DaisySP

Generated by Doxygen 1.8.13

Contents

Chapter 1

libdaisy

Multi-layer hardware abstraction library for Daisy Product family

On STM32H7 MCUs

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

Prefixes and their meanings:

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

Also included is a core/ folder containing:

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

1.1 Using libdaisy

Due to the amount of hardware configuration and flexibility of the daisy platform, (in the present, and the future), a user can use libdaisy to define their own custom hardware, or include one of our supported board files to jumpstart the creativity, and hack on an existing piece of hardware.

If you are getting started, and have one of the Daisy Family Products, you can skip ahead to that section below.

1.1.1 daisy.h

The base-level include file. This is all you need to include to create your own custom hardware that uses libdaisy.

`daisy_seed.h` is an example of a board level file that utilizes libdaisy to define some hardware, and provide flexible access.

1.1.2 daisy_seed.h

The SOM-level include file. This can be used with any boards that use the Daisy Seed hardware.

Additional configuration files, with more specific hardware access are provided below for our supported hardware platforms.

1.1.3 daisy_platform.h

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

These are:

- daisy_field
- daisy_patch
- daisy_petal
- daisy_pod

With these files a number of additional initialization, and configuration is done by the library.

This allows a user to jump right into their new product with a simple api to do things without having a full understanding of what's going on under the hood.

With this flexible approach to the hardware configuration, we hope to promote a lot of fantastic hardware along with code to go with it.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Namespace Index

3.1 Namespace List

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

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

Chapter 6

Module Documentation

6.1 USBD_CDC_IF

Usb VCP device module.

Modules

- USBD_CDC_IF_Exported_Defines

  Defines.
- USBD_CDC_IF_Exported_Types

  Types.
- USBD_CDC_IF_Exported_Macros

  Aliases.
- USBD_CDC_IF_Exported_Variables

  Public variables.
- USBD_CDC_IF_Exported_FunctionsPrototype

  Public functions declaration.

6.1.1 Detailed Description

Usb VCP device module.

6.2 USBD_CDC_IF_Exported_Defines

Defines.

Defines.

6.3 USBD_CDC_IF_Exported_Types

Types.

Typedefs

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

6.3.1 Detailed Description

Types.

6.4 USBD_CDC_IF_Exported_Macros

Aliases.

Aliases.

6.5 USBD_CDC_IF_Exported_Variables

Public variables.

Variables

- USBD_CDC_ItfTypeDef USBD_Interface_fops_FS
- USBD_CDC_ItfTypeDef USBD_Interface_fops_HS

6.5.1 Detailed Description

Public variables.

6.5.2 Variable Documentation

6.5.2.1 USBD_Interface_fops_FS

```
USBD_CDC_ItfTypeDef USBD_Interface_fops_FS
```

CDC Interface callback.

6.5.2.2 USBD_Interface_fops_HS

```
USBD_CDC_ItfTypeDef USBD_Interface_fops_HS
```

CDC Interface callback.

6.6 USBD_CDC_IF_Exported_FunctionsPrototype

Public functions declaration.

Functions

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

6.6.1 Detailed Description

Public functions declaration.

6.7 USBD_CONF

Configuration file for Usb otg low level driver.

Modules

- USBD_CONF_Exported_Variables

  Public variables.
- USBD_CONF_Exported_Defines

  Defines for configuration of the Usb device.
- USBD_CONF_Exported_Macros

  Aliases.
- USBD_CONF_Exported_Types

  Types.
- USBD_CONF_Exported_FunctionsPrototype

  Declaration of public functions for Usb device.

6.7.1 Detailed Description

Configuration file for Usb otg low level driver.

6.8 USBD_CONF_Exported_Variables

Public variables.

Public variables.

6.9 USBD_CONF_Exported_Defines

Defines for configuration of the Usb device.

Macros

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

6.9.1 Detailed Description

Defines for configuration of the Usb device.

6.10 USBD_CONF_Exported_Macros

Aliases.

Macros

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

6.10.1 Detailed Description

Aliases.

6.10.2 Macro Definition Documentation

6.10.2.1 USBD_DbgLog

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

Value:

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

6.10.2.2 USBD_Delay

```
#define USBD_Delay HAL_Delay
```

Alias for delay.

6.10.2.3 USBD_ErrLog

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

Value:

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

6.10.2.4 USBD_free

```
#define USBD_free free
```

Alias for memory release.

6.10.2.5 USBD_malloc

```
#define USBD_malloc malloc
```

Alias for memory allocation.

6.10.2.6 USBD_memcpy

```
#define USBD_memcpy memcpy
```

Alias for memory copy.

6.10.2.7 USBD_memset

```
#define USBD_memset memset
```

Alias for memory set.

6.10.2.8 USBD_UsrLog

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

Value:

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

6.11 USBD_CONF_Exported_Types

Types.

Types.

6.12 USBD_CONF_Exported_FunctionsPrototype

Declaration of public functions for Usb device.

Declaration of public functions for Usb device.

6.13 USBD_DESC

Usb device descriptors module.

Modules

- USBD_DESC_Exported_Constants

  Constants.
- USBD_DESC_Exported_Defines

  Defines.
- USBD_DESC_Exported_TypesDefinitions

  Types.
- USBD_DESC_Exported_Macros

  Aliases.
- USBD_DESC_Exported_Variables

  Public variables.
- USBD_DESC_Exported_FunctionsPrototype

  Public functions declaration.

6.13.1 Detailed Description

Usb device descriptors module.

6.14 USBD_DESC_Exported_Constants

Constants.

Macros

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

6.14.1 Detailed Description

Constants.

6.15 USBD_DESC_Exported_Defines

Defines.

Defines.

6.16 USBD_DESC_Exported_TypesDefinitions

Types.

Types.

6.17 USBD_DESC_Exported_Macros

Aliases.

Aliases.

6.18 USBD_DESC_Exported_Variables

Public variables.

Variables

- USBD_DescriptorsTypeDef HS_Desc
- USBD_DescriptorsTypeDef FS_Desc

6.18.1 Detailed Description

Public variables.

6.18.2 Variable Documentation

6.18.2.1 FS_Desc

```
USBD_DescriptorsTypeDef FS_Desc
```

Descriptor for the Usb device.

6.18.2.2 HS_Desc

```
USBD_DescriptorsTypeDef HS_Desc
```

Descriptor for the Usb device.

6.19 USBD_DESC_Exported_FunctionsPrototype

Public functions declaration.

Public functions declaration.

6.20 STM32_USB_OTG_DEVICE_LIBRARY

For Usb device.

Modules

- USBD_CDC_IF

  Usb VCP device module.
- USBD_DESC

  Usb device descriptors module.

6.20.1 Detailed Description

For Usb device.

6.21 USBD_OTG_DRIVER

Modules

- USBD_CONF

  Configuration file for Usb otg low level driver.

6.21.1 Detailed Description

Chapter 7

Namespace Documentation

7.1 daisy Namespace Reference

Classes

- struct AdcChannelConfig
- class AdcHandle
- class AnalogControl

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

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

  Helpers and hardware definitions for daisy petal.
- class DaisyPod

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

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

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

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

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

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

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

  Interface for initializing and using the USB Peripherals on the daisy.
- struct WavFileInfo
- class WavPlayer

Enumerations

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

Functions

- FORCE_INLINE void daisy_field_init (daisy_field ∗p)

Variables

- const size_t **kUartMaxBufferSize** = 32

7.1.1 Detailed Description

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

Setup Hardware PWM for pins that have it

TODO:

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

7.1.2 Enumeration Type Documentation

7.1.2.1 anonymous enum

```
anonymous enum
```

enums for controls, etc.

Enumerator

SW_2	tactile switch
SW_1	tactile switch
SW_3	toggle
SW_LAST	**7.1.3 autotoc_md43**

7.1.3.1 anonymous enum

```
anonymous enum
```

All knobs connect to ADC1_INP10 via CD4051 mux

Enumerator

KNOB_1	**7.1.4 autotoc_md44**
KNOB_3	**7.1.5 autotoc_md45**
KNOB_5	**7.1.6 autotoc_md46**
KNOB_2	**7.1.7 autotoc_md47**
KNOB_4	**7.1.8 autotoc_md48**
KNOB_6	**7.1.9 autotoc_md49**
KNOB_7	**7.1.10 autotoc_md50**
KNOB_8	**7.1.11 autotoc_md51**
KNOB_LAST	**7.1.12 autotoc_md52**

7.1.12.1 anonymous enum

```
anonymous enum
```

Enumerator

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

7.1.12.2 anonymous enum

`anonymous enum`

Enumerator

LED_KEY_A8	**7.1.13 autotoc_md53**
LED_KEY_A7	**7.1.14 autotoc_md54**
LED_KEY_A6	**7.1.15 autotoc_md55**
LED_KEY_A5	**7.1.16 autotoc_md56**
LED_KEY_A4	**7.1.17 autotoc_md57**
LED_KEY_A3	**7.1.18 autotoc_md58**
LED_KEY_A2	**7.1.19 autotoc_md59**
LED_KEY_A1	**7.1.20 autotoc_md60**
LED_KEY_B1	**7.1.21 autotoc_md61**
LED_KEY_B2	**7.1.22 autotoc_md62**
LED_KEY_B3	**7.1.23 autotoc_md63**

Enumerator

LED_KEY_B4	7.1.24 autotoc_md64
LED_KEY_B5	7.1.25 autotoc_md65
LED_KEY_B6	7.1.26 autotoc_md66
LED_KEY_B7	7.1.27 autotoc_md67
LED_KEY_B8	7.1.28 autotoc_md68
LED_KNOB↩_1	7.1.29 autotoc_md69
LED_KNOB↩_2	7.1.30 autotoc_md70
LED_KNOB↩_3	7.1.31 autotoc_md71
LED_KNOB↩_4	7.1.32 autotoc_md72
LED_KNOB↩_5	7.1.33 autotoc_md73
LED_KNOB↩_6	7.1.34 autotoc_md74
LED_KNOB↩_7	7.1.35 autotoc_md75
LED_KNOB↩_8	7.1.36 autotoc_md76

Enumerator

LED_SW_1	**7.1.37 autotoc_md77**
LED_SW_2	**7.1.38 autotoc_md78**
LED_LAST	**7.1.39 autotoc_md79**

7.1.39.1 MidiMessageType

```
enum daisy::MidiMessageType
```

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

Enumerator

NoteOff	**7.1.40 autotoc_md178**
NoteOn	**7.1.41 autotoc_md179**
PolyphonicKeyPressure	**7.1.42 autotoc_md180**
ControlChange	**7.1.43 autotoc_md181**
ProgramChange	**7.1.44 autotoc_md182**
ChannelPressure	**7.1.45 autotoc_md183**
PitchBend	**7.1.46 autotoc_md184**

Enumerator

MessageLast	7.1.47 autotoc_md185

7.1.47.1 SdmmcBitWidth

```
enum daisy::SdmmcBitWidth
```

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

7.1.47.2 SdmmcMode

```
enum daisy::SdmmcMode
```

Operating ModeCurrently only FatFS is supported.

7.1.47.3 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.

7.1.47.4 SpiPeriph

```
enum daisy::SpiPeriph
```

Enumerator

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

7.1.47.5 SpiPin

```
enum daisy::SpiPin
```

Enumerator

SPI_PIN_SCK	CS pin
SPI_PIN_MOSI	SCK pin
SPI_PIN_MISO	MOSI pin

7.1.48 Function Documentation

7.1.48.1 daisy_field_init()

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

Initializes daisy field

Parameters

p	daisy_field struct to initialize

< #

< #

< #

< #

< #

< #

< #

< #

< #

< #

< #

< #

< #

< #

< #

< #

< #

< #

Chapter 8

Class Documentation

8.1 daisy::AdcChannelConfig Struct Reference

```
#include <per_adc.h>
```

Public Types

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

Public Member Functions

- void InitSingle (dsy_gpio_pin pin)
- void InitMux (dsy_gpio_pin adc_pin, dsy_gpio_pin mux_0, dsy_gpio_pin mux_1, dsy_gpio_pin mux_2, size↩ _t channels)

Public Attributes

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

8.1.1 Detailed Description

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

8.1.2 Member Function Documentation

8.1.2.1 InitMux()

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

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

8.1.2.2 InitSingle()

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

Initializes a single ADC pin as an ADC.

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

- src/per_adc.h

8.2 daisy::AdcHandle Class Reference

Public Types

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

Public Member Functions

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

8.2.1 Member Function Documentation

8.2.1.1 Get()

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

These are getters for a single channel

8.2.1.2 GetMux()

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

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

8.2.1.3 Init()

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

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

Parameters

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

8.2.1.4 Start()

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

Starts reading from the ADC

8.2.1.5 Stop()

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

Stops reading from the ADC

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

- src/per_adc.h

8.3 daisy::AnalogControl Class Reference

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

```
#include <hid_ctrl.h>
```

Public Member Functions

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

8.3.1 Detailed Description

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

Author

Stephen Hensley

Date

November 2019

8.3.2 Constructor & Destructor Documentation

8.3.2.1 AnalogControl()

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

Constructor

8.3.2.2 ∼AnalogControl()

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

destructor

8.3.3 Member Function Documentation

8.3.3.1 Init()

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

Initializes the control

Parameters

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

8.3.3.2 InitBipolarCv()

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

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

Parameters

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

8.3.3.3 Process()

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

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

8.3.3.4 Value()

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

Returns the current stored value, without reprocessing

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

- src/hid_ctrl.h

8.4 codec_frame_t Struct Reference

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

Public Attributes

- short l
- short r

8.4.1 Detailed Description

8.4.2 autotoc_md137

8.4.3 Member Data Documentation

8.4.3.1 l

```
short codec_frame_t::l
```

8.4.4 autotoc_md139

8.4.4.1 r

```
short codec_frame_t::r
```

8.4.5 autotoc_md140

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

- src/dev_codec_wm8731_frame.h

8.5 color Struct Reference

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

Public Attributes

- uint16_t red
- uint16_t green
- uint16_t blue

8.5.1 Detailed Description

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

8.5.2 Member Data Documentation

8.5.2.1 blue

```
uint16_t color::blue
```

8.5.3 autotoc_md153

8.5.3.1 green

```
uint16_t color::green
```

8.5.4 autotoc_md152

8.5.4.1 red

```
uint16_t color::red
```

8.5.5 autotoc_md151

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

- src/dev_leddriver.h

8.6 daisy::Color Class Reference

Public Types

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

Public Member Functions

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

8.6.1 Member Enumeration Documentation

8.6.1.1 PresetColor

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

List of colors that have a preset RGB value

8.6.2 Member Function Documentation

8.6.2.1 Init() [1/2]

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

Initializes the Color with a given preset.

8.6.2.2 Init() [2/2]

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

Initializes the Color with a specific RGB value

red, green, and blue should be floats between 0 and 1

8.6.2.3 Red()

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

Returns the 0-1 value for the given color

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

- src/util_color.h

8.7 daisy::ControlChangeEvent Struct Reference

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

Public Attributes

- int channel
- uint8_t control_number
- uint8_t value

8.7.1 Detailed Description

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

8.7.2 Member Data Documentation

8.7.2.1 channel

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

8.7.3 autotoc_md189

8.7.3.1 control_number

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

8.7.4 autotoc_md190

8.7.4.1 value

```
uint8_t daisy::ControlChangeEvent::value
```

8.7.5 autotoc_md191

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

- src/hid_midi.h

8.8 daisy::daisy_field Struct Reference

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

Public Attributes

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

8.8.1 Detailed Description

Struct containing hardware defines and daisy seed

8.8.2 Member Data Documentation

8.8.2.1 cvs

`AnalogControl daisy::daisy_field::cvs[CV_LAST]`

Array of cv ins

8.8.2.2 gate_in

`dsy_gpio daisy::daisy_field::gate_in`

Gate input.

8.8.2.3 gate_out

`dsy_gpio daisy::daisy_field::gate_out`

Gate output

8.8.2.4 keyboard_sr

`dsy_sr_4021_handle daisy::daisy_field::keyboard_sr`

Keyboard shift register

8.8.2.5 knobs

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

Array of hardware knobs

8.8.2.6 seed

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

Daisy seed

8.8.2.7 switches

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

Array of hardware switches

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

- src/daisy_field.h

8.9 daisy::DaisyPatch Class Reference

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

```
#include <daisy_patch.h>
```

Public Types

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

Public Member Functions

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

Public Attributes

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

8.9.1 Detailed Description

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

Author

Stephen Hensley

Date

November 2019

8.9.2 Member Enumeration Documentation

8.9.2.1 Ctrl

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

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

8.9.2.2 GateInput

```
enum daisy::DaisyPatch::GateInput
```

Daisy patch gate inputs

Enumerator

GATE_IN_LAST	<

8.9.3 Constructor & Destructor Documentation

8.9.3.1 DaisyPatch()

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

Constructor

8.9.3.2 ∼DaisyPatch()

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

Destructor

8.9.4 Member Function Documentation

8.9.4.1 AudioBlockSize()

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

Get block size

8.9.4.2 AudioCallbackRate()

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

Get callback rate

8.9.4.3 AudioSampleRate()

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

Get sample rate

8.9.4.4 ChangeAudioCallback()

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

Change to a different callback function.

Parameters

cb	New callback function.

8.9.4.5 DebounceControls()

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

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

8.9.4.6 DelayMs()

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

Wait some ms before going on.

Parameters

del	Delay time in ms.

8.9.4.7 DisplayControls()

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

Control the display

8.9.4.8 GetCtrlValue()

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

Get value for a partiular control

Parameters

k	Which control to get

8.9.4.9 Init()

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

Initializes the daisy seed, and patch hardware.

8.9.4.10 SetAudioBlockSize()

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

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

Parameters

size	Audio block size.

8.9.4.11 StartAdc()

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

Start analog to digital conversion.

8.9.4.12 StartAudio()

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

Start audio output.

Parameters

cb	Audio callback function

8.9.4.13 UpdateAnalogControls()

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

Call at same rate as reading controls for good reads.

8.9.5 Member Data Documentation

8.9.5.1 controls

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

Array of controls

8.9.5.2 display

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

8.9.6 autotoc_md80

8.9.6.1 encoder

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

Encoder object

8.9.6.2 gate_input

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

Gate inputs

8.9.6.3 gate_output

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

8.9.7 autotoc_md81

8.9.7.1 midi

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

Handles midi

8.9.7.2 seed

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

Seed object

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

- src/daisy_patch.h

8.10 daisy::DaisyPetal Class Reference

Helpers and hardware definitions for daisy petal.

```
#include <daisy_petal.h>
```

Public Types

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

Public Member Functions

- DaisyPetal ()
- ∼DaisyPetal ()
- void Init ()
- void DelayMs (size_t del)
- void SetAudioBlockSize (size_t size)
- void StartAudio (dsy_audio_callback cb)
- void ChangeAudioCallback (dsy_audio_callback cb)
- void StartAdc ()
- float AudioSampleRate ()
- size_t AudioBlockSize ()
- float AudioCallbackRate ()
- void UpdateAnalogControls ()
- float GetKnobValue (Knob k)
- float GetExpression ()
- void DebounceControls ()
- void ClearLeds ()
- void UpdateLeds ()
- void SetRingLed (RingLed idx, float r, float g, float b)
- void SetFootswitchLed (FootswitchLed idx, float bright)

Public Attributes

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

8.10.1 Detailed Description

Helpers and hardware definitions for daisy petal.

8.10.2 Member Enumeration Documentation

8.10.2.1 FootswitchLed

```
enum daisy::DaisyPetal::FootswitchLed
```

footswitch leds

Enumerator

FOOTSWITCH_LED_1	8.10.3 autotoc_md106
FOOTSWITCH_LED_2	8.10.4 autotoc_md107
FOOTSWITCH_LED_3	8.10.5 autotoc_md108
FOOTSWITCH_LED_4	8.10.6 autotoc_md109
FOOTSWITCH_LED_LAST	8.10.7 autotoc_md110

8.10.7.1 Knob

```
enum daisy::DaisyPetal::Knob
```

Knobs

Enumerator

KNOB_1	8.10.8 autotoc_md90
KNOB_2	8.10.9 autotoc_md91
KNOB_3	8.10.10 autotoc_md92
KNOB_4	8.10.11 autotoc_md93
KNOB_5	8.10.12 autotoc_md94

Enumerator

KNOB_6	**8.10.13 autotoc_md95**
KNOB_LAST	**8.10.14 autotoc_md96**

8.10.14.1 RingLed

`enum daisy::DaisyPetal::RingLed`

Leds in ringled

Enumerator

RING_LED_1	**8.10.15 autotoc_md97**
RING_LED_2	**8.10.16 autotoc_md98**
RING_LED_3	**8.10.17 autotoc_md99**
RING_LED_4	**8.10.18 autotoc_md100**
RING_LED_5	**8.10.19 autotoc_md101**
RING_LED_6	**8.10.20 autotoc_md102**
RING_LED_7	**8.10.21 autotoc_md103**
RING_LED_8	**8.10.22 autotoc_md104**

Enumerator

RING_LED_LAST	

8.10.23 autotoc_md105

8.10.23.1 Sw

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

Switches

Enumerator

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

8.10.24 Constructor & Destructor Documentation

8.10.24.1 DaisyPetal()

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

Constructor

8.10.24.2 ∼DaisyPetal()

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

Destructor

8.10.25 Member Function Documentation

8.10.25.1 AudioBlockSize()

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

Get audio block size

8.10.25.2 AudioCallbackRate()

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

Get callback rate

8.10.25.3 AudioSampleRate()

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

Device audio sample rate.

8.10.25.4 ChangeAudioCallback()

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

Change callback function

Parameters

cb	New callback function.

8.10.25.5 ClearLeds()

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

Turn all leds off

8.10.25.6 DebounceControls()

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

Debounce inputs.

8.10.25.7 DelayMs()

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

Wait before moving on.

Parameters

del	Delay time in ms.

8.10.25.8 GetExpression()

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

8.10.26 autotoc_md83

8.10.26.1 GetKnobValue()

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

Get value per knob.

Parameters

k	Which knob to get

Returns

Floating point knob position.

8.10.26.2 Init()

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

Initialize daisy petal

8.10.26.3 SetAudioBlockSize()

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

Set size of audio blocks.

Parameters

size	Audio block size

8.10.26.4 SetFootswitchLed()

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

Set footswitch LED

Parameters

idx	Led Index
bright	Brightness

8.10.26.5 SetRingLed()

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

Set ring LED colors

Parameters

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

8.10.26.6 StartAdc()

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

Start analog to digital conversion.

8.10.26.7 StartAudio()

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

Start audio callback

Parameters

cb	Callback function.

8.10.26.8 UpdateAnalogControls()

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

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

8.10.26.9 UpdateLeds()

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

Update Leds to values you had set.

8.10.27 Member Data Documentation

8.10.27.1 encoder

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

8.10.28 autotoc_md85

8.10.28.1 expression

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

8.10.29 autotoc_md87

8.10.29.1 footswitch_led

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

8.10.30 autotoc_md89

8.10.30.1 knob

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

8.10.31 autotoc_md86

8.10.31.1 ring_led

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

8.10.32 autotoc_md88

8.10.32.1 seed

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

8.10.33 autotoc_md84

8.10.33.1 switches

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

< #

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

- src/daisy_petal.h

8.11 daisy::DaisyPod Class Reference

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

```
#include <daisy_pod.h>
```

Public Types

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

Public Member Functions

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

Public Attributes

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

8.11.1 Detailed Description

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

Author

Stephen Hensley

Date

November 2019

8.11.2 Member Enumeration Documentation

8.11.2.1 Knob

```
enum daisy::DaisyPod::Knob
```

Knobs

Enumerator

KNOB_2	**8.11.3 autotoc_md125**
KNOB_LAST	**8.11.4 autotoc_md126**

8.11.4.1 Sw

```
enum daisy::DaisyPod::Sw
```

Switches

Enumerator

BUTTON_2	**8.11.5 autotoc_md122**
BUTTON_LAST	**8.11.6 autotoc_md123**

8.11.7 Member Function Documentation

8.11.7.1 AudioBlockSize()

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

Get block size

8.11.7.2 AudioCallbackRate()

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

Get callback rate

8.11.7.3 AudioSampleRate()

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

Get sample rate

8.11.7.4 ChangeAudioCallback()

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

Switch callback functions

Parameters

cb	New callback function.

8.11.7.5 ClearLeds()

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

Reset Leds

8.11.7.6 DebounceControls()

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

8.11.8 autotoc_md112

8.11.8.1 DelayMs()

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

Wait for a bit

Parameters

del	Time to wait in ms.

8.11.8.2 GetKnobValue()

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

8.11.9 autotoc_md111

8.11.9.1 Init()

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

Init related stuff.

8.11.9.2 SetAudioBlockSize()

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

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

Parameters

size	Block size to set.

8.11.9.3 StartAdc()

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

Start analog to digital conversion.

8.11.9.4 StartAudio()

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

Start audio callback

Parameters

cb	Callback function.

8.11.9.5 UpdateAnalogControls()

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

Call at same rate as analog reads for smooth reading.

8.11.9.6 UpdateLeds()

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

Update Leds to set colors

8.11.10 Member Data Documentation

8.11.10.1 button1

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

8.11.11 autotoc_md117

8.11.11.1 button2

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

8.11.12 autotoc_md118

8.11.12.1 buttons

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

8.11.13 autotoc_md119

8.11.13.1 encoder

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

8.11.14 autotoc_md113

8.11.14.1 knob1

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

8.11.15 autotoc_md114

8.11.15.1 knob2

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

8.11.16 autotoc_md115

8.11.16.1 knobs

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

8.11.17 autotoc_md116

8.11.17.1 led1

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

8.11.18 autotoc_md120

8.11.18.1 led2

```
RgbLed daisy::DaisyPod::led2
```

8.11.19 autotoc_md121

8.11.19.1 seed

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

Public Members #

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

- src/daisy_pod.h

8.12 daisy::DaisySeed Class Reference

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

```
#include <daisy_seed.h>
```

Public Member Functions

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

Public Attributes

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

8.12.1 Detailed Description

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

8.12.2 Member Function Documentation

8.12.2.1 AudioSampleRate()

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

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

8.12.2.2 Configure()

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

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

8.12.2.3 GetPin()

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

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

8.12.2.4 Init()

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

Initializes the Daisy Seed and the following peripherals: SDRAM, QSPI, 24-bit 48kHz Audio via AK4556, Internal USB, as well as the built-in LED and Testpoint.

ADCs, DACs, and other special peripherals (such as I2C, SPI, etc.) can be initialized using their specific initializers within libdaisy for a specific application.

8.12.2.5 SetAudioBlockSize()

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

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

8.12.2.6 SetLed()

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

Sets the state of the built in LED

8.12.2.7 SetTestPoint()

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

Sets the state of the test point near pin 10

8.12.2.8 StartAudio()

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

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

8.12.3 Member Data Documentation

8.12.3.1 adc

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

8.12.4 autotoc_md134

8.12.4.1 audio_handle

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

8.12.5 autotoc_md130

8.12.5.1 dac_handle

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

8.12.6 autotoc_md135

8.12.6.1 i2c1_handle

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

8.12.7 autotoc_md132

8.12.7.1 i2c2_handle

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

8.12.8 autotoc_md133

8.12.8.1 qspi_handle

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

8.12.9 autotoc_md129

8.12.9.1 sai_handle

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

8.12.10 autotoc_md131

8.12.10.1 sdram_handle

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

8.12.11 autotoc_md128

8.12.11.1 usb_handle

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

8.12.12 autotoc_md136

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

- src/daisy_seed.h

8.13 dsy_audio_handle Struct Reference

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

Public Attributes

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

8.13.1 Detailed Description

Simple config struct that holds peripheral drivers.

8.13.2 Member Data Documentation

8.13.2.1 block_size

```
size_t dsy_audio_handle::block_size
```

8.13.3 autotoc_md174

8.13.3.1 dev0_i2c

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

8.13.4 autotoc_md176

8.13.4.1 dev1_i2c

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

8.13.5 autotoc_md177

8.13.5.1 sai

```
dsy_sai_handle* dsy_audio_handle::sai
```

8.13.6 autotoc_md175

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

- src/hid_audio.h

8.14 dsy_dac_handle Struct Reference

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

Public Attributes

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

8.14.1 Detailed Description

Configuration structure for DAC initialization and settings.

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

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

- src/per_dac.h

8.15 dsy_gpio Struct Reference

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

Public Attributes

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

8.15.1 Detailed Description

Struct for holding the pin, and configuration

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

- src/per_gpio.h

8.16 dsy_gpio_pin Struct Reference

```
#include <daisy_core.h>
```

Public Attributes

- dsy_gpio_port port
- uint8_t pin

8.16.1 Detailed Description

Hardware define pins

8.16.2 Member Data Documentation

8.16.2.1 pin

```
uint8_t dsy_gpio_pin::pin
```

number 0-15

8.16.2.2 port

```
dsy_gpio_port dsy_gpio_pin::port
```

8.16.3 autotoc_md20

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

- src/daisy_core.h

8.17 dsy_i2c_handle Struct Reference

```
#include <per_i2c.h>
```

Public Attributes

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

8.17.1 Detailed Description

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

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

- src/per_i2c.h

8.18 dsy_qspi_handle Struct Reference

```
#include <per_qspi.h>
```

Public Attributes

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

8.18.1 Detailed Description

Configuration structure for interfacing with QSPI Driver.

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

- src/per_qspi.h

8.19 dsy_sai_handle Struct Reference

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

Public Attributes

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

8.19.1 Detailed Description

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

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

- src/per_sai.h

8.20 DSY_SD_CardInfoTypeDef Struct Reference

```
#include <util_bsp_sd_diskio.h>
```

Public Attributes

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

8.20.1 Detailed Description

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

8.20.2 Member Data Documentation

8.20.2.1 BlockNbr

```
uint32_t DSY_SD_CardInfoTypeDef::BlockNbr
```

Specifies the Card Capacity in blocks

8.20.2.2 BlockSize

```
uint32_t DSY_SD_CardInfoTypeDef::BlockSize
```

Specifies one block size in bytes

8.20.2.3 CardSpeed

```
uint32_t DSY_SD_CardInfoTypeDef::CardSpeed
```

Specifies the card Speed

8.20.2.4 CardType

```
uint32_t DSY_SD_CardInfoTypeDef::CardType
```

Specifies the card Type

8.20.2.5 CardVersion

```
uint32_t DSY_SD_CardInfoTypeDef::CardVersion
```

Specifies the card version

8.20.2.6 Class

```
uint32_t DSY_SD_CardInfoTypeDef::Class
```

Specifies the class of the card class

8.20.2.7 LogBlockNbr

```
uint32_t DSY_SD_CardInfoTypeDef::LogBlockNbr
```

Specifies the Card logical Capacity in blocks

8.20.2.8 LogBlockSize

```
uint32_t DSY_SD_CardInfoTypeDef::LogBlockSize
```

Specifies logical block size in bytes

8.20.2.9 RelCardAdd

```
uint32_t DSY_SD_CardInfoTypeDef::RelCardAdd
```

Specifies the Relative Card Address

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

- src/util_bsp_sd_diskio.h

8.21 dsy_sdram_handle Struct Reference

```
#include <dev_sdram.h>
```

Public Attributes

- dsy_sdram_state state
- dsy_gpio_pin pin_config [DSY_SDRAM_PIN_LAST]

8.21.1 Detailed Description

Configuration struct for passing to initialization

8.21.2 Member Data Documentation

8.21.2.1 pin_config

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

8.21.3 autotoc_md162

8.21.3.1 state

```
dsy_sdram_state dsy_sdram_handle::state
```

8.21.4 autotoc_md161

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

- src/dev_sdram.h

8.22 dsy_sr_4021_handle Struct Reference

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

Public Attributes

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

8.22.1 Detailed Description

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

8.22.2 Member Data Documentation

8.22.2.1 clk

`dsy_gpio dsy_sr_4021_handle::clk`

clk pin

8.22.2.2 cs

`dsy_gpio dsy_sr_4021_handle::cs`

cs pin

8.22.2.3 data

`dsy_gpio dsy_sr_4021_handle::data[2]`

array of data pins

8.22.2.4 num_daisychained

`uint8_t dsy_sr_4021_handle::num_daisychained`

Number of devices daisy chained

8.22.2.5 num_parallel

`uint8_t dsy_sr_4021_handle::num_parallel`

number of devices connected

8.22.2.6 pin_config

`dsy_gpio_pin dsy_sr_4021_handle::pin_config[DSY_SR_4021_PIN_LAST]`

used to initialize the dsy_gpio

8.22.2.7 states

```
uint8_t dsy_sr_4021_handle::states[8 * 1 * 2]
```

array of states

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

- src/dev_sr_4021.h

8.23 daisy::Encoder Class Reference

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

```
#include <hid_encoder.h>
```

Public Member Functions

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

8.23.1 Detailed Description

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

Author

Stephen Hensley

Date

December 2019

8.23.2 Member Function Documentation

8.23.2.1 Debounce()

```
void daisy::Encoder::Debounce ( )
```

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

8.23.2.2 FallingEdge()

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

Returns true if the encoder was just released.

8.23.2.3 Increment()

```
int32_t daisy::Encoder::Increment ( ) const  [inline]
```

Returns +1 if the encoder was turned clockwise, -1 if it was turned counter-clockwise, or 0 if it was not just turned.

8.23.2.4 Init()

```
void daisy::Encoder::Init (
             dsy_gpio_pin a,
             dsy_gpio_pin b,
             dsy_gpio_pin click,
             float update_rate )
```

Initializes the encoder with the specified hardware pins. Update rate should be the rate at which Debounce() gets called in Hertz.

8.23.2.5 Pressed()

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

Returns true while the encoder is held down.

8.23.2.6 RisingEdge()

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

Returns true if the encoder was just pressed.

8.23.2.7 TimeHeldMs()

```
float daisy::Encoder::TimeHeldMs ( ) const  [inline]
```

Returns the time in milliseconds that the encoder has been held down.

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

- src/hid_encoder.h

8.24 FontDef Struct Reference

Public Attributes

- const uint8_t FontWidth
- uint8_t FontHeight
- const uint16_t ∗ data

8.24.1 Member Data Documentation

8.24.1.1 data

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

Pointer to data font data array

8.24.1.2 FontHeight

```
uint8_t FontDef::FontHeight
```

Font height in pixels

8.24.1.3 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

8.25 daisy::GateIn Class Reference

Generic Class for handling gate inputs through GPIO.

```
#include <hid_gatein.h>
```

Public Member Functions

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

8.25.1 Detailed Description

Generic Class for handling gate inputs through GPIO.

Author

Stephen Hensley

Date

March 2020

8.25.2 Constructor & Destructor Documentation

8.25.2.1 GateIn()

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

GateIn Constructor

8.25.2.2 ∼GateIn()

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

GateIn∼ Destructor

8.25.3 Member Function Documentation

8.25.3.1 Init()

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

Init Initializes the gate input with specified hardware pin

8.25.3.2 Trig()

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

Trig Checks current state of gate input.

Returns

FALSE if pin is low, and TRUE if high

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

- src/hid_gatein.h

8.26 daisy::Led Class Reference

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

```
#include <hid_led.h>
```

Public Member Functions

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

8.26.1 Detailed Description

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

Author

shensley

Date

March 2020

8.26.2 Member Function Documentation

8.26.2.1 Init()

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

Initializes an LED using the specified hardware pin.

Parameters

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

8.26.2.2 Set()

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

Sets the brightness of the Led.

Parameters

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

8.26.2.3 Update()

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

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

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

- src/hid_led.h

8.27 daisy::MidiEvent Struct Reference

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

Public Member Functions

- NoteOnEvent AsNoteOn ()
- ControlChangeEvent AsControlChange ()

Public Attributes

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

8.27.1 Detailed Description

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

8.27.2 Member Function Documentation

8.27.2.1 AsControlChange()

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

Returns the data within the MidiEvent as a NoteOnEvent struct.

8.27.2.2 AsNoteOn()

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

Returns the data within the MidiEvent as a NoteOnEvent struct

8.27.3 Member Data Documentation

8.27.3.1 channel

`int daisy::MidiEvent::channel`

8.27.4 autotoc_md193

8.27.4.1 data

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

8.27.5 autotoc_md194

8.27.5.1 type

`MidiMessageType daisy::MidiEvent::type`

8.27.6 autotoc_md192

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

- src/hid_midi.h

8.28 daisy::MidiHandler Class Reference

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

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

Public Types

- enum MidiInputMode { INPUT_MODE_NONE = 0x00, INPUT_MODE_UART1 = 0x01, INPUT_MODE_US↵ B_INT = 0x02, INPUT_MODE_USB_EXT = 0x04 }
- enum MidiOutputMode { OUTPUT_MODE_NONE = 0x00, OUTPUT_MODE_UART1 = 0x01, OUTPUT_M↵ ODE_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 ()

8.28.1 Detailed Description

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

Author

shensley

Date

March 2020

8.28.2 Member Enumeration Documentation

8.28.2.1 MidiInputMode

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

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

Enumerator

INPUT_MODE_NONE	8.28.3 autotoc_md195
INPUT_MODE_UART1	8.28.4 autotoc_md196
INPUT_MODE_USB_INT	8.28.5 autotoc_md197
INPUT_MODE_USB_EXT	8.28.6 autotoc_md198

8.28.6.1 MidiOutputMode

```
enum daisy::MidiHandler::MidiOutputMode
```

Output mode

Enumerator

OUTPUT_MODE_NONE	8.28.7 autotoc_md199
OUTPUT_MODE_UART1	8.28.8 autotoc_md200
OUTPUT_MODE_USB_INT	8.28.9 autotoc_md201
OUTPUT_MODE_USB_EXT	8.28.10 autotoc_md202

8.28.11 Member Function Documentation

8.28.11.1 HasEvents()

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

Checks if there are unhandled messages in the queue

Returns

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

8.28.11.2 Init()

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

Initializes the MidiHandler

Parameters

in_mode	Input mode
out_mode	Output mode

8.28.11.3 Listen()

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

Start listening

8.28.11.4 Parse()

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

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

Parameters

byte	#

8.28.11.5 PopEvent()

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

Pops the oldest unhandled MidiEvent from the internal queue

Returns

The event to be handled

8.28.11.6 StartReceive()

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

Starts listening on the selected input mode(s). MidiEvent Queue will begin to fill, and can be checked with

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

- src/hid_midi.h

8.29 daisy::NoteOnEvent Struct Reference

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

Public Attributes

- int channel
- uint8_t note
- uint8_t velocity

8.29.1 Detailed Description

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

8.29.2 Member Data Documentation

8.29.2.1 channel

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

8.29.3 autotoc_md186

8.29.3.1 note

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

8.29.4 autotoc_md187

8.29.4.1 velocity

```
uint8_t daisy::NoteOnEvent::velocity
```

8.29.5 autotoc_md188

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

- src/hid_midi.h

8.30 daisy::OledDisplay Class Reference

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

Public Types

- enum Pins { DATA_COMMAND, RESET, NUM_PINS }

Public Member Functions

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

8.30.1 Detailed Description

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

8.30.2 Member Enumeration Documentation

8.30.2.1 Pins

```
enum daisy::OledDisplay::Pins
```

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

Enumerator

DATA_COMMAND	Data command pin.
RESET	Reset pin
NUM_PINS	Num pins

8.30.3 Member Function Documentation

8.30.3.1 DrawPixel()

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

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

Parameters

x	x Coordinate
y	y coordinate
on	on or off

8.30.3.2 Fill()

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

Fills the entire display with either on/off.

Parameters

on	Sets on or off.

8.30.3.3 Init()

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

Takes an argument for the pin cfg

Parameters

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

8.30.3.4 SetCursor()

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

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

Parameters

x	x pos
y	y pos

8.30.3.5 Update()

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

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

8.30.3.6 WriteChar()

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

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

Parameters

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

Returns

#

8.30.3.7 WriteString()

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

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

Parameters

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

Returns

#

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

- src/hid_oled_display.h

8.31 daisy::Parameter Class Reference

```
#include <hid_parameter.h>
```

Public Types

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

Public Member Functions

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

8.31.1 Detailed Description

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

8.31.2 Member Enumeration Documentation

8.31.2.1 Curve

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

Curves are applied to the output signal

Enumerator

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

8.31.3 Constructor & Destructor Documentation

8.31.3.1 Parameter()

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

Constructor

8.31.3.2 ∼Parameter()

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

Destructor

8.31.4 Member Function Documentation

8.31.4.1 Init()

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

initialize a parameter using an hid_ctrl object.

Parameters

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

8.31.4.2 Process()

```
float daisy::Parameter::Process ( )
```

processes the input signal, this should be called at the samplerate of the hid_ctrl passed in.

Returns

a float with the specified transformation applied.

8.31.4.3 Value()

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

Returns

the current value from the parameter without processing another sample. this is useful if you need to use the value multiple times, and don't store the output of process in a local variable.

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

- src/hid_parameter.h

8.32 daisy::RgbLed Class Reference

```
#include <hid_rgb_led.h>
```

Public Member Functions

- void Init (dsy_gpio_pin red, dsy_gpio_pin green, dsy_gpio_pin blue, bool invert)
- void Set (float r, float g, float b)
- void SetColor (Color c)
- void Update ()

8.32.1 Detailed Description

3x LEDs configured as an RGB for ease of use.

8.32.2 Member Function Documentation

8.32.2.1 Init()

```
void daisy::RgbLed::Init (
            dsy_gpio_pin red,
            dsy_gpio_pin green,
            dsy_gpio_pin blue,
            bool invert )
```

Initializes 3x GPIO Pins as red, green, and blue elements of an RGB LED

Parameters

red	Red element
green	Green element
blue	Blue element
invert	Flips led polarity

8.32.2.2 Set()

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

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

Parameters

r	Red element
g	Green element
b	Blue element

8.32.2.3 SetColor()

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

Sets the RGB using a Color object.

Parameters

c	Color object to set.

8.32.2.4 Update()

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

Updates the PWM of the LED based on the current values. Should be called at a regular interval. (i.e. 1kHz/1ms)

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

- src/hid_rgb_led.h

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

Public Member Functions

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

8.33.1 Member Function Documentation

8.33.1.1 capacity()

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

Returns the total size of the ring buffer

8.33.1.2 Flush()

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

Flushes unread elements from the ring buffer

8.33.1.3 ImmediateRead() [1/2]

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

Reads next element from ring buffer immediately

8.33.1.4 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

8.33.1.5 Init()

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

Initializes the Ring Buffer

8.33.1.6 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.

8.33.1.7 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.

8.33.1.8 Read()

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

Reads the first available element from the ring buffer

8.33.1.9 readable()

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

Returns number of unread elements in ring buffer

8.33.1.10 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.

8.33.1.11 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.

8.33.1.12 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

8.34 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

8.35 daisy::SdmmcHandler Class Reference

Public Member Functions

- void Init ()

8.35.1 Member Function Documentation

8.35.1.1 Init()

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

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

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

- src/per_sdmmc.h

8.36 daisy::SdmmcHandlerInit Struct Reference

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

Public Attributes

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

8.36.1 Detailed Description

Structure for setting the options above.

Used to intiailize SdmmcHandler

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

- src/per_sdmmc.h

8.37 ShiftRegister595 Class Reference

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

```
#include <dev_sr_595.h>
```

Public Types

- enum Pins { **PIN_LATCH**, PIN_CLK, PIN_DATA, NUM_PINS }

Public Member Functions

- void Init (dsy_gpio_pin *pin_cfg, size_t num_daisy_chained=1)
- void Set (uint8_t idx, bool state)
- void Write ()

8.37.1 Detailed Description

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

Author

shensley

Date

May 2020

8.37.2 Member Enumeration Documentation

8.37.2.1 Pins

```
enum ShiftRegister595::Pins
```

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

Enumerator

PIN_CLK	LATCH corresonds to Pin 12 "RCLK"
PIN_DATA	CLK corresponds to Pin 11 "SRCLK"
NUM_PINS	DATA corresponds to Pin 14 "SER"

8.37.3 Member Function Documentation

8.37.3.1 Init()

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

Initializes the GPIO, and data for the ShiftRegister

Parameters

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

8.37.3.2 Set()

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

Sets the state of the specified output.

Parameters

idx	The index starts with QA on the first device and ends with QH on the last device.
state	A true state will set the output HIGH, while a false state will set the output LOW.

8.37.3.3 Write()

```
void ShiftRegister595::Write ( )
```

Writes the states of shift register out to the connected devices.

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

- src/dev_sr_595.h

8.38 daisy::SpiHandle Class Reference

```
#include <per_spi.h>
```

Public Member Functions

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

8.38.1 Detailed Description

Handler for serial peripheral interface

8.38.2 Member Function Documentation

8.38.2.1 BlockingTransmit()

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

Blocking transmit

Parameters

∗buff	input buffer
size	buffer size

8.38.2.2 Init()

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

Initializes handler

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

- src/per_spi.h

8.39 daisy::Switch Class Reference

```
#include <hid_switch.h>
```

Public Types

- enum Type { TYPE_TOGGLE, TYPE_MOMENTARY }
- enum Polarity { POLARITY_NORMAL, POLARITY_INVERTED }
- enum Pull { PULL_UP, PULL_DOWN, PULL_NONE }

Public Member Functions

- void Init (dsy_gpio_pin pin, float update_rate, Type t, Polarity pol, Pull pu)
- void Init (dsy_gpio_pin pin, float update_rate) void Debounce()
- bool RisingEdge () const
- bool FallingEdge () const
- bool Pressed () const
- float TimeHeldMs () const

8.39.1 Detailed Description

Generic Class for handling momentary/latching switches
Inspired/influenced by Mutable Instruments (pichenettes) Switch classes

Author

Stephen Hensley

Date

December 2019

8.39.2 Member Enumeration Documentation

8.39.2.1 Polarity

`enum daisy::Switch::Polarity`

Specifies whether the pressed is HIGH or LOW.

Enumerator

POLARITY_NORMAL	**8.39.3 autotoc_md205**
POLARITY_INVERTED	**8.39.4 autotoc_md206**

8.39.4.1 Pull

```
enum daisy::Switch::Pull
```

Specifies whether to use built-in Pull Up/Down resistors to hold button at a given state when not engaged.

Enumerator

PULL_UP	**8.39.5 autotoc_md207**
PULL_DOWN	**8.39.6 autotoc_md208**
PULL_NONE	**8.39.7 autotoc_md209**

8.39.7.1 Type

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

Specifies the expected behavior of the switch

Enumerator

TYPE_TOGGLE	**8.39.8 autotoc_md203**
TYPE_MOMENTARY	**8.39.9 autotoc_md204**

8.39.10 Member Function Documentation

8.39.10.1 FallingEdge()

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

Returns

true if the button was just released

8.39.10.2 Init() [1/2]

```
void daisy::Switch::Init (
            dsy_gpio_pin pin,
            float update_rate,
            Type t,
            Polarity pol,
            Pull pu )
```

Initializes the switch object with a given port/pin combo.

Parameters

pin	port/pin object to tell the switch which hardware pin to use.
update_rate	the rate at which the Debounce() function will be called. (used for timing).
t	switch type – Default: TYPE_MOMENTARY
pol	switch polarity – Default: POLARITY_INVERTED
pu	switch pull up/down – Default: PULL_UP

8.39.10.3 Init() [2/2]

```
void daisy::Switch::Init (
            dsy_gpio_pin pin,
            float update_rate )
```

Simplified Init.

Parameters

pin	port/pin object to tell the switch which hardware pin to use.
update_rate	the rate at which the Debounce() function will be called. (used for timing). Called at update_rate to debounce and handle timing for the switch. In order for events not to be missed, its important that the Edge/Pressed checks be made at the same rate as the debounce function is being called.

8.39.10.4 Pressed()

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

Returns

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

8.39.10.5 RisingEdge()

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

Returns

true if a button was just pressed.

8.39.10.6 TimeHeldMs()

```
float daisy::Switch::TimeHeldMs ( ) const  [inline]
```

Returns

the time in milliseconds that the button has been held (or toggle has been on)

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

- src/hid_switch.h

8.40 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 ()

8.40.1 Member Function Documentation

8.40.1.1 CheckError()

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

Returns the result of HAL_UART_GetError() to the user.

8.40.1.2 FlushRx()

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

Flushes the Receive Queue

8.40.1.3 Init()

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

Initializes the UART Peripheral

8.40.1.4 PollReceive()

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

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

8.40.1.5 PollTx()

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

Sends an amount of data in blocking mode.

8.40.1.6 PopRx()

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

Pops the oldest byte from the FIFO.

8.40.1.7 Readable()

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

Checks if there are any unread bytes in the FIFO

8.40.1.8 RxActive()

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

Returns whether Rx DMA is listening or not.

8.40.1.9 StartRx()

```
int daisy::UartHandler::StartRx (
             size_t size )
```

Starts a DMA Receive callback to fill a buffer of specified size. Data is populated into a FIFO queue, and can be queried with the functions below. Maximum Buffer size is defined above. If a value outside of the maximum is specified, the size will be set to the maximum.

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

- src/per_uart.h

8.41 daisy::UsbHandle Class Reference

Interface for initializing and using the USB Peripherals on the daisy.

```
#include <hid_usb.h>
```

Public Types

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

8.41.1 Detailed Description

Interface for initializing and using the USB Peripherals on the daisy.

Author

Stephen Hensley

Date

December 2019

8.41.2 Member Typedef Documentation

8.41.2.1 ReceiveCallback

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

Function called upon reception of a buffer

8.41.3 Member Enumeration Documentation

8.41.3.1 UsbPeriph

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

Specified which of the two USB Peripherals to initialize.

Enumerator

FS_INTERNAL	Internal pin
FS_EXTERNAL	FS External D+ pin is Pin 38 (GPIO32). FS External D- pin is Pin 37 (GPIO31)
FS_BOTH	Both

8.41.4 Member Function Documentation

8.41.4.1 Init()

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

Initializes the specified peripheral(s) as USB CDC Devices

Parameters

dev	Device to initialize

8.41.4.2 SetReceiveCallback()

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

sets the callback to be called upon reception of new data

Parameters

cb	Function to serve as callback

8.41.4.3 TransmitExternal()

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

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

Parameters

buff	Buffer to transmit
size	Buffer size

8.41.4.4 TransmitInternal()

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

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

Parameters

buff	Buffer to transmit
size	Buffer size

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

- src/hid_usb.h

8.42 WAV_FormatTypeDef Struct Reference

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

8.43 daisy::WavFileInfo Struct Reference

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

Public Attributes

- WAV_FormatTypeDef raw_data
- char name [256]

8.43.1 Detailed Description

Struct containing details of Wav File.

8.43.2 Member Data Documentation

8.43.2.1 name

```
char daisy::WavFileInfo::name[256]
```

Wav filename

8.43.2.2 raw_data

```
WAV_FormatTypeDef daisy::WavFileInfo::raw_data
```

Raw wav data

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

- src/hid_wavplayer.h

8.44 daisy::WavPlayer Class Reference

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

Public Member Functions

- void Init ()
- int Open (size_t sel)
- int Close ()
- int16_t Stream ()
- void Prepare ()
- void Restart ()
- void SetLooping (bool loop)
- bool GetLooping () const
- size_t GetNumberFiles () const
- size_t GetCurrentFile () const

8.44.1 Detailed Description

Wav Player that will load .wav files from an SD Card, and then provide a method of accessing the samples with double-buffering.

8.44.2 Member Function Documentation

8.44.2.1 Close()

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

Closes whatever file is currently open.

Returns

#

8.44.2.2 GetCurrentFile()

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

Returns

currently selected file.

8.44.2.3 GetLooping()

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

Returns

Whether the WavPlayer is looping or not.

8.44.2.4 GetNumberFiles()

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

Returns

The number of files loaded by the WavPlayer

8.44.2.5 Init()

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

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

8.44.2.6 Open()

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

Opens the file at index sel for reading.

Parameters

sel	File to open

8.44.2.7 Prepare()

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

Collects buffer for playback when needed.

8.44.2.8 Restart()

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

Resets the playback position to the beginning of the file immediately

8.44.2.9 SetLooping()

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

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

Parameters

loop	To loop or not to loop.

8.44.2.10 Stream()

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

Returns

The next sample if playing, otherwise returns 0

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

- src/hid_wavplayer.h

Chapter 9

File Documentation

9.1 src/daisy.h File Reference

```
#include <stdint.h>
#include "daisy_core.h"
#include "sys_system.h"
#include "per_qspi.h"
#include "per_dac.h"
#include "per_gpio.h"
#include "per_i2c.h"
#include "per_sai.h"
#include "per_tim.h"
#include "dev_leddriver.h"
#include "dev_sdram.h"
#include "dev_sr_4021.h"
#include "hid_audio.h"
#include "util_unique_id.h"
#include "per_adc.h"
#include "per_uart.h"
#include "hid_midi.h"
#include "hid_encoder.h"
#include "hid_switch.h"
#include "hid_ctrl.h"
#include "hid_gatein.h"
#include "hid_parameter.h"
#include "hid_usb.h"
#include "per_sdmmc.h"
#include "per_spi.h"
#include "hid_oled_display.h"
#include "hid_wavplayer.h"
#include "hid_led.h"
#include "hid_rgb_led.h"
#include "dev_sr_595.h"
```

Macros

- #define FBIPMAX 0.999985f
- #define FBIPMIN (-FBIPMAX)

- #define S162F_SCALE 3.0517578125e-05f
- #define F2S16_SCALE 32767.0f
- #define F2S24_SCALE 8388608.0f
- #define S242F_SCALE 1.192092896e-07f
- #define S24SIGN 0x800000

Functions

- FORCE_INLINE float s162f (int16_t x)
- FORCE_INLINE int16_t f2s16 (float x)
- FORCE_INLINE float s242f (int32_t x)
- FORCE_INLINE int32_t f2s24 (float x)

9.1.1 Macro Definition Documentation

9.1.1.1 F2S16_SCALE

```
#define F2S16_SCALE 32767.0f
```

(2 ** 15) - 1

9.1.1.2 F2S24_SCALE

```
#define F2S24_SCALE 8388608.0f
```

2 ** 23

9.1.1.3 FBIPMAX

```
#define FBIPMAX 0.999985f
```

close to 1.0f-LSB at 16 bit

9.1.1.4 FBIPMIN

```
#define FBIPMIN (-FBIPMAX)
```

- (1 - LSB)

9.1.1.5 S162F_SCALE

```
#define S162F_SCALE 3.0517578125e-05f
```

1 / (2$**$ 15)

9.1.1.6 S242F_SCALE

```
#define S242F_SCALE 1.192092896e-07f
```

1 / (2 $**$ 23)

9.1.1.7 S24SIGN

```
#define S24SIGN 0x800000
```

2 $**$ 23

9.1.2 Function Documentation

9.1.2.1 f2s16()

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

$<$ close to 1.0f-LSB at 16 bit

$<$ - (1 - LSB)

$<$ close to 1.0f-LSB at 16 bit

$<$ - (1 - LSB)

$<$ close to 1.0f-LSB at 16 bit

$<$ close to 1.0f-LSB at 16 bit

$<$ (2 $**$ 15) - 1

9.1.2.2 f2s24()

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

$<$ close to 1.0f-LSB at 16 bit

$<$ - (1 - LSB)

$<$ close to 1.0f-LSB at 16 bit

$<$ - (1 - LSB)

$<$ close to 1.0f-LSB at 16 bit

$<$ close to 1.0f-LSB at 16 bit

$<$ 2 $**$ 23

9.1.2.3 s162f()

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

Scales float by 1/(2 ^ 15)

Parameters

x	Number to be scaled.

Returns

Scaled number.

< 1 / (2$**$ 15)

9.1.2.4 s242f()

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

$< 2 ** 23$

$< 2 ** 23$

< 1 / (2 $**$ 23)

9.2 src/daisy_core.h File Reference

```
#include <stdint.h>
#include <stdlib.h>
```

Classes

- struct dsy_gpio_pin

Macros

- #define DSY_CORE_HW_H
- #define DMA_BUFFER_MEM_SECTION __attribute__((section(".sram1_bss")))
- #define DTCM_MEM_SECTION __attribute__((section(".dtcmram_bss")))

Enumerations

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

Functions

- FORCE_INLINE float cube (float x)
- FORCE_INLINE dsy_gpio_pin dsy_pin (dsy_gpio_port port, uint8_t pin)
- FORCE_INLINE uint8_t dsy_pin_cmp (dsy_gpio_pin ∗a, dsy_gpio_pin ∗b)

9.2.1 Macro Definition Documentation

9.2.1.1 DMA_BUFFER_MEM_SECTION

```
#define DMA_BUFFER_MEM_SECTION __attribute__((section(".sram1_bss")))
```

Macro for area of memory that is configured as cacheless This should be used primarily for DMA buffers, and the like.

9.2.1.2 DSY_CORE_HW_H

```
#define DSY_CORE_HW_H
```

9.2.2 autotoc_md8

9.2.2.1 DTCM_MEM_SECTION

```
#define DTCM_MEM_SECTION __attribute__((section(".dtcmram_bss")))
```

THE DTCM RAM section is also non-cached. However, is not suitable for DMA transfers. Performance is on par with internal SRAM w/ cache enabled.

9.2.3 Enumeration Type Documentation

9.2.3.1 dsy_gpio_port

```
enum dsy_gpio_port
```

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

Enumerator

DSY_GPIOA	**9.2.4 autotoc_md9**
DSY_GPIOB	**9.2.5 autotoc_md10**
DSY_GPIOC	**9.2.6 autotoc_md11**
DSY_GPIOD	**9.2.7 autotoc_md12**
DSY_GPIOE	**9.2.8 autotoc_md13**
DSY_GPIOF	**9.2.9 autotoc_md14**
DSY_GPIOG	**9.2.10 autotoc_md15**
DSY_GPIOH	**9.2.11 autotoc_md16**
DSY_GPIOI	**9.2.12 autotoc_md17**
DSY_GPIOJ	**9.2.13 autotoc_md18**
DSY_GPIOK	**9.2.14 autotoc_md19**
DSY_GPIO_LAST	This is a non-existant port for unsupported bits of hardware.

9.2.15 Function Documentation

9.2.15.1 cube()

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

Computes cube.

Parameters

x	Number to be cubed

Returns

x ^ 3

9.2.15.2 dsy_pin()

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

Helper for creating pins from port/pin combos easily

9.2.15.3 dsy_pin_cmp()

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

Helper for testing sameness of two dsy_gpio_pins

Returns

1 if same, 0 if different

9.3 src/daisy_field.h File Reference

Hardware defines and helpers for daisy field platform.

```
#include "daisy_seed.h"
```

Classes

- struct daisy::daisy_field

Namespaces

- daisy

Macros

- #define DSY_FIELD_BSP_H
- #define SAMPLE_RATE DSY_AUDIO_SAMPLE_RATE
- #define SW_1_PIN 29
- #define SW_2_PIN 28
- #define SW_3_PIN 27
- #define GATE_OUT_PIN 0
- #define GATE_IN_PIN 1
- #define KB_SW_SR_CS_PIN 8
- #define KB_SW_SR_CLK_PIN 9
- #define KB_SW_SR_D1_PIN 10
- #define KB_SW_SR_D2_PIN 11
- #define MIDI_OUT_PIN 14
- #define MIDI_IN_PIN 15
- #define MUX_SEL_0_PIN 21
- #define MUX_SEL_1_PIN 20
- #define MUX_SEL_2_PIN 19
- #define MUX_ADC_PIN 16
- #define CV1_ADC_PIN 17
- #define CV2_ADC_PIN 18
- #define CV3_ADC_PIN 23
- #define CV4_ADC_PIN 22
- #define LED_DRIVER_I2C i2c1_handle

Enumerations

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

Functions

- FORCE_INLINE void daisy::daisy_field_init (daisy_field ∗p)

9.3.1 Detailed Description

Hardware defines and helpers for daisy field platform.

9.3.2 Macro Definition Documentation

9.3.2.1 CV1_ADC_PIN

```
#define CV1_ADC_PIN 17
```

9.3.3 autotoc_md38

9.3.3.1 CV2_ADC_PIN

```
#define CV2_ADC_PIN 18
```

9.3.4 autotoc_md39

9.3.4.1 CV3_ADC_PIN

```
#define CV3_ADC_PIN 23
```

9.3.5 autotoc_md40

9.3.5.1 CV4_ADC_PIN

```
#define CV4_ADC_PIN 22
```

9.3.6 autotoc_md41

9.3.6.1 DSY_FIELD_BSP_H

```
#define DSY_FIELD_BSP_H
```

9.3.7 autotoc_md21

9.3.7.1 GATE_IN_PIN

```
#define GATE_IN_PIN 1
```

9.3.8 autotoc_md27

9.3.8.1 GATE_OUT_PIN

```
#define GATE_OUT_PIN 0
```

9.3.9 autotoc_md26

9.3.9.1 KB_SW_SR_CLK_PIN

```
#define KB_SW_SR_CLK_PIN 9
```

9.3.10 autotoc_md29

9.3.10.1 KB_SW_SR_CS_PIN

```
#define KB_SW_SR_CS_PIN 8
```

9.3.11 autotoc_md28

9.3.11.1 KB_SW_SR_D1_PIN

```
#define KB_SW_SR_D1_PIN 10
```

9.3.12 autotoc_md30

9.3.12.1 KB_SW_SR_D2_PIN

```
#define KB_SW_SR_D2_PIN 11
```

9.3.13 autotoc_md31

9.3.13.1 LED_DRIVER_I2C

```
#define LED_DRIVER_I2C i2c1_handle
```

9.3.14 autotoc_md42

9.3.14.1 MIDI_IN_PIN

```
#define MIDI_IN_PIN 15
```

9.3.15 autotoc_md33

9.3.15.1 MIDI_OUT_PIN

```
#define MIDI_OUT_PIN 14
```

9.3.16 autotoc_md32

9.3.16.1 MUX_ADC_PIN

```
#define MUX_ADC_PIN 16
```

9.3.17 autotoc_md37

9.3.17.1 MUX_SEL_0_PIN

```
#define MUX_SEL_0_PIN 21
```

9.3.18 autotoc_md34

9.3.18.1 MUX_SEL_1_PIN

```
#define MUX_SEL_1_PIN 20
```

9.3.19 autotoc_md35

9.3.19.1 MUX_SEL_2_PIN

```
#define MUX_SEL_2_PIN 19
```

9.3.20 autotoc_md36

9.3.20.1 SAMPLE_RATE

```
#define SAMPLE_RATE DSY_AUDIO_SAMPLE_RATE
```

9.3.21 autotoc_md22

9.3.21.1 SW_1_PIN

```
#define SW_1_PIN 29
```

9.3.22 autotoc_md23

9.3.22.1 SW_2_PIN

```
#define SW_2_PIN 28
```

9.3.23 autotoc_md24

9.3.23.1 SW_3_PIN

```
#define SW_3_PIN 27
```

9.3.24 autotoc_md25

9.4 src/daisy_patch.h File Reference

```
#include "daisy_seed.h"
```

Classes

- class daisy::DaisyPatch

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

Namespaces

- daisy

9.5 src/daisy_petal.h File Reference

```
#include "daisy_seed.h"
```

Classes

- class daisy::DaisyPetal

  Helpers and hardware definitions for daisy petal.

Namespaces

- daisy

Macros

- #define DSY_PETAL_H

9.5.1 Macro Definition Documentation

9.5.1.1 DSY_PETAL_H

```
#define DSY_PETAL_H
```

9.5.2 autotoc_md82

9.6 src/daisy_pod.h File Reference

```
#include "daisy_seed.h"
```

Classes

- class daisy::DaisyPod

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

Namespaces

- daisy

9.7 src/daisy_seed.h File Reference

```
#include "daisy.h"
```

Classes

- class daisy::DaisySeed

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

Namespaces

- daisy

9.8 src/dev_codec_ak4556.h File Reference

Driver for the AK4556 Stereo Codec.

```
#include "daisy_core.h"
```

Functions

- void codec_ak4556_init (dsy_gpio_pin reset_pin)

9.8.1 Detailed Description

Driver for the AK4556 Stereo Codec.

9.8.2 Function Documentation

9.8.2.1 codec_ak4556_init()

```
void codec_ak4556_init (
            dsy_gpio_pin reset_pin )
```

Resets the AK4556

Parameters

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

9.9 src/dev_codec_pcm3060.h File Reference

Driver for the PCM3060 Codec.

```
#include "per_i2c.h"
```

Functions

- void codec_pcm3060_init (dsy_i2c_handle ∗hi2c)

9.9.1 Detailed Description

Driver for the PCM3060 Codec.

9.9.2 Function Documentation

9.9.2.1 codec_pcm3060_init()

```
void codec_pcm3060_init (
            dsy_i2c_handle ∗ hi2c )
```

Resets the PCM060

Parameters

∗*hi2c*	array of pins handling i2c?

9.10 src/dev_codec_wm8731.h File Reference

Driver for the WM8731 Codec.

```
#include <stddef.h>
#include "per_sai.h"
#include "per_i2c.h"
```

Functions

- uint8_t codec_wm8731_init (dsy_i2c_handle ∗hi2c, uint8_t mcu_is_master, int32_t sample_rate, uint8_↩
  t bitdepth)
- uint8_t codec_wm8731_enter_bypass (dsy_i2c_handle ∗hi2c)
- uint8_t codec_wm8731_exit_bypass (dsy_i2c_handle ∗hi2c)

9.10.1 Detailed Description

Driver for the WM8731 Codec.

9.10.2 Function Documentation

9.10.2.1 codec_wm8731_enter_bypass()

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

Put codec into bypass mode

Parameters

∗*hi2c*	pins handling i2c

9.10.2.2 codec_wm8731_exit_bypass()

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

Take codec out of bypass mode

Parameters

∗*hi2c*	pins handling i2c

9.10.2.3 codec_wm8731_init()

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

Resets the WM8731

Parameters

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

9.11 src/dev_codec_wm8731_frame.h File Reference

WM8731 Codec framework.

```
#include <stddef.h>
```

Classes

- struct codec_frame_t

Typedefs

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

9.11.1 Detailed Description

WM8731 Codec framework.

9.11.2 Typedef Documentation

9.11.2.1 sa_audio_callback

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

9.11.3 autotoc_md138

9.12 src/dev_flash_IS25LP064A.h File Reference

IS25LP08D Commands.

Macros

- #define **IS25LP064A_FLASH_SIZE** 0x800000 /∗ 2 ∗ 8 MBits => 1 ∗ 1MBytes => 1MBytes∗/
- #define **IS25LP064A_SECTOR_SIZE** 0x10000 /∗ 2 ∗ 1024 sectors of 64KBytes ∗/
- #define **IS25LP064A_SUBSECTOR_SIZE** 0x1000 /∗ 2 ∗ 16384 subsectors of 4kBytes ∗/
- #define **IS25LP064A_PAGE_SIZE** 0x100 /∗ 2 ∗ 262144 pages of 256 bytes ∗/
- #define **IS25LP064A_DUMMY_CYCLES_READ_QUAD** 8
- #define **IS25LP064A_DUMMY_CYCLES_READ** 8
- #define **IS25LP064A_DUMMY_CYCLES_READ_DTR** 6
- #define **IS25LP064A_DUMMY_CYCLES_READ_QUAD_DTR** 6
- #define **IS25LP064A_DIE_ERASE_MAX_TIME** 460000
- #define **IS25LP064A_SECTOR_ERASE_MAX_TIME** 1000
- #define **IS25LP064A_SUBSECTOR_ERASE_MAX_TIME** 400
- #define **RESET_ENABLE_CMD** 0x66
- #define **RESET_MEMORY_CMD** 0x99
- #define **READ_ID_CMD** 0x9E
- #define **READ_ID_CMD2** 0x9F
- #define **MULTIPLE_IO_READ_ID_CMD** 0xAF
- #define **READ_SERIAL_FLASH_DISCO_PARAM_CMD** 0x5A
- #define **READ_CMD** 0x03
- #define **READ_4_BYTE_ADDR_CMD** 0x13
- #define **FAST_READ_CMD** 0x0B
- #define **FAST_READ_DTR_CMD** 0x0D
- #define **FAST_READ_4_BYTE_ADDR_CMD** 0x0C
- #define **DUAL_OUT_FAST_READ_CMD** 0x3B
- #define **DUAL_OUT_FAST_READ_DTR_CMD** 0x3D
- #define **DUAL_OUT_FAST_READ_4_BYTE_ADDR_CMD** 0x3C
- #define **DUAL_INOUT_FAST_READ_CMD** 0xBB
- #define **DUAL_INOUT_FAST_READ_DTR_CMD** 0xBD
- #define **DUAL_INOUT_FAST_READ_4_BYTE_ADDR_CMD** 0xBC
- #define **QUAD_OUT_FAST_READ_CMD** 0x6B
- #define **QUAD_OUT_FAST_READ_DTR_CMD** 0x0D
- #define **QUAD_OUT_FAST_READ_4_BYTE_ADDR_CMD** 0x6C
- #define **QUAD_INOUT_FAST_READ_CMD** 0xEB
- #define **QUAD_INOUT_FAST_READ_DTR_CMD** 0xED
- #define **QUAD_INOUT_FAST_READ_4_BYTE_ADDR_CMD** 0xEC
- #define **WRITE_ENABLE_CMD** 0x06
- #define **WRITE_DISABLE_CMD** 0x04
- #define **READ_STATUS_REG_CMD** 0x05
- #define **WRITE_STATUS_REG_CMD** 0x01
- #define **READ_LOCK_REG_CMD** 0xE8
- #define **WRITE_LOCK_REG_CMD** 0xE5
- #define **READ_FLAG_STATUS_REG_CMD** 0x70
- #define **CLEAR_FLAG_STATUS_REG_CMD** 0x50
- #define **READ_NONVOL_CFG_REG_CMD** 0xB5
- #define **WRITE_NONVOL_CFG_REG_CMD** 0xB1
- #define **READ_READ_PARAM_REG_CMD** 0x61
- #define **WRITE_READ_PARAM_REG_CMD** 0xC0
- #define **READ_ENHANCED_VOL_CFG_REG_CMD** 0x81
- #define **WRITE_ENHANCED_VOL_CFG_REG_CMD** 0x85
- #define **READ_EXT_ADDR_REG_CMD** 0xC8
- #define **WRITE_EXT_ADDR_REG_CMD** 0xC5
- #define **PAGE_PROG_CMD** 0x02
- #define **PAGE_PROG_4_BYTE_ADDR_CMD** 0x12
- #define **DUAL_IN_FAST_PROG_CMD** 0xA2

- #define **EXT_DUAL_IN_FAST_PROG_CMD** 0xD2
- #define **QUAD_IN_FAST_PROG_CMD** 0x32
- #define **EXT_QUAD_IN_FAST_PROG_CMD** 0x38
- #define **QUAD_IN_FAST_PROG_4_BYTE_ADDR_CMD** 0x34
- #define **SUBSECTOR_ERASE_CMD** 0xd7
- #define **SUBSECTOR_ERASE_QPI_CMD** 0x20
- #define **SUBSECTOR_ERASE_4_BYTE_ADDR_CMD** 0x21
- #define **SECTOR_ERASE_CMD** 0xD8
- #define **SECTOR_ERASE_4_BYTE_ADDR_CMD** 0xDC
- #define **BLOCK_ERASE_32K_CMD** 0x52
- #define **DIE_ERASE_CMD** 0xC4
- #define **PROG_ERASE_RESUME_CMD** 0x7A
- #define **PROG_ERASE_SUSPEND_CMD** 0x75
- #define **READ_OTP_ARRAY_CMD** 0x4B
- #define **PROG_OTP_ARRAY_CMD** 0x42
- #define **ENTER_4_BYTE_ADDR_MODE_CMD** 0xB7
- #define **EXIT_4_BYTE_ADDR_MODE_CMD** 0xE9
- #define **ENTER_QUAD_CMD** 0x35
- #define **EXIT_QUAD_CMD** 0xF5
- #define IS25LP064A_SR_WIP ((uint8_t)0x01)

    IS25LP08D Registers.
- #define IS25LP064A_SR_WREN ((uint8_t)0x02)
- #define IS25LP064A_SR_SRWREN ((uint8_t)0x80)
- #define **IS25LP064A_SR_QE** ((uint8_t)0x40)
- #define IS25LP064A_NVCR_NBADDR ((uint16_t)0x0001)
- #define IS25LP064A_NVCR_SEGMENT ((uint16_t)0x0002)
- #define IS25LP064A_NVCR_DUAL ((uint16_t)0x0004)
- #define IS25LP064A_NVCR_QUAB ((uint16_t)0x0008)
- #define IS25LP064A_NVCR_RH ((uint16_t)0x0010)
- #define IS25LP064A_NVCR_DTRP ((uint16_t)0x0020)
- #define IS25LP064A_NVCR_ODS ((uint16_t)0x01C0)
- #define IS25LP064A_NVCR_XIP ((uint16_t)0x0E00)
- #define IS25LP064A_NVCR_NB_DUMMY ((uint16_t)0xF000)
- #define IS25LP064A_VCR_WRAP ((uint8_t)0x03)
- #define IS25LP064A_VCR_XIP ((uint8_t)0x08)
- #define IS25LP064A_VCR_NB_DUMMY ((uint8_t)0xF0)
- #define IS25LP064A_EAR_HIGHEST_SE ((uint8_t)0x03)
- #define IS25LP064A_EAR_THIRD_SEG ((uint8_t)0x02)
- #define IS25LP064A_EAR_SECOND_SEG ((uint8_t)0x01)
- #define IS25LP064A_EAR_LOWEST_SEG ((uint8_t)0x00)
- #define IS25LP064A_EVCR_ODS ((uint8_t)0x07)
- #define IS25LP064A_EVCR_RH ((uint8_t)0x10)
- #define IS25LP064A_EVCR_DTRP ((uint8_t)0x20)
- #define IS25LP064A_EVCR_DUAL ((uint8_t)0x40)
- #define IS25LP064A_EVCR_QUAD ((uint8_t)0x80)
- #define IS25LP064A_FSR_NBADDR ((uint8_t)0x01)
- #define IS25LP064A_FSR_PRERR ((uint8_t)0x02)
- #define IS25LP064A_FSR_PGSUS ((uint8_t)0x04)
- #define IS25LP064A_FSR_PGERR ((uint8_t)0x10)
- #define IS25LP064A_FSR_ERERR ((uint8_t)0x20)
- #define IS25LP064A_FSR_ERSUS ((uint8_t)0x40)
- #define IS25LP064A_FSR_READY ((uint8_t)0x80)

9.12.1 Detailed Description

IS25LP08D Commands.

9.12.2 Macro Definition Documentation

9.12.2.1 IS25LP064A_EAR_HIGHEST_SE

```
#define IS25LP064A_EAR_HIGHEST_SE ((uint8_t)0x03)
```

Select the Highest 128Mb segment

9.12.2.2 IS25LP064A_EAR_LOWEST_SEG

```
#define IS25LP064A_EAR_LOWEST_SEG ((uint8_t)0x00)
```

Select the Lowest 128Mb segment (default)

9.12.2.3 IS25LP064A_EAR_SECOND_SEG

```
#define IS25LP064A_EAR_SECOND_SEG ((uint8_t)0x01)
```

Select the Second 128Mb segment

9.12.2.4 IS25LP064A_EAR_THIRD_SEG

```
#define IS25LP064A_EAR_THIRD_SEG ((uint8_t)0x02)
```

Select the Third 128Mb segment

9.12.2.5 IS25LP064A_EVCR_DTRP

```
#define IS25LP064A_EVCR_DTRP ((uint8_t)0x20)
```

Double transfer rate protocol

9.12.2.6 IS25LP064A_EVCR_DUAL

```
#define IS25LP064A_EVCR_DUAL ((uint8_t)0x40)
```

Dual I/O protocol

9.12.2.7 IS25LP064A_EVCR_ODS

```
#define IS25LP064A_EVCR_ODS ((uint8_t)0x07)
```

Output driver strength

9.12.2.8 IS25LP064A_EVCR_QUAD

```
#define IS25LP064A_EVCR_QUAD ((uint8_t)0x80)
```

Quad I/O protocol

9.12.2.9 IS25LP064A_EVCR_RH

```
#define IS25LP064A_EVCR_RH ((uint8_t)0x10)
```

Reset/hold

9.12.2.10 IS25LP064A_FSR_ERERR

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

Erase error

9.12.2.11 IS25LP064A_FSR_ERSUS

```
#define IS25LP064A_FSR_ERSUS ((uint8_t)0x40)
```

Erase operation suspended

9.12.2.12 IS25LP064A_FSR_NBADDR

```
#define IS25LP064A_FSR_NBADDR ((uint8_t)0x01)
```

3-bytes or 4-bytes addressing

9.12.2.13 IS25LP064A_FSR_PGERR

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

Program error

9.12.2.14 IS25LP064A_FSR_PGSUS

```
#define IS25LP064A_FSR_PGSUS ((uint8_t)0x04)
```

Program operation suspended

9.12.2.15 IS25LP064A_FSR_PRERR

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

Protection error

9.12.2.16 IS25LP064A_FSR_READY

```
#define IS25LP064A_FSR_READY ((uint8_t)0x80)
```

Ready or command in progress

9.12.2.17 IS25LP064A_NVCR_DTRP

```
#define IS25LP064A_NVCR_DTRP ((uint16_t)0x0020)
```

Double transfer rate protocol

9.12.2.18 IS25LP064A_NVCR_DUAL

```
#define IS25LP064A_NVCR_DUAL ((uint16_t)0x0004)
```

Dual I/O protocol

9.12.2.19 IS25LP064A_NVCR_NB_DUMMY

```
#define IS25LP064A_NVCR_NB_DUMMY ((uint16_t)0xF000)
```

Number of dummy clock cycles

9.12.2.20 IS25LP064A_NVCR_NBADDR

```
#define IS25LP064A_NVCR_NBADDR ((uint16_t)0x0001)
```

3-bytes or 4-bytes addressing

9.12.2.21 IS25LP064A_NVCR_ODS

```
#define IS25LP064A_NVCR_ODS ((uint16_t)0x01C0)
```

Output driver strength

9.12.2.22 IS25LP064A_NVCR_QUAB

```
#define IS25LP064A_NVCR_QUAB ((uint16_t)0x0008)
```

Quad I/O protocol

9.12.2.23 IS25LP064A_NVCR_RH

```
#define IS25LP064A_NVCR_RH ((uint16_t)0x0010)
```

Reset/hold

9.12.2.24 IS25LP064A_NVCR_SEGMENT

```
#define IS25LP064A_NVCR_SEGMENT ((uint16_t)0x0002)
```

Upper or lower 128Mb segment selected by default

9.12.2.25 IS25LP064A_NVCR_XIP

```
#define IS25LP064A_NVCR_XIP ((uint16_t)0x0E00)
```

XIP mode at power-on reset

9.12.2.26 IS25LP064A_SR_SRWREN

```
#define IS25LP064A_SR_SRWREN ((uint8_t)0x80)
```

Status register write enable/disable

9.12.2.27 IS25LP064A_SR_WIP

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

IS25LP08D Registers.

Write in progress

9.12.2.28 IS25LP064A_SR_WREN

```
#define IS25LP064A_SR_WREN ((uint8_t)0x02)
```

Write enable latch

9.12.2.29 IS25LP064A_VCR_NB_DUMMY

```
#define IS25LP064A_VCR_NB_DUMMY ((uint8_t)0xF0)
```

Number of dummy clock cycles

9.12.2.30 IS25LP064A_VCR_WRAP

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

Wrap

9.12.2.31 IS25LP064A_VCR_XIP

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

XIP

9.13 src/dev_flash_IS25LP080D.h File Reference

IS25LP08D Commands.

Macros

- #define **IS25LP080D_FLASH_SIZE** 0x100000 /∗ 2 ∗ 8 MBits => 1 ∗ 1MBytes => 1MBytes∗/
- #define **IS25LP080D_SECTOR_SIZE** 0x10000 /∗ 2 ∗ 1024 sectors of 64KBytes ∗/
- #define **IS25LP080D_SUBSECTOR_SIZE** 0x1000 /∗ 2 ∗ 16384 subsectors of 4kBytes ∗/
- #define **IS25LP080D_PAGE_SIZE** 0x100 /∗ 2 ∗ 262144 pages of 256 bytes ∗/
- #define **IS25LP080D_DUMMY_CYCLES_READ_QUAD** 8
- #define **IS25LP080D_DUMMY_CYCLES_READ** 8
- #define **IS25LP080D_DUMMY_CYCLES_READ_DTR** 6
- #define **IS25LP080D_DUMMY_CYCLES_READ_QUAD_DTR** 6
- #define **IS25LP080D_DIE_ERASE_MAX_TIME** 460000
- #define **IS25LP080D_SECTOR_ERASE_MAX_TIME** 1000
- #define **IS25LP080D_SUBSECTOR_ERASE_MAX_TIME** 400
- #define **RESET_ENABLE_CMD** 0x66
- #define **RESET_MEMORY_CMD** 0x99
- #define **READ_ID_CMD** 0x9E
- #define **READ_ID_CMD2** 0x9F
- #define **MULTIPLE_IO_READ_ID_CMD** 0xAF
- #define **READ_SERIAL_FLASH_DISCO_PARAM_CMD** 0x5A
- #define **READ_CMD** 0x03
- #define **READ_4_BYTE_ADDR_CMD** 0x13
- #define **FAST_READ_CMD** 0x0B
- #define **FAST_READ_DTR_CMD** 0x0D
- #define **FAST_READ_4_BYTE_ADDR_CMD** 0x0C
- #define **DUAL_OUT_FAST_READ_CMD** 0x3B
- #define **DUAL_OUT_FAST_READ_DTR_CMD** 0x3D
- #define **DUAL_OUT_FAST_READ_4_BYTE_ADDR_CMD** 0x3C
- #define **DUAL_INOUT_FAST_READ_CMD** 0xBB
- #define **DUAL_INOUT_FAST_READ_DTR_CMD** 0xBD
- #define **DUAL_INOUT_FAST_READ_4_BYTE_ADDR_CMD** 0xBC
- #define **QUAD_OUT_FAST_READ_CMD** 0x6B
- #define **QUAD_OUT_FAST_READ_DTR_CMD** 0x0D
- #define **QUAD_OUT_FAST_READ_4_BYTE_ADDR_CMD** 0x6C
- #define **QUAD_INOUT_FAST_READ_CMD** 0xEB

- #define **QUAD_INOUT_FAST_READ_DTR_CMD** 0xED
- #define **QUAD_INOUT_FAST_READ_4_BYTE_ADDR_CMD** 0xEC
- #define **WRITE_ENABLE_CMD** 0x06
- #define **WRITE_DISABLE_CMD** 0x04
- #define **READ_STATUS_REG_CMD** 0x05
- #define **WRITE_STATUS_REG_CMD** 0x01
- #define **READ_LOCK_REG_CMD** 0xE8
- #define **WRITE_LOCK_REG_CMD** 0xE5
- #define **READ_FLAG_STATUS_REG_CMD** 0x70
- #define **CLEAR_FLAG_STATUS_REG_CMD** 0x50
- #define **READ_NONVOL_CFG_REG_CMD** 0xB5
- #define **WRITE_NONVOL_CFG_REG_CMD** 0xB1
- #define **READ_READ_PARAM_REG_CMD** 0x61
- #define **WRITE_READ_PARAM_REG_CMD** 0xC0
- #define **READ_ENHANCED_VOL_CFG_REG_CMD** 0x81
- #define **WRITE_ENHANCED_VOL_CFG_REG_CMD** 0x85
- #define **READ_EXT_ADDR_REG_CMD** 0xC8
- #define **WRITE_EXT_ADDR_REG_CMD** 0xC5
- #define **PAGE_PROG_CMD** 0x02
- #define **PAGE_PROG_4_BYTE_ADDR_CMD** 0x12
- #define **DUAL_IN_FAST_PROG_CMD** 0xA2
- #define **EXT_DUAL_IN_FAST_PROG_CMD** 0xD2
- #define **QUAD_IN_FAST_PROG_CMD** 0x32
- #define **EXT_QUAD_IN_FAST_PROG_CMD** 0x38
- #define **QUAD_IN_FAST_PROG_4_BYTE_ADDR_CMD** 0x34
- #define **SUBSECTOR_ERASE_CMD** 0xd7
- #define **SUBSECTOR_ERASE_QPI_CMD** 0x20
- #define **SUBSECTOR_ERASE_4_BYTE_ADDR_CMD** 0x21
- #define **SECTOR_ERASE_CMD** 0xD8
- #define **SECTOR_ERASE_4_BYTE_ADDR_CMD** 0xDC
- #define **BLOCK_ERASE_32K_CMD** 0x52
- #define **DIE_ERASE_CMD** 0xC4
- #define **PROG_ERASE_RESUME_CMD** 0x7A
- #define **PROG_ERASE_SUSPEND_CMD** 0x75
- #define **READ_OTP_ARRAY_CMD** 0x4B
- #define **PROG_OTP_ARRAY_CMD** 0x42
- #define **ENTER_4_BYTE_ADDR_MODE_CMD** 0xB7
- #define **EXIT_4_BYTE_ADDR_MODE_CMD** 0xE9
- #define **ENTER_QUAD_CMD** 0x35
- #define **EXIT_QUAD_CMD** 0xF5
- #define IS25LP080D_SR_WIP ((uint8_t)0x01)

  IS25LP08D Registers.
- #define IS25LP080D_SR_WREN ((uint8_t)0x02)
- #define IS25LP080D_SR_SRWREN ((uint8_t)0x80)
- #define **IS25LP080D_SR_QE** ((uint8_t)0x40)
- #define IS25LP080D_NVCR_NBADDR ((uint16_t)0x0001)
- #define IS25LP080D_NVCR_SEGMENT ((uint16_t)0x0002)
- #define IS25LP080D_NVCR_DUAL ((uint16_t)0x0004)
- #define IS25LP080D_NVCR_QUAB ((uint16_t)0x0008)
- #define IS25LP080D_NVCR_RH ((uint16_t)0x0010)
- #define IS25LP080D_NVCR_DTRP ((uint16_t)0x0020)
- #define IS25LP080D_NVCR_ODS ((uint16_t)0x01C0)
- #define IS25LP080D_NVCR_XIP ((uint16_t)0x0E00)
- #define IS25LP080D_NVCR_NB_DUMMY ((uint16_t)0xF000)

- #define IS25LP080D_VCR_WRAP ((uint8_t)0x03)
- #define IS25LP080D_VCR_XIP ((uint8_t)0x08)
- #define IS25LP080D_VCR_NB_DUMMY ((uint8_t)0xF0)
- #define IS25LP080D_EAR_HIGHEST_SE ((uint8_t)0x03)
- #define IS25LP080D_EAR_THIRD_SEG ((uint8_t)0x02)
- #define IS25LP080D_EAR_SECOND_SEG ((uint8_t)0x01)
- #define IS25LP080D_EAR_LOWEST_SEG ((uint8_t)0x00)
- #define IS25LP080D_EVCR_ODS ((uint8_t)0x07)
- #define IS25LP080D_EVCR_RH ((uint8_t)0x10)
- #define IS25LP080D_EVCR_DTRP ((uint8_t)0x20)
- #define IS25LP080D_EVCR_DUAL ((uint8_t)0x40)
- #define IS25LP080D_EVCR_QUAD ((uint8_t)0x80)
- #define IS25LP080D_FSR_NBADDR ((uint8_t)0x01)
- #define IS25LP080D_FSR_PRERR ((uint8_t)0x02)
- #define IS25LP080D_FSR_PGSUS ((uint8_t)0x04)
- #define IS25LP080D_FSR_PGERR ((uint8_t)0x10)
- #define IS25LP080D_FSR_ERERR ((uint8_t)0x20)
- #define IS25LP080D_FSR_ERSUS ((uint8_t)0x40)
- #define IS25LP080D_FSR_READY ((uint8_t)0x80)

9.13.1 Detailed Description

IS25LP08D Commands.

9.13.2 Macro Definition Documentation

9.13.2.1 IS25LP080D_EAR_HIGHEST_SE

```
#define IS25LP080D_EAR_HIGHEST_SE ((uint8_t)0x03)
```

Select the Highest 128Mb segment

9.13.2.2 IS25LP080D_EAR_LOWEST_SEG

```
#define IS25LP080D_EAR_LOWEST_SEG ((uint8_t)0x00)
```

Select the Lowest 128Mb segment (default)

9.13.2.3 IS25LP080D_EAR_SECOND_SEG

```
#define IS25LP080D_EAR_SECOND_SEG ((uint8_t)0x01)
```

Select the Second 128Mb segment

9.13.2.4 IS25LP080D_EAR_THIRD_SEG

```
#define IS25LP080D_EAR_THIRD_SEG ((uint8_t)0x02)
```

Select the Third 128Mb segment

9.13.2.5 IS25LP080D_EVCR_DTRP

```
#define IS25LP080D_EVCR_DTRP ((uint8_t)0x20)
```

Double transfer rate protocol

9.13.2.6 IS25LP080D_EVCR_DUAL

```
#define IS25LP080D_EVCR_DUAL ((uint8_t)0x40)
```

Dual I/O protocol

9.13.2.7 IS25LP080D_EVCR_ODS

```
#define IS25LP080D_EVCR_ODS ((uint8_t)0x07)
```

Output driver strength

9.13.2.8 IS25LP080D_EVCR_QUAD

```
#define IS25LP080D_EVCR_QUAD ((uint8_t)0x80)
```

Quad I/O protocol

9.13.2.9 IS25LP080D_EVCR_RH

```
#define IS25LP080D_EVCR_RH ((uint8_t)0x10)
```

Reset/hold

9.13.2.10 IS25LP080D_FSR_ERERR

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

Erase error

9.13.2.11 IS25LP080D_FSR_ERSUS

```
#define IS25LP080D_FSR_ERSUS ((uint8_t)0x40)
```

Erase operation suspended

9.13.2.12 IS25LP080D_FSR_NBADDR

```
#define IS25LP080D_FSR_NBADDR ((uint8_t)0x01)
```

3-bytes or 4-bytes addressing

9.13.2.13 IS25LP080D_FSR_PGERR

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

Program error

9.13.2.14 IS25LP080D_FSR_PGSUS

```
#define IS25LP080D_FSR_PGSUS ((uint8_t)0x04)
```

Program operation suspended

9.13.2.15 IS25LP080D_FSR_PRERR

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

Protection error

9.13.2.16 IS25LP080D_FSR_READY

```
#define IS25LP080D_FSR_READY ((uint8_t)0x80)
```

Ready or command in progress

9.13.2.17 IS25LP080D_NVCR_DTRP

```
#define IS25LP080D_NVCR_DTRP ((uint16_t)0x0020)
```

Double transfer rate protocol

9.13.2.18 IS25LP080D_NVCR_DUAL

```
#define IS25LP080D_NVCR_DUAL ((uint16_t)0x0004)
```

Dual I/O protocol

9.13.2.19 IS25LP080D_NVCR_NB_DUMMY

```
#define IS25LP080D_NVCR_NB_DUMMY ((uint16_t)0xF000)
```

Number of dummy clock cycles

9.13.2.20 IS25LP080D_NVCR_NBADDR

```
#define IS25LP080D_NVCR_NBADDR ((uint16_t)0x0001)
```

3-bytes or 4-bytes addressing

9.13.2.21 IS25LP080D_NVCR_ODS

```
#define IS25LP080D_NVCR_ODS ((uint16_t)0x01C0)
```

Output driver strength

9.13.2.22 IS25LP080D_NVCR_QUAB

```
#define IS25LP080D_NVCR_QUAB ((uint16_t)0x0008)
```

Quad I/O protocol

9.13.2.23 IS25LP080D_NVCR_RH

```
#define IS25LP080D_NVCR_RH ((uint16_t)0x0010)
```

Reset/hold

9.13.2.24 IS25LP080D_NVCR_SEGMENT

```
#define IS25LP080D_NVCR_SEGMENT ((uint16_t)0x0002)
```

Upper or lower 128Mb segment selected by default

9.13.2.25 IS25LP080D_NVCR_XIP

```
#define IS25LP080D_NVCR_XIP ((uint16_t)0x0E00)
```

XIP mode at power-on reset

9.13.2.26 IS25LP080D_SR_SRWREN

```
#define IS25LP080D_SR_SRWREN ((uint8_t)0x80)
```

Status register write enable/disable

9.13.2.27 IS25LP080D_SR_WIP

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

IS25LP08D Registers.

Write in progress

9.13.2.28 IS25LP080D_SR_WREN

```
#define IS25LP080D_SR_WREN ((uint8_t)0x02)
```

Write enable latch

9.13.2.29 IS25LP080D_VCR_NB_DUMMY

```
#define IS25LP080D_VCR_NB_DUMMY ((uint8_t)0xF0)
```

Number of dummy clock cycles

9.13.2.30 IS25LP080D_VCR_WRAP

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

Wrap

9.13.2.31 IS25LP080D_VCR_XIP

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

XIP

9.14 src/dev_leddriver.h File Reference

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

```
#include <stdint.h>
#include "per_i2c.h"
```

Classes

- struct color

Macros

- #define SA_LED_DRIVER_H
- #define DSY_LED_DRIVER_MAX_DRIVERS 8

Enumerations

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

Functions

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

9.14.1 Detailed Description

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

9.14.2 Macro Definition Documentation

9.14.2.1 DSY_LED_DRIVER_MAX_DRIVERS

```
#define DSY_LED_DRIVER_MAX_DRIVERS 8
```

Maximum number of drivers

9.14.2.2 SA_LED_DRIVER_H

```
#define SA_LED_DRIVER_H
```

9.14.3 autotoc_md141

9.14.4 Enumeration Type Documentation

9.14.4.1 anonymous enum

```
anonymous enum
```

Different Led colors

Enumerator

LED_COLOR_RED	9.14.5 autotoc_md142
LED_COLOR_GREEN	9.14.6 autotoc_md143
LED_COLOR_BLUE	9.14.7 autotoc_md144
LED_COLOR_WHITE	9.14.8 autotoc_md145
LED_COLOR_PURPLE	9.14.9 autotoc_md146
LED_COLOR_CYAN	9.14.10 autotoc_md147
LED_COLOR_GOLD	9.14.11 autotoc_md148
LED_COLOR_OFF	9.14.12 autotoc_md149
LED_COLOR_LAST	9.14.13 autotoc_md150

9.14.14 Function Documentation

9.14.14.1 dsy_led_driver_color_by_name()

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

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

Parameters

name	Preset color

9.14.14.2 dsy_led_driver_init()

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

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

Parameters

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

9.14.14.3 dsy_led_driver_set_led()

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

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

Parameters

idx	Index
bright	Brightness

9.14.14.4 dsy_led_driver_update()

```
void dsy_led_driver_update ( )
```

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

9.15 src/dev_sdram.h File Reference

```
#include <stdint.h>
#include "daisy_core.h"
```

Classes

- struct dsy_sdram_handle

Macros

- #define RAM_AS4C16M16SA_H
- #define DSY_SDRAM_DATA __attribute__((section(".sdram_data")))
- #define DSY_SDRAM_BSS __attribute__((section(".sdram_bss")))

Enumerations

- enum { DSY_SDRAM_OK, DSY_SDRAM_ERR }
- enum dsy_sdram_state { DSY_SDRAM_STATE_ENABLE, DSY_SDRAM_STATE_DISABLE, DSY_SDR←
  AM_STATE_LAST }
- enum dsy_sdram_pin { DSY_SDRAM_PIN_SDNWE, DSY_SDRAM_PIN_LAST }

Functions

- uint8_t dsy_sdram_init (dsy_sdram_handle ∗dsy_hsdram)

9.15.1 Macro Definition Documentation

9.15.1.1 DSY_SDRAM_BSS

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

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

9.15.1.2 DSY_SDRAM_DATA

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

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

9.15.1.3 RAM_AS4C16M16SA_H

```
#define RAM_AS4C16M16SA_H
```

SDRAM for 32MB AS4C16M16SA (and 64MB equivalent). Thanks to whoever this awesome person is: http↵://main.lv/writeup/stm32f4_sdram_configuration.md The Init function is basically a copy and paste. He has references to timing, etc. RAM is configured at 100MHz (fastest possible on the MCU). To use these the .sdram_data/_bss sections must be configured correctly in the LINKER SCRIPT. using BSS is advised for most things, since the DATA section must also fit in flash in order to be initialized. Data section init not properly set up, as SDRAM is not initialized until after startup code.#

9.15.2 Enumeration Type Documentation

9.15.2.1 anonymous enum

```
anonymous enum
```

Enumerator

DSY_SDRAM_OK	**9.15.3 autotoc_md154**
DSY_SDRAM_ERR	**9.15.4 autotoc_md155**

9.15.4.1 dsy_sdram_pin

```
enum dsy_sdram_pin
```

This is PH5 on Daisy

Enumerator

DSY_SDRAM_PIN_SDNWE	**9.15.5 autotoc_md159**
DSY_SDRAM_PIN_LAST	**9.15.6 autotoc_md160**

9.15.6.1 dsy_sdram_state

```
enum dsy_sdram_state
```

Determines whether chip is initialized, and activated.

Enumerator

DSY_SDRAM_STATE_ENABLE	**9.15.7 autotoc_md156**
DSY_SDRAM_STATE_DISABLE	**9.15.8 autotoc_md157**
DSY_SDRAM_STATE_LAST	**9.15.9 autotoc_md158**

9.15.10 Function Documentation

9.15.10.1 dsy_sdram_init()

```
uint8_t dsy_sdram_init (
            dsy_sdram_handle * dsy_hsdram )
```

Initializes the SDRAM peripheral

9.16 src/dev_sr_4021.h File Reference

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

```
#include "per_gpio.h"
```

Classes

- struct dsy_sr_4021_handle

Macros

- #define DEV_SR_4021_H
- #define SR_4021_MAX_PARALLEL 2
- #define SR_4021_MAX_DAISYCHAIN 1

Enumerations

- enum {
  DSY_SR_4021_PIN_CS, DSY_SR_4021_PIN_CLK, DSY_SR_4021_PIN_DATA, DSY_SR_4021_PIN_D↩
  ATA2,
  DSY_SR_4021_PIN_LAST }

Functions

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

9.16.1 Detailed Description

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

9.16.2 Macro Definition Documentation

9.16.2.1 DEV_SR_4021_H

```
#define DEV_SR_4021_H
```

9.16.3 autotoc_md163

9.16.3.1 SR_4021_MAX_DAISYCHAIN

```
#define SR_4021_MAX_DAISYCHAIN 1
```

fixed maximum for daisychained use

9.16.3.2 SR_4021_MAX_PARALLEL

```
#define SR_4021_MAX_PARALLEL 2
```

Fixed maximums for parallel/daisychained use
These could be expanded, but haven't been tested beyond this

9.16.4 Enumeration Type Documentation

9.16.4.1 anonymous enum

```
anonymous enum
```

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

Enumerator

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

9.16.5 Function Documentation

9.16.5.1 dsy_sr_4021_init()

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

Initialize CD4021 with settings from sr_4021_handle

Parameters

sr	handle to initialize

9.16.5.2 dsy_sr_4021_state()

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

Returns the state of a pin at a given index.

Parameters

∗sr	Handle containing desired pin
idx	Pin index

9.16.5.3 dsy_sr_4021_update()

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

Fills internal states with CD4021 data states.

Parameters

**sr*	Handle to update

9.17 src/dev_sr_595.h File Reference

```
#include "daisy_core.h"
#include "per_gpio.h"
```

Classes

- class ShiftRegister595

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

Variables

- const size_t **kMaxSr595DaisyChain** = 16

9.17.1 Detailed Description

Maximum Number of chained devices Connect device's QH' pin to the next chips serial input

9.18 src/fatfs.h File Reference

fatfs support.

```
#include "ff.h"
#include "ff_gen_drv.h"
#include "util_sd_diskio.h"
```

Macros

- #define __fatfs_H

Functions

- void dsy_fatfs_init (void)

Variables

- uint8_t retSD
- char SDPath [4]
- FATFS SDFatFS
- FIL SDFile

9.18.1 Detailed Description

fatfs support.

9.18.2 Macro Definition Documentation

9.18.2.1 __fatfs_H

```
#define __fatfs_H
```

9.18.3 autotoc_md164

9.18.4 Function Documentation

9.18.4.1 dsy_fatfs_init()

```
void dsy_fatfs_init (
             void  )
```

9.18.5 autotoc_md169

9.18.6 Variable Documentation

9.18.6.1 retSD

```
uint8_t retSD
```

9.18.7 autotoc_md165

9.18.7.1 SDFatFS

```
FATFS SDFatFS
```

9.18.8 autotoc_md167

9.18.8.1 SDFile

```
FIL SDFile
```

9.18.9 autotoc_md168

9.18.9.1 SDPath

```
char SDPath[4]
```

9.18.10 autotoc_md166

9.19 src/ffconf.h File Reference

```
#include "util_bsp_sd_diskio.h"
#include <stdlib.h>
```

Macros

- #define _FFCONF 68300 /∗ Revision ID ∗/
- #define **_FS_READONLY** 0 /∗ 0:Read/Write or 1:Read only ∗/
- #define **_FS_MINIMIZE** 0 /∗ 0 to 3 ∗/
- #define **_USE_STRFUNC** 2 /∗ 0:Disable or 1-2:Enable ∗/
- #define **_USE_FIND** 0
- #define **_USE_MKFS** 1
- #define **_USE_FASTSEEK** 1
- #define **_USE_EXPAND** 0
- #define **_USE_CHMOD** 0
- #define **_USE_LABEL** 0
- #define **_USE_FORWARD** 0
- #define **_CODE_PAGE** 850
- #define **_USE_LFN** 1 /∗ 0 to 3 ∗/
- #define **_MAX_LFN** 255 /∗ Maximum LFN length to handle (12 to 255) ∗/
- #define **_LFN_UNICODE** 0 /∗ 0:ANSI/OEM or 1:Unicode ∗/
- #define **_STRF_ENCODE** 3
- #define **_FS_RPATH** 0 /∗ 0 to 2 ∗/
- #define **_VOLUMES** 1
- #define **_STR_VOLUME_ID** 0 /∗ 0:Use only 0-9 for drive ID, 1:Use strings for drive ID ∗/
- #define **_VOLUME_STRS** "RAM", "NAND", "CF", "SD1", "SD2", "USB1", "USB2", "USB3"
- #define **_MULTI_PARTITION** 0 /∗ 0:Single partition, 1:Multiple partition ∗/
- #define **_MIN_SS** 512 /∗ 512, 1024, 2048 or 4096 ∗/
- #define **_MAX_SS** 512 /∗ 512, 1024, 2048 or 4096 ∗/
- #define **_USE_TRIM** 0
- #define **_FS_NOFSINFO** 0 /∗ 0,1,2 or 3 ∗/
- #define **_FS_TINY** 0 /∗ 0:Normal or 1:Tiny ∗/
- #define **_FS_EXFAT** 0
- #define **_FS_NORTC** 0
- #define **_NORTC_MON** 6
- #define **_NORTC_MDAY** 4
- #define **_NORTC_YEAR** 2015
- #define **_FS_LOCK** 2 /∗ 0:Disable or >=1:Enable ∗/
- #define **_FS_REENTRANT** 0 /∗ 0:Disable or 1:Enable ∗/
- #define **_FS_TIMEOUT** 1000 /∗ Timeout period in unit of time ticks ∗/
- #define **_SYNC_t** osSemaphoreId
- #define **ff_malloc** malloc
- #define **ff_free** free

9.19.1 Detailed Description

Further fatfs support.

9.19.2 Macro Definition Documentation

9.19.2.1 _FFCONF

```
#define _FFCONF 68300 /* Revision ID */
```

FatFs - Generic FAT file system module R0.12c (C)ChaN, 2017

Attention

9.20 src/hid_audio.h File Reference

Audio Driver
Configures Audio Device and provides callback for signal processing.
Many of the hard-coded values here will change (increase), and/or
be replaced by configurable options
.

```
#include <stddef.h>
#include <stdint.h>
#include "per_sai.h"
#include "per_i2c.h"
```

Classes

- struct dsy_audio_handle

Macros

- #define DSY_AUDIO_H
- #define DSY_AUDIO_BLOCK_SIZE_MAX 128
- #define DSY_AUDIO_CHANNELS_MAX 2
- #define DSY_AUDIO_SAMPLE_RATE 48000.0f

Typedefs

- typedef void(∗ dsy_audio_mc_callback) (float ∗∗, float ∗∗, size_t)

Enumerations

- enum { DSY_AUDIO_INTERNAL, DSY_AUDIO_EXTERNAL, DSY_AUDIO_LAST }

Functions

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

9.20.1 Detailed Description

Audio Driver
Configures Audio Device and provides callback for signal processing.
Many of the hard-coded values here will change (increase), and/or
be replaced by configurable options
.

9.20.2 Macro Definition Documentation

9.20.2.1 DSY_AUDIO_BLOCK_SIZE_MAX

```
#define DSY_AUDIO_BLOCK_SIZE_MAX 128
```

Defines for generic maximums While 'Audio Channels Max' is set to 2, this is per-SAI 4x4 Audio I/O is possible using the dsy_audio_mc_callback Hard-coded samplerate is calculated from original clock tree. The new clock tree has less than 0.01% error for all supported sampleratesMax block size

9.20.2.2 DSY_AUDIO_CHANNELS_MAX

```
#define DSY_AUDIO_CHANNELS_MAX 2
```

Max number of audio channels

9.20.2.3 DSY_AUDIO_H

```
#define DSY_AUDIO_H
```

9.20.3 autotoc_md170

9.20.3.1 DSY_AUDIO_SAMPLE_RATE

```
#define DSY_AUDIO_SAMPLE_RATE 48000.0f
```

Default audio engine rate

9.20.4 Typedef Documentation

9.20.4.1 dsy_audio_mc_callback

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

These are user-defineable callbacks that are called when audio data is ready to be received/transmitted. This function is called at samplerate/blocksize (e.g. 1kHz when Function to define for using a single Stereo device for I/Oaudio is packed as: { LEFT | RIGHT | LEFT | RIGHT } typical example:

```
void AudioCallback(float *in, float *out, size_t size)
{        for (size_t i = 0; i < size; i+=2)
{
    out[i] = in[i]; // Left
    out[i+1] = in[i+1]; // Right
}        }
```

*/ typedef void (*dsy_audio_callback)(float*, float*, size_t);*

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

```
void AudioCallback(float **in, float **out, size_t size)
{
*/
```

9.20.5 Enumeration Type Documentation

9.20.5.1 anonymous enum

```
anonymous enum
```

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

Enumerator

DSY_AUDIO_INTERNAL	**9.20.6 autotoc_md171**
DSY_AUDIO_EXTERNAL	**9.20.7 autotoc_md172**
DSY_AUDIO_LAST	**9.20.8 autotoc_md173**

9.20.9 Function Documentation

9.20.9.1 dsy_audio_enter_bypass()

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

If the device supports hardware bypass, enter that mode.∗∗Only minimally tested with WM8731 codec.∗∗

9.20.9.2 dsy_audio_exit_bypass()

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

If the device supports hardware bypass, exit that mode.∗∗Only minimally tested with WM8731 codec.∗∗

9.20.9.3 dsy_audio_init()

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

Initializes the Audio Engine using configurations set to the sai_handlei2c_handles can be set to NULL if not needed.

9.20.9.4 dsy_audio_passthru()

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

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

9.20.9.5 dsy_audio_set_blocksize()

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

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

9.20.9.6 dsy_audio_set_callback()

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

Sets the user defined, interleaving callback to be called when audio data is ready.

intext is a specifier for DSY_AUDIO_INT/EXT (which audio peripheral to use).

When using this, each 'audio block' can have completely independent callbacks.

9.20.9.7 dsy_audio_set_mc_callback()

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

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

9.20.9.8 dsy_audio_silence()

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

sets outputs to 0 without stopping the Audio Engine.

9.20.9.9 dsy_audio_start()

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

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

9.20.9.10 dsy_audio_stop()

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

Stops transmitting/receiving audio on the specified audio block.

9.21 src/hid_ctrl.h File Reference

```
#include <stdint.h>
```

Classes

- class daisy::AnalogControl

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

Namespaces

- daisy

9.22 src/hid_encoder.h File Reference

```
#include "daisy_core.h"
#include "per_gpio.h"
#include "hid_switch.h"
```

Classes

- class daisy::Encoder

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

Namespaces

- daisy

9.23 src/hid_gatein.h File Reference

```
#include "per_gpio.h"
```

Classes

- class daisy::GateIn

  Generic Class for handling gate inputs through GPIO.

Namespaces

- daisy

9.24 src/hid_led.h File Reference

```
#include "daisy_core.h"
#include "per_gpio.h"
```

Classes

- class daisy::Led

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

Namespaces

- daisy

9.25 src/hid_midi.h File Reference

```
#include <stdint.h>
#include <stdlib.h>
#include "per_uart.h"
#include "util_ringbuffer.h"
```

Classes

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

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

Namespaces

- daisy

Enumerations

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

9.26 src/hid_oled_display.h File Reference

```
#include <stdlib.h>
#include <stdint.h>
#include "util_oled_fonts.h"
#include "daisy_core.h"
```

Classes

- class daisy::OledDisplay

Namespaces

- daisy

Macros

- #define DSY_OLED_DISPLAY_H
- #define SSD1309_HEIGHT 64
- #define SSD1309_WIDTH 128

9.26.1 Macro Definition Documentation

9.26.1.1 DSY_OLED_DISPLAY_H

```
#define DSY_OLED_DISPLAY_H
```

Macro

9.26.1.2 SSD1309_HEIGHT

```
#define SSD1309_HEIGHT 64
```

SSD1309 height in pixels

9.26.1.3 SSD1309_WIDTH

```
#define SSD1309_WIDTH 128
```

SSD1309 width in pixels

9.27 src/hid_parameter.h File Reference

```
#include <stdint.h>
#include "hid_ctrl.h"
```

Classes

- class daisy::Parameter

Namespaces

- daisy

9.28 src/hid_rgb_led.h File Reference

```
#include "hid_led.h"
#include "util_color.h"
```

Classes

- class daisy::RgbLed

Namespaces

- daisy

9.29 src/hid_switch.h File Reference

```
#include "daisy_core.h"
#include "per_gpio.h"
```

Classes

- class daisy::Switch

Namespaces

- daisy

9.30 src/hid_usb.h File Reference

```
#include <stdint.h>
#include <stdlib.h>
```

Classes

- class daisy::UsbHandle

  Interface for initializing and using the USB Peripherals on the daisy.

Namespaces

- daisy

9.31 src/hid_wavplayer.h File Reference

```
#include "daisy_core.h"
#include "util_wav_format.h"
```

Classes

- struct daisy::WavFileInfo
- class daisy::WavPlayer

Namespaces

- daisy

Macros

- #define DSY_WAVPLAYER_H
- #define WAV_FILENAME_MAX 256

9.31.1 Macro Definition Documentation

9.31.1.1 DSY_WAVPLAYER_H

```
#define DSY_WAVPLAYER_H
```

Macro

9.31.1.2 WAV_FILENAME_MAX

```
#define WAV_FILENAME_MAX 256
```

Maximum LFN (set to same in FatFs (ffconf.h)

9.32 src/usbd_cdc_if.h File Reference

: Header for usbd_cdc_if.c file.

```
#include "usbd_cdc.h"
```

Typedefs

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

Functions

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

Variables

- USBD_CDC_ItfTypeDef USBD_Interface_fops_FS
- USBD_CDC_ItfTypeDef USBD_Interface_fops_HS

9.32.1 Detailed Description

: Header for usbd_cdc_if.c file.

Version

: v1.0_Cube

Attention

9.33 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**(...)

9.33.1 Detailed Description

: Header for usbd_conf.c file.

Version

: v1.0_Cube

Attention

Index