DaisySP

Generated by Doxygen 1.8.13

Contents

1 libdaisy **1**

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

2 Module Index **3**

2.1 Modules 3

3 Namespace Index **5**

3.1 Namespace List 5

4 Class Index **7**

4.1 Class List 7

5 File Index **9**

5.1 File List 9

6 Module Documentation **11**

6.1 USBD_CDC_IF_Private_TypesDefinitions 11

6.2 USBD_CDC_IF_Private_Defines 12

6.2.1 Detailed Description 12

6.3 USBD_CDC_IF_Private_Macros 13

6.4 USBD_CDC_IF_Private_Variables 14

6.4.1 Detailed Description 14

6.4.2 Variable Documentation 14

6.4.2.1 UserRxBufferFS 14
6.4.2.2 UserRxBufferHS 14
6.4.2.3 UserTxBufferFS 14
6.4.2.4 UserTxBufferHS 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_Private_FunctionPrototypes 16
6.6.1 Detailed Description 16
6.7 USBD_CDC_IF 17
6.7.1 Detailed Description 17
6.7.2 Function Documentation 18
6.7.2.1 CDC_Transmit_FS() 18
6.7.2.2 CDC_Transmit_HS() 18
6.7.3 Variable Documentation 18
6.7.3.1 USBD_Interface_fops_FS 19
6.7.3.2 USBD_Interface_fops_HS 19
6.8 USBD_CDC_IF_Exported_Defines 20
6.9 USBD_CDC_IF_Exported_Types 21
6.9.1 Detailed Description 21
6.10 USBD_CDC_IF_Exported_Macros 22
6.11 USBD_CDC_IF_Exported_FunctionsPrototype 23
6.11.1 Detailed Description 23
6.11.2 Function Documentation 23
6.11.2.1 CDC_Transmit_FS() 23
6.11.2.2 CDC_Transmit_HS() 24
6.12 USBD_CONF 25
6.12.1 Detailed Description 25

6.13 USBD_CONF_Exported_Variables 26
6.14 USBD_CONF_Exported_Defines 27
6.14.1 Detailed Description 27
6.15 USBD_CONF_Exported_Macros 28
6.15.1 Detailed Description 28
6.15.2 Macro Definition Documentation 28
6.15.2.1 USBD_DbgLog 28
6.15.2.2 USBD_Delay 28
6.15.2.3 USBD_ErrLog 29
6.15.2.4 USBD_free 29
6.15.2.5 USBD_malloc 29
6.15.2.6 USBD_memcpy 29
6.15.2.7 USBD_memset 29
6.15.2.8 USBD_UsrLog 29
6.16 USBD_CONF_Exported_Types 30
6.17 USBD_CONF_Exported_FunctionsPrototype 31
6.18 USBD_DESC_Private_TypesDefinitions 32
6.19 USBD_DESC_Private_Defines 33
6.19.1 Detailed Description 33
6.20 USBD_DESC_Private_Macros 34
6.21 USBD_DESC_Private_FunctionPrototypes 35
6.21.1 Detailed Description 35
6.22 USBD_DESC_Private_Variables 36
6.22.1 Detailed Description 36
6.22.2 Variable Documentation 36
6.22.2.1 __ALIGN_END 36
6.22.2.2 FS_Desc 37
6.22.2.3 HS_Desc 37
6.23 USBD_DESC_Private_Functions 38
6.23.1 Detailed Description 38

6.23.2 Function Documentation 38
6.23.2.1 USBD_FS_ConfigStrDescriptor() 38
6.23.2.2 USBD_FS_DeviceDescriptor() 39
6.23.2.3 USBD_FS_InterfaceStrDescriptor() 39
6.23.2.4 USBD_FS_LangIDStrDescriptor() 40
6.23.2.5 USBD_FS_ManufacturerStrDescriptor() 40
6.23.2.6 USBD_FS_ProductStrDescriptor() 40
6.23.2.7 USBD_FS_SerialStrDescriptor() 41
6.23.2.8 USBD_HS_ConfigStrDescriptor() 41
6.23.2.9 USBD_HS_DeviceDescriptor() 42
6.23.2.10 USBD_HS_InterfaceStrDescriptor() 42
6.23.2.11 USBD_HS_LangIDStrDescriptor() 42
6.23.2.12 USBD_HS_ManufacturerStrDescriptor() 43
6.23.2.13 USBD_HS_ProductStrDescriptor() 43
6.23.2.14 USBD_HS_SerialStrDescriptor() 44
6.24 USBD_DESC 45
6.24.1 Detailed Description 45
6.25 USBD_DESC_Exported_Constants 46
6.25.1 Detailed Description 46
6.26 USBD_DESC_Exported_Defines 47
6.27 USBD_DESC_Exported_TypesDefinitions 48
6.28 USBD_DESC_Exported_Macros 49
6.29 USBD_DESC_Exported_Variables 50
6.29.1 Detailed Description 50
6.29.2 Variable Documentation 50
6.29.2.1 FS_Desc 50
6.29.2.2 HS_Desc 50
6.30 USBD_DESC_Exported_FunctionsPrototype 51
6.31 CMSIS 52
6.31.1 Detailed Description 52

6.32 Stm32h7xx_system 53
6.32.1 Detailed Description 53
6.33 STM32H7xx_System_Private_Includes 54
6.33.1 Detailed Description 54
6.33.2 Macro Definition Documentation 54
6.33.2.1 CSI_VALUE 54
6.33.2.2 HSE_VALUE 54
6.33.2.3 HSI_VALUE 54
6.34 STM32H7xx_System_Private_TypesDefinitions 55
6.35 STM32H7xx_System_Private_Defines 56
6.35.1 Detailed Description 56
6.35.2 Macro Definition Documentation 56
6.35.2.1 VECT_TAB_OFFSET 56
6.36 STM32H7xx_System_Private_Macros 57
6.37 STM32H7xx_System_Private_Variables 58
6.37.1 Detailed Description 58
6.38 STM32H7xx_System_Private_FunctionPrototypes 59
6.39 STM32H7xx_System_Private_Functions 60
6.39.1 Detailed Description 60
6.39.2 Function Documentation 60
6.39.2.1 SystemCoreClockUpdate() 60
6.39.2.2 SystemInit() 61
6.40 STM32_USB_OTG_DEVICE_LIBRARY 62
6.40.1 Detailed Description 62
6.41 USBD_OTG_DRIVER 63
6.41.1 Detailed Description 63

7 Namespace Documentation 65

7.1 daisy Namespace Reference 65

7.1.1 Detailed Description 66

7.1.2 Enumeration Type Documentation 67

7.1.2.1 anonymous enum 67

7.1.2.2 anonymous enum 67

7.1.2.3 MidiMessageType 67

7.1.2.4 SdmmcBitWidth 67

7.1.2.5 SdmmcMode 67

7.1.2.6 SdmmcSpeed 67

7.1.3 Function Documentation 67

7.1.3.1 daisy_field_init() 68

8 Class Documentation 69

8.1 daisy::AdcChannelConfig Struct Reference 69

8.1.1 Detailed Description 69

8.1.2 Member Function Documentation 69

8.1.2.1 InitMux() 70

8.1.2.2 InitSingle() 70

8.2 daisy::AdcHandle Class Reference 70

8.2.1 Member Function Documentation 70

8.2.1.1 Get() 71

8.2.1.2 GetMux() 71

8.2.1.3 Init() 71

8.2.1.4 Start() 71

8.2.1.5 Stop() 71

8.3 daisy::AnalogControl Class Reference 71

8.3.1 Member Function Documentation 72

8.3.1.1 Init() 72

8.3.1.2 InitBipolarCv() 72

8.3.1.3 Process() 72

8.3.1.4 Value() . . . 72
8.4 codec_frame_t Struct Reference . . . 73
8.5 color Struct Reference . . . 73
8.5.1 Detailed Description . . . 73
8.6 daisy::Color Class Reference . . . 73
8.6.1 Member Enumeration Documentation . . . 74
8.6.1.1 PresetColor . . . 74
8.6.2 Member Function Documentation . . . 74
8.6.2.1 Init() [1/2] . . . 74
8.6.2.2 Init() [2/2] . . . 74
8.6.2.3 Red() . . . 74
8.7 daisy::ControlChangeEvent Struct Reference . . . 74
8.7.1 Detailed Description . . . 75
8.8 daisy::daisy_field Struct Reference . . . 75
8.9 daisy::DaisyPatch Class Reference . . . 75
8.9.1 Member Enumeration Documentation . . . 76
8.9.1.1 Ctrl . . . 76
8.9.2 Member Function Documentation . . . 76
8.9.2.1 AudioSampleRate() . . . 76
8.9.2.2 Init() . . . 76
8.9.2.3 SetAudioBlockSize() . . . 76
8.9.3 Member Data Documentation . . . 76
8.9.3.1 gate_output . . . 77
8.9.3.2 seed . . . 77
8.10 daisy::DaisyPetal Class Reference . . . 77
8.11 daisy::DaisyPod Class Reference . . . 78
8.11.1 Member Function Documentation . . . 79
8.11.1.1 AudioSampleRate() . . . 79
8.11.1.2 Init() . . . 79
8.11.1.3 SetAudioBlockSize() . . . 79

8.11.2 Member Data Documentation 79
8.11.2.1 seed 79
8.12 daisy::DaisySeed Class Reference 79
8.12.1 Member Function Documentation 80
8.12.1.1 AudioSampleRate() 80
8.12.1.2 Configure() 80
8.12.1.3 GetPin() 80
8.12.1.4 Init() 80
8.12.1.5 SetAudioBlockSize() 81
8.12.1.6 SetLed() 81
8.12.1.7 SetTestPoint() 81
8.12.1.8 StartAudio() 81
8.12.2 Member Data Documentation 81
8.12.2.1 sdram_handle 81
8.13 dsy_adc Struct Reference 82
8.14 dsy_audio Struct Reference 82
8.15 dsy_audio_handle Struct Reference 82
8.15.1 Detailed Description 83
8.16 dsy_dac_handle Struct Reference 83
8.16.1 Detailed Description 83
8.17 dsy_dac_t Struct Reference 83
8.18 dsy_gpio Struct Reference 84
8.18.1 Detailed Description 84
8.19 dsy_gpio_pin Struct Reference 84
8.20 dsy_i2c_handle Struct Reference 84
8.20.1 Detailed Description 84
8.21 dsy_led_driver_t Struct Reference 85
8.22 dsy_qspi Struct Reference 85
8.23 dsy_qspi_handle Struct Reference 85
8.23.1 Detailed Description 86

8.24 dsy_sai_handle Struct Reference 86
8.24.1 Detailed Description 86
8.25 DSY_SD_CardInfoTypeDef Struct Reference 86
8.25.1 Detailed Description 87
8.25.2 Member Data Documentation 87
8.25.2.1 BlockNbr 87
8.25.2.2 BlockSize 87
8.25.2.3 CardSpeed 87
8.25.2.4 CardType 87
8.25.2.5 CardVersion 87
8.25.2.6 Class 87
8.25.2.7 LogBlockNbr 88
8.25.2.8 LogBlockSize 88
8.25.2.9 RelCardAdd 88
8.26 dsy_sdram_t Struct Reference 88
8.27 dsy_sr_4021_handle Struct Reference 88
8.27.1 Detailed Description 89
8.28 dsy_tim Struct Reference 89
8.29 dsy_wm8731_handle_t Struct Reference 89
8.30 daisy::Encoder Class Reference 89
8.30.1 Member Function Documentation 90
8.30.1.1 Debounce() 90
8.30.1.2 FallingEdge() 90
8.30.1.3 Increment() 90
8.30.1.4 Init() 90
8.30.1.5 Pressed() 90
8.30.1.6 RisingEdge() 90
8.30.1.7 TimeHeldMs() 91
8.31 FontDef Struct Reference 91
8.31.1 Member Data Documentation 91

8.31.1.1 data 91
8.31.1.2 FontHeight 91
8.31.1.3 FontWidth 91
8.32 daisy::GateIn Class Reference 92
8.32.1 Member Function Documentation 92
8.32.1.1 Init() 92
8.32.1.2 Trig() 92
8.33 led Struct Reference 92
8.34 daisy::Led Class Reference 93
8.34.1 Member Function Documentation 93
8.34.1.1 Init() 93
8.34.1.2 Set() 93
8.34.1.3 Update() 93
8.35 daisy::MidiEvent Struct Reference 93
8.35.1 Detailed Description 94
8.35.2 Member Function Documentation 94
8.35.2.1 AsControlChange() 94
8.35.2.2 AsNoteOn() 94
8.35.3 Member Data Documentation 94
8.35.3.1 type 94
8.36 daisy::MidiHandler Class Reference 95
8.36.1 Member Enumeration Documentation 95
8.36.1.1 MidiInputMode 95
8.36.2 Member Function Documentation 95
8.36.2.1 HasEvents() 95
8.36.2.2 Init() 95
8.36.2.3 Parse() 96
8.36.2.4 PopEvent() 96
8.36.2.5 StartReceive() 96
8.37 daisy::NoteOnEvent Struct Reference 96

8.37.1 Detailed Description 96
8.38 NVIC_TypeDef Struct Reference 97
8.39 daisy::OledDisplay Class Reference 97
8.39.1 Member Enumeration Documentation 97
8.39.1.1 Pins 97
8.39.2 Member Function Documentation 98
8.39.2.1 DrawPixel() 98
8.39.2.2 Fill() 98
8.39.2.3 Init() 98
8.39.2.4 SetCursor() 98
8.39.2.5 Update() 98
8.39.2.6 WriteChar() 99
8.39.2.7 WriteString() 99
8.40 daisy::Parameter Class Reference 99
8.40.1 Member Enumeration Documentation 99
8.40.1.1 Curve 99
8.40.2 Member Function Documentation 100
8.40.2.1 Init() 100
8.40.2.2 Process() 100
8.40.2.3 Value() 100
8.41 rgb_led Struct Reference 100
8.42 daisy::RgbLed Class Reference 101
8.42.1 Member Function Documentation 101
8.42.1.1 Init() 101
8.42.1.2 Set() 101
8.42.1.3 SetColor() 101
8.42.1.4 Update() 101
8.43 daisy::RingBuffer< T, size > Class Template Reference 102
8.43.1 Member Function Documentation 102
8.43.1.1 capacity() 102

8.43.1.2 Flush() 102
8.43.1.3 ImmediateRead() [1/2] 102
8.43.1.4 ImmediateRead() [2/2] 102
8.43.1.5 Init() 103
8.43.1.6 Overwrite() [1/2] 103
8.43.1.7 Overwrite() [2/2] 103
8.43.1.8 Read() 103
8.43.1.9 readable() 103
8.43.1.10 Swallow() 103
8.43.1.11 writable() 104
8.43.1.12 Write() 104
8.44 daisy::RingBuffer< T, 0 > Class Template Reference 104
8.45 daisy::SdmmcHandler Class Reference 104
8.45.1 Member Function Documentation 105
8.45.1.1 Init() 105
8.46 daisy::SdmmcHandlerInit Struct Reference 105
8.46.1 Detailed Description 105
8.47 ShiftRegister595 Class Reference 105
8.47.1 Member Enumeration Documentation 106
8.47.1.1 Pins 106
8.47.2 Member Function Documentation 106
8.47.2.1 Init() 106
8.47.2.2 Set() 106
8.47.2.3 Write() 106
8.48 daisy::SpiHandle Class Reference 107
8.49 SSD1309_t Struct Reference 107
8.50 daisy::Switch Class Reference 107
8.50.1 Member Enumeration Documentation 107
8.50.1.1 Polarity 108
8.50.1.2 Pull 108

8.50.1.3 Type 108
8.50.2 Member Function Documentation 108
8.50.2.1 Debounce() 108
8.50.2.2 FallingEdge() 108
8.50.2.3 Init() 108
8.50.2.4 Pressed() 109
8.50.2.5 RisingEdge() 109
8.50.2.6 TimeHeldMs() 109
8.51 uart_handle Struct Reference 109
8.52 daisy::UartHandler Class Reference 109
8.52.1 Member Function Documentation 110
8.52.1.1 CheckError() 110
8.52.1.2 FlushRx() 110
8.52.1.3 Init() 110
8.52.1.4 PollReceive() 110
8.52.1.5 PollTx() 110
8.52.1.6 PopRx() 110
8.52.1.7 Readable() 111
8.52.1.8 RxActive() 111
8.52.1.9 StartRx() 111
8.53 daisy::UsbHandle Class Reference 111
8.53.1 Member Typedef Documentation 111
8.53.1.1 ReceiveCallback 112
8.53.2 Member Enumeration Documentation 112
8.53.2.1 UsbPeriph 112
8.53.3 Member Function Documentation 112
8.53.3.1 Init() 112
8.53.3.2 SetReceiveCallback() 112
8.53.3.3 TransmitExternal() 112
8.53.3.4 TransmitInternal() 113

8.54 WAV_FormatTypeDef Struct Reference 113
8.55 daisy::WavFileInfo Struct Reference 113
8.55.1 Detailed Description 114
8.56 daisy::WavPlayer Class Reference 114
8.56.1 Detailed Description 114
8.56.2 Member Function Documentation 114
8.56.2.1 Close() 114
8.56.2.2 GetCurrentFile() 115
8.56.2.3 GetLooping() 115
8.56.2.4 GetNumberFiles() 115
8.56.2.5 Init() 115
8.56.2.6 Open() 115
8.56.2.7 Prepare() 115
8.56.2.8 Restart() 115
8.56.2.9 SetLooping() 115
8.56.2.10 Stream() 115

9 File Documentation 117

9.1 src/system_stm32h7xx.c File Reference 117
9.1.1 Detailed Description 118
9.2 src/usbd_cdc_if.c File Reference 118
9.2.1 Detailed Description 119
9.3 src/usbd_cdc_if.h File Reference 119
9.3.1 Detailed Description 120
9.4 src/usbd_conf.c File Reference 120
9.4.1 Detailed Description 122
9.4.2 Function Documentation 122
9.4.2.1 HAL_PCD_ConnectCallback() 122
9.4.2.2 HAL_PCD_DataInStageCallback() 123
9.4.2.3 HAL_PCD_DataOutStageCallback() 123
9.4.2.4 HAL_PCD_DisconnectCallback() 124

9.4.2.5 HAL_PCD_ISOINIncompleteCallback() 124
9.4.2.6 HAL_PCD_ISOOUTIncompleteCallback() 124
9.4.2.7 HAL_PCD_MspDeInit() 125
9.4.2.8 HAL_PCD_MspInit() 125
9.4.2.9 HAL_PCD_ResetCallback() 125
9.4.2.10 HAL_PCD_ResumeCallback() 125
9.4.2.11 HAL_PCD_SetupStageCallback() 127
9.4.2.12 HAL_PCD_SOFCallback() 127
9.4.2.13 HAL_PCD_SuspendCallback() 127
9.4.2.14 USBD_Get_USB_Status() 128
9.4.2.15 USBD_LL_ClearStallEP() 128
9.4.2.16 USBD_LL_CloseEP() 129
9.4.2.17 USBD_LL_DeInit() 129
9.4.2.18 USBD_LL_Delay() 129
9.4.2.19 USBD_LL_FlushEP() 130
9.4.2.20 USBD_LL_GetRxDataSize() 130
9.4.2.21 USBD_LL_Init() 130
9.4.2.22 USBD_LL_IsStallEP() 131
9.4.2.23 USBD_LL_OpenEP() 131
9.4.2.24 USBD_LL_PrepareReceive() 132
9.4.2.25 USBD_LL_SetUSBAddress() 132
9.4.2.26 USBD_LL_StallEP() 133
9.4.2.27 USBD_LL_Start() 133
9.4.2.28 USBD_LL_Stop() 133
9.4.2.29 USBD_LL_Transmit() 134
9.5 src/usbd_conf.h File Reference 134
9.5.1 Detailed Description 135
9.6 src/usbd_desc.c File Reference 135
9.6.1 Detailed Description 136

Index **137**

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:

CMSIS 52
- Stm32h7xx_system 53
  - STM32H7xx_System_Private_Includes 54
  - STM32H7xx_System_Private_TypesDefinitions 55
  - STM32H7xx_System_Private_Defines 56
  - STM32H7xx_System_Private_Macros 57
  - STM32H7xx_System_Private_Variables 58
  - STM32H7xx_System_Private_FunctionPrototypes 59
  - STM32H7xx_System_Private_Functions 60

STM32_USB_OTG_DEVICE_LIBRARY 62
- USBD_CDC_IF 17
  - USBD_CDC_IF_Private_TypesDefinitions 11
  - USBD_CDC_IF_Private_Defines 12
  - USBD_CDC_IF_Private_Macros 13
  - USBD_CDC_IF_Private_Variables 14
  - USBD_CDC_IF_Exported_Variables 15
  - USBD_CDC_IF_Private_FunctionPrototypes 16
  - USBD_CDC_IF_Exported_Defines 20
  - USBD_CDC_IF_Exported_Types 21
  - USBD_CDC_IF_Exported_Macros 22
  - USBD_CDC_IF_Exported_FunctionsPrototype 23
- USBD_DESC 45
  - USBD_DESC_Private_TypesDefinitions 32
  - USBD_DESC_Private_Defines 33
  - USBD_DESC_Private_Macros 34
  - USBD_DESC_Private_FunctionPrototypes 35
  - USBD_DESC_Private_Variables 36
  - USBD_DESC_Private_Functions 38
  - USBD_DESC_Exported_Constants 46
  - USBD_DESC_Exported_Defines 47
  - USBD_DESC_Exported_TypesDefinitions 48
  - USBD_DESC_Exported_Macros 49
  - USBD_DESC_Exported_Variables 50
  - USBD_DESC_Exported_FunctionsPrototype 51

USBD_OTG_DRIVER 63
USBD_CONF 25
USBD_CONF_Exported_Variables 26
USBD_CONF_Exported_Defines 27
USBD_CONF_Exported_Macros 28
USBD_CONF_Exported_Types 30
USBD_CONF_Exported_FunctionsPrototype 31

Chapter 3

Namespace Index

3.1 Namespace List

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

daisy 65

Chapter 4

Class Index

4.1 Class List

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

daisy::AdcChannelConfig 69
daisy::AdcHandle 70
daisy::AnalogControl 71
codec_frame_t 73
color 73
daisy::Color 73
daisy::ControlChangeEvent 74
daisy::daisy_field 75
daisy::DaisyPatch 75
daisy::DaisyPetal 77
daisy::DaisyPod 78
daisy::DaisySeed 79
dsy_adc 82
dsy_audio 82
dsy_audio_handle 82
dsy_dac_handle 83
dsy_dac_t 83
dsy_gpio 84
dsy_gpio_pin 84
dsy_i2c_handle 84
dsy_led_driver_t 85
dsy_qspi 85
dsy_qspi_handle 85
dsy_sai_handle 86
DSY_SD_CardInfoTypeDef 86
dsy_sdram_t 88
dsy_sr_4021_handle 88
dsy_tim 89
dsy_wm8731_handle_t 89
daisy::Encoder 89
FontDef 91
daisy::GateIn 92
led 92
daisy::Led 93
daisy::MidiEvent 93

daisy::MidiHandler 95
daisy::NoteOnEvent 96
NVIC_TypeDef 97
daisy::OledDisplay 97
daisy::Parameter 99
rgb_led 100
daisy::RgbLed 101
daisy::RingBuffer< T, size > 102
daisy::RingBuffer< T, 0 > 104
daisy::SdmmcHandler 104
daisy::SdmmcHandlerInit 105
ShiftRegister595 105
daisy::SpiHandle 107
SSD1309_t 107
daisy::Switch 107
uart_handle 109
daisy::UartHandler 109
daisy::UsbHandle 111
WAV_FormatTypeDef 113
daisy::WavFileInfo 113
daisy::WavPlayer 114

Chapter 5

File Index

5.1 File List

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

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

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/system_stm32h7xx.c
CMSIS Cortex-Mx Device Peripheral Access Layer System Source File 117
src/usbd_cdc_if.c
: Usb device for Virtual Com Port 118
src/usbd_cdc_if.h
: Header for usbd_cdc_if.c file 119
src/usbd_conf.c
: This file implements the board support package for the USB device library 120
src/usbd_conf.h
: Header for usbd_conf.c file 134
src/usbd_desc.c
: This file implements the USB device descriptors 135
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 USBD_CDC_IF_Private_TypesDefinitions

Private types.

Private types.

6.2 USBD_CDC_IF_Private_Defines

Private defines.

Macros

- #define **APP_RX_DATA_SIZE** 2048
- #define **APP_TX_DATA_SIZE** 2048

6.2.1 Detailed Description

Private defines.

6.3 USBD_CDC_IF_Private_Macros

Private macros.

Private macros.

6.4 USBD_CDC_IF_Private_Variables

Private variables.

Functions

- void **dummy_rx_callback** (uint8_t ∗buf, uint32_t ∗len)

Variables

- uint8_t UserRxBufferFS [2048]
- uint8_t UserTxBufferFS [2048]
- uint8_t UserRxBufferHS [2048]
- uint8_t UserTxBufferHS [2048]
- CDC_ReceiveCallback **rx_callback_fs**

6.4.1 Detailed Description

Private variables.

6.4.2 Variable Documentation

6.4.2.1 UserRxBufferFS

```
uint8_t UserRxBufferFS[2048]
```

Received data over USB are stored in this buffer

6.4.2.2 UserRxBufferHS

```
uint8_t UserRxBufferHS[2048]
```

Received data over USB are stored in this buffer

6.4.2.3 UserTxBufferFS

```
uint8_t UserTxBufferFS[2048]
```

Data to send over USB CDC are stored in this buffer

6.4.2.4 UserTxBufferHS

```
uint8_t UserTxBufferHS[2048]
```

Data to send over USB CDC are stored in this buffer

6.5 USBD_CDC_IF_Exported_Variables

Public variables.

Variables

- USBD_HandleTypeDef **hUsbDeviceFS**
- USBD_HandleTypeDef **hUsbDeviceHS**
- USBD_CDC_ItfTypeDef USBD_Interface_fops_FS
- USBD_CDC_ItfTypeDef USBD_Interface_fops_HS

6.5.1 Detailed Description

Public variables.

6.5.2 Variable Documentation

6.5.2.1 USBD_Interface_fops_FS

```
USBD_CDC_ItfTypeDef USBD_Interface_fops_FS
```

CDC Interface callback.

6.5.2.2 USBD_Interface_fops_HS

```
USBD_CDC_ItfTypeDef USBD_Interface_fops_HS
```

CDC Interface callback.

6.6 USBD_CDC_IF_Private_FunctionPrototypes

Private functions declaration.

6.6.1 Detailed Description

Private functions declaration.

6.7 USBD_CDC_IF

Usb VCP device module.

Modules

- USBD_CDC_IF_Private_TypesDefinitions

  Private types.
- USBD_CDC_IF_Private_Defines

  Private defines.
- USBD_CDC_IF_Private_Macros

  Private macros.
- USBD_CDC_IF_Private_Variables

  Private variables.
- USBD_CDC_IF_Exported_Variables

  Public variables.
- USBD_CDC_IF_Private_FunctionPrototypes

  Private functions declaration.
- USBD_CDC_IF_Exported_Defines

  Defines.
- USBD_CDC_IF_Exported_Types

  Types.
- USBD_CDC_IF_Exported_Macros

  Aliases.
- USBD_CDC_IF_Exported_FunctionsPrototype

  Public functions declaration.

Functions

- uint8_t CDC_Transmit_FS (uint8_t ∗Buf, uint16_t Len)

  CDC_Transmit_FS Data to send over USB IN endpoint are sent over CDC interface through this function.
- uint8_t CDC_Transmit_HS (uint8_t ∗Buf, uint16_t Len)

  Data to send over USB IN endpoint are sent over CDC interface through this function.
- void **CDC_Set_Rx_Callback_FS** (CDC_ReceiveCallback cb)

Variables

- USBD_CDC_ItfTypeDef USBD_Interface_fops_FS = {CDC_Init_FS, CDC_DeInit_FS, CDC_Control_FS, C←
  DC_Receive_FS}
- USBD_CDC_ItfTypeDef USBD_Interface_fops_HS = {CDC_Init_HS, CDC_DeInit_HS, CDC_Control_HS, CDC_Receive_HS}

6.7.1 Detailed Description

Usb VCP device module.

6.7.2 Function Documentation

6.7.2.1 CDC_Transmit_FS()

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

CDC_Transmit_FS Data to send over USB IN endpoint are sent over CDC interface through this function.

Note

Parameters

Buf	Buffer of data to be sent
Len	Number of data to be sent (in bytes)

Return values

USBD_OK	if all operations are OK else USBD_FAIL or USBD_BUSY

6.7.2.2 CDC_Transmit_HS()

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

Data to send over USB IN endpoint are sent over CDC interface through this function.

Parameters

Buf	Buffer of data to be sent
Len	Number of data to be sent (in bytes)

Return values

Result	of the operation: USBD_OK if all operations are OK else USBD_FAIL or USBD_BUSY

6.7.3 Variable Documentation

6.7.3.1 USBD_Interface_fops_FS

```
USBD_CDC_ItfTypeDef USBD_Interface_fops_FS = {CDC_Init_FS, CDC_DeInit_FS, CDC_Control_FS, CD←
C_Receive_FS}
```

CDC Interface callback.

6.7.3.2 USBD_Interface_fops_HS

```
USBD_CDC_ItfTypeDef USBD_Interface_fops_HS = {CDC_Init_HS, CDC_DeInit_HS, CDC_Control_HS, CD←
C_Receive_HS}
```

CDC Interface callback.

6.8 USBD_CDC_IF_Exported_Defines

Defines.

Defines.

6.9 USBD_CDC_IF_Exported_Types

Types.

Typedefs

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

6.9.1 Detailed Description

Types.

6.10 USBD_CDC_IF_Exported_Macros

Aliases.

Aliases.

6.11 USBD_CDC_IF_Exported_FunctionsPrototype

Public functions declaration.

Functions

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

  CDC_Transmit_FS Data to send over USB IN endpoint are sent over CDC interface through this function.
- uint8_t CDC_Transmit_HS (uint8_t ∗Buf, uint16_t Len)

  Data to send over USB IN endpoint are sent over CDC interface through this function.

6.11.1 Detailed Description

Public functions declaration.

6.11.2 Function Documentation

6.11.2.1 CDC_Transmit_FS()

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

CDC_Transmit_FS Data to send over USB IN endpoint are sent over CDC interface through this function.

Note

Parameters

Buf	Buffer of data to be sent
Len	Number of data to be sent (in bytes)

Return values

USBD_OK	if all operations are OK else USBD_FAIL or USBD_BUSY

6.11.2.2 CDC_Transmit_HS()

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

Data to send over USB IN endpoint are sent over CDC interface through this function.

Parameters

Buf	Buffer of data to be sent
Len	Number of data to be sent (in bytes)

Return values

Result	of the operation: USBD_OK if all operations are OK else USBD_FAIL or USBD_BUSY

6.12 USBD_CONF

Configuration file for Usb otg low level driver.

Modules

- USBD_CONF_Exported_Variables

  Public variables.
- USBD_CONF_Exported_Defines

  Defines for configuration of the Usb device.
- USBD_CONF_Exported_Macros

  Aliases.
- USBD_CONF_Exported_Types

  Types.
- USBD_CONF_Exported_FunctionsPrototype

  Declaration of public functions for Usb device.

6.12.1 Detailed Description

Configuration file for Usb otg low level driver.

6.13 USBD_CONF_Exported_Variables

Public variables.

Public variables.

6.14 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

6.14.1 Detailed Description

Defines for configuration of the Usb device.

6.15 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**(...)

6.15.1 Detailed Description

Aliases.

6.15.2 Macro Definition Documentation

6.15.2.1 USBD_DbgLog

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

Value:

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

6.15.2.2 USBD_Delay

```
#define USBD_Delay HAL_Delay
```

Alias for delay.

6.15.2.3 USBD_ErrLog

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

Value:

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

6.15.2.4 USBD_free

```
#define USBD_free free
```

Alias for memory release.

6.15.2.5 USBD_malloc

```
#define USBD_malloc malloc
```

Alias for memory allocation.

6.15.2.6 USBD_memcpy

```
#define USBD_memcpy memcpy
```

Alias for memory copy.

6.15.2.7 USBD_memset

```
#define USBD_memset memset
```

Alias for memory set.

6.15.2.8 USBD_UsrLog

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

Value:

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

6.16 USBD_CONF_Exported_Types

Types.

Types.

6.17 USBD_CONF_Exported_FunctionsPrototype

Declaration of public functions for Usb device.

Declaration of public functions for Usb device.

6.18 USBD_DESC_Private_TypesDefinitions

Private types.

Private types.

6.19 USBD_DESC_Private_Defines

Private defines.

Macros

- #define **USBD_VID** 1155
- #define **USBD_LANGID_STRING** 1033
- #define **USBD_MANUFACTURER_STRING** "Electrosmith"
- #define **USBD_PID_HS** 22336
- #define **USBD_PRODUCT_STRING_HS** "Daisy Seed External"
- #define **USBD_CONFIGURATION_STRING_HS** "CDC Config"
- #define **USBD_INTERFACE_STRING_HS** "CDC Interface"
- #define **USBD_PID_FS** 22336
- #define **USBD_PRODUCT_STRING_FS** "Daisy Seed Built In"
- #define **USBD_CONFIGURATION_STRING_FS** "CDC Config"
- #define **USBD_INTERFACE_STRING_FS** "CDC Interface"

6.19.1 Detailed Description

Private defines.

6.20 USBD_DESC_Private_Macros

Private macros.

Private macros.

6.21 USBD_DESC_Private_FunctionPrototypes

Private functions declaration.

6.21.1 Detailed Description

Private functions declaration.

Private functions declaration for HS.

Private functions declaration for FS.

6.22 USBD_DESC_Private_Variables

Private variables.

Variables

- USBD_DescriptorsTypeDef FS_Desc
- __ALIGN_BEGIN uint8_t USBD_FS_DeviceDesc [USB_LEN_DEV_DESC] __ALIGN_END
- USBD_DescriptorsTypeDef HS_Desc

6.22.1 Detailed Description

Private variables.

6.22.2 Variable Documentation

6.22.2.1 __ALIGN_END

```
__ALIGN_BEGIN uint8_t USBD_StringSerial [ 0x1A ] __ALIGN_END
```

Initial value:

```
=
{
  0x12,
  USB_DESC_TYPE_DEVICE,
  0x00,
  0x02,
  0x02,
  0x02,
  0x00,
  USB_MAX_EP0_SIZE,
  LOBYTE( 1155 ),
  HIBYTE( 1155 ),
  LOBYTE( 22336 ),
  HIBYTE( 22336 ),
  0x00,
  0x02,
  USBD_IDX_MFC_STR,
  USBD_IDX_PRODUCT_STR,
  USBD_IDX_SERIAL_STR,
   1U
}
```

USB standard device descriptor.

USB lang indentifier descriptor.

IAR Compiler

6.22.2.2 FS_Desc

```
USBD_DescriptorsTypeDef FS_Desc
```

Initial value:

```
=
{
  USBD_FS_DeviceDescriptor
, USBD_FS_LangIDStrDescriptor
, USBD_FS_ManufacturerStrDescriptor
, USBD_FS_ProductStrDescriptor
, USBD_FS_SerialStrDescriptor
, USBD_FS_ConfigStrDescriptor
, USBD_FS_InterfaceStrDescriptor
}
```

Descriptor for the Usb device.

6.22.2.3 HS_Desc

```
USBD_DescriptorsTypeDef HS_Desc
```

Initial value:

```
=
{
  USBD_HS_DeviceDescriptor
, USBD_HS_LangIDStrDescriptor
, USBD_HS_ManufacturerStrDescriptor
, USBD_HS_ProductStrDescriptor
, USBD_HS_SerialStrDescriptor
, USBD_HS_ConfigStrDescriptor
, USBD_HS_InterfaceStrDescriptor
}
```

Descriptor for the Usb device.

6.23 USBD_DESC_Private_Functions

Private functions.

Functions

- uint8_t ∗ USBD_HS_DeviceDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)
  Return the device descriptor.
- uint8_t ∗ USBD_HS_LangIDStrDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)
  Return the LangID string descriptor.
- uint8_t ∗ USBD_HS_ProductStrDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)
  Return the product string descriptor.
- uint8_t ∗ USBD_HS_ManufacturerStrDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)
  Return the manufacturer string descriptor.
- uint8_t ∗ USBD_HS_SerialStrDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)
  Return the serial number string descriptor.
- uint8_t ∗ USBD_HS_ConfigStrDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)
  Return the configuration string descriptor.
- uint8_t ∗ USBD_HS_InterfaceStrDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)
  Return the interface string descriptor.
- uint8_t ∗ USBD_FS_DeviceDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)
  Return the device descriptor.
- uint8_t ∗ USBD_FS_LangIDStrDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)
  Return the LangID string descriptor.
- uint8_t ∗ USBD_FS_ProductStrDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)
  Return the product string descriptor.
- uint8_t ∗ USBD_FS_ManufacturerStrDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)
  Return the manufacturer string descriptor.
- uint8_t ∗ USBD_FS_SerialStrDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)
  Return the serial number string descriptor.
- uint8_t ∗ USBD_FS_ConfigStrDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)
  Return the configuration string descriptor.
- uint8_t ∗ USBD_FS_InterfaceStrDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)
  Return the interface string descriptor.

6.23.1 Detailed Description

Private functions.

6.23.2 Function Documentation

6.23.2.1 USBD_FS_ConfigStrDescriptor()

```
uint8_t ∗ USBD_FS_ConfigStrDescriptor (
            USBD_SpeedTypeDef speed,
            uint16_t ∗ length )
```

Return the configuration string descriptor.

Parameters

speed	: Current device speed
length	: Pointer to data length variable

Return values

Pointer	to descriptor buffer

6.23.2.2 USBD_FS_DeviceDescriptor()

```
uint8_t * USBD_FS_DeviceDescriptor (
            USBD_SpeedTypeDef speed,
            uint16_t * length )
```

Return the device descriptor.

Parameters

speed	: Current device speed
length	: Pointer to data length variable

Return values

Pointer	to descriptor buffer

6.23.2.3 USBD_FS_InterfaceStrDescriptor()

```
uint8_t * USBD_FS_InterfaceStrDescriptor (
            USBD_SpeedTypeDef speed,
            uint16_t * length )
```

Return the interface string descriptor.

Parameters

speed	: Current device speed
length	: Pointer to data length variable

Return values

Pointer	to descriptor buffer

6.23.2.4 USBD_FS_LangIDStrDescriptor()

```
uint8_t * USBD_FS_LangIDStrDescriptor (
            USBD_SpeedTypeDef speed,
            uint16_t * length )
```

Return the LangID string descriptor.

Parameters

speed	: Current device speed
length	: Pointer to data length variable

Return values

Pointer	to descriptor buffer

6.23.2.5 USBD_FS_ManufacturerStrDescriptor()

```
uint8_t * USBD_FS_ManufacturerStrDescriptor (
            USBD_SpeedTypeDef speed,
            uint16_t * length )
```

Return the manufacturer string descriptor.

Parameters

speed	: Current device speed
length	: Pointer to data length variable

Return values

Pointer	to descriptor buffer

6.23.2.6 USBD_FS_ProductStrDescriptor()

```
uint8_t * USBD_FS_ProductStrDescriptor (
            USBD_SpeedTypeDef speed,
            uint16_t * length )
```

Return the product string descriptor.

Parameters

speed	: Current device speed
length	: Pointer to data length variable

Return values

Pointer	to descriptor buffer

6.23.2.7 USBD_FS_SerialStrDescriptor()

```
uint8_t * USBD_FS_SerialStrDescriptor (
             USBD_SpeedTypeDef speed,
             uint16_t * length )
```

Return the serial number string descriptor.

Parameters

speed	: Current device speed
length	: Pointer to data length variable

Return values

Pointer	to descriptor buffer

6.23.2.8 USBD_HS_ConfigStrDescriptor()

```
uint8_t * USBD_HS_ConfigStrDescriptor (
             USBD_SpeedTypeDef speed,
             uint16_t * length )
```

Return the configuration string descriptor.

Parameters

speed	: Current device speed
length	: Pointer to data length variable

Return values

Pointer	to descriptor buffer

6.23.2.9 USBD_HS_DeviceDescriptor()

```
uint8_t * USBD_HS_DeviceDescriptor (
            USBD_SpeedTypeDef speed,
            uint16_t * length )
```

Return the device descriptor.

Parameters

speed	: Current device speed
length	: Pointer to data length variable

Return values

Pointer	to descriptor buffer

6.23.2.10 USBD_HS_InterfaceStrDescriptor()

```
uint8_t * USBD_HS_InterfaceStrDescriptor (
            USBD_SpeedTypeDef speed,
            uint16_t * length )
```

Return the interface string descriptor.

Parameters

speed	: Current device speed
length	: Pointer to data length variable

Return values

Pointer	to descriptor buffer

6.23.2.11 USBD_HS_LangIDStrDescriptor()

```
uint8_t * USBD_HS_LangIDStrDescriptor (
            USBD_SpeedTypeDef speed,
            uint16_t * length )
```

Return the LangID string descriptor.

Parameters

speed	: Current device speed
length	: Pointer to data length variable

Return values

Pointer	to descriptor buffer

6.23.2.12 USBD_HS_ManufacturerStrDescriptor()

```
uint8_t * USBD_HS_ManufacturerStrDescriptor (
             USBD_SpeedTypeDef speed,
             uint16_t * length )
```

Return the manufacturer string descriptor.

Parameters

speed	: Current device speed
length	: Pointer to data length variable

Return values

Pointer	to descriptor buffer

6.23.2.13 USBD_HS_ProductStrDescriptor()

```
uint8_t * USBD_HS_ProductStrDescriptor (
             USBD_SpeedTypeDef speed,
             uint16_t * length )
```

Return the product string descriptor.

Parameters

speed	: current device speed
length	: pointer to data length variable

Return values

pointer	to descriptor buffer

6.23.2.14 USBD_HS_SerialStrDescriptor()

```
uint8_t * USBD_HS_SerialStrDescriptor (
            USBD_SpeedTypeDef speed,
            uint16_t * length )
```

Return the serial number string descriptor.

Parameters

speed	: Current device speed
length	: Pointer to data length variable

Return values

Pointer	to descriptor buffer

6.24 USBD_DESC

Usb device descriptors module.

Modules

- USBD_DESC_Private_TypesDefinitions

  Private types.
- USBD_DESC_Private_Defines

  Private defines.
- USBD_DESC_Private_Macros

  Private macros.
- USBD_DESC_Private_FunctionPrototypes

  Private functions declaration.
- USBD_DESC_Private_Variables

  Private variables.
- USBD_DESC_Private_Functions

  Private functions.
- USBD_DESC_Exported_Constants

  Constants.
- USBD_DESC_Exported_Defines

  Defines.
- USBD_DESC_Exported_TypesDefinitions

  Types.
- USBD_DESC_Exported_Macros

  Aliases.
- USBD_DESC_Exported_Variables

  Public variables.
- USBD_DESC_Exported_FunctionsPrototype

  Public functions declaration.

6.24.1 Detailed Description

Usb device descriptors module.

6.25 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

6.25.1 Detailed Description

Constants.

6.26 USBD_DESC_Exported_Defines

Defines.

Defines.

6.27 USBD_DESC_Exported_TypesDefinitions

Types.

Types.

6.28 USBD_DESC_Exported_Macros

Aliases.

Aliases.

6.29 USBD_DESC_Exported_Variables

Public variables.

Variables

- USBD_DescriptorsTypeDef HS_Desc
- USBD_DescriptorsTypeDef FS_Desc

6.29.1 Detailed Description

Public variables.

6.29.2 Variable Documentation

6.29.2.1 FS_Desc

```
USBD_DescriptorsTypeDef FS_Desc
```

Descriptor for the Usb device.

6.29.2.2 HS_Desc

```
USBD_DescriptorsTypeDef HS_Desc
```

Descriptor for the Usb device.

6.30 USBD_DESC_Exported_FunctionsPrototype

Public functions declaration.

Public functions declaration.

6.31 CMSIS

Modules

- Stm32h7xx_system

6.31.1 Detailed Description

6.32 Stm32h7xx_system

Modules

- STM32H7xx_System_Private_Includes
- STM32H7xx_System_Private_TypesDefinitions
- STM32H7xx_System_Private_Defines
- STM32H7xx_System_Private_Macros
- STM32H7xx_System_Private_Variables
- STM32H7xx_System_Private_FunctionPrototypes
- STM32H7xx_System_Private_Functions

6.32.1 Detailed Description

6.33 STM32H7xx_System_Private_Includes

Macros

- #define HSE_VALUE ((uint32_t)25000000)
- #define CSI_VALUE ((uint32_t)4000000)
- #define HSI_VALUE ((uint32_t)64000000)

6.33.1 Detailed Description

6.33.2 Macro Definition Documentation

6.33.2.1 CSI_VALUE

```
#define CSI_VALUE ((uint32_t)4000000)
```

Value of the Internal oscillator in Hz

6.33.2.2 HSE_VALUE

```
#define HSE_VALUE ((uint32_t)25000000)
```

Value of the External oscillator in Hz

6.33.2.3 HSI_VALUE

```
#define HSI_VALUE ((uint32_t)64000000)
```

Value of the Internal oscillator in Hz

6.34 STM32H7xx_System_Private_TypesDefinitions

6.35 STM32H7xx_System_Private_Defines

Macros

- #define VECT_TAB_OFFSET 0x00000000UL

6.35.1 Detailed Description

6.35.2 Macro Definition Documentation

6.35.2.1 VECT_TAB_OFFSET

```
#define VECT_TAB_OFFSET 0x00000000UL
```

< Uncomment the following line if you need to use initialized data in D2 domain SRAM (AHB SRAM)

< Uncomment the following line if you need to relocate your vector Table in Internal SRAM. Vector Table base offset field. This value must be a multiple of 0x200.

6.36 STM32H7xx_System_Private_Macros

6.37 STM32H7xx_System_Private_Variables

Variables

- uint32_t **SystemCoreClock** = 64000000
- uint32_t **SystemD2Clock** = 64000000
- const uint8_t **D1CorePrescTable** [16] = {0, 0, 0, 0, 1, 2, 3, 4, 1, 2, 3, 4, 6, 7, 8, 9}

6.37.1 Detailed Description

6.38 STM32H7xx_System_Private_FunctionPrototypes

6.39 STM32H7xx_System_Private_Functions

Functions

- void SystemInit (void)

  Setup the microcontroller system Initialize the FPU setting and vector table location configuration.
- void SystemCoreClockUpdate (void)

  Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock , it can be used by the user application to setup the SysTick timer or configure other parameters.

6.39.1 Detailed Description

6.39.2 Function Documentation

6.39.2.1 SystemCoreClockUpdate()

```
void SystemCoreClockUpdate (
            void   )
```

Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock , it can be used by the user application to setup the SysTick timer or configure other parameters.

Note

Each time the core clock changes, this function must be called to update SystemCoreClock variable value. Otherwise, any configuration based on this variable will be incorrect.
- The system frequency computed by this function is not the real frequency in the chip. It is calculated based on the predefined constant and the selected clock source:

- If SYSCLK source is CSI, SystemCoreClock will contain the CSI_VALUE(∗)
- If SYSCLK source is HSI, SystemCoreClock will contain the HSI_VALUE(∗∗)
- If SYSCLK source is HSE, SystemCoreClock will contain the HSE_VALUE(∗∗∗)
- If SYSCLK source is PLL, SystemCoreClock will contain the CSI_VALUE(∗), HSI_VALUE(∗∗) or HSE_VA←LUE(∗∗∗) multiplied/divided by the PLL factors.

(∗) CSI_VALUE is a constant defined in stm32h7xx_hal.h file (default value 4 MHz) but the real value may vary depending on the variations in voltage and temperature. (∗∗) HSI_VALUE is a constant defined in stm32h7xx_hal.h file (default value 64 MHz) but the real value may vary depending on the variations in voltage and temperature.

(∗∗∗)HSE_VALUE is a constant defined in stm32h7xx_hal.h file (default value 25 MHz), user has to ensure that HSE_VALUE is same as the real frequency of the crystal used. Otherwise, this function may have wrong result.

- The result of this function could be not correct when using fractional value for HSE crystal.

Parameters

None	

Return values

None	

< Value of the Internal oscillator in Hz

< Value of the Internal oscillator in Hz

< Value of the External oscillator in Hz

< Value of the Internal oscillator in Hz

< Value of the Internal oscillator in Hz

< Value of the External oscillator in Hz

< Value of the Internal oscillator in Hz

< Value of the Internal oscillator in Hz

6.39.2.2 SystemInit()

```
void SystemInit (
             void  )
```

Setup the microcontroller system Initialize the FPU setting and vector table location configuration.

Parameters

None	

Return values

None	

< Vector Table base offset field. This value must be a multiple of 0x200.

6.40 STM32_USB_OTG_DEVICE_LIBRARY

Usb device library.

Modules

- USBD_CDC_IF

  Usb VCP device module.
- USBD_DESC

  Usb device descriptors module.

6.40.1 Detailed Description

Usb device library.

For Usb device.

6.41 USBD_OTG_DRIVER

Modules

- USBD_CONF

  Configuration file for Usb otg low level driver.

6.41.1 Detailed Description

Chapter 7

Namespace Documentation

7.1 daisy Namespace Reference

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
- class Led
- 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

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

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.

7.1.3 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_**
- dsy_gpio **mux_pin_** [MUX_SEL_LAST]
- uint8_t **mux_channels_**

8.1.1 Detailed Description

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

8.1.2 Member Function Documentation

8.1.2.1 InitMux()

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

Initializes a single ADC pin as a Multiplexed ADC.Requires a CD4051 Multiplexor connected to the pinInternal Callbacks handle the pin addressing.channels must be 1-8

8.1.2.2 InitSingle()

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

Initializes a single ADC pin as an ADC.

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

- src/per_adc.h
- src/per_adc.cpp

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 AdcHandle::Get (
             uint8_t chn )
```

These are getters for a single channel

8.2.1.2 GetMux()

```
uint16_t 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 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 num_↩
channels: number of ADC channels to initialize ovs: Oversampling amount - Defaults to OVS_32

8.2.1.4 Start()

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

Starts reading from the ADC

8.2.1.5 Stop()

```
void AdcHandle::Stop ( )
```

Stops reading from the ADC

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

- src/per_adc.h
- src/per_adc.cpp

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

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

8.3.1.2 InitBipolarCv()

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

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

8.3.1.3 Process()

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

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

8.3.1.4 Value()

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

Returns the current stored value, without reprocessing

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

- src/hid_ctrl.h
- src/hid_ctrl.cpp

8.4 codec_frame_t Struct Reference

Public Attributes

- short **l**
- short **r**

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

- src/dev_codec_wm8731_frame.h

8.5 color Struct Reference

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

Public Attributes

- uint16_t **red**
- uint16_t **green**
- uint16_t **blue**

8.5.1 Detailed Description

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

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

- src/dev_leddriver.h

8.6 daisy::Color Class Reference

Public Types

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

Public Member Functions

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

8.6.1 Member Enumeration Documentation

8.6.1.1 PresetColor

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

List of colors that have a preset RGB value

8.6.2 Member Function Documentation

8.6.2.1 Init() [1/2]

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

Initializes the Color with a given preset.

8.6.2.2 Init() [2/2]

```
void 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 files:

- src/util_color.h
- src/util_color.cpp

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 DaisyPatch::AudioSampleRate ( )
```

Hardware Accessors

8.9.2.2 Init()

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

Initializes the daisy seed, and patch hardware.

8.9.2.3 SetAudioBlockSize()

```
void 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 directlyHelper functions above provide easier access to much of what they are capable of.

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

- src/daisy_patch.h
- src/daisy_patch.cpp

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 files:

- src/daisy_petal.h
- src/daisy_petal.cpp

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 DaisyPod::AudioSampleRate ( )
```

Hardware Accessors

8.11.1.2 Init()

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

Functions Init related stuff.

8.11.1.3 SetAudioBlockSize()

```
void 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 files:

- src/daisy_pod.h
- src/daisy_pod.cpp

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 DaisySeed::AudioSampleRate ( )
```

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

8.12.1.2 Configure()

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

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

8.12.1.6 SetLed()

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

Sets the state of the built in LED

8.12.1.7 SetTestPoint()

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

Sets the state of the test point near pin 10

8.12.1.8 StartAudio()

```
void 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 files:

- src/daisy_seed.h
- src/daisy_seed.cpp

8.13 dsy_adc Struct Reference

Public Attributes

- AdcChannelConfig **pin_cfg** [14]
- uint8_t **channels**
- uint8_t **mux_channels** [14]
- uint16_t **mux_index** [14]
- uint16_t ∗ **dma_buffer**
- uint16_t(∗ **mux_cache**)[14][8]
- ADC_HandleTypeDef **hadc1**
- DMA_HandleTypeDef **hdma_adc1**

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

- src/per_adc.cpp

8.14 dsy_audio Struct Reference

Public Attributes

- dsy_audio_callback **callback**
- dsy_audio_mc_callback **mc_callback**
- int32_t ∗ **dma_buffer_rx**
- int32_t ∗ **dma_buffer_tx**
- float **in** [128 ∗2]
- float **out** [128 ∗2]
- size_t **block_size**
- size_t **offset**
- size_t **dma_size**
- uint8_t **bitdepth**
- uint8_t **device**
- uint8_t **channels**
- dsy_i2c_handle ∗ **device_control_hi2c**
- dsy_audio_handle ∗ **config_handle**

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

- src/hid_audio.c

8.15 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**

8.15.1 Detailed Description

Simple config struct that holds peripheral drivers.

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

- src/hid_audio.h

8.16 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]

8.16.1 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

8.17 dsy_dac_t Struct Reference

Public Attributes

- DAC_HandleTypeDef **hdac1**
- dsy_dac_handle ∗ **dsy_hdac**
- uint8_t **initialized**
- dsy_dac_channel **active_channels**

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

- src/per_dac.c

8.18 dsy_gpio Struct Reference

```
#include <per_gpio.h>
```

Public Attributes

- dsy_gpio_pin **pin**
- dsy_gpio_mode **mode**
- dsy_gpio_pull **pull**

8.18.1 Detailed Description

Struct for holding the pin, and configuration

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

- src/per_gpio.h

8.19 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

8.20 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**

8.20.1 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

8.21 dsy_led_driver_t Struct Reference

Public Attributes

- led **leds** [16 ∗8]
- uint16_t ∗ **sorted_bright** [8][16]
- uint16_t **dummy_bright**
- float **master_dim**
- uint8_t **temp_buff** [((16 ∗4)+1)]
- uint8_t **current_drv**
- color **standard_colors** [LED_COLOR_LAST]
- uint8_t **num_drivers**
- uint8_t **driver_addr** [8]
- I2C_HandleTypeDef ∗ **i2c**
- dsy_i2c_handle ∗ **dsy_i2c**

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

- src/dev_leddriver.c

8.22 dsy_qspi Struct Reference

Public Attributes

- QSPI_HandleTypeDef **hqspi**
- uint8_t **board**
- dsy_qspi_handle ∗ **dsy_hqspi**

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

- src/per_qspi.c

8.23 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]

8.23.1 Detailed Description

Configuration structure for interfacing with QSPI Driver.

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

- src/per_qspi.h

8.24 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]

8.24.1 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

8.25 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

8.25.1 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.

8.25.2 Member Data Documentation

8.25.2.1 BlockNbr

`uint32_t DSY_SD_CardInfoTypeDef::BlockNbr`

Specifies the Card Capacity in blocks

8.25.2.2 BlockSize

`uint32_t DSY_SD_CardInfoTypeDef::BlockSize`

Specifies one block size in bytes

8.25.2.3 CardSpeed

`uint32_t DSY_SD_CardInfoTypeDef::CardSpeed`

Specifies the card Speed

8.25.2.4 CardType

`uint32_t DSY_SD_CardInfoTypeDef::CardType`

Specifies the card Type

8.25.2.5 CardVersion

`uint32_t DSY_SD_CardInfoTypeDef::CardVersion`

Specifies the card version

8.25.2.6 Class

`uint32_t DSY_SD_CardInfoTypeDef::Class`

Specifies the class of the card class

8.25.2.7 LogBlockNbr

`uint32_t DSY_SD_CardInfoTypeDef::LogBlockNbr`

Specifies the Card logical Capacity in blocks

8.25.2.8 LogBlockSize

`uint32_t DSY_SD_CardInfoTypeDef::LogBlockSize`

Specifies logical block size in bytes

8.25.2.9 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

8.26 dsy_sdram_t Struct Reference

Public Attributes

- uint8_t **board**
- SDRAM_HandleTypeDef **hsdram**
- dsy_sdram_handle ∗ **dsy_hsdram**

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

- src/dev_sdram.c

8.27 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]

8.27.1 Detailed Description

configuration strucutre for 4021

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

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

- src/dev_sr_4021.h

8.28 dsy_tim Struct Reference

Public Attributes

- uint32_t **scale** [SCALE_LAST]
- TIM_HandleTypeDef **htim2**

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

- src/per_tim.c

8.29 dsy_wm8731_handle_t Struct Reference

Public Attributes

- uint8_t **mcu_is_master**
- uint8_t **bitdepth**
- int32_t **sample_rate**
- size_t **block_size**
- size_t **stride**
- I2C_HandleTypeDef ∗ **i2c**

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

- src/dev_codec_wm8731.c

8.30 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

8.30.1 Member Function Documentation

8.30.1.1 Debounce()

```
void 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.

8.30.1.2 FallingEdge()

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

Returns true if the encoder was just released.

8.30.1.3 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.

8.30.1.4 Init()

```
void 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.

8.30.1.5 Pressed()

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

Returns true while the encoder is held down.

8.30.1.6 RisingEdge()

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

Returns true if the encoder was just pressed.

8.30.1.7 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 files:

- src/hid_encoder.h
- src/hid_encoder.cpp

8.31 FontDef Struct Reference

Public Attributes

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

8.31.1 Member Data Documentation

8.31.1.1 data

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

Pointer to data font data array

8.31.1.2 FontHeight

```
uint8_t FontDef::FontHeight
```

Font height in pixels

8.31.1.3 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

8.32 daisy::GateIn Class Reference

Public Member Functions

- void Init (dsy_gpio_pin ∗pin_cfg)
- bool Trig ()

8.32.1 Member Function Documentation

8.32.1.1 Init()

```
void GateIn::Init (
            dsy_gpio_pin ∗ pin_cfg )
```

Initializes the gate input with specified hardware pin

8.32.1.2 Trig()

```
bool GateIn::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 files:

- src/hid_gatein.h
- src/hid_gatein.cpp

8.33 led Struct Reference

Public Attributes

- uint16_t **bright**
- uint16_t **addr**
- uint16_t **drv**

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

- src/dev_leddriver.c

8.34 daisy::Led Class Reference

Public Member Functions

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

8.34.1 Member Function Documentation

8.34.1.1 Init()

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

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

8.34.1.2 Set()

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

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

8.34.1.3 Update()

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

This processes the pwm of the LED sets the hardware accordingly.

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

- src/hid_led.h
- src/hid_led.cpp

8.35 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]

8.35.1 Detailed Description

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

8.35.2 Member Function Documentation

8.35.2.1 AsControlChange()

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

Returns the data within the MidiEvent as a NoteOnEvent struct.

8.35.2.2 AsNoteOn()

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

Returns the data within the MidiEvent as a NoteOnEvent struct.

8.35.3 Member Data Documentation

8.35.3.1 type

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

Newer ish.

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

- src/hid_midi.h

8.36 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 ()

8.36.1 Member Enumeration Documentation

8.36.1.1 MidiInputMode

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

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

8.36.2 Member Function Documentation

8.36.2.1 HasEvents()

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

Checks if there are unhandled messages in the queue

8.36.2.2 Init()

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

Initializes the MidiHandler

8.36.2.3 Parse()

```
void 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());

8.36.2.4 PopEvent()

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

Pops the oldest unhandled MidiEvent from the internal queue

8.36.2.5 StartReceive()

```
void 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 files:

- src/hid_midi.h
- src/hid_midi.cpp

8.37 daisy::NoteOnEvent Struct Reference

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

Public Attributes

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

8.37.1 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

8.38 NVIC_TypeDef Struct Reference

Public Attributes

- volatile uint32_t **ISER** [2]
- uint32_t **RESERVED0** [30]
- volatile uint32_t **ICER** [2]
- uint32_t **RSERVED1** [30]
- volatile uint32_t **ISPR** [2]
- uint32_t **RESERVED2** [30]
- volatile uint32_t **ICPR** [2]
- uint32_t **RESERVED3** [30]
- volatile uint32_t **IABR** [2]
- uint32_t **RESERVED4** [62]
- volatile uint32_t **IPR** [15]

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

- src/sys_system.c

8.39 daisy::OledDisplay Class Reference

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 ()

8.39.1 Member Enumeration Documentation

8.39.1.1 Pins

`enum daisy::OledDisplay::Pins`

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

8.39.2 Member Function Documentation

8.39.2.1 DrawPixel()

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

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

8.39.2.2 Fill()

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

Fills the entire display with either on/off.

8.39.2.3 Init()

```
void 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 Takes an argument for the pin cfg pin_cfg should be a pointer to an array of OledDisplay::NUM_PINS dsy_gpio_pins

8.39.2.4 SetCursor()

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

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

8.39.2.5 Update()

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

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

8.39.2.6 WriteChar()

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

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

8.39.2.7 WriteString()

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

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

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

- src/hid_oled_display.h
- src/hid_oled_display.cpp

8.40 daisy::Parameter Class Reference

Public Types

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

Public Member Functions

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

8.40.1 Member Enumeration Documentation

8.40.1.1 Curve

```
enum daisy::Parameter::Curve
```

Curves are applied to the output signal

8.40.2 Member Function Documentation

8.40.2.1 Init()

```
void 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.

8.40.2.2 Process()

```
float 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.

8.40.2.3 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 files:

- src/hid_parameter.h
- src/hid_parameter.cpp

8.41 rgb_led Struct Reference

Public Attributes

- color **c**
- uint16_t **addr_r**
- uint16_t **addr_g**
- uint16_t **addr_b**
- uint16_t **drv_r**
- uint16_t **drv_g**
- uint16_t **drv_b**

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

- src/dev_leddriver.c

8.42 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 ()

8.42.1 Member Function Documentation

8.42.1.1 Init()

```
void 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.

8.42.1.2 Set()

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

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

8.42.1.3 SetColor()

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

Sets the RGB using a Color object.

8.42.1.4 Update()

```
void RgbLed::Update (
             void )
```

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 files:

- src/hid_rgb_led.h
- src/hid_rgb_led.cpp

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

Public Member Functions

- void Init ()
- size_t capacity () const
- size_t writable () const
- size_t readable () const
- void Write (T v)
- void Overwrite (T v)
- T Read ()
- T ImmediateRead ()
- void Flush ()
- void Swallow (size_t n)
- void ImmediateRead (T ∗destination, size_t num_elements)
- void Overwrite (const T ∗source, size_t num_elements)

8.43.1 Member Function Documentation

8.43.1.1 capacity()

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

Returns the total size of the ring buffer

8.43.1.2 Flush()

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

Flushes unread elements from the ring buffer

8.43.1.3 ImmediateRead() [1/2]

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

Reads next element from ring buffer immediately

8.43.1.4 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

8.43.1.5 Init()

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

Initializes the Ring Buffer

8.43.1.6 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.

8.43.1.7 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.

8.43.1.8 Read()

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

Reads the first available element from the ring buffer

8.43.1.9 readable()

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

Returns number of unread elements in ring buffer

8.43.1.10 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.

8.43.1.11 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.

8.43.1.12 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

8.44 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

8.45 daisy::SdmmcHandler Class Reference

Public Member Functions

- void Init ()

8.45.1 Member Function Documentation

8.45.1.1 Init()

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

Initializes the SD Card InterfaceFor now all settings are fixed (See todo at top of section)

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

- src/per_sdmmc.h
- src/per_sdmmc.cpp

8.46 daisy::SdmmcHandlerInit Struct Reference

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

Public Attributes

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

8.46.1 Detailed Description

Structure for setting the options above.

Used to intiailize SdmmcHandler

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

- src/per_sdmmc.h

8.47 ShiftRegister595 Class Reference

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 ()

8.47.1 Member Enumeration Documentation

8.47.1.1 Pins

```
enum ShiftRegister595::Pins
```

The following pins correspond to the hardware connections to the 595.

LATCH corresonds to Pin 12 "RCLK" CLK corresponds to Pin 11 "SRCLK" DATA corresponds to Pin 14 "SER" *SRCLR* is not added here, but is tied to 3v3 on test hardware.

8.47.2 Member Function Documentation

8.47.2.1 Init()

```
void ShiftRegister595::Init (
            dsy_gpio_pin * pin_cfg,
            size_t num_daisy_chained = 1 )
```

Initializes the GPIO, and data for the ShiftRegister

Arguments:

∗pin_cfg is an array of dsy_gpio_pin corresponding the the Pins enum above. num_daisy_chained (default = 1) is the number of 595 devices daisy chained together.

8.47.2.2 Set()

```
void ShiftRegister595::Set (
            uint8_t idx,
            bool state )
```

Sets the state of the specified output. The index starts with QA on the first device and ends with QH on the last device.

a true state will set the output HIGH, while a false state will set the output LOW.

8.47.2.3 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 files:

- src/dev_sr_595.h
- src/dev_sr_595.cpp

8.48 daisy::SpiHandle Class Reference

Public Member Functions

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

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

- src/per_spi.h
- src/per_spi.cpp

8.49 SSD1309_t Struct Reference

Public Attributes

- uint16_t **CurrentX**
- uint16_t **CurrentY**
- uint8_t **Inverted**
- uint8_t **Initialized**

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

- src/hid_oled_display.cpp

8.50 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

8.50.1 Member Enumeration Documentation

8.50.1.1 Polarity

```
enum daisy::Switch::Polarity
```

Specifies whether the pressed is HIGH or LOW.

8.50.1.2 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.

8.50.1.3 Type

```
enum daisy::Switch::Type
```

Specifies the expected behavior of the switch

8.50.2 Member Function Documentation

8.50.2.1 Debounce()

```
void 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.

8.50.2.2 FallingEdge()

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

Returns true if the button was just released

8.50.2.3 Init()

```
void 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

8.50.2.4 Pressed()

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

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

8.50.2.5 RisingEdge()

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

Returns true if a button was just pressed.

8.50.2.6 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 files:

- src/hid_switch.h
- src/hid_switch.cpp

8.51 uart_handle Struct Reference

Public Attributes

- UART_HandleTypeDef **huart1**
- DMA_HandleTypeDef **hdma_usart1_rx**
- uint8_t ∗ **dma_buffer_rx**
- bool **receiving**
- size_t **rx_size**
- RingBuffer< uint8_t, 64 > **queue_rx**
- bool **rx_active**
- bool **tx_active**

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

- src/per_uart.cpp

8.52 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 ()

8.52.1 Member Function Documentation

8.52.1.1 CheckError()

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

Returns the result of HAL_UART_GetError() to the user.

8.52.1.2 FlushRx()

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

Flushes the Receive Queue

8.52.1.3 Init()

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

Initializes the UART Peripheral

8.52.1.4 PollReceive()

```
int UartHandler::PollReceive (
             uint8_t * buff,
             size_t size,
             uint32_t timeout )
```

Reads the amount of bytes in blocking mode with a 10ms timeout.

8.52.1.5 PollTx()

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

Sends an amount of data in blocking mode.

8.52.1.6 PopRx()

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

Pops the oldest byte from the FIFO.

8.52.1.7 Readable()

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

Checks if there are any unread bytes in the FIFO

8.52.1.8 RxActive()

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

Returns whether Rx DMA is listening or not.

8.52.1.9 StartRx()

```
int 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 files:

- src/per_uart.h
- src/per_uart.cpp

8.53 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)

8.53.1 Member Typedef Documentation

8.53.1.1 ReceiveCallback

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

Function called upon reception of a buffer

8.53.2 Member Enumeration Documentation

8.53.2.1 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)

8.53.3 Member Function Documentation

8.53.3.1 Init()

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

Initializes the specified peripheral(s) as USB CDC Devices

8.53.3.2 SetReceiveCallback()

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

sets the callback to be called upon reception of new data

8.53.3.3 TransmitExternal()

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

Transmits a buffer of 'size' bytes from a USB port connected to the external USB Pins of the daisy seed.

8.53.3.4 TransmitInternal()

```
void 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 files:

- src/hid_usb.h
- src/hid_usb.cpp

8.54 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

8.55 daisy::WavFileInfo Struct Reference

```
#include <hid_wavplayer.h>
```

Public Attributes

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

8.55.1 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

8.56 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

8.56.1 Detailed Description

Class for handling playback of WAV files.

TODO:

- Make template-y to reduce memory usage.

8.56.2 Member Function Documentation

8.56.2.1 Close()

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

Closes whatever file is currently open.

8.56.2.2 GetCurrentFile()

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

Returns currently selected file.

8.56.2.3 GetLooping()

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

Returns whether the WavPlayer is looping or not.

8.56.2.4 GetNumberFiles()

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

Returns the number of files loaded by the WavPlayer

8.56.2.5 Init()

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

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

8.56.2.6 Open()

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

Opens the file at index sel for reading.

8.56.2.7 Prepare()

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

Collects buffer for playback when needed.

8.56.2.8 Restart()

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

Resets the playback position to the beginning of the file immediately

8.56.2.9 SetLooping()

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

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

8.56.2.10 Stream()

```
int16_t WavPlayer::Stream ( )
```

Returns the next sample if playing, otherwise returns 0

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

- src/hid_wavplayer.h
- src/hid_wavplayer.cpp

Chapter 9

File Documentation

9.1 src/system_stm32h7xx.c File Reference

CMSIS Cortex-Mx Device Peripheral Access Layer System Source File.

```
#include "stm32h7xx.h"
#include <math.h>
```

Macros

- #define HSE_VALUE ((uint32_t)25000000)
- #define CSI_VALUE ((uint32_t)4000000)
- #define HSI_VALUE ((uint32_t)64000000)
- #define VECT_TAB_OFFSET 0x00000000UL

Functions

- void SystemInit (void)

  Setup the microcontroller system Initialize the FPU setting and vector table location configuration.
- void SystemCoreClockUpdate (void)

  Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock , it can be used by the user application to setup the SysTick timer or configure other parameters.

Variables

- uint32_t **SystemCoreClock** = 64000000
- uint32_t **SystemD2Clock** = 64000000
- const uint8_t **D1CorePrescTable** [16] = {0, 0, 0, 0, 1, 2, 3, 4, 1, 2, 3, 4, 6, 7, 8, 9}

9.1.1 Detailed Description

CMSIS Cortex-Mx Device Peripheral Access Layer System Source File.

Author

MCD Application Team This file provides two functions and one global variable to be called from user application:

- SystemInit(): This function is called at startup just after reset and before branch to main program. This call is made inside the "startup_stm32h7xx.s" file.
- SystemCoreClock variable: Contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.
- SystemCoreClockUpdate(): Updates the variable SystemCoreClock and must be called whenever the core clock is changed during program execution.

Attention

9.2 src/usbd_cdc_if.c File Reference

: Usb device for Virtual Com Port.

```
#include "usbd_cdc_if.h"
```

Macros

- #define **APP_RX_DATA_SIZE** 2048
- #define **APP_TX_DATA_SIZE** 2048

Functions

- void **dummy_rx_callback** (uint8_t ∗buf, uint32_t ∗len)
- uint8_t CDC_Transmit_FS (uint8_t ∗Buf, uint16_t Len)

  CDC_Transmit_FS Data to send over USB IN endpoint are sent over CDC interface through this function.
- uint8_t CDC_Transmit_HS (uint8_t ∗Buf, uint16_t Len)

  Data to send over USB IN endpoint are sent over CDC interface through this function.
- void **CDC_Set_Rx_Callback_FS** (CDC_ReceiveCallback cb)

Variables

- uint8_t UserRxBufferFS [2048]
- uint8_t UserTxBufferFS [2048]
- uint8_t UserRxBufferHS [2048]
- uint8_t UserTxBufferHS [2048]
- CDC_ReceiveCallback **rx_callback_fs**
- USBD_HandleTypeDef **hUsbDeviceFS**
- USBD_HandleTypeDef **hUsbDeviceHS**
- USBD_CDC_ItfTypeDef USBD_Interface_fops_FS = {CDC_Init_FS, CDC_DeInit_FS, CDC_Control_FS, C←DC_Receive_FS}
- USBD_CDC_ItfTypeDef USBD_Interface_fops_HS = {CDC_Init_HS, CDC_DeInit_HS, CDC_Control_HS, CDC_Receive_HS}

9.2.1 Detailed Description

: Usb device for Virtual Com Port.

Version

: v1.0_Cube

Attention

9.3 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)

  CDC_Transmit_FS Data to send over USB IN endpoint are sent over CDC interface through this function.
- uint8_t CDC_Transmit_HS (uint8_t ∗Buf, uint16_t Len)

  Data to send over USB IN endpoint are sent over CDC interface through this function.

Variables

- USBD_CDC_ItfTypeDef USBD_Interface_fops_FS
- USBD_CDC_ItfTypeDef USBD_Interface_fops_HS

9.3.1 Detailed Description

: Header for usbd_cdc_if.c file.

Version

: v1.0_Cube

Attention

9.4 src/usbd_conf.c File Reference

: This file implements the board support package for the USB device library

```
#include "stm32h7xx.h"
#include "stm32h7xx_hal.h"
#include "usbd_def.h"
#include "usbd_core.h"
```

Functions

- void **Error_Handler** (void)
- USBD_StatusTypeDef USBD_Get_USB_Status (HAL_StatusTypeDef hal_status)

  Retuns the USB status depending on the HAL status:
- void HAL_PCD_MspInit (PCD_HandleTypeDef ∗pcdHandle)
- void HAL_PCD_MspDeInit (PCD_HandleTypeDef ∗pcdHandle)
- void HAL_PCD_SetupStageCallback (PCD_HandleTypeDef ∗hpcd)

  Setup stage callback.
- void HAL_PCD_DataOutStageCallback (PCD_HandleTypeDef ∗hpcd, uint8_t epnum)

  Data Out stage callback.
- void HAL_PCD_DataInStageCallback (PCD_HandleTypeDef ∗hpcd, uint8_t epnum)

  Data In stage callback.
- void HAL_PCD_SOFCallback (PCD_HandleTypeDef ∗hpcd)

  SOF callback.
- void HAL_PCD_ResetCallback (PCD_HandleTypeDef ∗hpcd)

  Reset callback.
- void HAL_PCD_SuspendCallback (PCD_HandleTypeDef ∗hpcd)

  Suspend callback. When Low power mode is enabled the debug cannot be used (IAR, Keil doesn't support it)
- void HAL_PCD_ResumeCallback (PCD_HandleTypeDef ∗hpcd)

  Resume callback. When Low power mode is enabled the debug cannot be used (IAR, Keil doesn't support it)
- void HAL_PCD_ISOOUTIncompleteCallback (PCD_HandleTypeDef ∗hpcd, uint8_t epnum)

  ISOOUTIncomplete callback.
- void HAL_PCD_ISOINIncompleteCallback (PCD_HandleTypeDef ∗hpcd, uint8_t epnum)

  ISOINIncomplete callback.
- void HAL_PCD_ConnectCallback (PCD_HandleTypeDef ∗hpcd)

  Connect callback.
- void HAL_PCD_DisconnectCallback (PCD_HandleTypeDef ∗hpcd)

  Disconnect callback.
- USBD_StatusTypeDef USBD_LL_Init (USBD_HandleTypeDef ∗pdev)

  Initializes the low level portion of the device driver.
- USBD_StatusTypeDef USBD_LL_DeInit (USBD_HandleTypeDef ∗pdev)

  De-Initializes the low level portion of the device driver.
- USBD_StatusTypeDef USBD_LL_Start (USBD_HandleTypeDef ∗pdev)

  Starts the low level portion of the device driver.
- USBD_StatusTypeDef USBD_LL_Stop (USBD_HandleTypeDef ∗pdev)

  Stops the low level portion of the device driver.
- USBD_StatusTypeDef USBD_LL_OpenEP (USBD_HandleTypeDef ∗pdev, uint8_t ep_addr, uint8_t ep_type, uint16_t ep_mps)

  Opens an endpoint of the low level driver.
- USBD_StatusTypeDef USBD_LL_CloseEP (USBD_HandleTypeDef ∗pdev, uint8_t ep_addr)

  Closes an endpoint of the low level driver.
- USBD_StatusTypeDef USBD_LL_FlushEP (USBD_HandleTypeDef ∗pdev, uint8_t ep_addr)

  Flushes an endpoint of the Low Level Driver.
- USBD_StatusTypeDef USBD_LL_StallEP (USBD_HandleTypeDef ∗pdev, uint8_t ep_addr)

  Sets a Stall condition on an endpoint of the Low Level Driver.
- USBD_StatusTypeDef USBD_LL_ClearStallEP (USBD_HandleTypeDef ∗pdev, uint8_t ep_addr)

  Clears a Stall condition on an endpoint of the Low Level Driver.
- uint8_t USBD_LL_IsStallEP (USBD_HandleTypeDef ∗pdev, uint8_t ep_addr)

  Returns Stall condition.
- USBD_StatusTypeDef USBD_LL_SetUSBAddress (USBD_HandleTypeDef ∗pdev, uint8_t dev_addr)

  Assigns a USB address to the device.
- USBD_StatusTypeDef USBD_LL_Transmit (USBD_HandleTypeDef ∗pdev, uint8_t ep_addr, uint8_t ∗pbuf, uint16_t size)

  Transmits data over an endpoint.
- USBD_StatusTypeDef USBD_LL_PrepareReceive (USBD_HandleTypeDef ∗pdev, uint8_t ep_addr, uint8_t ∗pbuf, uint16_t size)

  Prepares an endpoint for reception.
- uint32_t USBD_LL_GetRxDataSize (USBD_HandleTypeDef ∗pdev, uint8_t ep_addr)

  Returns the last transfered packet size.
- void USBD_LL_Delay (uint32_t Delay)

  Delays routine for the USB device library.

Variables

- PCD_HandleTypeDef **hpcd_USB_OTG_FS**
- PCD_HandleTypeDef **hpcd_USB_OTG_HS**

9.4.1 Detailed Description

: This file implements the board support package for the USB device library

Version

: v1.0_Cube

Attention

9.4.2 Function Documentation

9.4.2.1 HAL_PCD_ConnectCallback()

```
void HAL_PCD_ConnectCallback (
             PCD_HandleTypeDef * hpcd )
```

Connect callback.

Parameters

hpcd	PCD handle

Return values

None	

9.4.2.2 HAL_PCD_DataInStageCallback()

```
void HAL_PCD_DataInStageCallback (
             PCD_HandleTypeDef * hpcd,
             uint8_t epnum )
```

Data In stage callback.

Parameters

hpcd	PCD handle
epnum	Endpoint number

Return values

None	

9.4.2.3 HAL_PCD_DataOutStageCallback()

```
void HAL_PCD_DataOutStageCallback (
             PCD_HandleTypeDef * hpcd,
             uint8_t epnum )
```

Data Out stage callback.

Parameters

hpcd	PCD handle
epnum	Endpoint number

Return values

None	

9.4.2.4 HAL_PCD_DisconnectCallback()

```
void HAL_PCD_DisconnectCallback (
            PCD_HandleTypeDef * hpcd )
```

Disconnect callback.

Parameters

hpcd	PCD handle

Return values

None	

9.4.2.5 HAL_PCD_ISOINIncompleteCallback()

```
void HAL_PCD_ISOINIncompleteCallback (
            PCD_HandleTypeDef * hpcd,
            uint8_t epnum )
```

ISOINIncomplete callback.

Parameters

hpcd	PCD handle
epnum	Endpoint number

Return values

None	

9.4.2.6 HAL_PCD_ISOOUTIncompleteCallback()

```
void HAL_PCD_ISOOUTIncompleteCallback (
            PCD_HandleTypeDef * hpcd,
            uint8_t epnum )
```

ISOOUTIncomplete callback.

Parameters

hpcd	PCD handle
epnum	Endpoint number

Return values

None	

9.4.2.7 HAL_PCD_MspDeInit()

```
void HAL_PCD_MspDeInit (
            PCD_HandleTypeDef * pcdHandle )
```

USB_OTG_FS GPIO Configuration PA12 ——> USB_OTG_FS_DP PA11 ——> USB_OTG_FS_DM PA9 ——> USB_OTG_FS_VBUS

USB_OTG_HS GPIO Configuration PB14 -——> USB_OTG_HS_DM PB15 -——> USB_OTG_HS_DP

9.4.2.8 HAL_PCD_MspInit()

```
void HAL_PCD_MspInit (
            PCD_HandleTypeDef * pcdHandle )
```

USB_OTG_FS GPIO Configuration PA12 ——> USB_OTG_FS_DP PA11 ——> USB_OTG_FS_DM PA9 ——> USB_OTG_FS_VBUS

USB_OTG_HS GPIO Configuration PB14 -——> USB_OTG_HS_DM PB15 -——> USB_OTG_HS_DP

9.4.2.9 HAL_PCD_ResetCallback()

```
void HAL_PCD_ResetCallback (
            PCD_HandleTypeDef * hpcd )
```

Reset callback.

Parameters

hpcd	PCD handle

Return values

None	

9.4.2.10 HAL_PCD_ResumeCallback()

```
void HAL_PCD_ResumeCallback (
            PCD_HandleTypeDef * hpcd )
```

Resume callback. When Low power mode is enabled the debug cannot be used (IAR, Keil doesn't support it)

Parameters

hpcd	PCD handle

Return values

None	

9.4.2.11 HAL_PCD_SetupStageCallback()

```
void HAL_PCD_SetupStageCallback (
            PCD_HandleTypeDef * hpcd )
```

Setup stage callback.

Parameters

hpcd	PCD handle

Return values

None	

9.4.2.12 HAL_PCD_SOFCallback()

```
void HAL_PCD_SOFCallback (
            PCD_HandleTypeDef * hpcd )
```

SOF callback.

Parameters

hpcd	PCD handle

Return values

None	

9.4.2.13 HAL_PCD_SuspendCallback()

```
void HAL_PCD_SuspendCallback (
            PCD_HandleTypeDef * hpcd )
```

Suspend callback. When Low power mode is enabled the debug cannot be used (IAR, Keil doesn't support it)

Parameters

hpcd	PCD handle

Return values

None	

9.4.2.14 USBD_Get_USB_Status()

```
USBD_StatusTypeDef USBD_Get_USB_Status (
            HAL_StatusTypeDef hal_status )
```

Retuns the USB status depending on the HAL status:

Parameters

hal_status	HAL status

Return values

USB	status

9.4.2.15 USBD_LL_ClearStallEP()

```
USBD_StatusTypeDef USBD_LL_ClearStallEP (
            USBD_HandleTypeDef * pdev,
            uint8_t ep_addr )
```

Clears a Stall condition on an endpoint of the Low Level Driver.

Parameters

pdev	Device handle
ep_addr	Endpoint number

Return values

USBD	status

9.4.2.16 USBD_LL_CloseEP()

```
USBD_StatusTypeDef USBD_LL_CloseEP (
            USBD_HandleTypeDef * pdev,
            uint8_t ep_addr )
```

Closes an endpoint of the low level driver.

Parameters

pdev	Device handle
ep_addr	Endpoint number

Return values

USBD	status

9.4.2.17 USBD_LL_DeInit()

```
USBD_StatusTypeDef USBD_LL_DeInit (
            USBD_HandleTypeDef * pdev )
```

De-Initializes the low level portion of the device driver.

Parameters

pdev	Device handle

Return values

USBD	status

9.4.2.18 USBD_LL_Delay()

```
void USBD_LL_Delay (
            uint32_t Delay )
```

Delays routine for the USB device library.

Parameters

Delay	Delay in ms

Return values

None	

9.4.2.19 USBD_LL_FlushEP()

```
USBD_StatusTypeDef USBD_LL_FlushEP (
            USBD_HandleTypeDef * pdev,
            uint8_t ep_addr )
```

Flushes an endpoint of the Low Level Driver.

Parameters

pdev	Device handle
ep_addr	Endpoint number

Return values

USBD	status

9.4.2.20 USBD_LL_GetRxDataSize()

```
uint32_t USBD_LL_GetRxDataSize (
            USBD_HandleTypeDef * pdev,
            uint8_t ep_addr )
```

Returns the last transfered packet size.

Parameters

pdev	Device handle
ep_addr	Endpoint number

Return values

Recived	Data Size

9.4.2.21 USBD_LL_Init()

```
USBD_StatusTypeDef USBD_LL_Init (
            USBD_HandleTypeDef * pdev )
```

Initializes the low level portion of the device driver.

Parameters

pdev	Device handle

Return values

USBD	status

9.4.2.22 USBD_LL_IsStallEP()

```
uint8_t USBD_LL_IsStallEP (
            USBD_HandleTypeDef * pdev,
            uint8_t ep_addr )
```

Returns Stall condition.

Parameters

pdev	Device handle
ep_addr	Endpoint number

Return values

Stall	(1: Yes, 0: No)

9.4.2.23 USBD_LL_OpenEP()

```
USBD_StatusTypeDef USBD_LL_OpenEP (
            USBD_HandleTypeDef * pdev,
            uint8_t ep_addr,
            uint8_t ep_type,
            uint16_t ep_mps )
```

Opens an endpoint of the low level driver.

Parameters

pdev	Device handle
ep_addr	Endpoint number
ep_type	Endpoint type
ep_mps	Endpoint max packet size

Return values

USBD	status

9.4.2.24 USBD_LL_PrepareReceive()

```
USBD_StatusTypeDef USBD_LL_PrepareReceive (
            USBD_HandleTypeDef * pdev,
            uint8_t ep_addr,
            uint8_t * pbuf,
            uint16_t size )
```

Prepares an endpoint for reception.

Parameters

pdev	Device handle
ep_addr	Endpoint number
pbuf	Pointer to data to be received
size	Data size

Return values

USBD	status

9.4.2.25 USBD_LL_SetUSBAddress()

```
USBD_StatusTypeDef USBD_LL_SetUSBAddress (
            USBD_HandleTypeDef * pdev,
            uint8_t dev_addr )
```

Assigns a USB address to the device.

Parameters

pdev	Device handle
dev_addr	Device address

Return values

USBD	status

9.4.2.26 USBD_LL_StallEP()

```
USBD_StatusTypeDef USBD_LL_StallEP (
            USBD_HandleTypeDef * pdev,
            uint8_t ep_addr )
```

Sets a Stall condition on an endpoint of the Low Level Driver.

Parameters

pdev	Device handle
ep_addr	Endpoint number

Return values

USBD	status

9.4.2.27 USBD_LL_Start()

```
USBD_StatusTypeDef USBD_LL_Start (
            USBD_HandleTypeDef * pdev )
```

Starts the low level portion of the device driver.

Parameters

pdev	Device handle

Return values

USBD	status

9.4.2.28 USBD_LL_Stop()

```
USBD_StatusTypeDef USBD_LL_Stop (
            USBD_HandleTypeDef * pdev )
```

Stops the low level portion of the device driver.

Parameters

pdev	Device handle

Return values

USBD	status

9.4.2.29 USBD_LL_Transmit()

```
USBD_StatusTypeDef USBD_LL_Transmit (
            USBD_HandleTypeDef * pdev,
            uint8_t ep_addr,
            uint8_t * pbuf,
            uint16_t size )
```

Transmits data over an endpoint.

Parameters

pdev	Device handle
ep_addr	Endpoint number
pbuf	Pointer to data to be sent
size	Data size

Return values

USBD	status

9.5 src/usbd_conf.h File Reference

: Header for usbd_conf.c file.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "stm32h7xx.h"
#include "stm32h7xx_hal.h"
```

Macros

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

- #define **DEVICE_FS** 0
- #define **DEVICE_HS** 1
- #define USBD_malloc malloc
- #define USBD_free free
- #define USBD_memset memset
- #define USBD_memcpy memcpy
- #define USBD_Delay HAL_Delay
- #define **USBD_UsrLog**(...)
- #define **USBD_ErrLog**(...)
- #define **USBD_DbgLog**(...)

9.5.1 Detailed Description

: Header for usbd_conf.c file.

Version

: v1.0_Cube

Attention

9.6 src/usbd_desc.c File Reference

: This file implements the USB device descriptors.

```
#include "usbd_core.h"
#include "usbd_desc.h"
#include "usbd_conf.h"
```

Macros

- #define **USBD_VID** 1155
- #define **USBD_LANGID_STRING** 1033
- #define **USBD_MANUFACTURER_STRING** "Electrosmith"
- #define **USBD_PID_HS** 22336
- #define **USBD_PRODUCT_STRING_HS** "Daisy Seed External"
- #define **USBD_CONFIGURATION_STRING_HS** "CDC Config"
- #define **USBD_INTERFACE_STRING_HS** "CDC Interface"
- #define **USBD_PID_FS** 22336
- #define **USBD_PRODUCT_STRING_FS** "Daisy Seed Built In"
- #define **USBD_CONFIGURATION_STRING_FS** "CDC Config"
- #define **USBD_INTERFACE_STRING_FS** "CDC Interface"

Functions

- uint8_t ∗ USBD_FS_DeviceDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)

  Return the device descriptor.
- uint8_t ∗ USBD_FS_LangIDStrDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)

  Return the LangID string descriptor.
- uint8_t ∗ USBD_FS_ManufacturerStrDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)

  Return the manufacturer string descriptor.
- uint8_t ∗ USBD_FS_ProductStrDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)

  Return the product string descriptor.
- uint8_t ∗ USBD_FS_SerialStrDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)

  Return the serial number string descriptor.
- uint8_t ∗ USBD_FS_ConfigStrDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)

  Return the configuration string descriptor.
- uint8_t ∗ USBD_FS_InterfaceStrDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)

  Return the interface string descriptor.
- uint8_t ∗ USBD_HS_DeviceDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)

  Return the device descriptor.
- uint8_t ∗ USBD_HS_LangIDStrDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)

  Return the LangID string descriptor.
- uint8_t ∗ USBD_HS_ManufacturerStrDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)

  Return the manufacturer string descriptor.
- uint8_t ∗ USBD_HS_ProductStrDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)

  Return the product string descriptor.
- uint8_t ∗ USBD_HS_SerialStrDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)

  Return the serial number string descriptor.
- uint8_t ∗ USBD_HS_ConfigStrDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)

  Return the configuration string descriptor.
- uint8_t ∗ USBD_HS_InterfaceStrDescriptor (USBD_SpeedTypeDef speed, uint16_t ∗length)

  Return the interface string descriptor.

Variables

- USBD_DescriptorsTypeDef FS_Desc
- __ALIGN_BEGIN uint8_t USBD_FS_DeviceDesc [USB_LEN_DEV_DESC] __ALIGN_END
- USBD_DescriptorsTypeDef HS_Desc

9.6.1 Detailed Description

: This file implements the USB device descriptors.

: Header for usbd_conf.c file.

Version

: v1.0_Cube

Attention

Index

__ALIGN_END
    USBD_DESC_Private_Variables, 36

AsControlChange
    daisy::MidiEvent, 94
AsNoteOn
    daisy::MidiEvent, 94
AudioSampleRate
    daisy::DaisyPatch, 76
    daisy::DaisyPod, 79
    daisy::DaisySeed, 80

BlockNbr
    DSY_SD_CardInfoTypeDef, 87
BlockSize
    DSY_SD_CardInfoTypeDef, 87

CDC_Transmit_FS
    USBD_CDC_IF_Exported_FunctionsPrototype, 23
    USBD_CDC_IF, 18
CDC_Transmit_HS
    USBD_CDC_IF_Exported_FunctionsPrototype, 23
    USBD_CDC_IF, 18
CMSIS, 52
CSI_VALUE
    STM32H7xx_System_Private_Includes, 54
capacity
    daisy::RingBuffer, 102
CardSpeed
    DSY_SD_CardInfoTypeDef, 87
CardType
    DSY_SD_CardInfoTypeDef, 87
CardVersion
    DSY_SD_CardInfoTypeDef, 87
CheckError
    daisy::UartHandler, 110
Class
    DSY_SD_CardInfoTypeDef, 87
Close
    daisy::WavPlayer, 114
codec_frame_t, 73
color, 73
Configure
    daisy::DaisySeed, 80
Ctrl
    daisy::DaisyPatch, 76
Curve
    daisy::Parameter, 99

DSY_SD_CardInfoTypeDef, 86
    BlockNbr, 87
    BlockSize, 87
    CardSpeed, 87
    CardType, 87
    CardVersion, 87
    Class, 87
    LogBlockNbr, 87
    LogBlockSize, 88
    RelCardAdd, 88
daisy, 65
    daisy_field_init, 67
    MidiMessageType, 67
    SdmmcBitWidth, 67
    SdmmcMode, 67
    SdmmcSpeed, 67
daisy::AdcChannelConfig, 69
    InitMux, 69
    InitSingle, 70
daisy::AdcHandle, 70
    Get, 70
    GetMux, 71
    Init, 71
    Start, 71
    Stop, 71
daisy::AnalogControl, 71
    Init, 72
    InitBipolarCv, 72
    Process, 72
    Value, 72
daisy::Color, 73
    Init, 74
    PresetColor, 74
    Red, 74
daisy::ControlChangeEvent, 74
daisy::DaisyPatch, 75
    AudioSampleRate, 76
    Ctrl, 76
    gate_output, 76
    Init, 76
    seed, 77
    SetAudioBlockSize, 76
daisy::DaisyPetal, 77
daisy::DaisyPod, 78
    AudioSampleRate, 79
    Init, 79
    seed, 79
    SetAudioBlockSize, 79
daisy::DaisySeed, 79
    AudioSampleRate, 80

Configure, 80
GetPin, 80
Init, 80
sdram_handle, 81
SetAudioBlockSize, 80
SetLed, 81
SetTestPoint, 81
StartAudio, 81
daisy::Encoder, 89
Debounce, 90
FallingEdge, 90
Increment, 90
Init, 90
Pressed, 90
RisingEdge, 90
TimeHeldMs, 90
daisy::GateIn, 92
Init, 92
Trig, 92
daisy::Led, 93
Init, 93
Set, 93
Update, 93
daisy::MidiEvent, 93
AsControlChange, 94
AsNoteOn, 94
type, 94
daisy::MidiHandler, 95
HasEvents, 95
Init, 95
MidiInputMode, 95
Parse, 95
PopEvent, 96
StartReceive, 96
daisy::NoteOnEvent, 96
daisy::OledDisplay, 97
DrawPixel, 98
Fill, 98
Init, 98
Pins, 97
SetCursor, 98
Update, 98
WriteChar, 98
WriteString, 99
daisy::Parameter, 99
Curve, 99
Init, 100
Process, 100
Value, 100
daisy::RgbLed, 101
Init, 101
Set, 101
SetColor, 101
Update, 101
daisy::RingBuffer
capacity, 102
Flush, 102
ImmediateRead, 102
Init, 102
Overwrite, 103
Read, 103
readable, 103
Swallow, 103
writable, 103
Write, 104
daisy::RingBuffer$<$ T, 0 $>$, 104
daisy::RingBuffer$<$ T, size $>$, 102
daisy::SdmmcHandler, 104
Init, 105
daisy::SdmmcHandlerInit, 105
daisy::SpiHandle, 107
daisy::Switch, 107
Debounce, 108
FallingEdge, 108
Init, 108
Polarity, 107
Pressed, 108
Pull, 108
RisingEdge, 109
TimeHeldMs, 109
Type, 108
daisy::UartHandler, 109
CheckError, 110
FlushRx, 110
Init, 110
PollReceive, 110
PollTx, 110
PopRx, 110
Readable, 110
RxActive, 111
StartRx, 111
daisy::UsbHandle, 111
Init, 112
ReceiveCallback, 111
SetReceiveCallback, 112
TransmitExternal, 112
TransmitInternal, 112
UsbPeriph, 112
daisy::WavFileInfo, 113
daisy::WavPlayer, 114
Close, 114
GetCurrentFile, 114
GetLooping, 115
GetNumberFiles, 115
Init, 115
Open, 115
Prepare, 115
Restart, 115
SetLooping, 115
Stream, 115
daisy::daisy_field, 75
daisy_field_init
daisy, 67
data
FontDef, 91
Debounce

daisy::Encoder, 90
daisy::Switch, 108
DrawPixel
daisy::OledDisplay, 98
dsy_adc, 82
dsy_audio, 82
dsy_audio_handle, 82
dsy_dac_handle, 83
dsy_dac_t, 83
dsy_gpio, 84
dsy_gpio_pin, 84
dsy_i2c_handle, 84
dsy_led_driver_t, 85
dsy_qspi, 85
dsy_qspi_handle, 85
dsy_sai_handle, 86
dsy_sdram_t, 88
dsy_sr_4021_handle, 88
dsy_tim, 89
dsy_wm8731_handle_t, 89

FS_Desc
USBD_DESC_Exported_Variables, 50
USBD_DESC_Private_Variables, 36
FallingEdge
daisy::Encoder, 90
daisy::Switch, 108
Fill
daisy::OledDisplay, 98
Flush
daisy::RingBuffer, 102
FlushRx
daisy::UartHandler, 110
FontDef, 91
data, 91
FontHeight, 91
FontWidth, 91
FontHeight
FontDef, 91
FontWidth
FontDef, 91

gate_output
daisy::DaisyPatch, 76
Get
daisy::AdcHandle, 70
GetCurrentFile
daisy::WavPlayer, 114
GetLooping
daisy::WavPlayer, 115
GetMux
daisy::AdcHandle, 71
GetNumberFiles
daisy::WavPlayer, 115
GetPin
daisy::DaisySeed, 80

HAL_PCD_ConnectCallback
usbd_conf.c, 122
HAL_PCD_DataInStageCallback
usbd_conf.c, 123
HAL_PCD_DataOutStageCallback
usbd_conf.c, 123
HAL_PCD_DisconnectCallback
usbd_conf.c, 123
HAL_PCD_ISOINIncompleteCallback
usbd_conf.c, 124
HAL_PCD_ISOOUTIncompleteCallback
usbd_conf.c, 124
HAL_PCD_MspDeInit
usbd_conf.c, 125
HAL_PCD_MspInit
usbd_conf.c, 125
HAL_PCD_ResetCallback
usbd_conf.c, 125
HAL_PCD_ResumeCallback
usbd_conf.c, 125
HAL_PCD_SOFCallback
usbd_conf.c, 127
HAL_PCD_SetupStageCallback
usbd_conf.c, 127
HAL_PCD_SuspendCallback
usbd_conf.c, 127
HS_Desc
USBD_DESC_Exported_Variables, 50
USBD_DESC_Private_Variables, 37
HSE_VALUE
STM32H7xx_System_Private_Includes, 54
HSI_VALUE
STM32H7xx_System_Private_Includes, 54
HasEvents
daisy::MidiHandler, 95

ImmediateRead
daisy::RingBuffer, 102
Increment
daisy::Encoder, 90
Init
daisy::AdcHandle, 71
daisy::AnalogControl, 72
daisy::Color, 74
daisy::DaisyPatch, 76
daisy::DaisyPod, 79
daisy::DaisySeed, 80
daisy::Encoder, 90
daisy::GateIn, 92
daisy::Led, 93
daisy::MidiHandler, 95
daisy::OledDisplay, 98
daisy::Parameter, 100
daisy::RgbLed, 101
daisy::RingBuffer, 102
daisy::SdmmcHandler, 105
daisy::Switch, 108
daisy::UartHandler, 110
daisy::UsbHandle, 112
daisy::WavPlayer, 115
ShiftRegister595, 106

InitBipolarCv
daisy::AnalogControl, 72
InitMux
daisy::AdcChannelConfig, 69
InitSingle
daisy::AdcChannelConfig, 70

led, 92
LogBlockNbr
DSY_SD_CardInfoTypeDef, 87
LogBlockSize
DSY_SD_CardInfoTypeDef, 88

MidiInputMode
daisy::MidiHandler, 95
MidiMessageType
daisy, 67

NVIC_TypeDef, 97

Open
daisy::WavPlayer, 115
Overwrite
daisy::RingBuffer, 103

Parse
daisy::MidiHandler, 95
Pins
daisy::OledDisplay, 97
ShiftRegister595, 106
Polarity
daisy::Switch, 107
PollReceive
daisy::UartHandler, 110
PollTx
daisy::UartHandler, 110
PopEvent
daisy::MidiHandler, 96
PopRx
daisy::UartHandler, 110
Prepare
daisy::WavPlayer, 115
PresetColor
daisy::Color, 74
Pressed
daisy::Encoder, 90
daisy::Switch, 108
Process
daisy::AnalogControl, 72
daisy::Parameter, 100
Pull
daisy::Switch, 108

Read
daisy::RingBuffer, 103
Readable
daisy::UartHandler, 110
readable
daisy::RingBuffer, 103
ReceiveCallback
daisy::UsbHandle, 111
Red
daisy::Color, 74
RelCardAdd
DSY_SD_CardInfoTypeDef, 88
Restart
daisy::WavPlayer, 115
rgb_led, 100
RisingEdge
daisy::Encoder, 90
daisy::Switch, 109
RxActive
daisy::UartHandler, 111

SSD1309_t, 107
STM32_USB_OTG_DEVICE_LIBRARY, 62
STM32H7xx_System_Private_Defines, 56
VECT_TAB_OFFSET, 56
STM32H7xx_System_Private_FunctionPrototypes, 59
STM32H7xx_System_Private_Functions, 60
SystemCoreClockUpdate, 60
SystemInit, 61
STM32H7xx_System_Private_Includes, 54
CSI_VALUE, 54
HSE_VALUE, 54
HSI_VALUE, 54
STM32H7xx_System_Private_Macros, 57
STM32H7xx_System_Private_TypesDefinitions, 55
STM32H7xx_System_Private_Variables, 58
SdmmcBitWidth
daisy, 67
SdmmcMode
daisy, 67
SdmmcSpeed
daisy, 67
sdram_handle
daisy::DaisySeed, 81
seed
daisy::DaisyPatch, 77
daisy::DaisyPod, 79
Set
daisy::Led, 93
daisy::RgbLed, 101
ShiftRegister595, 106
SetAudioBlockSize
daisy::DaisyPatch, 76
daisy::DaisyPod, 79
daisy::DaisySeed, 80
SetColor
daisy::RgbLed, 101
SetCursor
daisy::OledDisplay, 98
SetLed
daisy::DaisySeed, 81
SetLooping
daisy::WavPlayer, 115
SetReceiveCallback
daisy::UsbHandle, 112
SetTestPoint

daisy::DaisySeed, 81
ShiftRegister595, 105
Init, 106
Pins, 106
Set, 106
Write, 106
src/system_stm32h7xx.c, 117
src/usbd_cdc_if.c, 118
src/usbd_cdc_if.h, 119
src/usbd_conf.c, 120
src/usbd_conf.h, 134
src/usbd_desc.c, 135
Start
daisy::AdcHandle, 71
StartAudio
daisy::DaisySeed, 81
StartReceive
daisy::MidiHandler, 96
StartRx
daisy::UartHandler, 111
Stm32h7xx_system, 53
Stop
daisy::AdcHandle, 71
Stream
daisy::WavPlayer, 115
Swallow
daisy::RingBuffer, 103
SystemCoreClockUpdate
STM32H7xx_System_Private_Functions, 60
SystemInit
STM32H7xx_System_Private_Functions, 61

TimeHeldMs
daisy::Encoder, 90
daisy::Switch, 109
TransmitExternal
daisy::UsbHandle, 112
TransmitInternal
daisy::UsbHandle, 112
Trig
daisy::GateIn, 92
Type
daisy::Switch, 108
type
daisy::MidiEvent, 94

USBD_CDC_IF_Exported_Defines, 20
USBD_CDC_IF_Exported_FunctionsPrototype, 23
CDC_Transmit_FS, 23
CDC_Transmit_HS, 23
USBD_CDC_IF_Exported_Macros, 22
USBD_CDC_IF_Exported_Types, 21
USBD_CDC_IF_Exported_Variables, 15
USBD_Interface_fops_FS, 15
USBD_Interface_fops_HS, 15
USBD_CDC_IF_Private_Defines, 12
USBD_CDC_IF_Private_FunctionPrototypes, 16
USBD_CDC_IF_Private_Macros, 13
USBD_CDC_IF_Private_TypesDefinitions, 11
USBD_CDC_IF_Private_Variables, 14
UserRxBufferFS, 14
UserRxBufferHS, 14
UserTxBufferFS, 14
UserTxBufferHS, 14
USBD_CDC_IF, 17
CDC_Transmit_FS, 18
CDC_Transmit_HS, 18
USBD_Interface_fops_FS, 18
USBD_Interface_fops_HS, 19
USBD_CONF_Exported_Defines, 27
USBD_CONF_Exported_FunctionsPrototype, 31
USBD_CONF_Exported_Macros, 28
USBD_DbgLog, 28
USBD_Delay, 28
USBD_ErrLog, 28
USBD_UsrLog, 29
USBD_free, 29
USBD_malloc, 29
USBD_memcpy, 29
USBD_memset, 29
USBD_CONF_Exported_Types, 30
USBD_CONF_Exported_Variables, 26
USBD_CONF, 25
USBD_DESC_Exported_Constants, 46
USBD_DESC_Exported_Defines, 47
USBD_DESC_Exported_FunctionsPrototype, 51
USBD_DESC_Exported_Macros, 49
USBD_DESC_Exported_TypesDefinitions, 48
USBD_DESC_Exported_Variables, 50
FS_Desc, 50
HS_Desc, 50
USBD_DESC_Private_Defines, 33
USBD_DESC_Private_FunctionPrototypes, 35
USBD_DESC_Private_Functions, 38
USBD_FS_ConfigStrDescriptor, 38
USBD_FS_DeviceDescriptor, 39
USBD_FS_InterfaceStrDescriptor, 39
USBD_FS_LangIDStrDescriptor, 40
USBD_FS_ManufacturerStrDescriptor, 40
USBD_FS_ProductStrDescriptor, 40
USBD_FS_SerialStrDescriptor, 41
USBD_HS_ConfigStrDescriptor, 41
USBD_HS_DeviceDescriptor, 42
USBD_HS_InterfaceStrDescriptor, 42
USBD_HS_LangIDStrDescriptor, 42
USBD_HS_ManufacturerStrDescriptor, 43
USBD_HS_ProductStrDescriptor, 43
USBD_HS_SerialStrDescriptor, 44
USBD_DESC_Private_Macros, 34
USBD_DESC_Private_TypesDefinitions, 32
USBD_DESC_Private_Variables, 36
__ALIGN_END, 36
FS_Desc, 36
HS_Desc, 37
USBD_DESC, 45
USBD_DbgLog
USBD_CONF_Exported_Macros, 28

USBD_Delay
  USBD_CONF_Exported_Macros, 28
USBD_ErrLog
  USBD_CONF_Exported_Macros, 28
USBD_FS_ConfigStrDescriptor
  USBD_DESC_Private_Functions, 38
USBD_FS_DeviceDescriptor
  USBD_DESC_Private_Functions, 39
USBD_FS_InterfaceStrDescriptor
  USBD_DESC_Private_Functions, 39
USBD_FS_LangIDStrDescriptor
  USBD_DESC_Private_Functions, 40
USBD_FS_ManufacturerStrDescriptor
  USBD_DESC_Private_Functions, 40
USBD_FS_ProductStrDescriptor
  USBD_DESC_Private_Functions, 40
USBD_FS_SerialStrDescriptor
  USBD_DESC_Private_Functions, 41
USBD_Get_USB_Status
  usbd_conf.c, 128
USBD_HS_ConfigStrDescriptor
  USBD_DESC_Private_Functions, 41
USBD_HS_DeviceDescriptor
  USBD_DESC_Private_Functions, 42
USBD_HS_InterfaceStrDescriptor
  USBD_DESC_Private_Functions, 42
USBD_HS_LangIDStrDescriptor
  USBD_DESC_Private_Functions, 42
USBD_HS_ManufacturerStrDescriptor
  USBD_DESC_Private_Functions, 43
USBD_HS_ProductStrDescriptor
  USBD_DESC_Private_Functions, 43
USBD_HS_SerialStrDescriptor
  USBD_DESC_Private_Functions, 44
USBD_Interface_fops_FS
  USBD_CDC_IF_Exported_Variables, 15
  USBD_CDC_IF, 18
USBD_Interface_fops_HS
  USBD_CDC_IF_Exported_Variables, 15
  USBD_CDC_IF, 19
USBD_LL_ClearStallEP
  usbd_conf.c, 128
USBD_LL_CloseEP
  usbd_conf.c, 129
USBD_LL_DeInit
  usbd_conf.c, 129
USBD_LL_Delay
  usbd_conf.c, 129
USBD_LL_FlushEP
  usbd_conf.c, 130
USBD_LL_GetRxDataSize
  usbd_conf.c, 130
USBD_LL_Init
  usbd_conf.c, 130
USBD_LL_IsStallEP
  usbd_conf.c, 131
USBD_LL_OpenEP
  usbd_conf.c, 131
USBD_LL_PrepareReceive
  usbd_conf.c, 132
USBD_LL_SetUSBAddress
  usbd_conf.c, 132
USBD_LL_StallEP
  usbd_conf.c, 132
USBD_LL_Start
  usbd_conf.c, 133
USBD_LL_Stop
  usbd_conf.c, 133
USBD_LL_Transmit
  usbd_conf.c, 134
USBD_OTG_DRIVER, 63
USBD_UsrLog
  USBD_CONF_Exported_Macros, 29
USBD_free
  USBD_CONF_Exported_Macros, 29
USBD_malloc
  USBD_CONF_Exported_Macros, 29
USBD_memcpy
  USBD_CONF_Exported_Macros, 29
USBD_memset
  USBD_CONF_Exported_Macros, 29
uart_handle, 109
Update
  daisy::Led, 93
  daisy::OledDisplay, 98
  daisy::RgbLed, 101
UsbPeriph
  daisy::UsbHandle, 112
usbd_conf.c
  HAL_PCD_ConnectCallback, 122
  HAL_PCD_DataInStageCallback, 123
  HAL_PCD_DataOutStageCallback, 123
  HAL_PCD_DisconnectCallback, 123
  HAL_PCD_ISOINIncompleteCallback, 124
  HAL_PCD_ISOOUTIncompleteCallback, 124
  HAL_PCD_MspDeInit, 125
  HAL_PCD_MspInit, 125
  HAL_PCD_ResetCallback, 125
  HAL_PCD_ResumeCallback, 125
  HAL_PCD_SOFCallback, 127
  HAL_PCD_SetupStageCallback, 127
  HAL_PCD_SuspendCallback, 127
  USBD_Get_USB_Status, 128
  USBD_LL_ClearStallEP, 128
  USBD_LL_CloseEP, 129
  USBD_LL_DeInit, 129
  USBD_LL_Delay, 129
  USBD_LL_FlushEP, 130
  USBD_LL_GetRxDataSize, 130
  USBD_LL_Init, 130
  USBD_LL_IsStallEP, 131
  USBD_LL_OpenEP, 131
  USBD_LL_PrepareReceive, 132
  USBD_LL_SetUSBAddress, 132
  USBD_LL_StallEP, 132
  USBD_LL_Start, 133

USBD_LL_Stop, 133
USBD_LL_Transmit, 134
UserRxBufferFS
USBD_CDC_IF_Private_Variables, 14
UserRxBufferHS
USBD_CDC_IF_Private_Variables, 14
UserTxBufferFS
USBD_CDC_IF_Private_Variables, 14
UserTxBufferHS
USBD_CDC_IF_Private_Variables, 14

VECT_TAB_OFFSET
STM32H7xx_System_Private_Defines, 56
Value
daisy::AnalogControl, 72
daisy::Parameter, 100

WAV_FormatTypeDef, 113
writable
daisy::RingBuffer, 103
Write
daisy::RingBuffer, 104
ShiftRegister595, 106
WriteChar
daisy::OledDisplay, 98
WriteString
daisy::OledDisplay, 99