

DaisySP

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>libdaisy</b>	<b>1</b>
1.1	Using libdaisy . . . . .	1
1.1.1	daisy.h . . . . .	2
1.1.2	daisy_seed.h . . . . .	2
1.1.3	daisy_platform.h . . . . .	2
<b>2</b>	<b>Module Index</b>	<b>3</b>
2.1	Modules . . . . .	3
<b>3</b>	<b>Namespace Index</b>	<b>5</b>
3.1	Namespace List . . . . .	5
<b>4</b>	<b>Class Index</b>	<b>7</b>
4.1	Class List . . . . .	7
<b>5</b>	<b>File Index</b>	<b>9</b>
5.1	File List . . . . .	9
<b>6</b>	<b>Module Documentation</b>	<b>11</b>
6.1	USBD_CDC_IF . . . . .	11
6.1.1	Detailed Description . . . . .	11
6.2	USBD_CDC_IF_Exported_Defines . . . . .	12
6.3	USBD_CDC_IF_Exported_Types . . . . .	13
6.3.1	Detailed Description . . . . .	13
6.4	USBD_CDC_IF_Exported_Macros . . . . .	14
6.5	USBD_CDC_IF_Exported_Variables . . . . .	15

6.5.1	Detailed Description	15
6.5.2	Variable Documentation	15
6.5.2.1	USBD_Interface_fops_FS	15
6.5.2.2	USBD_Interface_fops_HS	15
6.6	USBD_CDC_IF_Exported_FunctionsPrototype	16
6.6.1	Detailed Description	16
6.7	USBD_CONF	17
6.7.1	Detailed Description	17
6.8	USBD_CONF_Exported_Variables	18
6.9	USBD_CONF_Exported_Defines	19
6.9.1	Detailed Description	19
6.10	USBD_CONF_Exported_Macros	20
6.10.1	Detailed Description	20
6.10.2	Macro Definition Documentation	20
6.10.2.1	USBD_DbgLog	20
6.10.2.2	USBD_Delay	20
6.10.2.3	USBD_ErrLog	21
6.10.2.4	USBD_free	21
6.10.2.5	USBD_malloc	21
6.10.2.6	USBD_memcpy	21
6.10.2.7	USBD_memset	21
6.10.2.8	USBD_UsrLog	21
6.11	USBD_CONF_Exported_Types	22
6.12	USBD_CONF_Exported_FunctionsPrototype	23
6.13	USBD_DESC	24
6.13.1	Detailed Description	24
6.14	USBD_DESC_Exported_Constants	25
6.14.1	Detailed Description	25
6.15	USBD_DESC_Exported_Defines	26
6.16	USBD_DESC_Exported_TypesDefinitions	27
6.17	USBD_DESC_Exported_Macros	28
6.18	USBD_DESC_Exported_Variables	29
6.18.1	Detailed Description	29
6.18.2	Variable Documentation	29
6.18.2.1	FS_Desc	29
6.18.2.2	HS_Desc	29
6.19	USBD_DESC_Exported_FunctionsPrototype	30
6.20	STM32_USB_OTG_DEVICE_LIBRARY	31
6.20.1	Detailed Description	31
6.21	USBD_OTG_DRIVER	32
6.21.1	Detailed Description	32

<b>7 Namespace Documentation</b>	<b>33</b>
7.1 daisy Namespace Reference	33
7.1.1 Detailed Description	34
7.1.2 Enumeration Type Documentation	35
7.1.2.1 anonymous enum	35
7.1.2.2 anonymous enum	35
7.1.2.3 MidiMessageType	35
7.1.2.4 SdmmcBitWidth	35
7.1.2.5 SdmmcMode	35
7.1.2.6 SdmmcSpeed	35
7.1.2.7 SpiPeriph	35
7.1.2.8 SpiPin	36
7.1.3 Function Documentation	36
7.1.3.1 daisy_field_init()	36
<b>8 Class Documentation</b>	<b>37</b>
8.1 daisy::AdcChannelConfig Struct Reference	37
8.1.1 Detailed Description	37
8.1.2 Member Function Documentation	37
8.1.2.1 InitMux()	38
8.1.2.2 InitSingle()	38
8.2 daisy::AdcHandle Class Reference	38
8.2.1 Member Function Documentation	38
8.2.1.1 Get()	39
8.2.1.2 GetMux()	39
8.2.1.3 Init()	39
8.2.1.4 Start()	39
8.2.1.5 Stop()	39
8.3 daisy::AnalogControl Class Reference	40
8.3.1 Member Function Documentation	40
8.3.1.1 Init()	40

8.3.1.2	InitBipolarCv()	40
8.3.1.3	Process()	40
8.3.1.4	Value()	40
8.4	codec_frame_t Struct Reference	41
8.5	color Struct Reference	41
8.5.1	Detailed Description	41
8.6	daisy::Color Class Reference	41
8.6.1	Member Enumeration Documentation	42
8.6.1.1	PresetColor	42
8.6.2	Member Function Documentation	42
8.6.2.1	Init() [1/2]	42
8.6.2.2	Init() [2/2]	42
8.6.2.3	Red()	42
8.7	daisy::ControlChangeEvent Struct Reference	42
8.7.1	Detailed Description	43
8.8	daisy::daisy_field Struct Reference	43
8.9	daisy::DaisyPatch Class Reference	43
8.9.1	Member Enumeration Documentation	44
8.9.1.1	Ctrl	44
8.9.2	Member Function Documentation	44
8.9.2.1	AudioSampleRate()	44
8.9.2.2	Init()	44
8.9.2.3	SetAudioBlockSize()	44
8.9.3	Member Data Documentation	44
8.9.3.1	gate_output	45
8.9.3.2	seed	45
8.10	daisy::DaisyPetal Class Reference	45
8.11	daisy::DaisyPod Class Reference	46
8.11.1	Member Function Documentation	47
8.11.1.1	AudioSampleRate()	47

8.11.1.2	<a href="#">Init()</a>	47
8.11.1.3	<a href="#">SetAudioBlockSize()</a>	47
8.11.2	<a href="#">Member Data Documentation</a>	47
8.11.2.1	<a href="#">seed</a>	47
8.12	<a href="#">daisy::DaisySeed Class Reference</a>	47
8.12.1	<a href="#">Member Function Documentation</a>	48
8.12.1.1	<a href="#">AudioSampleRate()</a>	48
8.12.1.2	<a href="#">Configure()</a>	48
8.12.1.3	<a href="#">GetPin()</a>	48
8.12.1.4	<a href="#">Init()</a>	48
8.12.1.5	<a href="#">SetAudioBlockSize()</a>	49
8.12.1.6	<a href="#">SetLed()</a>	49
8.12.1.7	<a href="#">SetTestPoint()</a>	49
8.12.1.8	<a href="#">StartAudio()</a>	49
8.12.2	<a href="#">Member Data Documentation</a>	49
8.12.2.1	<a href="#">sdram_handle</a>	49
8.13	<a href="#">dsy_audio_handle Struct Reference</a>	49
8.13.1	<a href="#">Detailed Description</a>	50
8.14	<a href="#">dsy_dac_handle Struct Reference</a>	50
8.14.1	<a href="#">Detailed Description</a>	50
8.15	<a href="#">dsy_gpio Struct Reference</a>	50
8.15.1	<a href="#">Detailed Description</a>	51
8.16	<a href="#">dsy_gpio_pin Struct Reference</a>	51
8.17	<a href="#">dsy_i2c_handle Struct Reference</a>	51
8.17.1	<a href="#">Detailed Description</a>	51
8.18	<a href="#">dsy_qspi_handle Struct Reference</a>	51
8.18.1	<a href="#">Detailed Description</a>	52
8.19	<a href="#">dsy_sai_handle Struct Reference</a>	52
8.19.1	<a href="#">Detailed Description</a>	52
8.20	<a href="#">DSY_SD_CardInfoTypeDef Struct Reference</a>	52

8.20.1 Detailed Description . . . . .	53
8.20.2 Member Data Documentation . . . . .	53
8.20.2.1 BlockNbr . . . . .	53
8.20.2.2 BlockSize . . . . .	53
8.20.2.3 CardSpeed . . . . .	53
8.20.2.4 CardType . . . . .	53
8.20.2.5 CardVersion . . . . .	54
8.20.2.6 Class . . . . .	54
8.20.2.7 LogBlockNbr . . . . .	54
8.20.2.8 LogBlockSize . . . . .	54
8.20.2.9 RelCardAdd . . . . .	54
8.21 dsy_sr_4021_handle Struct Reference . . . . .	54
8.21.1 Detailed Description . . . . .	55
8.22 daisy::Encoder Class Reference . . . . .	55
8.22.1 Member Function Documentation . . . . .	55
8.22.1.1 Debounce() . . . . .	55
8.22.1.2 FallingEdge() . . . . .	55
8.22.1.3 Increment() . . . . .	55
8.22.1.4 Init() . . . . .	56
8.22.1.5 Pressed() . . . . .	56
8.22.1.6 RisingEdge() . . . . .	56
8.22.1.7 TimeHeldMs() . . . . .	56
8.23 FontDef Struct Reference . . . . .	56
8.23.1 Member Data Documentation . . . . .	56
8.23.1.1 data . . . . .	57
8.23.1.2 FontHeight . . . . .	57
8.23.1.3 FontWidth . . . . .	57
8.24 daisy::GateIn Class Reference . . . . .	57
8.24.1 Detailed Description . . . . .	57
8.24.2 Constructor & Destructor Documentation . . . . .	58



8.24.2.1	<a href="#">GateIn()</a>	58
8.24.2.2	<a href="#">~GateIn()</a>	58
8.24.3	<a href="#">Member Function Documentation</a>	58
8.24.3.1	<a href="#">Init()</a>	58
8.24.3.2	<a href="#">Trig()</a>	58
8.25	<a href="#">daisy::Led Class Reference</a>	58
8.25.1	<a href="#">Detailed Description</a>	59
8.25.2	<a href="#">Member Function Documentation</a>	59
8.25.2.1	<a href="#">Init()</a>	59
8.25.2.2	<a href="#">Set()</a>	59
8.25.2.3	<a href="#">Update()</a>	60
8.26	<a href="#">daisy::MidiEvent Struct Reference</a>	60
8.26.1	<a href="#">Detailed Description</a>	60
8.26.2	<a href="#">Member Function Documentation</a>	60
8.26.2.1	<a href="#">AsControlChange()</a>	60
8.26.2.2	<a href="#">AsNoteOn()</a>	61
8.26.3	<a href="#">Member Data Documentation</a>	61
8.26.3.1	<a href="#">type</a>	61
8.27	<a href="#">daisy::MidiHandler Class Reference</a>	61
8.27.1	<a href="#">Member Enumeration Documentation</a>	61
8.27.1.1	<a href="#">MidiInputMode</a>	61
8.27.2	<a href="#">Member Function Documentation</a>	62
8.27.2.1	<a href="#">HasEvents()</a>	62
8.27.2.2	<a href="#">Init()</a>	62
8.27.2.3	<a href="#">Parse()</a>	62
8.27.2.4	<a href="#">PopEvent()</a>	62
8.27.2.5	<a href="#">StartReceive()</a>	62
8.28	<a href="#">daisy::NoteOnEvent Struct Reference</a>	62
8.28.1	<a href="#">Detailed Description</a>	63
8.29	<a href="#">daisy::OledDisplay Class Reference</a>	63

8.29.1 Detailed Description . . . . .	63
8.29.2 Member Enumeration Documentation . . . . .	63
8.29.2.1 Pins . . . . .	63
8.29.3 Member Function Documentation . . . . .	64
8.29.3.1 DrawPixel() . . . . .	64
8.29.3.2 Fill() . . . . .	64
8.29.3.3 Init() . . . . .	64
8.29.3.4 SetCursor() . . . . .	65
8.29.3.5 Update() . . . . .	65
8.29.3.6 WriteChar() . . . . .	65
8.29.3.7 WriteString() . . . . .	65
8.30 daisy::Parameter Class Reference . . . . .	66
8.30.1 Detailed Description . . . . .	66
8.30.2 Member Enumeration Documentation . . . . .	66
8.30.2.1 Curve . . . . .	66
8.30.3 Constructor & Destructor Documentation . . . . .	67
8.30.3.1 Parameter() . . . . .	67
8.30.3.2 ~Parameter() . . . . .	67
8.30.4 Member Function Documentation . . . . .	67
8.30.4.1 Init() . . . . .	67
8.30.4.2 Process() . . . . .	67
8.30.4.3 Value() . . . . .	68
8.31 daisy::RgbLed Class Reference . . . . .	68
8.31.1 Member Function Documentation . . . . .	68
8.31.1.1 Init() . . . . .	68
8.31.1.2 Set() . . . . .	68
8.31.1.3 SetColor() . . . . .	69
8.31.1.4 Update() . . . . .	69
8.32 daisy::RingBuffer< T, size > Class Template Reference . . . . .	69
8.32.1 Member Function Documentation . . . . .	69

8.32.1.1	<a href="#">capacity()</a> . . . . .	69
8.32.1.2	<a href="#">Flush()</a> . . . . .	70
8.32.1.3	<a href="#">ImmediateRead()</a> [1/2] . . . . .	70
8.32.1.4	<a href="#">ImmediateRead()</a> [2/2] . . . . .	70
8.32.1.5	<a href="#">Init()</a> . . . . .	70
8.32.1.6	<a href="#">Overwrite()</a> [1/2] . . . . .	70
8.32.1.7	<a href="#">Overwrite()</a> [2/2] . . . . .	70
8.32.1.8	<a href="#">Read()</a> . . . . .	71
8.32.1.9	<a href="#">readable()</a> . . . . .	71
8.32.1.10	<a href="#">Swallow()</a> . . . . .	71
8.32.1.11	<a href="#">writable()</a> . . . . .	71
8.32.1.12	<a href="#">Write()</a> . . . . .	71
8.33	<a href="#">daisy::RingBuffer&lt; T, 0 &gt; Class Template Reference</a> . . . . .	72
8.34	<a href="#">daisy::SdmmcHandler Class Reference</a> . . . . .	72
8.34.1	<a href="#">Member Function Documentation</a> . . . . .	72
8.34.1.1	<a href="#">Init()</a> . . . . .	72
8.35	<a href="#">daisy::SdmmcHandlerInit Struct Reference</a> . . . . .	72
8.35.1	<a href="#">Detailed Description</a> . . . . .	73
8.36	<a href="#">ShiftRegister595 Class Reference</a> . . . . .	73
8.36.1	<a href="#">Detailed Description</a> . . . . .	73
8.36.2	<a href="#">Member Enumeration Documentation</a> . . . . .	73
8.36.2.1	<a href="#">Pins</a> . . . . .	73
8.36.3	<a href="#">Member Function Documentation</a> . . . . .	74
8.36.3.1	<a href="#">Init()</a> . . . . .	74
8.36.3.2	<a href="#">Set()</a> . . . . .	74
8.36.3.3	<a href="#">Write()</a> . . . . .	74
8.37	<a href="#">daisy::SpiHandle Class Reference</a> . . . . .	75
8.37.1	<a href="#">Detailed Description</a> . . . . .	75
8.37.2	<a href="#">Member Function Documentation</a> . . . . .	75
8.37.2.1	<a href="#">BlockingTransmit()</a> . . . . .	75

8.37.2.2	Init()	75
8.38	daisy::Switch Class Reference	76
8.38.1	Member Enumeration Documentation	76
8.38.1.1	Polarity	76
8.38.1.2	Pull	76
8.38.1.3	Type	76
8.38.2	Member Function Documentation	76
8.38.2.1	Debounce()	77
8.38.2.2	FallingEdge()	77
8.38.2.3	Init()	77
8.38.2.4	Pressed()	77
8.38.2.5	RisingEdge()	77
8.38.2.6	TimeHeldMs()	78
8.39	daisy::UartHandler Class Reference	78
8.39.1	Member Function Documentation	78
8.39.1.1	CheckError()	78
8.39.1.2	FlushRx()	78
8.39.1.3	Init()	78
8.39.1.4	PollReceive()	79
8.39.1.5	PollTx()	79
8.39.1.6	PopRx()	79
8.39.1.7	Readable()	79
8.39.1.8	RxActive()	79
8.39.1.9	StartRx()	79
8.40	daisy::UsbHandle Class Reference	80
8.40.1	Member Typedef Documentation	80
8.40.1.1	ReceiveCallback	80
8.40.2	Member Enumeration Documentation	80
8.40.2.1	UsbPeriph	80
8.40.3	Member Function Documentation	80

8.40.3.1	Init()	80
8.40.3.2	SetReceiveCallback()	81
8.40.3.3	TransmitExternal()	81
8.40.3.4	TransmitInternal()	81
8.41	WAV_FormatTypeDef Struct Reference	81
8.42	daisy::WavFileInfo Struct Reference	82
8.42.1	Detailed Description	82
8.43	daisy::WavPlayer Class Reference	82
8.43.1	Detailed Description	82
8.43.2	Member Function Documentation	82
8.43.2.1	Close()	83
8.43.2.2	GetCurrentFile()	83
8.43.2.3	GetLooping()	83
8.43.2.4	GetNumberFiles()	83
8.43.2.5	Init()	83
8.43.2.6	Open()	83
8.43.2.7	Prepare()	83
8.43.2.8	Restart()	84
8.43.2.9	SetLooping()	84
8.43.2.10	Stream()	84
<b>9</b>	<b>File Documentation</b>	<b>85</b>
9.1	src/usbd_cdc_if.h File Reference	85
9.1.1	Detailed Description	85
9.2	src/usbd_conf.h File Reference	86
9.2.1	Detailed Description	86
<b>Index</b>		<b>87</b>



# 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:

STM32_USB_OTG_DEVICE_LIBRARY . . . . .	31
USBDCDCIF . . . . .	11
USBDCDCIF_Exported_Defines . . . . .	12
USBDCDCIF_Exported_Types . . . . .	13
USBDCDCIF_Exported_Macros . . . . .	14
USBDCDCIF_Exported_Variables . . . . .	15
USBDCDCIF_Exported_FunctionsPrototype . . . . .	16
USBDESC . . . . .	24
USBDESC_Exported_Constants . . . . .	25
USBDESC_Exported_Defines . . . . .	26
USBDESC_Exported_TypesDefinitions . . . . .	27
USBDESC_Exported_Macros . . . . .	28
USBDESC_Exported_Variables . . . . .	29
USBDESC_Exported_FunctionsPrototype . . . . .	30
USB_OTG_DRIVER . . . . .	32
USBCONF . . . . .	17
USBCONF_Exported_Variables . . . . .	18
USBCONF_Exported_Defines . . . . .	19
USBCONF_Exported_Macros . . . . .	20
USBCONF_Exported_Types . . . . .	22
USBCONF_Exported_FunctionsPrototype . . . . .	23



## Chapter 3

# Namespace Index

### 3.1 Namespace List

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

<a href="#">daisy</a> . . . . .	<a href="#">33</a>
---------------------------------	--------------------



## Chapter 4

# Class Index

### 4.1 Class List

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

daisy::AdcChannelConfig	37
daisy::AdcHandle	38
daisy::AnalogControl	40
codec_frame_t	41
color	41
daisy::Color	41
daisy::ControlChangeEvent	42
daisy::daisy_field	43
daisy::DaisyPatch	43
daisy::DaisyPetal	45
daisy::DaisyPod	46
daisy::DaisySeed	47
dsy_audio_handle	49
dsy_dac_handle	50
dsy_gpio	50
dsy_gpio_pin	51
dsy_i2c_handle	51
dsy_qspi_handle	51
dsy_sai_handle	52
DSY_SD_CardInfoTypeDef	52
dsy_sr_4021_handle	54
daisy::Encoder	55
FontDef	56
daisy::GateIn	
Generic Class for handling gate inputs through GPIO	57
daisy::Led	
LED Class providing simple Software PWM ability, etc Eventually this will work with hardware PWM, and external LED Driver devices as well	58
daisy::MidiEvent	60
daisy::MidiHandler	61
daisy::NoteOnEvent	62
daisy::OledDisplay	63
daisy::Parameter	66
daisy::RgbLed	68
daisy::RingBuffer< T, size >	69

<a href="#">daisy::RingBuffer&lt; T, 0 &gt;</a>	<a href="#">72</a>
<a href="#">daisy::SdmmcHandler</a>	<a href="#">72</a>
<a href="#">daisy::SdmmcHandlerInit</a>	<a href="#">72</a>
<a href="#">ShiftRegister595</a>	<a href="#">73</a>
<a href="#">daisy::SpiHandle</a>	<a href="#">75</a>
<a href="#">daisy::Switch</a>	<a href="#">76</a>
<a href="#">daisy::UartHandler</a>	<a href="#">78</a>
<a href="#">daisy::UsbHandle</a>	<a href="#">80</a>
<a href="#">WAV_FormatTypeDef</a>	<a href="#">81</a>
<a href="#">daisy::WavFileInfo</a>	<a href="#">82</a>
<a href="#">daisy::WavPlayer</a>	<a href="#">82</a>

## Chapter 5

# File Index

### 5.1 File List

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

src/ <b>daisy.h</b>	??
src/ <b>daisy_core.h</b>	??
src/ <b>daisy_field.h</b>	??
src/ <b>daisy_patch.h</b>	??
src/ <b>daisy_petal.h</b>	??
src/ <b>daisy_pod.h</b>	??
src/ <b>daisy_seed.h</b>	??
src/ <b>dev_codec_ak4556.h</b>	??
src/ <b>dev_codec_pcm3060.h</b>	??
src/ <b>dev_codec_wm8731.h</b>	??
src/ <b>dev_codec_wm8731_frame.h</b>	??
src/ <b>dev_flash_IS25LP064A.h</b>	??
src/ <b>dev_flash_IS25LP080D.h</b>	??
src/ <b>dev_leddriver.h</b>	??
src/ <b>dev_sdram.h</b>	??
src/ <b>dev_sr_4021.h</b>	??
src/ <b>dev_sr_595.h</b>	??
src/ <b>fatfs.h</b>	??
src/ <b>ffconf.h</b>	??
src/ <b>hid_audio.h</b>	??
src/ <b>hid_ctrl.h</b>	??
src/ <b>hid_encoder.h</b>	??
src/ <b>hid_gatein.h</b>	??
src/ <b>hid_led.h</b>	??
src/ <b>hid_midi.h</b>	??
src/ <b>hid_oled_display.h</b>	??
src/ <b>hid_parameter.h</b>	??
src/ <b>hid_rgb_led.h</b>	??
src/ <b>hid_switch.h</b>	??
src/ <b>hid_usb.h</b>	??
src/ <b>hid_wavplayer.h</b>	??
src/ <b>per_adc.h</b>	??
src/ <b>per_dac.h</b>	??
src/ <b>per_gpio.h</b>	??
src/ <b>per_i2c.h</b>	??

src/per_qspi.h	??
src/per_sai.h	??
src/per_sdmmc.h	??
src/per_spi.h	??
src/per_tim.h	??
src/per_uart.h	??
src/stm32h7xx_hal_conf.h	??
src/sys_dma.h	??
src/sys_system.h	??
src/usbd_cdc_if.h	
: Header for usbd_cdc_if.c file	85
src/usbd_conf.h	
: Header for usbd_conf.c file	86
src/usbd_desc.h	??
src/util_bsp_sd_diskio.h	??
src/util_color.h	??
src/util_hal_map.h	??
src/util_oled_fonts.h	??
src/util_ringbuffer.h	??
src/util_sd_diskio.h	??
src/util_unique_id.h	??
src/util_wav_format.h	??



## Chapter 6

# Module Documentation

### 6.1 USB\_D\_CDC\_IF

Usb VCP device module.

#### Modules

- [USB\\_D\\_CDC\\_IF\\_Exported\\_Defines](#)  
*Defines.*
- [USB\\_D\\_CDC\\_IF\\_Exported\\_Types](#)  
*Types.*
- [USB\\_D\\_CDC\\_IF\\_Exported\\_Macros](#)  
*Aliases.*
- [USB\\_D\\_CDC\\_IF\\_Exported\\_Variables](#)  
*Public variables.*
- [USB\\_D\\_CDC\\_IF\\_Exported\\_FunctionsPrototype](#)  
*Public functions declaration.*

#### 6.1.1 Detailed Description

Usb VCP device module.

## 6.2 USB\_D\_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 USB\_D\_CDC\_IF\_Exported\_Macros

Aliases.

Aliases.

## 6.5 USB\_D\_CDC\_IF\_Exported\_Variables

Public variables.

### Variables

- USB\_D\_CDC\_ItfTypeDef [USB\\_Interface\\_fops\\_FS](#)
- USB\_D\_CDC\_ItfTypeDef [USB\\_Interface\\_fops\\_HS](#)

### 6.5.1 Detailed Description

Public variables.

### 6.5.2 Variable Documentation

#### 6.5.2.1 USB\_Interface\_fops\_FS

USB\_D\_CDC\_ItfTypeDef USB\_Interface\_fops\_FS

CDC Interface callback.

#### 6.5.2.2 USB\_Interface\_fops\_HS

USB\_D\_CDC\_ItfTypeDef USB\_Interface\_fops\_HS

CDC Interface callback.

## 6.6 USB\_D\_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

- [USB\\_CONF\\_Exported\\_Variables](#)  
*Public variables.*
- [USB\\_CONF\\_Exported\\_Defines](#)  
*Defines for configuration of the Usb device.*
- [USB\\_CONF\\_Exported\\_Macros](#)  
*Aliases.*
- [USB\\_CONF\\_Exported\\_Types](#)  
*Types.*
- [USB\\_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

- [USB\\_D\\_DESC\\_Exported\\_Constants](#)  
*Constants.*
- [USB\\_D\\_DESC\\_Exported\\_Defines](#)  
*Defines.*
- [USB\\_D\\_DESC\\_Exported\\_TypesDefinitions](#)  
*Types.*
- [USB\\_D\\_DESC\\_Exported\\_Macros](#)  
*Aliases.*
- [USB\\_D\\_DESC\\_Exported\\_Variables](#)  
*Public variables.*
- [USB\\_D\\_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

```
USB_DescriptorsTypeDef FS_Desc
```

Descriptor for the Usb device.

#### 6.18.2.2 HS\_Desc

```
USB_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

- [USB\\_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 [DaisyPetal](#)
- 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

- 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.

### 7.1.2.2 anonymous enum

anonymous enum

All knobs connect to ADC1\_INP10 via CD4051 mux

### 7.1.2.3 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.

### 7.1.2.4 SdmmcBitWidth

enum `daisy::SdmmcBitWidth`

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

### 7.1.2.5 SdmmcMode

enum `daisy::SdmmcMode`

Operating ModeCurrently only FatFS is supported.

### 7.1.2.6 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.

### 7.1.2.7 SpiPeriph

enum `daisy::SpiPeriph`

**Enumerator**

SPI_PERIPH↔ _3	SPI peripheral 1
SPI_PERIPH↔ _6	SPI peripheral 3

**7.1.2.8 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 )
```

```
dsy_gpio_port sw_ports[SW_LAST] = {SW_1_PORT, SW_2_PORT, SW_3_PORT};
```

Init Daisy Seed

Init Switches

Init Gate Input

Init Gate Output

Init LED Driver

2x PCA9685 addresses 0x00, and 0x01 TODO: add multidriver support

Init Keyboard Switches

TODO: add cd4021 with parallel data support

Init ADC (currently in daisy\_seed).

Set up mux pin

Set up CV inputs

Init all 5 channels

Setup Knob/CV Analog Controls

Mapped to ADCs

Start timer

## 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](#) **pin\_**
- [dsy\\_gpio\\_mux\\_pin](#) **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 pin. Internal 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

<i>num_channels</i>	number of ADC channels to initialize
<i>ovs</i>	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 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. `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.  $1.f - \text{input}$ ) or not before being processed. `invert` determines whether the input is inverted (i.e.  $-1.f * \text{input}$ ) or not 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 input. All 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 -> 1.0. Bi-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 struct Different 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

### 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]

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

- src/daisy\_field.h

## 8.9 daisy::DaisyPatch Class Reference

### 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 Member Enumeration Documentation

### 8.9.1.1 Ctrl

```
enum daisy::DaisyPatch::Ctrl
```

Enum of Ctrls to represent the four CV/Knob combos on the Patch

## 8.9.2 Member Function Documentation

### 8.9.2.1 AudioSampleRate()

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

Hardware Accessors

### 8.9.2.2 Init()

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

Initializes the daisy seed, and patch hardware.

### 8.9.2.3 SetAudioBlockSize()

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

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

## 8.9.3 Member Data Documentation

### 8.9.3.1 gate\_output

`dsy_gpio` `daisy::DaisyPatch::gate_output`

TODO: Add class for Gate output

### 8.9.3.2 seed

`DaisySeed` `daisy::DaisyPatch::seed`

These are exposed for the user to access and manipulate directly. Helper functions above provide easier access to much of what they are capable of.

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

- `src/daisy_patch.h`

## 8.10 daisy::DaisyPetal Class Reference

### 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]

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

- `src/daisy_petal.h`

## 8.11 daisy::DaisyPod Class Reference

### 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 Member Function Documentation

#### 8.11.1.1 AudioSampleRate()

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

Hardware Accessors

#### 8.11.1.2 Init()

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

Functions Init related stuff.

#### 8.11.1.3 SetAudioBlockSize()

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

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

### 8.11.2 Member Data Documentation

#### 8.11.2.1 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

### 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 Member Function Documentation

### 8.12.1.1 AudioSampleRate()

```
float daisy::DaisySeed::AudioSampleRate ( )
```

Returns the audio sample rate in Hz as a floating point number.

### 8.12.1.2 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

### 8.12.1.3 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).

### 8.12.1.4 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.

#### 8.12.1.5 SetAudioBlockSize()

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

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

#### 8.12.1.6 SetLed()

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

Sets the state of the built in LED

#### 8.12.1.7 SetTestPoint()

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

Sets the state of the test point near pin 10

#### 8.12.1.8 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.

### 8.12.2 Member Data Documentation

#### 8.12.2.1 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 data num\_daisy chained 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 )
```

Init Initializes an LED using the specified hardware pin.

#### Parameters

<i>pin</i>	chooses LED pin
<i>invert</i>	will set whether to internally invert the brightness due to hardware config.
<i>samplerate</i>	sets the rate at which ' <a href="#">Update()</a> ' will be called (used for software PWM)

#### 8.25.2.2 Set()

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

Set Sets the brightness of the [Led](#).

**Parameters**

<i>val</i>	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 ( )
```

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_US↵  
B_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 MidilInputMode

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](https://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

<i>x</i>	x Coordinate
<i>y</i>	y coordinate
<i>on</i>	on or off

## 8.29.3.2 Fill()

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

Fill Fills the entire display with either on/off.

## Parameters

<i>on</i>	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

<i>x</i>	x pos
<i>y</i>	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

<i>char</i>	character to be written
<i>font</i>	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

<i>str</i>	string to be written
<i>font</i>	font to use
<i>on</i>	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 &amp; 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&lt; T, size &gt; 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 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

### 8.35 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 intiaillize [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 corresponds 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

<i>pin_cfg</i>	is an array of <a href="#">dsy_gpio_pin</a> corresponding the the Pins enum above.
<i>num_daisy_chained</i>	(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

<i>idx</i>	The index starts with QA on the first device and ends with QH on the last device.
<i>state</i>	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

<i>*buff</i>	input buffer
<i>size</i>	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

© 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](http://www.st.com/SLA0044)

## 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

© 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](http://www.st.com/SLA0044)

# Index

- ~GateIn
  - daisy::GateIn, [58](#)
- ~Parameter
  - daisy::Parameter, [67](#)
- AsControlChange
  - daisy::MidiEvent, [60](#)
- AsNoteOn
  - daisy::MidiEvent, [60](#)
- AudioSampleRate
  - daisy::DaisyPatch, [44](#)
  - daisy::DaisyPod, [47](#)
  - daisy::DaisySeed, [48](#)
- BlockNbr
  - DSY\_SD\_CardInfoTypeDef, [53](#)
- BlockSize
  - DSY\_SD\_CardInfoTypeDef, [53](#)
- BlockingTransmit
  - daisy::SpiHandle, [75](#)
- capacity
  - daisy::RingBuffer, [69](#)
- CardSpeed
  - DSY\_SD\_CardInfoTypeDef, [53](#)
- CardType
  - DSY\_SD\_CardInfoTypeDef, [53](#)
- CardVersion
  - DSY\_SD\_CardInfoTypeDef, [53](#)
- CheckError
  - daisy::UartHandler, [78](#)
- Class
  - DSY\_SD\_CardInfoTypeDef, [54](#)
- Close
  - daisy::WavPlayer, [82](#)
- codec\_frame\_t, [41](#)
- color, [41](#)
- Configure
  - daisy::DaisySeed, [48](#)
- Ctrl
  - daisy::DaisyPatch, [44](#)
- Curve
  - daisy::Parameter, [66](#)
- DSY\_SD\_CardInfoTypeDef, [52](#)
  - BlockNbr, [53](#)
  - BlockSize, [53](#)
  - CardSpeed, [53](#)
  - CardType, [53](#)
  - CardVersion, [53](#)
  - Class, [54](#)
  - LogBlockNbr, [54](#)
  - LogBlockSize, [54](#)
  - RelCardAdd, [54](#)
- daisy, [33](#)
  - daisy\_field\_init, [36](#)
  - MidiMessageType, [35](#)
  - SdmmcBitWidth, [35](#)
  - SdmmcMode, [35](#)
  - SdmmcSpeed, [35](#)
  - SpiPeriph, [35](#)
  - SpiPin, [36](#)
- daisy::AdcChannelConfig, [37](#)
  - InitMux, [37](#)
  - InitSingle, [38](#)
- daisy::AdcHandle, [38](#)
  - Get, [38](#)
  - GetMux, [39](#)
  - Init, [39](#)
  - Start, [39](#)
  - Stop, [39](#)
- daisy::AnalogControl, [40](#)
  - Init, [40](#)
  - InitBipolarCv, [40](#)
  - Process, [40](#)
  - Value, [40](#)
- daisy::Color, [41](#)
  - Init, [42](#)
  - PresetColor, [42](#)
  - Red, [42](#)
- daisy::ControlChangeEvent, [42](#)
- daisy::DaisyPatch, [43](#)
  - AudioSampleRate, [44](#)
  - Ctrl, [44](#)
  - gate\_output, [44](#)
  - Init, [44](#)
  - seed, [45](#)
  - SetAudioBlockSize, [44](#)
- daisy::DaisyPetal, [45](#)
- daisy::DaisyPod, [46](#)
  - AudioSampleRate, [47](#)
  - Init, [47](#)
  - seed, [47](#)
  - SetAudioBlockSize, [47](#)
- daisy::DaisySeed, [47](#)
  - AudioSampleRate, [48](#)
  - Configure, [48](#)
  - GetPin, [48](#)
  - Init, [48](#)

- sdram\_handle, 49
- SetAudioBlockSize, 48
- SetLed, 49
- SetTestPoint, 49
- StartAudio, 49
- daisy::Encoder, 55
  - Debounce, 55
  - FallingEdge, 55
  - Increment, 55
  - Init, 55
  - Pressed, 56
  - RisingEdge, 56
  - TimeHeldMs, 56
- daisy::GateIn, 57
  - ~GateIn, 58
  - GateIn, 58
  - Init, 58
  - Trig, 58
- daisy::Led, 58
  - Init, 59
  - Set, 59
  - Update, 60
- daisy::MidiEvent, 60
  - AsControlChange, 60
  - AsNoteOn, 60
  - type, 61
- daisy::MidiHandler, 61
  - HasEvents, 62
  - Init, 62
  - MidiInputMode, 61
  - Parse, 62
  - PopEvent, 62
  - StartReceive, 62
- daisy::NoteOnEvent, 62
- daisy::OledDisplay, 63
  - DrawPixel, 64
  - Fill, 64
  - Init, 64
  - Pins, 63
  - SetCursor, 65
  - Update, 65
  - WriteChar, 65
  - WriteString, 65
- daisy::Parameter, 66
  - ~Parameter, 67
  - Curve, 66
  - Init, 67
  - Parameter, 67
  - Process, 67
  - Value, 67
- daisy::RgbLed, 68
  - Init, 68
  - Set, 68
  - SetColor, 68
  - Update, 69
- daisy::RingBuffer
  - capacity, 69
  - Flush, 69
  - ImmediateRead, 70
  - Init, 70
  - Overwrite, 70
  - Read, 70
  - readable, 71
  - Swallow, 71
  - writable, 71
  - Write, 71
- daisy::RingBuffer< T, 0 >, 72
- daisy::RingBuffer< T, size >, 69
- daisy::SdmmcHandler, 72
  - Init, 72
- daisy::SdmmcHandlerInit, 72
- daisy::SpiHandle, 75
  - BlockingTransmit, 75
  - Init, 75
- daisy::Switch, 76
  - Debounce, 76
  - FallingEdge, 77
  - Init, 77
  - Polarity, 76
  - Pressed, 77
  - Pull, 76
  - RisingEdge, 77
  - TimeHeldMs, 77
  - Type, 76
- daisy::UartHandler, 78
  - CheckError, 78
  - FlushRx, 78
  - Init, 78
  - PollReceive, 78
  - PollTx, 79
  - PopRx, 79
  - Readable, 79
  - RxActive, 79
  - StartRx, 79
- daisy::UsbHandle, 80
  - Init, 80
  - ReceiveCallback, 80
  - SetReceiveCallback, 80
  - TransmitExternal, 81
  - TransmitInternal, 81
  - UsbPeriph, 80
- daisy::WavFileInfo, 82
- daisy::WavPlayer, 82
  - Close, 82
  - GetCurrentFile, 83
  - GetLooping, 83
  - GetNumberFiles, 83
  - Init, 83
  - Open, 83
  - Prepare, 83
  - Restart, 83
  - SetLooping, 84
  - Stream, 84
- daisy::daisy\_field, 43
- daisy\_field\_init
  - daisy, 36

- data
  - FontDef, [56](#)
- Debounce
  - daisy::Encoder, [55](#)
  - daisy::Switch, [76](#)
- DrawPixel
  - daisy::OledDisplay, [64](#)
- dsy\_audio\_handle, [49](#)
- dsy\_dac\_handle, [50](#)
- dsy\_gpio, [50](#)
- dsy\_gpio\_pin, [51](#)
- dsy\_i2c\_handle, [51](#)
- dsy\_qspi\_handle, [51](#)
- dsy\_sai\_handle, [52](#)
- dsy\_sr\_4021\_handle, [54](#)
- FS\_Desc
  - USBD\_DESC\_Exported\_Variables, [29](#)
- FallingEdge
  - daisy::Encoder, [55](#)
  - daisy::Switch, [77](#)
- Fill
  - daisy::OledDisplay, [64](#)
- Flush
  - daisy::RingBuffer, [69](#)
- FlushRx
  - daisy::UartHandler, [78](#)
- FontDef, [56](#)
  - data, [56](#)
  - FontHeight, [57](#)
  - FontWidth, [57](#)
- FontHeight
  - FontDef, [57](#)
- FontWidth
  - FontDef, [57](#)
- gate\_output
  - daisy::DaisyPatch, [44](#)
- Gateln
  - daisy::Gateln, [58](#)
- Get
  - daisy::AdcHandle, [38](#)
- GetCurrentFile
  - daisy::WavPlayer, [83](#)
- GetLooping
  - daisy::WavPlayer, [83](#)
- GetMux
  - daisy::AdcHandle, [39](#)
- GetNumberFiles
  - daisy::WavPlayer, [83](#)
- GetPin
  - daisy::DaisySeed, [48](#)
- HS\_Desc
  - USBD\_DESC\_Exported\_Variables, [29](#)
- HasEvents
  - daisy::MidiHandler, [62](#)
- ImmediateRead
  - daisy::RingBuffer, [70](#)
- Increment
  - daisy::Encoder, [55](#)
- Init
  - daisy::AdcHandle, [39](#)
  - daisy::AnalogControl, [40](#)
  - daisy::Color, [42](#)
  - daisy::DaisyPatch, [44](#)
  - daisy::DaisyPod, [47](#)
  - daisy::DaisySeed, [48](#)
  - daisy::Encoder, [55](#)
  - daisy::Gateln, [58](#)
  - daisy::Led, [59](#)
  - daisy::MidiHandler, [62](#)
  - daisy::OledDisplay, [64](#)
  - daisy::Parameter, [67](#)
  - daisy::RgbLed, [68](#)
  - daisy::RingBuffer, [70](#)
  - daisy::SdmmcHandler, [72](#)
  - daisy::SpiHandle, [75](#)
  - daisy::Switch, [77](#)
  - daisy::UartHandler, [78](#)
  - daisy::UsbHandle, [80](#)
  - daisy::WavPlayer, [83](#)
  - ShiftRegister595, [74](#)
- InitBipolarCv
  - daisy::AnalogControl, [40](#)
- InitMux
  - daisy::AdcChannelConfig, [37](#)
- InitSingle
  - daisy::AdcChannelConfig, [38](#)
- LogBlockNbr
  - DSY\_SD\_CardInfoTypeDef, [54](#)
- LogBlockSize
  - DSY\_SD\_CardInfoTypeDef, [54](#)
- MidiInputMode
  - daisy::MidiHandler, [61](#)
- MidiMessageType
  - daisy, [35](#)
- Open
  - daisy::WavPlayer, [83](#)
- Overwrite
  - daisy::RingBuffer, [70](#)
- Parameter
  - daisy::Parameter, [67](#)
- Parse
  - daisy::MidiHandler, [62](#)
- Pins
  - daisy::OledDisplay, [63](#)
  - ShiftRegister595, [73](#)
- Polarity
  - daisy::Switch, [76](#)
- PollReceive
  - daisy::UartHandler, [78](#)
- PollTx

- daisy::UartHandler, 79
- PopEvent
  - daisy::MidiHandler, 62
- PopRx
  - daisy::UartHandler, 79
- Prepare
  - daisy::WavPlayer, 83
- PresetColor
  - daisy::Color, 42
- Pressed
  - daisy::Encoder, 56
  - daisy::Switch, 77
- Process
  - daisy::AnalogControl, 40
  - daisy::Parameter, 67
- Pull
  - daisy::Switch, 76
- Read
  - daisy::RingBuffer, 70
- Readable
  - daisy::UartHandler, 79
- readable
  - daisy::RingBuffer, 71
- ReceiveCallback
  - daisy::UsbHandle, 80
- Red
  - daisy::Color, 42
- RelCardAdd
  - DSY\_SD\_CardInfoTypeDef, 54
- Restart
  - daisy::WavPlayer, 83
- RisingEdge
  - daisy::Encoder, 56
  - daisy::Switch, 77
- RxActive
  - daisy::UartHandler, 79
- STM32\_USB\_OTG\_DEVICE\_LIBRARY, 31
- SdmmcBitWidth
  - daisy, 35
- SdmmcMode
  - daisy, 35
- SdmmcSpeed
  - daisy, 35
- sdram\_handle
  - daisy::DaisySeed, 49
- seed
  - daisy::DaisyPatch, 45
  - daisy::DaisyPod, 47
- Set
  - daisy::Led, 59
  - daisy::RgbLed, 68
  - ShiftRegister595, 74
- SetAudioBlockSize
  - daisy::DaisyPatch, 44
  - daisy::DaisyPod, 47
  - daisy::DaisySeed, 48
- SetColor
  - daisy::RgbLed, 68
- SetCursor
  - daisy::OledDisplay, 65
- SetLed
  - daisy::DaisySeed, 49
- SetLooping
  - daisy::WavPlayer, 84
- SetReceiveCallback
  - daisy::UsbHandle, 80
- SetTestPoint
  - daisy::DaisySeed, 49
- ShiftRegister595, 73
  - Init, 74
  - Pins, 73
  - Set, 74
  - Write, 74
- SpiPeriph
  - daisy, 35
- SpiPin
  - daisy, 36
- src/usbd\_cdc\_if.h, 85
- src/usbd\_conf.h, 86
- Start
  - daisy::AdcHandle, 39
- StartAudio
  - daisy::DaisySeed, 49
- StartReceive
  - daisy::MidiHandler, 62
- StartRx
  - daisy::UartHandler, 79
- Stop
  - daisy::AdcHandle, 39
- Stream
  - daisy::WavPlayer, 84
- Swallow
  - daisy::RingBuffer, 71
- TimeHeldMs
  - daisy::Encoder, 56
  - daisy::Switch, 77
- TransmitExternal
  - daisy::UsbHandle, 81
- TransmitInternal
  - daisy::UsbHandle, 81
- Trig
  - daisy::Gateln, 58
- Type
  - daisy::Switch, 76
- type
  - daisy::MidiEvent, 61
- USBD\_CDC\_IF\_Exported\_Defines, 12
- USBD\_CDC\_IF\_Exported\_FunctionsPrototype, 16
- USBD\_CDC\_IF\_Exported\_Macros, 14
- USBD\_CDC\_IF\_Exported\_Types, 13
- USBD\_CDC\_IF\_Exported\_Variables, 15
  - USBD\_Interface\_fops\_FS, 15
  - USBD\_Interface\_fops\_HS, 15
- USBD\_CDC\_IF, 11



- USBD\_CONF\_Exported\_Defines, [19](#)
- USBD\_CONF\_Exported\_FunctionsPrototype, [23](#)
- USBD\_CONF\_Exported\_Macros, [20](#)
  - USBD\_DbgLog, [20](#)
  - USBD\_Delay, [20](#)
  - USBD\_ErrLog, [20](#)
  - USBD\_UsrLog, [21](#)
  - USBD\_free, [21](#)
  - USBD\_malloc, [21](#)
  - USBD\_memcpy, [21](#)
  - USBD\_memset, [21](#)
- USBD\_CONF\_Exported\_Types, [22](#)
- USBD\_CONF\_Exported\_Variables, [18](#)
- USBD\_CONF, [17](#)
- USBD\_DESC\_Exported\_Constants, [25](#)
- USBD\_DESC\_Exported\_Defines, [26](#)
- USBD\_DESC\_Exported\_FunctionsPrototype, [30](#)
- USBD\_DESC\_Exported\_Macros, [28](#)
- USBD\_DESC\_Exported\_TypesDefinitions, [27](#)
- USBD\_DESC\_Exported\_Variables, [29](#)
  - FS\_Desc, [29](#)
  - HS\_Desc, [29](#)
- USBD\_DESC, [24](#)
- USBD\_DbgLog
  - USBD\_CONF\_Exported\_Macros, [20](#)
- USBD\_Delay
  - USBD\_CONF\_Exported\_Macros, [20](#)
- USBD\_ErrLog
  - USBD\_CONF\_Exported\_Macros, [20](#)
- USBD\_Interface\_fops\_FS
  - USBD\_CDC\_IF\_Exported\_Variables, [15](#)
- USBD\_Interface\_fops\_HS
  - USBD\_CDC\_IF\_Exported\_Variables, [15](#)
- USBD\_OTG\_DRIVER, [32](#)
- USBD\_UsrLog
  - USBD\_CONF\_Exported\_Macros, [21](#)
- USBD\_free
  - USBD\_CONF\_Exported\_Macros, [21](#)
- USBD\_malloc
  - USBD\_CONF\_Exported\_Macros, [21](#)
- USBD\_memcpy
  - USBD\_CONF\_Exported\_Macros, [21](#)
- USBD\_memset
  - USBD\_CONF\_Exported\_Macros, [21](#)
- Update
  - daisy::Led, [60](#)
  - daisy::OledDisplay, [65](#)
  - daisy::RgbLed, [69](#)
- UsbPeriph
  - daisy::UsbHandle, [80](#)
- Value
  - daisy::AnalogControl, [40](#)
  - daisy::Parameter, [67](#)
- WAV\_FormatTypeDef, [81](#)
- writable
  - daisy::RingBuffer, [71](#)
- Write
  - daisy::RingBuffer, [71](#)
  - ShiftRegister595, [74](#)
- WriteChar
  - daisy::OledDisplay, [65](#)
- WriteString
  - daisy::OledDisplay, [65](#)