as USB CDC Devices

Parameters

dev	Device to initialize

8.41.4.2 Init() [2/2]

```
void UsbHandle::Init (
            UsbPeriph dev )
```

Initializes the specified peripheral(s) as USB CDC Devices

Parameters

dev	Device to initialize

8.41.4.3 SetReceiveCallback() [1/2]

```
void UsbHandle::SetReceiveCallback (
            ReceiveCallback cb )
```

sets the callback to be called upon reception of new data

Parameters

cb	Function to serve as callback

8.41.4.4 SetReceiveCallback() [2/2]

```
void UsbHandle::SetReceiveCallback (
            ReceiveCallback cb )
```

sets the callback to be called upon reception of new data

Parameters

cb	Function to serve as callback

8.41.4.5 TransmitExternal() [1/2]

```
void UsbHandle::TransmitExternal (
            uint8_t * buff,
            size_t size )
```

Transmits a buffer of 'size' bytes from a USB port connected to the external USB Pins of the daisy seed.

Parameters

buff	Buffer to transmit
size	Buffer size

8.41.4.6 TransmitExternal() [2/2]

```
void UsbHandle::TransmitExternal (
            uint8_t * buff,
            size_t size )
```

Transmits a buffer of 'size' bytes from a USB port connected to the external USB Pins of the daisy seed.

Parameters

buff	Buffer to transmit
size	Buffer size

8.41.4.7 TransmitInternal() [1/2]

```
void UsbHandle::TransmitInternal (
            uint8_t * buff,
            size_t size )
```

Transmits a buffer of 'size' bytes from the on board USB FS port.

Parameters

buff	Buffer to transmit
size	Buffer size

8.41.4.8 TransmitInternal() [2/2]

```
void UsbHandle::TransmitInternal (
            uint8_t * buff,
            size_t size )
```

Transmits a buffer of 'size' bytes from the on board USB FS port.

Parameters

buff	Buffer to transmit
size	Buffer size

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

- src/hid_usb.h

8.42 WAV_FormatTypeDef Struct Reference

```
#include <util_wav_format.h>
```

Public Attributes

- uint32_t ChunkId
- uint32_t FileSize
- uint32_t FileFormat
- uint32_t SubChunk1ID
- uint32_t SubChunk1Size
- uint16_t AudioFormat
- uint16_t NbrChannels
- uint32_t SampleRate
- uint32_t ByteRate
- uint16_t BlockAlign
- uint16_t BitPerSample
- uint32_t SubChunk2ID
- uint32_t SubCHunk2Size

8.42.1 Detailed Description

Helper struct for handling the WAV file format

8.42.2 Member Data Documentation

8.42.2.1 AudioFormat

```
uint16_t WAV_FormatTypeDef::AudioFormat
```
&

8.42.2.2 BitPerSample

```
uint16_t WAV_FormatTypeDef::BitPerSample
```
&

8.42.2.3 BlockAlign

```
uint16_t WAV_FormatTypeDef::BlockAlign
```
&

8.42.2.4 ByteRate

```
uint32_t WAV_FormatTypeDef::ByteRate
```
&

8.42.2.5 ChunkId

```
uint32_t WAV_FormatTypeDef::ChunkId
```
&

8.42.2.6 FileFormat

```
uint32_t WAV_FormatTypeDef::FileFormat
```
&

8.42.2.7 FileSize

```
uint32_t WAV_FormatTypeDef::FileSize
```
&

8.42.2.8 NbrChannels

```
uint16_t WAV_FormatTypeDef::NbrChannels
```
&

8.42.2.9 SampleRate

```
uint32_t WAV_FormatTypeDef::SampleRate
```
&

8.42.2.10 SubChunk1ID

```
uint32_t WAV_FormatTypeDef::SubChunk1ID
```
&

8.42.2.11 SubChunk1Size

```
uint32_t WAV_FormatTypeDef::SubChunk1Size
```
&

8.42.2.12 SubChunk2ID

```
uint32_t WAV_FormatTypeDef::SubChunk2ID
```
&

8.42.2.13 SubCHunk2Size

```
uint32_t WAV_FormatTypeDef::SubCHunk2Size
```
&
The documentation for this struct was generated from the following file:

- src/util_wav_format.h

8.43 daisy::WavFileInfo Struct Reference

```
#include <hid_wavplayer.h>
```

Public Attributes

- WAV_FormatTypeDef raw_data
- char name [256]

8.43.1 Detailed Description

Struct containing details of Wav File.

8.43.2 Member Data Documentation

8.43.2.1 name

```
char daisy::WavFileInfo::name[256]
```
Wav filename

8.43.2.2 raw_data

`WAV_FormatTypeDef daisy::WavFileInfo::raw_data`
Raw wav data
The documentation for this struct was generated from the following file:

- src/hid_wavplayer.h

8.44 daisy::WavPlayer Class Reference

`#include <hid_wavplayer.h>`

Public Member Functions

- void Init ()
- int Open (size_t sel)
- int Close ()
- int16_t Stream ()
- void Prepare ()
- void Restart ()
- void SetLooping (bool loop)
- bool GetLooping () const
- size_t GetNumberFiles () const
- size_t GetCurrentFile () const

8.44.1 Detailed Description

Wav Player that will load .wav files from an SD Card, and then provide a method of accessing the samples with double-buffering.

8.44.2 Member Function Documentation

8.44.2.1 Close()

`int daisy::WavPlayer::Close ( )`
Closes whatever file is currently open.

Returns

&

8.44.2.2 GetCurrentFile()

`size_t daisy::WavPlayer::GetCurrentFile ( ) const [inline]`

Returns

currently selected file.

8.44.2.3 GetLooping()

`bool daisy::WavPlayer::GetLooping ( ) const [inline]`

Returns

Whether the WavPlayer is looping or not.

8.44.2.4 GetNumberFiles()

```
size_t daisy::WavPlayer::GetNumberFiles ( ) const  [inline]
```

Returns

The number of files loaded by the WavPlayer

8.44.2.5 Init()

```
void daisy::WavPlayer::Init ( )
```

Initializes the WavPlayer, loading up to max_files of wav files from an SD Card.

8.44.2.6 Open()

```
int daisy::WavPlayer::Open (
            size_t sel )
```

Opens the file at index sel for reading.

Parameters

sel	File to open

8.44.2.7 Prepare()

```
void daisy::WavPlayer::Prepare ( )
```

Collects buffer for playback when needed.

8.44.2.8 Restart()

```
void daisy::WavPlayer::Restart ( )
```

Resets the playback position to the beginning of the file immediately

8.44.2.9 SetLooping()

```
void daisy::WavPlayer::SetLooping (
            bool loop )  [inline]
```

Sets whether or not the current file will repeat after completing playback.

Parameters

loop	To loop or not to loop.

8.44.2.10 Stream()

```
int16_t daisy::WavPlayer::Stream ( )
```

Returns

The next sample if playing, otherwise returns 0

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

- src/hid_wavplayer.h

Chapter 9

File Documentation

9.1 src/ffconf.h File Reference

```
#include "util_bsp_sd_diskio.h"
#include <stdlib.h>
```

Macros

- #define _FFCONF 68300
- #define _FS_READONLY 0
- #define _FS_MINIMIZE 0
- #define _USE_STRFUNC 2
- #define _USE_FIND 0
- #define _USE_MKFS 1
- #define _USE_FASTSEEK 1
- #define _USE_EXPAND 0
- #define _USE_CHMOD 0
- #define _USE_LABEL 0
- #define _USE_FORWARD 0
- #define _CODE_PAGE 850
- #define _USE_LFN 1
- #define _MAX_LFN 255
- #define _LFN_UNICODE 0
- #define _STRF_ENCODE 3
- #define _FS_RPATH 0
- #define _VOLUMES 1
- #define _STR_VOLUME_ID 0
- #define _VOLUME_STRS
- #define _MULTI_PARTITION 0
- #define _MIN_SS 512
- #define _MAX_SS 512
- #define _USE_TRIM 0
- #define _FS_NOFSINFO 0
- #define _FS_TINY 0
- #define _FS_EXFAT 0
- #define _FS_NORTC 0
- #define _NORTC_MON 6
- #define _NORTC_MDAY 4
- #define _NORTC_YEAR 2015
- #define _FS_LOCK 2
- #define _FS_REENTRANT 0

- #define _FS_TIMEOUT 1000
- #define _SYNC_t osSemaphoreId
- #define ff_malloc malloc
- #define ff_free free

9.1.1 Detailed Description

Further fatfs support.

9.1.2 Macro Definition Documentation

9.1.2.1 _CODE_PAGE

```
#define _CODE_PAGE 850
```

This option specifies the OEM code page to be used on the target system. / Incorrect setting of the code page can cause a file open failure. / / 1 - ASCII (No extended character. Non-LFN cfg. only) / 437 - U.S. / 720 - Arabic / 737 - Greek / 771 - KBL / 775 - Baltic / 850 - Latin 1 / 852 - Latin 2 / 855 - Cyrillic / 857 - Turkish / 860 - Portuguese / 861 - Icelandic / 862 - Hebrew / 863 - Canadian French / 864 - Arabic / 865 - Nordic / 866 - Russian / 869 - Greek 2 / 932 - Japanese (DBCS) / 936 - Simplified Chinese (DBCS) / 949 - Korean (DBCS) / 950 - Traditional Chinese (DBCS)

9.1.2.2 _FFCONF

```
#define _FFCONF 68300
```

FatFs - Generic FAT file system module R0.12c (C)ChaN, 2017

Attention

9.1.2.3 _FS_EXFAT

```
#define _FS_EXFAT 0
```

This option switches support of exFAT file system. (0:Disable or 1:Enable) / When enable exFAT, also LFN needs to be enabled. (_USE_LFN >= 1) / Note that enabling exFAT discards C89 compatibility.

9.1.2.4 _FS_LOCK

```
#define _FS_LOCK 2
```

0:Disable or >=1:Enable The option _FS_LOCK switches file lock function to control duplicated file open / and illegal operation to open objects. This option must be 0 when _FS_READONLY / is 1. / / 0: Disable file lock function. To avoid volume corruption, application program / should avoid illegal open, remove and rename to the open objects. / >0: Enable file lock function. The value defines how many files/sub-directories / can be opened simultaneously under file lock control. Note that the file / lock control is independent of re-entrancy.

9.1.2.5 _FS_MINIMIZE

```
#define _FS_MINIMIZE 0
```

0 to 3 This option defines minimization level to remove some basic API functions. / / 0: All basic functions are enabled. / 1: f_stat(), f_getfree(), f_unlink(), f_mkdir(), f_truncate() and f_rename() / are removed. / 2: f_opendir(), f_readdir() and f_closedir() are removed in addition to 1. / 3: f_lseek() function is removed in addition to 2.

9.1.2.6 _FS_NOFSINFO

```
#define _FS_NOFSINFO 0
```

0,1,2 or 3 If you need to know correct free space on the FAT32 volume, set bit 0 of this / option, and f_getfree() function at first time after volume mount will force / a full FAT scan. Bit 1 controls the use of last allocated cluster number. / / bit0=0: Use free cluster count in the FSINFO if available. / bit0=1: Do not trust free cluster count in the FSINFO. / bit1=0: Use last allocated cluster number in the FSINFO if available. / bit1=1: Do not trust last allocated cluster number in the FSINFO.

9.1.2.7 _FS_NORTC

```
#define _FS_NORTC 0
```

&

9.1.2.8 _FS_READONLY

```
#define _FS_READONLY 0
```

0:Read/Write or 1:Read only This option switches read-only configuration. (0:Read/Write or 1:Read-only) / Read-only configuration removes writing API functions, f_write(), f_sync(), / f_unlink(), f_mkdir(), f_chmod(), f_rename(), f_truncate(), f_getfree() / and optional writing functions as well.

9.1.2.9 _FS_REENTRANT

```
#define _FS_REENTRANT 0
```

0:Disable or 1:Enable

9.1.2.10 _FS_RPATH

```
#define _FS_RPATH 0
```

0 to 2 This option configures support of relative path. / / 0: Disable relative path and remove related functions. / 1: Enable relative path. f_chdir() and f_chdrive() are available. / 2: f_getcwd() function is available in addition to 1.

9.1.2.11 _FS_TIMEOUT

```
#define _FS_TIMEOUT 1000
```

Timeout period in unit of time ticks

9.1.2.12 _FS_TINY

```
#define _FS_TINY 0
```

0:Normal or 1:Tiny This option switches tiny buffer configuration. (0:Normal or 1:Tiny) / At the tiny configuration, size of file object (FIL) is reduced _MAX_SS bytes. / Instead of private sector buffer eliminated from the file object, common sector / buffer in the file system object (FATFS) is used for the file data transfer.

9.1.2.13 _LFN_UNICODE

```
#define _LFN_UNICODE 0
```

0:ANSI/OEM or 1:Unicode This option switches character encoding on the API. (0:ANSI/OEM or 1:UTF-16) / To use Unicode string for the path name, enable LFN and set _LFN_UNICODE = 1. / This option also affects behavior of string I/O functions.

9.1.2.14 _MAX_LFN

```
#define _MAX_LFN 255
```

Maximum LFN length to handle (12 to 255) The _USE_LFN switches the support of long file name (LFN). / / 0: Disable support of LFN. _MAX_LFN has no effect. / 1: Enable LFN with static working buffer on the BSS. Always NOT thread-safe. / 2: Enable LFN with dynamic working buffer on the STACK. / 3: Enable LFN with dynamic working buffer on the HEAP. / / To enable the LFN, Unicode handling functions (option/unicode.c) must be added / to the project. The working buffer occupies (_MAX_LFN + 1) $*$ 2 bytes and / additional 608 bytes at exFAT enabled. _MAX_LFN can be in range from 12 to 255. / It should be set 255 to support full featured LFN operations. / When

use stack for the working buffer, take care on stack overflow. When use heap / memory for the working buffer, memory management functions, ff_memalloc() and / ff_memfree(), must be added to the project.

9.1.2.15 _MAX_SS

```
#define _MAX_SS 512
```

512, 1024, 2048 or 4096 These options configure the range of sector size to be supported. (512, 1024, / 2048 or 4096) Always set both 512 for most systems, all type of memory cards and / harddisk. But a larger value may be required for on-board flash memory and some / type of optical media. When _MAX_SS is larger than _MIN_SS, FatFs is configured / to variable sector size and GET_SECTOR_SIZE command must be implemented to the / disk_ioctl() function.

9.1.2.16 _MIN_SS

```
#define _MIN_SS 512
```

512, 1024, 2048 or 4096

9.1.2.17 _MULTI_PARTITION

```
#define _MULTI_PARTITION 0
```

0:Single partition, 1:Multiple partition This option switches support of multi-partition on a physical drive. / By default (0), each logical drive number is bound to the same physical drive / number and only an FAT volume found on the physical drive will be mounted. / When multi-partition is enabled (1), each logical drive number can be bound to / arbitrary physical drive and partition listed in the VolToPart[]. Also f_fdisk() / funciton will be available.

9.1.2.18 _NORTC_MDAY

```
#define _NORTC_MDAY 4
```

&

9.1.2.19 _NORTC_MON

```
#define _NORTC_MON 6
```

&

9.1.2.20 _NORTC_YEAR

```
#define _NORTC_YEAR 2015
```

The option _FS_NORTC switches timestamp functiton. If the system does not have / any RTC function or valid timestamp is not needed, set _FS_NORTC = 1 to disable / the timestamp function. All objects modified by FatFs will have a fixed timestamp / defined by _NORTC_MON, _NORTC_MDAY and _NORTC_YEAR in local time. / To enable timestamp function (_FS_NORTC = 0), get_fattime() function need to be / added to the project to get current time form real-time clock. _NORTC_MON, / _NORTC_MDAY and _NORTC_YEAR have no effect. / These options have no effect at read-only configuration (_FS_READONLY = 1).

9.1.2.21 _STR_VOLUME_ID

```
#define _STR_VOLUME_ID 0
```

0:Use only 0-9 for drive ID, 1:Use strings for drive ID

9.1.2.22 _STRF_ENCODE

```
#define _STRF_ENCODE 3
```

When _LFN_UNICODE == 1, this option selects the character encoding ON THE FILE to / be read/written via string I/O functions, f_gets(), f_putc(), f_puts and f_printf(). / / 0: ANSI/OEM / 1: UTF-16LE / 2: UTF-16BE / 3: UTF-8 / / This option has no effect when _LFN_UNICODE == 0.

9.1.2.23 _SYNC_t

```
#define _SYNC_t osSemaphoreId
```

The option _FS_REENTRANT switches the re-entrancy (thread safe) of the FatFs / module itself. Note that regardless of this option, file access to different / volume is always re-entrant and volume control functions, f_mount(), f_mkfs() / and f_fdisk() function, are always not re-entrant. Only file/directory access / to the same volume is under control of this function. / / 0: Disable re-entrancy. _FS_TIMEOUT and _SYNC_t have no effect. / 1: Enable re-entrancy. Also user provided synchronization handlers, / ff_req_grant(), ff_rel_grant(), ff_del_syncobj() and ff_cre_syncobj() / function, must be added to the project. Samples are available in / option/syscall.c. / / The _FS↩ _TIMEOUT defines timeout period in unit of time tick. / The _SYNC_t defines O/S dependent sync object type. e.g. HANDLE, ID, OS_EVENT∗, / SemaphoreHandle_t and etc.. A header file for O/S definitions needs to be / included somewhere in the scope of ff.h.

9.1.2.24 _USE_CHMOD

```
#define _USE_CHMOD 0
```

This option switches attribute manipulation functions, f_chmod() and f_utime(). / (0:Disable or 1:Enable) Also _F↩ S_READONLY needs to be 0 to enable this option.

9.1.2.25 _USE_EXPAND

```
#define _USE_EXPAND 0
```

This option switches f_expand function. (0:Disable or 1:Enable)

9.1.2.26 _USE_FASTSEEK

```
#define _USE_FASTSEEK 1
```

This option switches fast seek feature. (0:Disable or 1:Enable)

9.1.2.27 _USE_FIND

```
#define _USE_FIND 0
```

This option switches filtered directory read functions, f_findfirst() and / f_findnext(). (0:Disable, 1:Enable 2:Enable with matching altname[] too)

9.1.2.28 _USE_FORWARD

```
#define _USE_FORWARD 0
```

This option switches f_forward() function. (0:Disable or 1:Enable)

9.1.2.29 _USE_LABEL

```
#define _USE_LABEL 0
```

This option switches volume label functions, f_getlabel() and f_setlabel(). / (0:Disable or 1:Enable)

9.1.2.30 _USE_LFN

```
#define _USE_LFN 1
```

0 to 3

9.1.2.31 _USE_MKFS

```
#define _USE_MKFS 1
```

This option switches f_mkfs() function. (0:Disable or 1:Enable)

9.1.2.32 _USE_STRFUNC

```
#define _USE_STRFUNC 2
```

0:Disable or 1-2:Enable This option switches string functions, f_gets(), f_putc(), f_puts() and / f_printf(). / / 0: Disable string functions. / 1: Enable without LF-CRLF conversion. / 2: Enable with LF-CRLF conversion.

9.1.2.33 _USE_TRIM

```
#define _USE_TRIM 0
```

This option switches support of ATA-TRIM. (0:Disable or 1:Enable) / To enable Trim function, also CTRL_TRIM command should be implemented to the / disk_ioctl() function.

9.1.2.34 _VOLUME_STRS

```
#define _VOLUME_STRS
```

Value:

```
    "RAM", "NAND", "CF", "SD1", "SD2", "USB1", "USB2", \
        "USB3"
```

_STR_VOLUME_ID switches string support of volume ID. / When _STR_VOLUME_ID is set to 1, also pre-defined strings can be used as drive / number in the path name. _VOLUME_STRS defines the drive ID strings for each / logical drives. Number of items must be equal to _VOLUMES. Valid characters for / the drive ID strings are: A-Z and 0-9.

9.1.2.35 _VOLUMES

```
#define _VOLUMES 1
```

Number of volumes (logical drives) to be used.

9.1.2.36 ff_free

```
#define ff_free free
```

define the ff_malloc ff_free macros as standard malloc free

9.1.2.37 ff_malloc

```
#define ff_malloc malloc
```

define the ff_malloc ff_free macros as standard malloc free

9.2 src/hid_gatein.h File Reference

```
#include "per_gpio.h"
```

Classes

- class daisy::GateIn

  Generic Class for handling gate inputs through GPIO.

Namespaces

- daisy

  Hardware defines and helpers for daisy field platform.

9.3 src/hid_wavplayer.h File Reference

```
#include "daisy_core.h"
#include "util_wav_format.h"
```

Classes

- struct daisy::WavFileInfo
- class daisy::WavPlayer

Namespaces

- daisy

  Hardware defines and helpers for daisy field platform.

Macros

- #define DSY_WAVPLAYER_H
- #define WAV_FILENAME_MAX 256

9.3.1 Macro Definition Documentation

9.3.1.1 DSY_WAVPLAYER_H

```
#define DSY_WAVPLAYER_H
```

Macro

9.3.1.2 WAV_FILENAME_MAX

```
#define WAV_FILENAME_MAX 256
```

Maximum LFN (set to same in FatFs (ffconf.h)

9.4 src/usbd_cdc_if.h File Reference

: Header for usbd_cdc_if.c file.

```
#include "usbd_cdc.h"
```

Typedefs

- typedef void(∗ CDC_ReceiveCallback) (uint8_t ∗buf, uint32_t ∗size)

Functions

- void CDC_Set_Rx_Callback_FS (CDC_ReceiveCallback cb)
- uint8_t CDC_Transmit_FS (uint8_t ∗Buf, uint16_t Len)
- uint8_t CDC_Transmit_HS (uint8_t ∗Buf, uint16_t Len)

Variables

- USBD_CDC_ItfTypeDef USBD_Interface_fops_FS
- USBD_CDC_ItfTypeDef USBD_Interface_fops_HS

9.4.1 Detailed Description

: Header for usbd_cdc_if.c file.

Version

: v1.0_Cube

Attention

9.5 src/usbd_conf.h File Reference

: Header for usbd_conf.c file.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "stm32h7xx.h"
#include "stm32h7xx_hal.h"
```

Macros

- #define USBD_MAX_NUM_INTERFACES 1U
- #define USBD_MAX_NUM_CONFIGURATION 1U
- #define USBD_MAX_STR_DESC_SIZ 512U
- #define USBD_SUPPORT_USER_STRING 0U
- #define USBD_DEBUG_LEVEL 3U
- #define USBD_LPM_ENABLED 0U
- #define USBD_SELF_POWERED 1U
- #define DEVICE_FS 0
- #define DEVICE_HS 1
- #define USBD_malloc malloc
- #define USBD_free free
- #define USBD_memset memset
- #define USBD_memcpy memcpy
- #define USBD_Delay HAL_Delay
- #define USBD_UsrLog(...)
- #define USBD_ErrLog(...)
- #define USBD_DbgLog(...)

9.5.1 Detailed Description

: Header for usbd_conf.c file.

Version

: v1.0_Cube

Attention

© Copyright (c) 2019 STMicroelectronics. All rights reserved.

This software component is licensed by ST under Ultimate Liberty license SLA0044, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: www.st.com/SLA0044

9.6 src/usbd_desc.h File Reference

: Header for usbd_conf.c file.

```
#include "usbd_def.h"
```

Macros

- #define DEVICE_ID1 (UID_BASE)
- #define DEVICE_ID2 (UID_BASE + 0x4)
- #define DEVICE_ID3 (UID_BASE + 0x8)
- #define USB_SIZ_STRING_SERIAL 0x1A

Variables

- USBD_DescriptorsTypeDef HS_Desc
- USBD_DescriptorsTypeDef FS_Desc

9.6.1 Detailed Description

: Header for usbd_conf.c file.

Version

: v1.0_Cube

Attention

Index