DaisySP

Generated by Doxygen 1.8.13

Contents

Chapter 1

libdaisy

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.)

1.1 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.

1.1.1 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.

1.1.2 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.

1.1.3 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 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.

6.1.1 Detailed Description

Usb VCP device module.

6.2 USBD_CDC_IF_Exported_Defines

Defines.

Defines.

6.3 USBD_CDC_IF_Exported_Types

Types.

Typedefs

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

6.3.1 Detailed Description

Types.

6.4 USBD_CDC_IF_Exported_Macros

Aliases.

Aliases.

6.5 USBD_CDC_IF_Exported_Variables

Public variables.

Variables

- USBD_CDC_ItfTypeDef USBD_Interface_fops_FS
- USBD_CDC_ItfTypeDef USBD_Interface_fops_HS

6.5.1 Detailed Description

Public variables.

6.5.2 Variable Documentation

6.5.2.1 USBD_Interface_fops_FS

```
USBD_CDC_ItfTypeDef USBD_Interface_fops_FS
```

CDC Interface callback.

6.5.2.2 USBD_Interface_fops_HS

```
USBD_CDC_ItfTypeDef USBD_Interface_fops_HS
```

CDC Interface callback.

6.6 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)

6.6.1 Detailed Description

Public functions declaration.

6.7 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.

6.7.1 Detailed Description

Configuration file for Usb otg low level driver.

6.8 USBD_CONF_Exported_Variables

Public variables.

Public variables.

6.9 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

6.9.1 Detailed Description

Defines for configuration of the Usb device.

6.10 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**(...)

6.10.1 Detailed Description

Aliases.

6.10.2 Macro Definition Documentation

6.10.2.1 USBD_DbgLog

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

Value:

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

6.10.2.2 USBD_Delay

```
#define USBD_Delay HAL_Delay
```

Alias for delay.

6.10.2.3 USBD_ErrLog

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

Value:

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

6.10.2.4 USBD_free

```
#define USBD_free free
```

Alias for memory release.

6.10.2.5 USBD_malloc

```
#define USBD_malloc malloc
```

Alias for memory allocation.

6.10.2.6 USBD_memcpy

```
#define USBD_memcpy memcpy
```

Alias for memory copy.

6.10.2.7 USBD_memset

```
#define USBD_memset memset
```

Alias for memory set.

6.10.2.8 USBD_UsrLog

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

Value:

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

6.11 USBD_CONF_Exported_Types

Types.

Types.

6.12 USBD_CONF_Exported_FunctionsPrototype

Declaration of public functions for Usb device.

Declaration of public functions for Usb device.

6.13 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.

6.13.1 Detailed Description

Usb device descriptors module.

6.14 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

6.14.1 Detailed Description

Constants.

6.15 USBD_DESC_Exported_Defines

Defines.

Defines.

6.16 USBD_DESC_Exported_TypesDefinitions

Types.

Types.

6.17 USBD_DESC_Exported_Macros

Aliases.

Aliases.

6.18 USBD_DESC_Exported_Variables

Public variables.

Variables

- USBD_DescriptorsTypeDef HS_Desc
- USBD_DescriptorsTypeDef FS_Desc

6.18.1 Detailed Description

Public variables.

6.18.2 Variable Documentation

6.18.2.1 FS_Desc

```
USBD_DescriptorsTypeDef FS_Desc
```

Descriptor for the Usb device.

6.18.2.2 HS_Desc

```
USBD_DescriptorsTypeDef HS_Desc
```

Descriptor for the Usb device.

6.19 USBD_DESC_Exported_FunctionsPrototype

Public functions declaration.

Public functions declaration.

6.20 STM32_USB_OTG_DEVICE_LIBRARY

For Usb device.

Modules

- USBD_CDC_IF

  Usb VCP device module.
- USBD_DESC

  Usb device descriptors module.

6.20.1 Detailed Description

For Usb device.

6.21 USBD_OTG_DRIVER

Modules

- USBD_CONF

  Configuration file for Usb otg low level driver.

6.21.1 Detailed Description

Chapter 7

Namespace Documentation

7.1 daisy Namespace Reference

Classes

- struct AdcChannelConfig
- class AdcHandle
- class AnalogControl
- 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 DaisySeed
- class Encoder
- 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
- struct NoteOnEvent
- class OledDisplay
- class Parameter
- class RgbLed
- class RingBuffer
- class RingBuffer< T, 0 >
- class SdmmcHandler
- struct SdmmcHandlerInit
- class SpiHandle
- class Switch
- class UartHandler
- class UsbHandle
- 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

daisy_field.h Hardware defines and helpers for daisy field platform.

- Get this set up to work with the dev_leddriver stuff as well

Setup Hardware PWM for pins that have it

TODO:

- Add documentation
- Add configuration
- Add reception
- Add IT
- Add DMA

7.1.2 Enumeration Type Documentation

7.1.2.1 anonymous enum

`anonymous enum`

enums for controls, etc.

Enumerator

SW_1	tactile switch
SW_3	tactile switch
SW_LAST	toggle

7.1.2.2 anonymous enum

`anonymous enum`

All knobs connect to ADC1_INP10 via CD4051 mux

7.1.2.3 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

7.1.2.4 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.

7.1.2.5 SdmmcBitWidth

`enum daisy::SdmmcBitWidth`

Sets whether 4-bit mode or 1-bit mode is used for the SDMMC

7.1.2.6 SdmmcMode

`enum daisy::SdmmcMode`

Operating ModeCurrently only FatFS is supported.

7.1.2.7 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.

7.1.2.8 SpiPeriph

`enum daisy::SpiPeriph`

Enumerator

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

7.1.2.9 SpiPin

`enum daisy::SpiPin`

Enumerator

SPI_PIN_SCK	CS pin
SPI_PIN_MOSI	SCK pin
SPI_PIN_MISO	MOSI pin

7.1.3 Function Documentation

7.1.3.1 daisy_field_init()

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

Initializes daisy field

Parameters

p	daisy_field struct to initialize

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 While there may not be many configuration options here, using a struct like this allows us to add more configuration later without breaking existing functionality.

8.1.2 Member Function Documentation

8.1.2.1 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 pinInternal Callbacks handle the pin addressing.channels must be 1-8

8.1.2.2 InitSingle()

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

Initializes a single ADC pin as an ADC.

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

- src/per_adc.h

8.2 daisy::AdcHandle Class Reference

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 Member Function Documentation

8.2.1.1 Get()

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

These are getters for a single channel

8.2.1.2 GetMux()

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

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

8.2.1.3 Init()

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

Initializes the ADC with the pins passed in. ∗ ∗cfg: an array of AdcChannelConfig of the desired channel

Parameters

num_channels	number of ADC channels to initialize
ovs	Oversampling amount - Defaults to OVS_32

8.2.1.4 Start()

```
void daisy::AdcHandle::Start ( )
```

Starts reading from the ADC

8.2.1.5 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

Public Member Functions

- 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 Member Function Documentation

8.3.1.1 Init()

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

Initializes the control adcptr is a pointer to the raw adc read value – This can 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.slew_seconds is the slew time in seconds that it takes for the control to change to a new value.flip determines whether the input is flipped (i.e. 1.f - input) or not before being processed.invert determines whether the input is inverted (i.e. -1.f * input) or note before being processed.

8.3.1.2 InitBipolarCv()

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

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

8.3.1.3 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 -> 1.0Bi-polar CV inputs will return -1.0 -> 1.0

8.3.1.4 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

Public Attributes

- short **l**
- short **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 structDifferent from util_color only in type (0-4095 vs 0-1)This could easily be migrated to work with those instead.

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

- src/dev_leddriver.h

8.6 daisy::Color Class Reference

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 Member Enumeration Documentation

8.6.1.1 PresetColor

```
enum daisy::Color::PresetColor
```

List of colors that have a preset RGB value

8.6.2 Member Function Documentation

8.6.2.1 Init() [1/2]

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

Initializes the Color with a given preset.

8.6.2.2 Init() [2/2]

```
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

8.6.2.3 Red()

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

Returns the 0-1 value for the given color

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

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 hardware knobs

8.8.2.2 gate_in

```
dsy_gpio daisy::daisy_field::gate_in
```

Array of hardware switches

8.8.2.3 gate_out

```
dsy_gpio daisy::daisy_field::gate_out
```

Gate input.

8.8.2.4 keyboard_sr

```
dsy_sr_4021_handle daisy::daisy_field::keyboard_sr
```

Gate output

8.8.2.5 knobs

```
AnalogControl daisy::daisy_field::knobs[KNOB_LAST]
```

Keyboard shift register

8.8.2.6 switches

```
daisy::Switch daisy::daisy_field::switches[SW_LAST]
```

Daisy seed

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

- 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.

daisy_patch.h

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

8.9.3 Member Function Documentation

8.9.3.1 AudioBlockSize()

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

Get block size

8.9.3.2 AudioCallbackRate()

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

Get callback rate

8.9.3.3 AudioSampleRate()

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

Get sample rate

8.9.3.4 ChangeAudioCallback()

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

Change to a different callback function.

Parameters

cb	New callback function.

8.9.3.5 DelayMs()

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

Wait some ms before going on.

Parameters

del	Delay time in ms.

8.9.3.6 Init()

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

Initializes the daisy seed, and patch hardware.

8.9.3.7 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.

8.9.3.8 StartAdc()

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

Start analog to digital conversion.

8.9.3.9 StartAudio()

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

Start audio output.

Parameters

cb	Audio callback function

8.9.3.10 UpdateAnalogControls()

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

Call at same rate as reading controls for good reads.

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

- 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.

daisy_petal.h

8.10.2 Member Enumeration Documentation

8.10.2.1 Sw

```
enum daisy::DaisyPetal::Sw
```

Enumerator

SW_2	Footswitch
SW_3	Footswitch
SW_4	Footswitch
SW_5	Footswitch
SW_6	Toggle
SW_7	Toggle
SW_LAST	Toggle

8.10.3 Member Function Documentation

8.10.3.1 AudioBlockSize()

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

Get audio block size

8.10.3.2 AudioCallbackRate()

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

Get callback rate

8.10.3.3 AudioSampleRate()

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

Device audio sample rate.

8.10.3.4 ChangeAudioCallback()

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

Change callback function

Parameters

cb	New callback function.

8.10.3.5 ClearLeds()

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

Reset Leds to default values.

8.10.3.6 DebounceControls()

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

Debounce inputs.

8.10.3.7 DelayMs()

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

Wait before moving on.

Parameters

Delay	time in seconds.

8.10.3.8 GetKnobValue()

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

Get value per knob.

Parameters

knob	Which knob to get

Returns

Floating point knob position.

8.10.3.9 Init()

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

Initialize daisy petal

8.10.3.10 SetAudioBlockSize()

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

Set size of audio blocks.

Parameters

Audio	block size

8.10.3.11 SetFootswitchLed()

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

Set footswitch LED

Parameters

idx	Led Index
bright	Brightness

8.10.3.12 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

8.10.3.13 StartAdc()

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

Start analog to digital conversion.

8.10.3.14 StartAudio()

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

Start audio callback

Parameters

cb	Callback function.

8.10.3.15 UpdateAnalogControls()

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

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

8.10.3.16 UpdateLeds()

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

Udate Leds to values you had set.

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

- src/daisy_petal.h

8.11 daisy::DaisyPod Class Reference

```
#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

daisy_seed.h 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 Function Documentation

8.11.2.1 AudioBlockSize()

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

Get block size

8.11.2.2 AudioCallbackRate()

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

Get callback rate

8.11.2.3 AudioSampleRate()

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

Get sample rate

8.11.2.4 ChangeAudioCallback()

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

Switch callback functions

Parameters

cb	New callback function.

8.11.2.5 ClearLeds()

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

Reset Leds

8.11.2.6 DelayMs()

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

Wait for a bit Time to wait in ms.

8.11.2.7 Init()

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

Init related stuff.

8.11.2.8 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.

8.11.2.9 StartAdc()

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

Start analog to digital conversion.

8.11.2.10 StartAudio()

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

Start audio callback

Parameters

cb	Callback function.

8.11.2.11 UpdateAnalogControls()

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

Call at same rate as analog reads for smooth reading.

8.11.2.12 UpdateLeds()

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

Update Leds to set colors

8.11.3 Member Data Documentation

8.11.3.1 seed

```
DaisySeed daisy::DaisyPod::seed
```

Public Members

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

- src/daisy_pod.h

8.12 daisy::DaisySeed Class Reference

```
#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

daisy_seed.h 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. Defaults listed below: TODO: Add defaults

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 sdram_handle

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

While the library is still in heavy development, most of the configuration handles will remain public.

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.

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.

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

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

- src/per_gpio.h

8.16 dsy_gpio_pin Struct Reference

Public Attributes

- dsy_gpio_port **port**
- uint8_t **pin**

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.

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.

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.

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

This struct is identical to the struct provided as "HAL_SD_CardInfoTypeDef" I'm using this to allow users to link to the fatfs middleware without having to then link in the entire HAL to their project.

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_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]

8.21.1 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 datanum_daisychained is the number of devices in a daisy-chain configuration

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

- src/dev_sr_4021.h

8.22 daisy::Encoder Class Reference

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

8.22.1 Member Function Documentation

8.22.1.1 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.

8.22.1.2 FallingEdge()

```
bool daisy::Encoder::FallingEdge ( ) const  [inline]
```

Returns true if the encoder was just released.

8.22.1.3 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.

8.22.1.4 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.

8.22.1.5 Pressed()

```
bool daisy::Encoder::Pressed ( ) const  [inline]
```

Returns true while the encoder is held down.

8.22.1.6 RisingEdge()

```
bool daisy::Encoder::RisingEdge ( ) const  [inline]
```

Returns true if the encoder was just pressed.

8.22.1.7 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

8.23 FontDef Struct Reference

Public Attributes

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

8.23.1 Member Data Documentation

8.23.1.1 data

```
const uint16_t* FontDef::data
```

Pointer to data font data array

8.23.1.2 FontHeight

```
uint8_t FontDef::FontHeight
```

Font height in pixels

8.23.1.3 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

8.24 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 ()

8.24.1 Detailed Description

Generic Class for handling gate inputs through GPIO.

hid_gatein.h

Author

Stephen Hensley

Date

March 2020

8.24.2 Constructor & Destructor Documentation

8.24.2.1 GateIn()

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

GateIn Constructor

8.24.2.2 ~GateIn()

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

GateIn~ Destructor

8.24.3 Member Function Documentation

8.24.3.1 Init()

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

Init Initializes the gate input with specified hardware pin

8.24.3.2 Trig()

```
bool daisy::GateIn::Trig ( )
```

Trig Checks current state of gate input.

Returns

FALSE if pin is low, and TRUE if high

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

- src/hid_gatein.h

8.25 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 ()

8.25.1 Detailed Description

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

hid_led.h

Author

shensley

Date

March 2020

8.25.2 Member Function Documentation

8.25.2.1 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)

8.25.2.2 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.

8.25.2.3 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

8.26 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]

8.26.1 Detailed Description

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

8.26.2 Member Function Documentation

8.26.2.1 AsControlChange()

```
ControlChangeEvent daisy::MidiEvent::AsControlChange ( )  [inline]
```

Returns the data within the MidiEvent as a NoteOnEvent struct.

8.26.2.2 AsNoteOn()

```
NoteOnEvent daisy::MidiEvent::AsNoteOn ( )  [inline]
```

Returns the data within the MidiEvent as a NoteOnEvent struct.

8.26.3 Member Data Documentation

8.26.3.1 type

```
MidiMessageType daisy::MidiEvent::type
```

Newer ish.

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

- src/hid_midi.h

8.27 daisy::MidiHandler Class Reference

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 ()

8.27.1 Member Enumeration Documentation

8.27.1.1 MidiInputMode

```
enum daisy::MidiHandler::MidiInputMode
```

Input and Output can be configured separately Multiple Input modes can be selected by OR'ing the values.

8.27.2 Member Function Documentation

8.27.2.1 HasEvents()

```
bool daisy::MidiHandler::HasEvents ( ) const  [inline]
```

Checks if there are unhandled messages in the queue

8.27.2.2 Init()

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

Initializes the MidiHandler

8.27.2.3 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());

8.27.2.4 PopEvent()

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

Pops the oldest unhandled MidiEvent from the internal queue

8.27.2.5 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

8.28 daisy::NoteOnEvent Struct Reference

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

Public Attributes

- int **channel**
- uint8_t **note**
- uint8_t **velocity**

8.28.1 Detailed Description

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

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

- src/hid_midi.h

8.29 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 ()

8.29.1 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

8.29.2 Member Enumeration Documentation

8.29.2.1 Pins

`enum daisy::OledDisplay::Pins`

GPIO Pins that need to be used independent of peripheral used.

Enumerator

RESET	Data command pi.
NUM_PINS	Reset pin

8.29.3 Member Function Documentation

8.29.3.1 DrawPixel()

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

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

Parameters

x	x Coordinate
y	y coordinate
on	on or off

8.29.3.2 Fill()

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

Fill Fills the entire display with either on/off.

Parameters

on	Sets on or off.

8.29.3.3 Init()

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

TODO: - add I2C Support.

- add configuration for specific spi/i2c peripherals (currently only uses SPI1, w/ hardware controlled chip select.
- re-add support for SSD1306 displays Init Takes an argument for the pin cfg should be a pointer to an array of OledDisplay::NUM_PINS dsy_gpio_pins

8.29.3.4 SetCursor()

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

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

Parameters

x	x pos
y	y pos

8.29.3.5 Update()

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

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

8.29.3.6 WriteChar()

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

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

Parameters

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

8.29.3.7 WriteString()

```
char daisy::OledDisplay::WriteString (
            char * str,
```

```
            FontDef font,
            bool on )
```

WriteString 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

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

- src/hid_oled_display.h

8.30 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 ()

8.30.1 Detailed Description

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

8.30.2 Member Enumeration Documentation

8.30.2.1 Curve

`enum daisy::Parameter::Curve`

Curves are applied to the output signal

Enumerator

EXPONENTIAL	Linear curve
LOGARITHMIC	Exponential curve
CUBE	Logarithmic curve
LAST	Cubic curve

8.30.3 Constructor & Destructor Documentation

8.30.3.1 Parameter()

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

Constructor

8.30.3.2 ~Parameter()

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

Destructor

8.30.4 Member Function Documentation

8.30.4.1 Init()

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

initialize a parameter using an hid_ctrl object. hid_ctrl 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.

8.30.4.2 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.

8.30.4.3 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

8.31 daisy::RgbLed Class Reference

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 ()

8.31.1 Member Function Documentation

8.31.1.1 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

Invert will flip polarity of LED.

8.31.1.2 Set()

```
void daisy::RgbLed::Set (
            float r,
            float g,
            float b )
```

Sets each element of the LED with a floating point number 0-1

8.31.1.3 SetColor()

```
void daisy::RgbLed::SetColor (
            Color c )
```

Sets the RGB using a Color object.

8.31.1.4 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

8.32 daisy::RingBuffer< T, size > Class Template Reference

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)

8.32.1 Member Function Documentation

8.32.1.1 capacity()

```
template<typename T, size_t size>
size_t daisy::RingBuffer< T, size >::capacity ( ) const  [inline]
```

Returns the total size of the ring buffer

8.32.1.2 Flush()

```
template<typename T, size_t size>
void daisy::RingBuffer< T, size >::Flush ( )  [inline]
```

Flushes unread elements from the ring buffer

8.32.1.3 ImmediateRead() [1/2]

```
template<typename T, size_t size>
T daisy::RingBuffer< T, size >::ImmediateRead ( )  [inline]
```

Reads next element from ring buffer immediately

8.32.1.4 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

8.32.1.5 Init()

```
template<typename T, size_t size>
void daisy::RingBuffer< T, size >::Init ( )  [inline]
```

Initializes the Ring Buffer

8.32.1.6 Overwrite() [1/2]

```
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.

8.32.1.7 Overwrite() [2/2]

```
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.

8.32.1.8 Read()

```
template<typename T, size_t size>
T daisy::RingBuffer< T, size >::Read ( )  [inline]
```

Reads the first available element from the ring buffer

8.32.1.9 readable()

```
template<typename T, size_t size>
size_t daisy::RingBuffer< T, size >::readable ( ) const  [inline]
```

Returns number of unread elements in ring buffer

8.32.1.10 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.

8.32.1.11 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.

8.32.1.12 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

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

- src/util_ringbuffer.h

8.33 daisy::RingBuffer< T, 0 > Class Template Reference

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)

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

- src/util_ringbuffer.h

8.34 daisy::SdmmcHandler Class Reference

Public Member Functions

- void Init ()

8.34.1 Member Function Documentation

8.34.1.1 Init()

```
void daisy::SdmmcHandler::Init ( )
```

Initializes the SD Card InterfaceFor 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

8.35 daisy::SdmmcHandlerInit Struct Reference

```
#include <per_sdmmc.h>
```

Public Attributes

- SdmmcBitWidth **bitdepth**
- SdmmcSpeed **speed**

8.35.1 Detailed Description

Structure for setting the options above.

Used to intiailize SdmmcHandler

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

- src/per_sdmmc.h

8.36 ShiftRegister595 Class Reference

`#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 ()

8.36.1 Detailed Description

Maximum Number of chained devices Connect device's QH' pin to the next chips serial input Device Driver for 8-bit shift register CD74HC595 - 8-bit serial to parallel output shift Author**: shensley Date Added**: May 2020

8.36.2 Member Enumeration Documentation

8.36.2.1 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"

8.36.3 Member Function Documentation

8.36.3.1 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.

8.36.3.2 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.

8.36.3.3 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

8.37 daisy::SpiHandle Class Reference

`#include <per_spi.h>`

Public Member Functions

- void Init ()
- void BlockingTransmit (uint8_t *buff, size_t size)

8.37.1 Detailed Description

Handler for serial peripheral interface

8.37.2 Member Function Documentation

8.37.2.1 BlockingTransmit()

```
void daisy::SpiHandle::BlockingTransmit (
            uint8_t * buff,
            size_t size )
```

Blocking transmit

Parameters

*buff	input buffer
size	buffer size

8.37.2.2 Init()

```
void daisy::SpiHandle::Init ( )
```

Initializes handler

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

- src/per_spi.h

8.38 daisy::Switch Class Reference

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

8.38.1 Member Enumeration Documentation

8.38.1.1 Polarity

`enum daisy::Switch::Polarity`

Specifies whether the pressed is HIGH or LOW.

8.38.1.2 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.

8.38.1.3 Type

`enum daisy::Switch::Type`

Specifies the expected behavior of the switch

8.38.2 Member Function Documentation

8.38.2.1 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.

8.38.2.2 FallingEdge()

```
bool daisy::Switch::FallingEdge ( ) const  [inline]
```

Returns true if the button was just released

8.38.2.3 Init()

```
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

8.38.2.4 Pressed()

```
bool daisy::Switch::Pressed ( ) const  [inline]
```

Returns true if the button is held down (or if the toggle is on).

8.38.2.5 RisingEdge()

```
bool daisy::Switch::RisingEdge ( ) const  [inline]
```

Returns true if a button was just pressed.

8.38.2.6 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

8.39 daisy::UartHandler Class Reference

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 ()

8.39.1 Member Function Documentation

8.39.1.1 CheckError()

```
int daisy::UartHandler::CheckError ( )
```

Returns the result of HAL_UART_GetError() to the user.

8.39.1.2 FlushRx()

```
int daisy::UartHandler::FlushRx ( )
```

Flushes the Receive Queue

8.39.1.3 Init()

```
void daisy::UartHandler::Init ( )
```

Initializes the UART Peripheral

8.39.1.4 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.

8.39.1.5 PollTx()

```
int daisy::UartHandler::PollTx (
            uint8_t * buff,
            size_t size )
```

Sends an amount of data in blocking mode.

8.39.1.6 PopRx()

```
uint8_t daisy::UartHandler::PopRx ( )
```

Pops the oldest byte from the FIFO.

8.39.1.7 Readable()

```
size_t daisy::UartHandler::Readable ( )
```

Checks if there are any unread bytes in the FIFO

8.39.1.8 RxActive()

```
bool daisy::UartHandler::RxActive ( )
```

Returns whether Rx DMA is listening or not.

8.39.1.9 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.

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

- src/per_uart.h

8.40 daisy::UsbHandle Class Reference

Public Types

- enum UsbPeriph { **FS_INTERNAL**, **FS_EXTERNAL**, **FS_BOTH** }
- 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)

8.40.1 Member Typedef Documentation

8.40.1.1 ReceiveCallback

```
typedef void(* daisy::UsbHandle::ReceiveCallback) (uint8_t *buff, uint32_t *len)
```

Function called upon reception of a buffer

8.40.2 Member Enumeration Documentation

8.40.2.1 UsbPeriph

```
enum daisy::UsbHandle::UsbPeriph
```

Specified which of the two USB Peripherals to initialize.FS External D- pin is Pin 37 (GPIO31)FS External D+ pin is Pin 38 (GPIO32)

8.40.3 Member Function Documentation

8.40.3.1 Init()

```
void daisy::UsbHandle::Init (
            UsbPeriph dev )
```

Initializes the specified peripheral(s) as USB CDC Devices

8.40.3.2 SetReceiveCallback()

```
void daisy::UsbHandle::SetReceiveCallback (
            ReceiveCallback cb )
```

sets the callback to be called upon reception of new data

8.40.3.3 TransmitExternal()

```
void daisy::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.

8.40.3.4 TransmitInternal()

```
void daisy::UsbHandle::TransmitInternal (
            uint8_t * buff,
            size_t size )
```

Transmits a buffer of 'size' bytes from the on board USB FS port.

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

- src/hid_usb.h

8.41 WAV_FormatTypeDef Struct Reference

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**

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

- src/util_wav_format.h

8.42 daisy::WavFileInfo Struct Reference

`#include <hid_wavplayer.h>`

Public Attributes

- WAV_FormatTypeDef **raw_data**
- char **name** [256]

8.42.1 Detailed Description

Struct containing details of Wav File.TODO: add bitrate, samplerate, length, etc.

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

- src/hid_wavplayer.h

8.43 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

8.43.1 Detailed Description

Class for handling playback of WAV files.

TODO:

- Make template-y to reduce memory usage.

8.43.2 Member Function Documentation

8.43.2.1 Close()

```
int daisy::WavPlayer::Close ( )
```

Closes whatever file is currently open.

8.43.2.2 GetCurrentFile()

```
size_t daisy::WavPlayer::GetCurrentFile ( ) const  [inline]
```

Returns currently selected file.

8.43.2.3 GetLooping()

```
bool daisy::WavPlayer::GetLooping ( ) const  [inline]
```

Returns whether the WavPlayer is looping or not.

8.43.2.4 GetNumberFiles()

```
size_t daisy::WavPlayer::GetNumberFiles ( ) const  [inline]
```

Returns the number of files loaded by the WavPlayer

8.43.2.5 Init()

```
void daisy::WavPlayer::Init ( )
```

Initializes the WavPlayer, loading up to max_files of wav files from an SD Card.

8.43.2.6 Open()

```
int daisy::WavPlayer::Open (
            size_t sel )
```

Opens the file at index sel for reading.

8.43.2.7 Prepare()

```
void daisy::WavPlayer::Prepare ( )
```

Collects buffer for playback when needed.

8.43.2.8 Restart()

```
void daisy::WavPlayer::Restart ( )
```

Resets the playback position to the beginning of the file immediately

8.43.2.9 SetLooping()

```
void daisy::WavPlayer::SetLooping (
            bool loop )  [inline]
```

Sets whether or not the current file will repeat after completing playback.

8.43.2.10 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/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

9.1.1 Detailed Description

: Header for usbd_cdc_if.c file.

Version

: v1.0_Cube

Attention

9.2 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**(...)

9.2.1 Detailed Description

: Header for usbd_conf.c file.

Version

: v1.0_Cube

Attention

Index