libDaisy

Generated by Doxygen 1.9.1

Chapter 1

Main Page

libDaisy

Hardware Abstraction Library for the Daisy Audio Platform

libDaisy provides easy access to things such as Audio, Controls, GPIO, MIDI, USB communication, and more.

1.0.1 Features

- Configurable Audio Callback
- MIDI Drivers
- USB Communication (Audio, MIDI, Serial, etc.)
- Peripheral Device Drivers (SPI, I2S, I2C, etc.)

1.0.2 Code Example

```
 {c++}
int main(void)
{
    // Init
    float samplerate;
    hw.Init();
    samplerate = hw.AudioSampleRate();
    midi.Init(MidiHandler::INPUT_MODE_UART1, MidiHandler::OUTPUT_MODE_NONE);
    midi.StartReceive();
    hw.StartAdc();
    hw.StartAudio(AudioCallback);

    for(;;)
    {
        midi.Listen();
        // Handle MIDI Events
        while (midi.HasEvents())
        {
            HandleMidiMessage(midi.PopEvent());
        }
    }
}
```

1.0.3 Getting Started

- Check out our Getting Started Wiki Page
- Browse the reference documentation on the web
- Make some sound!

1.0.4 Project Overview

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 and 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.)

1.0.4.1 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.

1.0.4.2 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 supported hardware platforms.

1.0.4.3 daisy_platform.h

Several other pairs of files exist 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 in without needing a complete understanding of what's going on under the hood.

1.0.5 Contributing

Here are some ways that you can get involved:

- Proof read the documentation and suggest improvements
- Test existing functionality and make `issues`
- Add new functionality to the library. See issues labeled "feature"
- Fix problems with existing codebase. See issues labeled "bug" and/or "polish"

Before working on code, please check out our `Contribution Guidelines` and `Style Guide.`

1.0.6 Support

Here are some ways to get support and connect with other users and developers:

- Join the `Daisy Forum`
- Make a `GitHub Issue`
- Join the `Daisy Slack Workspace`

1.0.7 License

libDaisy is licensed with the permissive MIT open source license.

This allows for modification and reuse in both commercial and personal projects. It does not provide a warranty of any kind.

For the full license, read the `LICENSE` file in the root directory.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Namespace Index

3.1 Namespace List

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

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

Chapter 6

Module Documentation

6.1 LIBDAISY

The daisy library.

Modules

- HUMAN_INTERFACE

  Interface with the world.
- PERIPHERAL

  Peripheral devices, not meant for human interaction.
- SYSTEM

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

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

  Daisy devices. Pod, seed, etc.
- UTILITY

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

6.1.1 Detailed Description

The daisy library.

6.2 HUMAN_INTERFACE

Interface with the world.

Modules

- AUDIO

  Embedded Audio Engine.
- CONTROLS

  Hardware Controls.
- FEEDBACK

  Screens, leds, etc.
- EXTERNAL

  External interface devices.

6.2.1 Detailed Description

Interface with the world.

6.3 AUDIO

Embedded Audio Engine.

Embedded Audio Engine.

6.4 CONTROLS

Hardware Controls.

Classes

- class daisy::AnalogControl

  Hardware Interface for control inputs
  Primarily designed for ADC input controls such as
  potentiometers, and control voltage.
  .
- class daisy::Encoder

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

  Generic Class for handling gate inputs through GPIO.
- class daisy::Parameter
- class daisy::Switch

6.4.1 Detailed Description

Hardware Controls.

6.5 FEEDBACK

Screens, leds, etc.

Classes

- class daisy::Led

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

6.5.1 Detailed Description

Screens, leds, etc.

6.6 EXTERNAL

External interface devices.

Classes

- class daisy::Logger< dest >

  Interface for simple USB logging.
- class daisy::Logger< LOGGER_NONE >
- 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.

Macros

- #define LOGGER_NEWLINE "\r\n"
- #define LOGGER_BUFFER 128

Enumerations

- enum daisy::Logger< dest >::LoggerConsts { **LOGGER_SYNC_OUT** = 0 , daisy::Logger< dest >::LOGGER_SYNC_IN = 2 }
- enum daisy::MidiMessageType {
  daisy::NoteOff , daisy::NoteOn , daisy::PolyphonicKeyPressure , daisy::ControlChange ,
  daisy::ProgramChange , daisy::ChannelPressure , daisy::PitchBend , daisy::MessageLast }

Functions

- daisy::Logger< dest >::Logger ()
- static void daisy::Logger< dest >::Print (const char *format,...)
- static void daisy::Logger< dest >::PrintLine (const char *format,...)
- static void daisy::Logger< dest >::StartLog (bool wait_for_pc=false)
- static void daisy::Logger< dest >::PrintV (const char *format, va_list va)
- static void daisy::Logger< dest >::PrintLineV (const char *format, va_list va)
- static void daisy::Logger< dest >::TransmitSync (const void *buffer, size_t bytes)
- static void daisy::Logger< dest >::TransmitBuf ()
- static void daisy::Logger< dest >::AppendNewLine ()
- static constexpr size_t daisy::Logger< dest >::NewLineSeqLength ()
- static void **daisy::Logger**< **LOGGER_NONE** >**::Print** (const char *format,...)
- static void **daisy::Logger**< **LOGGER_NONE** >**::PrintLine** (const char *format,...)
- static void **daisy::Logger**< **LOGGER_NONE** >**::StartLog** (bool wait_for_pc=false)
- static void **daisy::Logger**< **LOGGER_NONE** >**::PrintV** (const char *format, va_list va)
- static void **daisy::Logger**< **LOGGER_NONE** >**::PrintLineV** (const char *format, va_list va)

- static char daisy::Logger< dest >::tx_buff_ [128]
- static size_t daisy::Logger< dest >::tx_ptr_ = 0
- static size_t daisy::Logger< dest >::pc_sync_ = LOGGER_SYNC_OUT
- static LoggerImpl< dest > daisy::Logger< dest >::impl_
- #define PPCAT_NX(A, B) A##B
- #define PPCAT(A, B) PPCAT_NX(A, B)
- #define STRINGIZE_NX(A) #A
- #define STRINGIZE(A) STRINGIZE_NX(A)
- #define FLT_FMT(_n) STRINGIZE(PPCAT(PPCAT(%c%d.%0, _n), d))
- #define FLT_VAR(_n, _x)
- #define FLT_FMT3 FLT_FMT(3)
- #define FLT_VAR3(_x) FLT_VAR(3, _x)

6.6.1 Detailed Description

External interface devices.

6.6.2 Macro Definition Documentation

6.6.2.1 FLT_FMT

```
#define FLT_FMT(
              _n ) STRINGIZE(PPCAT(PPCAT(%c%d.%0, _n), d))
```

Floating point output formatting string. Include in your printf-style format string example: printf("float value = " FLT_FMT(3) " continue like that", FLT_VAR(3, x));

6.6.2.2 FLT_FMT3

```
#define FLT_FMT3 FLT_FMT(3)
```

Shorthand for 10^-3 fraction, output equivalent to %.3f

6.6.2.3 FLT_VAR

```
#define FLT_VAR(
              _n,
              _x )
```

Value:

```
    (_x < 0 ? '-' : ' '), (int)(abs(_x)), \
        (int)(((abs(_x)) - (int)(abs(_x))) * pow(10, (_n)))
```

Floating point output variable preprocessing Note: uses truncation instead of rounding -> the last digit may be off

6.6.2.4 FLT_VAR3

```
#define FLT_VAR3(
              _x ) FLT_VAR(3, _x)
```

Shorthand for 10^-3 fraction

6.6.2.5 LOGGER_BUFFER

```
#define LOGGER_BUFFER 128
```

size in bytes

6.6.2.6 LOGGER_NEWLINE

```
#define LOGGER_NEWLINE "\r\n"
```

Logger configuration custom newline character sequence

6.6.2.7 PPCAT

```
#define PPCAT(
              A,
              B ) PPCAT_NX(A, B)
```

concatenate tokens

6.6.2.8 PPCAT_NX

```
#define PPCAT_NX(
              A,
              B ) A##B
```

Helper macros for string concatenation and macro expansion non-expanding concatenation

6.6.2.9 STRINGIZE

```
#define STRINGIZE(
              A ) STRINGIZE_NX(A)
```

make a string

6.6.2.10 STRINGIZE_NX

```
#define STRINGIZE_NX(
              A ) #A
```

non-expanding stringize

6.6.3 Enumeration Type Documentation

6.6.3.1 LoggerConsts

```
template<LoggerDestination dest = LOGGER_INTERNAL>
enum daisy::Logger::LoggerConsts  [protected]
```

Internal constants

Enumerator

LOGGER_SYNC_IN	successfully transmit this many packets before switching to blocking transfers

6.6.3.2 MidiMessageType

```
enum daisy::MidiMessageType
```

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

Enumerator

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

6.6.4 Function Documentation

6.6.4.1 AppendNewLine()

```
template<LoggerDestination dest = LOGGER_INTERNAL>
static void daisy::Logger< dest >::AppendNewLine ( )  [static], [protected]
```

Trim control characters and append clean newline sequence, if there's room in the buffer

6.6.4.2 Logger()

```
template<LoggerDestination dest = LOGGER_INTERNAL>
daisy::Logger< dest >::Logger ( )  [inline]
```

Object constructor

6.6.4.3 NewLineSeqLength()

```
template<LoggerDestination dest = LOGGER_INTERNAL>
static constexpr size_t daisy::Logger< dest >::NewLineSeqLength ( )  [inline], [static], [constexpr],
[protected]
```

Constexpr function equivalent of strlen(LOGGER_NEWLINE) < custom newline character sequence

6.6.4.4 Print()

```
template<LoggerDestination dest = LOGGER_INTERNAL>
static void daisy::Logger< dest >::Print (
            const char * format,
             ... )  [static]
```

Print formatted string

6.6.4.5 PrintLine()

```
template<LoggerDestination dest = LOGGER_INTERNAL>
static void daisy::Logger< dest >::PrintLine (
            const char * format,
             ... ) [static]
```

Print formatted string appending line termination sequence

6.6.4.6 PrintLineV()

```
template<LoggerDestination dest = LOGGER_INTERNAL>
static void daisy::Logger< dest >::PrintLineV (
            const char * format,
            va_list va ) [static]
```

Variadic argument variant of PrintLine()

6.6.4.7 PrintV()

```
template<LoggerDestination dest = LOGGER_INTERNAL>
static void daisy::Logger< dest >::PrintV (
            const char * format,
            va_list va ) [static]
```

Variadic argument variant of Print()

6.6.4.8 StartLog()

```
template<LoggerDestination dest = LOGGER_INTERNAL>
static void daisy::Logger< dest >::StartLog (
            bool wait_for_pc = false ) [static]
```

Start the logging session.

Parameters

wait_for_pc	block until remote terminal is ready

6.6.4.9 TransmitBuf()

```
template<LoggerDestination dest = LOGGER_INTERNAL>
static void daisy::Logger< dest >::TransmitBuf ( ) [static], [protected]
```

Transfer accumulated data

6.6.4.10 TransmitSync()

```
template<LoggerDestination dest = LOGGER_INTERNAL>
static void daisy::Logger< dest >::TransmitSync (
             const void * buffer,
             size_t bytes )  [inline], [static], [protected]
```

Blocking wrapper for Transmit()

6.6.5 Variable Documentation

6.6.5.1 impl_

```
template<LoggerDestination dest>
LoggerImpl< dest > daisy::Logger< dest >::impl_  [static], [protected]
```

underlying trasnfer implementation

6.6.5.2 pc_sync_

```
template<LoggerDestination dest>
size_t daisy::Logger< dest >::pc_sync_ = LOGGER_SYNC_OUT  [static], [protected]
```

terminal synchronization state

start with non-blocking transfers to support startup-time printouts

6.6.5.3 tx_buff_

```
template<LoggerDestination dest>
char daisy::Logger< dest >::tx_buff_  [static], [protected]
```

member variables buffer for log data

member variable definition (could switch to inline statics in C++17) this needs to remain in SRAM to support startup-time printouts

6.6.5.4 tx_ptr_

```
template<LoggerDestination dest>
size_t daisy::Logger< dest >::tx_ptr_ = 0  [static], [protected]
```

current position in the buffer

6.7 PERIPHERAL

Peripheral devices, not meant for human interaction.

Modules

- SERIAL

  Serial Communications.
- ANALOG_DIGITAL_CONVERSION

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

  GPIO, timers, and SDMMC.

6.7.1 Detailed Description

Peripheral devices, not meant for human interaction.

6.8 SERIAL

Serial Communications.

Classes

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

Enumerations

- enum dsy_qspi_pin { DSY_QSPI_PIN_IO0 , DSY_QSPI_PIN_IO1 , DSY_QSPI_PIN_IO2 , DSY_QSPI_PIN_IO3 , DSY_QSPI_PIN_CLK , DSY_QSPI_PIN_NCS , DSY_QSPI_PIN_LAST }
- enum dsy_qspi_mode { DSY_QSPI_MODE_DSY_MEMORY_MAPPED , DSY_QSPI_MODE_INDIRECT_POLLING , DSY_QSPI_MODE_LAST }
- enum dsy_qspi_device { DSY_QSPI_DEVICE_IS25LP080D , DSY_QSPI_DEVICE_IS25LP064A , DSY_QSPI_DEVICE_LAST }
- enum daisy::SpiPeriph { daisy::SPI_PERIPH_1 , daisy::SPI_PERIPH_3 , daisy::SPI_PERIPH_6 }
- enum daisy::SpiPin { daisy::SPI_PIN_CS , daisy::SPI_PIN_SCK , daisy::SPI_PIN_MOSI , daisy::SPI_PIN_MISO }

Functions

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

6.8.1 Detailed Description

Serial Communications.

6.8.2 Enumeration Type Documentation

6.8.2.1 dsy_qspi_device

`enum dsy_qspi_device`

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

Enumerator

DSY_QSPI_DEVICE_IS25LP080D	&
DSY_QSPI_DEVICE_IS25LP064A	&
DSY_QSPI_DEVICE_LAST	&

6.8.2.2 dsy_qspi_mode

`enum dsy_qspi_mode`

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

Read/Write possible via dsy_qspi_$*$ functions

Enumerator

DSY_QSPI_MODE_DSY_MEMORY_MAPPED	&
DSY_QSPI_MODE_INDIRECT_POLLING	&
DSY_QSPI_MODE_LAST	&

6.8.2.3 dsy_qspi_pin

`enum dsy_qspi_pin`

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

Enumerator

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

6.8.2.4 SpiPeriph

`enum daisy::SpiPeriph`

SPI peripheral enum

Enumerator

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

6.8.2.5 SpiPin

`enum daisy::SpiPin`

SPI pins

Enumerator

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

6.8.3 Function Documentation

6.8.3.1 dsy_qspi_deinit()

```
int dsy_qspi_deinit ( )
```

Deinitializes the peripheral This should be called before reinitializing QSPI in a different mode.

Returns

DSY_MEMORY_OK or DSY_MEMORY_ERROR

6.8.3.2 dsy_qspi_erase()

```
int dsy_qspi_erase (
            uint32_t start_adr,
            uint32_t end_adr )
```

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

Parameters

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

Returns

DSY_MEMORY_OK or DSY_MEMORY_ERROR

6.8.3.3 dsy_qspi_erasesector()

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

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

Parameters

addr	Address of sector to erase

Returns

DSY_MEMORY_OK or DSY_MEMORY_ERROR

6.8.3.4 dsy_qspi_init()

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

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

Parameters

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

Returns

DSY_MEMORY_OK or DSY_MEMORY_ERROR

6.8.3.5 dsy_qspi_write()

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

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

Parameters

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

Returns

DSY_MEMORY_OK or DSY_MEMORY_ERROR

6.8.3.6 dsy_qspi_writepage()

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

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

Parameters

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

Returns

DSY_MEMORY_OK or DSY_MEMORY_ERROR

6.9 ANALOG_DIGITAL_CONVERSION

Convert from digital to analog, or vice-versa.

Classes

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

6.9.1 Detailed Description

Convert from digital to analog, or vice-versa.

6.10 OTHER

GPIO, timers, and SDMMC.

Classes

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

Enumerations

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

Functions

- void dsy_gpio_init (const dsy_gpio *p)
- void dsy_gpio_deinit (const dsy_gpio *p)
- uint8_t dsy_gpio_read (const dsy_gpio *p)
- void dsy_gpio_write (const dsy_gpio *p, uint8_t state)
- void dsy_gpio_toggle (const dsy_gpio *p)

6.10.1 Detailed Description

GPIO, timers, and SDMMC.

General Purpose IO driver

6.10.2 Enumeration Type Documentation

6.10.2.1 dsy_gpio_mode

`enum dsy_gpio_mode`

Sets the mode of the GPIO

Enumerator

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

6.10.2.2 dsy_gpio_pull

`enum dsy_gpio_pull`

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

Enumerator

DSY_GPIO_NOPULL	&
DSY_GPIO_PULLUP	&
DSY_GPIO_PULLDOWN	&

6.10.2.3 SdmmcBitWidth

`enum daisy::SdmmcBitWidth`

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

Enumerator

SDMMC_BITS↩ _1	&
SDMMC_BITS↩ _4	&

6.10.2.4 SdmmcMode

`enum daisy::SdmmcMode`

Operating Mode. Currently only FatFS is supported.

Enumerator

SDMMC_MODE_FATFS	&

6.10.2.5 SdmmcSpeed

`enum daisy::SdmmcSpeed`

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

Enumerator

SDMMC_SPEED_400KHZ	&
SDMMC_SPEED_12MHZ	&

6.10.3 Function Documentation

6.10.3.1 dsy_gpio_deinit()

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

Deinitializes the gpio pin

Parameters

*p	Pin pointer

6.10.3.2 dsy_gpio_init()

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

Initializes the gpio with the settings configured.

Parameters

*p	Pin pointer

6.10.3.3 dsy_gpio_read()

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

Reads the state of the gpio pin

Parameters

*p	Pin pointer

Returns

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

6.10.3.4 dsy_gpio_toggle()

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

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

Parameters

*p	Pin pointer

6.10.3.5 dsy_gpio_write()

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

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

Parameters

*p	Pin pointer
state	State to write

6.11 SYSTEM

Deals with system. DMA, clocks, etc.

Functions

- void dsy_dma_init (void)
- void dsy_dma_clear_cache_for_buffer (uint8_t *buffer, size_t size)
- void dsy_dma_invalidate_cache_for_buffer (uint8_t *buffer, size_t size)

6.11.1 Detailed Description

Deals with system. DMA, clocks, etc.

A handle for interacting with the Core System. This includes the Clock tree, MPU, global DMA initialization, cache handling, and any other necessary global initiailzation

Author

shensley

6.11.2 Function Documentation

6.11.2.1 dsy_dma_clear_cache_for_buffer()

```
void dsy_dma_clear_cache_for_buffer (
            uint8_t * buffer,
            size_t size )
```

DMA transfers require the buffers to be excluded from the cache because the DMA reads / writes directly to the SRAM whereas the processor itself accesses the cache. Otherwise the DMA will access whatever is in the SRAM at the time which may not be in sync with the data in the cache - resulting in wrong data transmitted / received. You can place buffers in the D2 memory domain, in a section that has the cache disabled like this: uint8_t DMA↩ _BUFFER_MEM_SECTION my_buffer[100]; If this is not possible for some reason, call this function to clear the cache (write cache contents to SRAM if required) before starting to transmit data via the DMA.

6.11.2.2 dsy_dma_init()

```
void dsy_dma_init (
            void  )
```

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

6.11.2.3 dsy_dma_invalidate_cache_for_buffer()

```
void dsy_dma_invalidate_cache_for_buffer (
            uint8_t * buffer,
            size_t size )
```

DMA transfers require the buffers to be excluded from the cache because the DMA reads / writes directly to the SRAM whereas the processor itself accesses the cache. Otherwise the DMA will access whatever is in the SRAM at the time which may not be in sync with the data in the cache - resulting in wrong data transmitted / received. You can place buffers in the D2 memory domain, in a section that has the cache disabled like this: uint8_t DMA_↩ BUFFER_MEM_SECTION my_buffer[100]; If this is not possible for some reason, call this function to invalidate the cache (read SRAM contents to cache if required) after reading data from peripherals via the DMA.

6.12 DEVICE

Low level devices. Led drivers, codecs, etc.

Modules

- SHIFTREGISTER
  Digital shift registers.
- FLASH
  Flash memory.
- CODEC
  Audio codecs.
- LED
  LED driver devices.
- SDRAM
  SDRAM devices.

6.12.1 Detailed Description

Low level devices. Led drivers, codecs, etc.

6.13 SHIFTREGISTER

Digital shift registers.

Classes

- class ShiftRegister595

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

6.13.1 Detailed Description

Digital shift registers.

Device Driver for CD4021 shift register.

Author

shensley

CD4021B-Q1: CMOS 8-STAGE STATIC SHIFT REGISTER

Supply Voltage: 3V to 18V Clock Freq: 3MHz at 5V (less at 3v3) -> 8.5MHz at 15V Pin Descriptions:

- Parallel Data[1-8] - 7, 6, 5, 4, 13, 14, 115, 1
- Serial Data - 11
- Clock - 10
- P/!S - 9
- Q[6-8] - 2, 12, 3

Driver has support for daisy chaining and running up to 2 same-sized chains in parallel from a single set of clk/latch pins to reduce pin/code overhead when using multiple devices.

When dealing with multiple parallel/daisy-chained devices the states of all inputs will be filled in the following order (example uses two chained and two parallel): data[chain0,parallel0], data[chain1,parallel0], data[chain0,parallel1], data[chain1,parallel1];

When combining multiple daisy chained and parallel devices the number of devices chained should match for each parallel device chain.

6.14 FLASH

Flash memory.

Macros

- #define ENTER_DEEP_POWER_DOWN 0XB9
- #define EXIT_DEEP_POWER_DOWN 0XB9
- #define RESET_ENABLE_CMD 0x66
- #define RESET_MEMORY_CMD 0x99
- #define READ_ID_CMD 0xAB
- #define READ_ID_CMD2 0x9F
- #define MULTIPLE_IO_READ_ID_CMD 0xAF
- #define READ_SERIAL_FLASH_DISCO_PARAM_CMD 0x5A
- #define READ_MANUFACT_AND_ID 0x90
- #define READ_UNIQUE_ID 0x4B
- #define NO_OP 0x00
- #define SECTOR_UNLOCK 0x26
- #define SECTOR_LOCK 0x24
- #define INFO_ROW_ERASE_CMD 0x64
- #define INFO_ROW_PROGRAM_CMD 0x62
- #define INFO_ROW_READ_CMD 0x68
- #define READ_CMD 0x03
- #define FAST_READ_CMD 0x0B
- #define FAST_READ_DTR_CMD 0x0D
- #define DUAL_OUT_FAST_READ_CMD 0x3B
- #define DUAL_INOUT_FAST_READ_CMD 0xBB
- #define DUAL_INOUT_FAST_READ_DTR_CMD 0xBD
- #define QUAD_OUT_FAST_READ_CMD 0x6B
- #define QUAD_INOUT_FAST_READ_CMD 0xEB
- #define QUAD_INOUT_FAST_READ_DTR_CMD 0xED
- #define WRITE_ENABLE_CMD 0x06
- #define WRITE_DISABLE_CMD 0x04
- #define READ_STATUS_REG_CMD 0x05
- #define WRITE_STATUS_REG_CMD 0x01
- #define READ_FUNCTION_REGISTER 0X48
- #define WRITE_FUNCTION_REGISTER 0x42
- #define WRITE_READ_PARAM_REG_CMD 0xC0
- #define PAGE_PROG_CMD 0x02
- #define QUAD_IN_PAGE_PROG_CMD 0x32
- #define EXT_QUAD_IN_PAGE_PROG_CMD 0x38
- #define SECTOR_ERASE_CMD 0xd7
- #define SECTOR_ERASE_QPI_CMD 0x20
- #define BLOCK_ERASE_CMD 0xD8
- #define BLOCK_ERASE_32K_CMD 0x52
- #define CHIP_ERASE_CMD 0xC7
- #define EXT_CHIP_ERASE_CMD 0x60
- #define PROG_ERASE_RESUME_CMD 0x7A
- #define EXT_PROG_ERASE_RESUME_CMD 0x30
- #define PROG_ERASE_SUSPEND_CMD 0x75
- #define EXT_PROG_ERASE_SUSPEND_CMD 0xB0
- #define ENTER_QUAD_CMD 0x35
- #define EXIT_QUAD_CMD 0xF5

- #define IS25LP064A_SR_WIP ((uint8_t)0x01)

  IS25LP08D Registers

- #define IS25LP064A_SR_WREN ((uint8_t)0x02)
- #define IS25LP064A_SR_SRWREN ((uint8_t)0x80)
- #define IS25LP064A_SR_QE ((uint8_t)0x40)
- #define IS25LP064A_NVCR_NBADDR ((uint16_t)0x0001)
- #define IS25LP064A_NVCR_SEGMENT ((uint16_t)0x0002)
- #define IS25LP064A_NVCR_DUAL ((uint16_t)0x0004)
- #define IS25LP064A_NVCR_QUAB ((uint16_t)0x0008)
- #define IS25LP064A_NVCR_RH ((uint16_t)0x0010)
- #define IS25LP064A_NVCR_DTRP ((uint16_t)0x0020)
- #define IS25LP064A_NVCR_ODS ((uint16_t)0x01C0)
- #define IS25LP064A_NVCR_XIP ((uint16_t)0x0E00)
- #define IS25LP064A_NVCR_NB_DUMMY ((uint16_t)0xF000)
- #define IS25LP064A_VCR_WRAP ((uint8_t)0x03)
- #define IS25LP064A_VCR_XIP ((uint8_t)0x08)
- #define IS25LP064A_VCR_NB_DUMMY ((uint8_t)0xF0)
- #define IS25LP064A_EAR_HIGHEST_SE ((uint8_t)0x03)
- #define IS25LP064A_EAR_THIRD_SEG ((uint8_t)0x02)
- #define IS25LP064A_EAR_SECOND_SEG ((uint8_t)0x01)
- #define IS25LP064A_EAR_LOWEST_SEG ((uint8_t)0x00)
- #define IS25LP064A_EVCR_ODS ((uint8_t)0x07)
- #define IS25LP064A_EVCR_RH ((uint8_t)0x10)
- #define IS25LP064A_EVCR_DTRP ((uint8_t)0x20)
- #define IS25LP064A_EVCR_DUAL ((uint8_t)0x40)
- #define IS25LP064A_EVCR_QUAD ((uint8_t)0x80)
- #define IS25LP064A_FSR_NBADDR ((uint8_t)0x01)
- #define IS25LP064A_FSR_PRERR ((uint8_t)0x02)
- #define IS25LP064A_FSR_PGSUS ((uint8_t)0x04)
- #define IS25LP064A_FSR_PGERR ((uint8_t)0x10)
- #define IS25LP064A_FSR_ERERR ((uint8_t)0x20)
- #define IS25LP064A_FSR_ERSUS ((uint8_t)0x40)
- #define IS25LP064A_FSR_READY ((uint8_t)0x80)
- #define ENTER_DEEP_POWER_DOWN 0XB9
- #define EXIT_DEEP_POWER_DOWN 0XB9
- #define RESET_ENABLE_CMD 0x66
- #define RESET_MEMORY_CMD 0x99
- #define READ_ID_CMD 0xAB
- #define READ_ID_CMD2 0x9F
- #define MULTIPLE_IO_READ_ID_CMD 0xAF
- #define READ_SERIAL_FLASH_DISCO_PARAM_CMD 0x5A
- #define READ_MANUFACT_AND_ID 0x90
- #define READ_UNIQUE_ID 0x4B
- #define NO_OP 0x00
- #define SECTOR_UNLOCK 0x26
- #define SECTOR_LOCK 0x24
- #define INFO_ROW_ERASE_CMD 0x64
- #define INFO_ROW_PROGRAM_CMD 0x62
- #define INFO_ROW_READ_CMD 0x68
- #define PAGE_PROG_CMD 0x02
- #define PAGE_PROG_CMD 0x02
- #define QUAD_IN_PAGE_PROG_CMD 0x32
- #define EXT_QUAD_IN_PAGE_PROG_CMD 0x38

- #define READ_CMD 0x03
- #define FAST_READ_CMD 0x0B
- #define FAST_READ_DTR_CMD 0x0D
- #define DUAL_OUT_FAST_READ_CMD 0x3B
- #define DUAL_INOUT_FAST_READ_CMD 0xBB
- #define DUAL_INOUT_FAST_READ_DTR_CMD 0xBD
- #define QUAD_OUT_FAST_READ_CMD 0x6B
- #define QUAD_INOUT_FAST_READ_CMD 0xEB
- #define QUAD_INOUT_FAST_READ_DTR_CMD 0xED
- #define WRITE_ENABLE_CMD 0x06
- #define WRITE_DISABLE_CMD 0x04
- #define READ_STATUS_REG_CMD 0x05
- #define WRITE_STATUS_REG_CMD 0x01
- #define READ_FUNCTION_REGISTER 0X48
- #define WRITE_FUNCTION_REGISTER 0x42
- #define READ_READ_PARAM_REG_CMD 0x61
- #define READ_EXT_READ_PARAM_CMD 0x81
- #define CLEAR_EXT_READ_PARAM_CMD 0x82
- #define WRITE_READ_PARAM_REG_CMD 0xC0
- #define WRITE_NV_READ_PARAM_REG_CMD 0x65
- #define EXT_WRITE_READ_PARAM_REG_CMD 0x63
- #define WRITE_EXT_READ_PARAM_REG_CMD 0x83
- #define WRITE_EXT_NV_READ_PARAM_REG_CMD 0x85
- #define QUAD_IN_FAST_PROG_CMD 0x32
- #define EXT_QUAD_IN_FAST_PROG_CMD 0x38
- #define SECTOR_ERASE_CMD 0xd7
- #define SECTOR_ERASE_QPI_CMD 0x20
- #define BLOCK_ERASE_CMD 0xD8
- #define BLOCK_ERASE_32K_CMD 0x52
- #define CHIP_ERASE_CMD 0xC7
- #define EXT_CHIP_ERASE_CMD 0x60
- #define PROG_ERASE_RESUME_CMD 0x7A
- #define EXT_PROG_ERASE_RESUME_CMD 0x30
- #define PROG_ERASE_SUSPEND_CMD 0x75
- #define EXT_PROG_ERASE_SUSPEND_CMD 0xB0
- #define ENTER_QUAD_CMD 0x35
- #define EXIT_QUAD_CMD 0xF5
- #define IS25LP080D_SR_WIP ((uint8_t)0x01)

  IS25LP08D Registers

- #define IS25LP080D_SR_WREN ((uint8_t)0x02)
- #define IS25LP080D_SR_SRWREN ((uint8_t)0x80)
- #define IS25LP080D_SR_QE ((uint8_t)0x40)
- #define IS25LP080D_NVCR_NBADDR ((uint16_t)0x0001)
- #define IS25LP080D_NVCR_SEGMENT ((uint16_t)0x0002)
- #define IS25LP080D_NVCR_DUAL ((uint16_t)0x0004)
- #define IS25LP080D_NVCR_QUAB ((uint16_t)0x0008)
- #define IS25LP080D_NVCR_RH ((uint16_t)0x0010)
- #define IS25LP080D_NVCR_DTRP ((uint16_t)0x0020)
- #define IS25LP080D_NVCR_ODS ((uint16_t)0x01C0)
- #define IS25LP080D_NVCR_XIP ((uint16_t)0x0E00)
- #define IS25LP080D_NVCR_NB_DUMMY ((uint16_t)0xF000)
- #define IS25LP080D_VCR_WRAP ((uint8_t)0x03)
- #define IS25LP080D_VCR_XIP ((uint8_t)0x08)

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

6.14.1 Detailed Description

Flash memory.

IS25LP08D Commands

6.14.2 Macro Definition Documentation

6.14.2.1 BLOCK_ERASE_32K_CMD [1/2]

```
#define BLOCK_ERASE_32K_CMD 0x52
```

&

6.14.2.2 BLOCK_ERASE_32K_CMD [2/2]

```
#define BLOCK_ERASE_32K_CMD 0x52
```

&

6.14.2.3 BLOCK_ERASE_CMD [1/2]

```
#define BLOCK_ERASE_CMD 0xD8
```

&

6.14.2.4 BLOCK_ERASE_CMD [2/2]

```
#define BLOCK_ERASE_CMD 0xD8
```

&

6.14.2.5 CHIP_ERASE_CMD [1/2]

```
#define CHIP_ERASE_CMD 0xC7
```

&

6.14.2.6 CHIP_ERASE_CMD [2/2]

```
#define CHIP_ERASE_CMD 0xC7
```

&

6.14.2.7 CLEAR_EXT_READ_PARAM_CMD

```
#define CLEAR_EXT_READ_PARAM_CMD 0x82
```

&

6.14.2.8 DUAL_INOUT_FAST_READ_CMD [1/2]

```
#define DUAL_INOUT_FAST_READ_CMD 0xBB
```

&

6.14.2.9 DUAL_INOUT_FAST_READ_CMD [2/2]

```
#define DUAL_INOUT_FAST_READ_CMD 0xBB
```

&

6.14.2.10 DUAL_INOUT_FAST_READ_DTR_CMD [1/2]

```
#define DUAL_INOUT_FAST_READ_DTR_CMD 0xBD
```

&

6.14.2.11 DUAL_INOUT_FAST_READ_DTR_CMD [2/2]

```
#define DUAL_INOUT_FAST_READ_DTR_CMD 0xBD
```

&

6.14.2.12 DUAL_OUT_FAST_READ_CMD [1/2]

```
#define DUAL_OUT_FAST_READ_CMD 0x3B
```

&

6.14.2.13 DUAL_OUT_FAST_READ_CMD [2/2]

```
#define DUAL_OUT_FAST_READ_CMD 0x3B
```

&

6.14.2.14 ENTER_DEEP_POWER_DOWN [1/2]

```
#define ENTER_DEEP_POWER_DOWN 0XB9
```

Low Power Modes &

6.14.2.15 ENTER_DEEP_POWER_DOWN [2/2]

```
#define ENTER_DEEP_POWER_DOWN 0XB9
```

Low Power Modes &

6.14.2.16 ENTER_QUAD_CMD [1/2]

```
#define ENTER_QUAD_CMD 0x35
```

Quad Operations

6.14.2.17 ENTER_QUAD_CMD [2/2]

```
#define ENTER_QUAD_CMD 0x35
```

Quad Operations

6.14.2.18 EXIT_DEEP_POWER_DOWN [1/2]

```
#define EXIT_DEEP_POWER_DOWN 0XB9
```

&

6.14.2.19 EXIT_DEEP_POWER_DOWN [2/2]

```
#define EXIT_DEEP_POWER_DOWN 0XB9
```

&

6.14.2.20 EXIT_QUAD_CMD [1/2]

```
#define EXIT_QUAD_CMD 0xF5
```

&

6.14.2.21 EXIT_QUAD_CMD [2/2]

```
#define EXIT_QUAD_CMD 0xF5
```

&

6.14.2.22 EXT_CHIP_ERASE_CMD [1/2]

```
#define EXT_CHIP_ERASE_CMD 0x60
```

&

6.14.2.23 EXT_CHIP_ERASE_CMD [2/2]

```
#define EXT_CHIP_ERASE_CMD 0x60
```

&

6.14.2.24 EXT_PROG_ERASE_RESUME_CMD [1/2]

```
#define EXT_PROG_ERASE_RESUME_CMD 0x30
```

&

6.14.2.25 EXT_PROG_ERASE_RESUME_CMD [2/2]

```
#define EXT_PROG_ERASE_RESUME_CMD 0x30
```

&

6.14.2.26 EXT_PROG_ERASE_SUSPEND_CMD [1/2]

```
#define EXT_PROG_ERASE_SUSPEND_CMD 0xB0
```

&

6.14.2.27 EXT_PROG_ERASE_SUSPEND_CMD [2/2]

```
#define EXT_PROG_ERASE_SUSPEND_CMD 0xB0
```

&

6.14.2.28 EXT_QUAD_IN_FAST_PROG_CMD

```
#define EXT_QUAD_IN_FAST_PROG_CMD 0x38
```

&

6.14.2.29 EXT_QUAD_IN_PAGE_PROG_CMD [1/2]

```
#define EXT_QUAD_IN_PAGE_PROG_CMD 0x38
```

&

6.14.2.30 EXT_QUAD_IN_PAGE_PROG_CMD [2/2]

```
#define EXT_QUAD_IN_PAGE_PROG_CMD 0x38
```

&

6.14.2.31 EXT_WRITE_READ_PARAM_REG_CMD

```
#define EXT_WRITE_READ_PARAM_REG_CMD 0x63
```

volatile

6.14.2.32 FAST_READ_CMD [1/2]

```
#define FAST_READ_CMD 0x0B
```

&

6.14.2.33 FAST_READ_CMD [2/2]

```
#define FAST_READ_CMD 0x0B
```

&

6.14.2.34 FAST_READ_DTR_CMD [1/2]

```
#define FAST_READ_DTR_CMD 0x0D
```

&

6.14.2.35 FAST_READ_DTR_CMD [2/2]

```
#define FAST_READ_DTR_CMD 0x0D
```

&

6.14.2.36 INFO_ROW_ERASE_CMD [1/2]

```
#define INFO_ROW_ERASE_CMD 0x64
```

Security Information Row &

6.14.2.37 INFO_ROW_ERASE_CMD [2/2]

```
#define INFO_ROW_ERASE_CMD 0x64
```

Security Information Row &

6.14.2.38 INFO_ROW_PROGRAM_CMD [1/2]

```
#define INFO_ROW_PROGRAM_CMD 0x62
```

&

6.14.2.39 INFO_ROW_PROGRAM_CMD [2/2]

```
#define INFO_ROW_PROGRAM_CMD 0x62
```

&

6.14.2.40 INFO_ROW_READ_CMD [1/2]

```
#define INFO_ROW_READ_CMD 0x68
```

&

6.14.2.41 INFO_ROW_READ_CMD [2/2]

```
#define INFO_ROW_READ_CMD 0x68
```

&

6.14.2.42 IS25LP064A_EAR_HIGHEST_SE

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

Select the Highest 128Mb segment

6.14.2.43 IS25LP064A_EAR_LOWEST_SEG

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

Select the Lowest 128Mb segment (default)

6.14.2.44 IS25LP064A_EAR_SECOND_SEG

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

Select the Second 128Mb segment

6.14.2.45 IS25LP064A_EAR_THIRD_SEG

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

Select the Third 128Mb segment

6.14.2.46 IS25LP064A_EVCR_DTRP

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

Double transfer rate protocol

6.14.2.47 IS25LP064A_EVCR_DUAL

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

Dual I/O protocol

6.14.2.48 IS25LP064A_EVCR_ODS

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

Output driver strength

6.14.2.49 IS25LP064A_EVCR_QUAD

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

Quad I/O protocol

6.14.2.50 IS25LP064A_EVCR_RH

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

Reset/hold

6.14.2.51 IS25LP064A_FSR_ERERR

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

Erase error

6.14.2.52 IS25LP064A_FSR_ERSUS

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

Erase operation suspended

6.14.2.53 IS25LP064A_FSR_NBADDR

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

3-bytes or 4-bytes addressing

6.14.2.54 IS25LP064A_FSR_PGERR

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

Program error

6.14.2.55 IS25LP064A_FSR_PGSUS

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

Program operation suspended

6.14.2.56 IS25LP064A_FSR_PRERR

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

Protection error

6.14.2.57 IS25LP064A_FSR_READY

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

Ready or command in progress

6.14.2.58 IS25LP064A_NVCR_DTRP

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

Double transfer rate protocol

6.14.2.59 IS25LP064A_NVCR_DUAL

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

Dual I/O protocol

6.14.2.60 IS25LP064A_NVCR_NB_DUMMY

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

Number of dummy clock cycles

6.14.2.61 IS25LP064A_NVCR_NBADDR

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

3-bytes or 4-bytes addressing

6.14.2.62 IS25LP064A_NVCR_ODS

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

Output driver strength

6.14.2.63 IS25LP064A_NVCR_QUAB

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

Quad I/O protocol

6.14.2.64 IS25LP064A_NVCR_RH

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

Reset/hold

6.14.2.65 IS25LP064A_NVCR_SEGMENT

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

Upper or lower 128Mb segment selected by default

6.14.2.66 IS25LP064A_NVCR_XIP

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

XIP mode at power-on reset

6.14.2.67 IS25LP064A_SR_QE

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

&

6.14.2.68 IS25LP064A_SR_SRWREN

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

Status register write enable/disable

6.14.2.69 IS25LP064A_SR_WIP

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

IS25LP08D Registers

Write in progress

6.14.2.70 IS25LP064A_SR_WREN

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

Write enable latch

6.14.2.71 IS25LP064A_VCR_NB_DUMMY

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

Number of dummy clock cycles

6.14.2.72 IS25LP064A_VCR_WRAP

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

Wrap

6.14.2.73 IS25LP064A_VCR_XIP

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

XIP

6.14.2.74 IS25LP080D_EAR_HIGHEST_SE

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

Select the Highest 128Mb segment

6.14.2.75 IS25LP080D_EAR_LOWEST_SEG

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

Select the Lowest 128Mb segment (default)

6.14.2.76 IS25LP080D_EAR_SECOND_SEG

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

Select the Second 128Mb segment

6.14.2.77 IS25LP080D_EAR_THIRD_SEG

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

Select the Third 128Mb segment

6.14.2.78 IS25LP080D_EVCR_DTRP

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

Double transfer rate protocol

6.14.2.79 IS25LP080D_EVCR_DUAL

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

Dual I/O protocol

6.14.2.80 IS25LP080D_EVCR_ODS

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

Output driver strength

6.14.2.81 IS25LP080D_EVCR_QUAD

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

Quad I/O protocol

6.14.2.82 IS25LP080D_EVCR_RH

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

Reset/hold

6.14.2.83 IS25LP080D_FSR_ERERR

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

Erase error

6.14.2.84 IS25LP080D_FSR_ERSUS

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

Erase operation suspended

6.14.2.85 IS25LP080D_FSR_NBADDR

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

3-bytes or 4-bytes addressing

6.14.2.86 IS25LP080D_FSR_PGERR

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

Program error

6.14.2.87 IS25LP080D_FSR_PGSUS

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

Program operation suspended

6.14.2.88 IS25LP080D_FSR_PRERR

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

Protection error

6.14.2.89 IS25LP080D_FSR_READY

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

Ready or command in progress

6.14.2.90 IS25LP080D_NVCR_DTRP

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

Double transfer rate protocol

6.14.2.91 IS25LP080D_NVCR_DUAL

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

Dual I/O protocol

6.14.2.92 IS25LP080D_NVCR_NB_DUMMY

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

Number of dummy clock cycles

6.14.2.93 IS25LP080D_NVCR_NBADDR

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

3-bytes or 4-bytes addressing

6.14.2.94 IS25LP080D_NVCR_ODS

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

Output driver strength

6.14.2.95 IS25LP080D_NVCR_QUAB

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

Quad I/O protocol

6.14.2.96 IS25LP080D_NVCR_RH

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

Reset/hold

6.14.2.97 IS25LP080D_NVCR_SEGMENT

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

Upper or lower 128Mb segment selected by default

6.14.2.98 IS25LP080D_NVCR_XIP

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

XIP mode at power-on reset

6.14.2.99 IS25LP080D_SR_QE

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

&

6.14.2.100 IS25LP080D_SR_SRWREN

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

Status register write enable/disable

6.14.2.101 IS25LP080D_SR_WIP

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

IS25LP08D Registers

Status Register Write in progress

6.14.2.102 IS25LP080D_SR_WREN

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

Write enable latch

6.14.2.103 IS25LP080D_VCR_NB_DUMMY

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

Number of dummy clock cycles

6.14.2.104 IS25LP080D_VCR_WRAP

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

Wrap

6.14.2.105 IS25LP080D_VCR_XIP

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

XIP

6.14.2.106 MULTIPLE_IO_READ_ID_CMD [1/2]

```
#define MULTIPLE_IO_READ_ID_CMD 0xAF
```

&

6.14.2.107 MULTIPLE_IO_READ_ID_CMD [2/2]

```
#define MULTIPLE_IO_READ_ID_CMD 0xAF
```

&

6.14.2.108 NO_OP [1/2]

```
#define NO_OP 0x00
```

Cancels Reset Enable

6.14.2.109 NO_OP [2/2]

```
#define NO_OP 0x00
```

Cancels Reset Enable

6.14.2.110 PAGE_PROG_CMD [1/3]

```
#define PAGE_PROG_CMD 0x02
```

Page Program Operations

6.14.2.111 PAGE_PROG_CMD [2/3]

```
#define PAGE_PROG_CMD 0x02
```

Page Operations

Program Operations

6.14.2.112 PAGE_PROG_CMD [3/3]

```
#define PAGE_PROG_CMD 0x02
```

Page Operations

Program Operations

6.14.2.113 PROG_ERASE_RESUME_CMD [1/2]

```
#define PROG_ERASE_RESUME_CMD 0x7A
```

&

6.14.2.114 PROG_ERASE_RESUME_CMD [2/2]

```
#define PROG_ERASE_RESUME_CMD 0x7A
```

&

6.14.2.115 PROG_ERASE_SUSPEND_CMD [1/2]

```
#define PROG_ERASE_SUSPEND_CMD 0x75
```

&

6.14.2.116 PROG_ERASE_SUSPEND_CMD [2/2]

```
#define PROG_ERASE_SUSPEND_CMD 0x75
```

&

6.14.2.117 QUAD_IN_FAST_PROG_CMD

```
#define QUAD_IN_FAST_PROG_CMD 0x32
```

&

6.14.2.118 QUAD_IN_PAGE_PROG_CMD [1/2]

```
#define QUAD_IN_PAGE_PROG_CMD 0x32
```

&

6.14.2.119 QUAD_IN_PAGE_PROG_CMD [2/2]

```
#define QUAD_IN_PAGE_PROG_CMD 0x32
```

&

6.14.2.120 QUAD_INOUT_FAST_READ_CMD [1/2]

```
#define QUAD_INOUT_FAST_READ_CMD 0xEB
```

&

6.14.2.121 QUAD_INOUT_FAST_READ_CMD [2/2]

```
#define QUAD_INOUT_FAST_READ_CMD 0xEB
```

&

6.14.2.122 QUAD_INOUT_FAST_READ_DTR_CMD [1/2]

```
#define QUAD_INOUT_FAST_READ_DTR_CMD 0xED
```

&

6.14.2.123 QUAD_INOUT_FAST_READ_DTR_CMD [2/2]

```
#define QUAD_INOUT_FAST_READ_DTR_CMD 0xED
```

&

6.14.2.124 QUAD_OUT_FAST_READ_CMD [1/2]

```
#define QUAD_OUT_FAST_READ_CMD 0x6B
```

&

6.14.2.125 QUAD_OUT_FAST_READ_CMD [2/2]

```
#define QUAD_OUT_FAST_READ_CMD 0x6B
```

&

6.14.2.126 READ_CMD [1/2]

```
#define READ_CMD 0x03
```

Read Operations

6.14.2.127 READ_CMD [2/2]

```
#define READ_CMD 0x03
```

Read Operations

6.14.2.128 READ_EXT_READ_PARAM_CMD

```
#define READ_EXT_READ_PARAM_CMD 0x81
```

&

6.14.2.129 READ_FUNCTION_REGISTER [1/2]

```
#define READ_FUNCTION_REGISTER 0X48
```

&

6.14.2.130 READ_FUNCTION_REGISTER [2/2]

```
#define READ_FUNCTION_REGISTER 0X48
```

&

6.14.2.131 READ_ID_CMD [1/2]

```
#define READ_ID_CMD 0xAB
```

Identification Operations

6.14.2.132 READ_ID_CMD [2/2]

```
#define READ_ID_CMD 0xAB
```

Identification Operations

6.14.2.133 READ_ID_CMD2 [1/2]

```
#define READ_ID_CMD2 0x9F
```

&

6.14.2.134 READ_ID_CMD2 [2/2]

```
#define READ_ID_CMD2 0x9F
```

&

6.14.2.135 READ_MANUFACT_AND_ID [1/2]

```
#define READ_MANUFACT_AND_ID 0x90
```

&

6.14.2.136 READ_MANUFACT_AND_ID [2/2]

```
#define READ_MANUFACT_AND_ID 0x90
```

&

6.14.2.137 READ_READ_PARAM_REG_CMD

```
#define READ_READ_PARAM_REG_CMD 0x61
```

&

6.14.2.138 READ_SERIAL_FLASH_DISCO_PARAM_CMD [1/2]

```
#define READ_SERIAL_FLASH_DISCO_PARAM_CMD 0x5A
```

&

6.14.2.139 READ_SERIAL_FLASH_DISCO_PARAM_CMD [2/2]

```
#define READ_SERIAL_FLASH_DISCO_PARAM_CMD 0x5A
```

&

6.14.2.140 READ_STATUS_REG_CMD [1/2]

```
#define READ_STATUS_REG_CMD 0x05
```

Register Operations

6.14.2.141 READ_STATUS_REG_CMD [2/2]

```
#define READ_STATUS_REG_CMD 0x05
```

Register Operations

6.14.2.142 READ_UNIQUE_ID [1/2]

```
#define READ_UNIQUE_ID 0x4B
```

&

6.14.2.143 READ_UNIQUE_ID [2/2]

```
#define READ_UNIQUE_ID 0x4B
```

&

6.14.2.144 RESET_ENABLE_CMD [1/2]

```
#define RESET_ENABLE_CMD 0x66
```

Reset Operations

6.14.2.145 RESET_ENABLE_CMD [2/2]

```
#define RESET_ENABLE_CMD 0x66
```

Reset Operations

6.14.2.146 RESET_MEMORY_CMD [1/2]

```
#define RESET_MEMORY_CMD 0x99
```

&

6.14.2.147 RESET_MEMORY_CMD [2/2]

```
#define RESET_MEMORY_CMD 0x99
```

&

6.14.2.148 SECTOR_ERASE_CMD [1/2]

```
#define SECTOR_ERASE_CMD 0xd7
```

Erase Operations

6.14.2.149 SECTOR_ERASE_CMD [2/2]

```
#define SECTOR_ERASE_CMD 0xd7
```

Erase Operations

6.14.2.150 SECTOR_ERASE_QPI_CMD [1/2]

```
#define SECTOR_ERASE_QPI_CMD 0x20
```

&

6.14.2.151 SECTOR_ERASE_QPI_CMD [2/2]

```
#define SECTOR_ERASE_QPI_CMD 0x20
```

&

6.14.2.152 SECTOR_LOCK [1/2]

```
#define SECTOR_LOCK 0x24
```

&

6.14.2.153 SECTOR_LOCK [2/2]

```
#define SECTOR_LOCK 0x24
```

&

6.14.2.154 SECTOR_UNLOCK [1/2]

```
#define SECTOR_UNLOCK 0x26
```

&

6.14.2.155 SECTOR_UNLOCK [2/2]

```
#define SECTOR_UNLOCK 0x26
```

&

6.14.2.156 WRITE_DISABLE_CMD [1/2]

```
#define WRITE_DISABLE_CMD 0x04
```

&

6.14.2.157 WRITE_DISABLE_CMD [2/2]

```
#define WRITE_DISABLE_CMD 0x04
```

&

6.14.2.158 WRITE_ENABLE_CMD [1/2]

```
#define WRITE_ENABLE_CMD 0x06
```

Write Operations

6.14.2.159 WRITE_ENABLE_CMD [2/2]

```
#define WRITE_ENABLE_CMD 0x06
```

Write Operations

6.14.2.160 WRITE_EXT_NV_READ_PARAM_REG_CMD

```
#define WRITE_EXT_NV_READ_PARAM_REG_CMD 0x85
```

non-volatile

6.14.2.161 WRITE_EXT_READ_PARAM_REG_CMD

```
#define WRITE_EXT_READ_PARAM_REG_CMD 0x83
```

volatile

6.14.2.162 WRITE_FUNCTION_REGISTER [1/2]

```
#define WRITE_FUNCTION_REGISTER 0x42
```

&

6.14.2.163 WRITE_FUNCTION_REGISTER [2/2]

```
#define WRITE_FUNCTION_REGISTER 0x42
```

&

6.14.2.164 WRITE_NV_READ_PARAM_REG_CMD

```
#define WRITE_NV_READ_PARAM_REG_CMD 0x65
```

non-volatile

6.14.2.165 WRITE_READ_PARAM_REG_CMD [1/2]

```
#define WRITE_READ_PARAM_REG_CMD 0xC0
```

&

6.14.2.166 WRITE_READ_PARAM_REG_CMD [2/2]

```
#define WRITE_READ_PARAM_REG_CMD 0xC0
```

volatile

6.14.2.167 WRITE_STATUS_REG_CMD [1/2]

```
#define WRITE_STATUS_REG_CMD 0x01
```

&

6.14.2.168 WRITE_STATUS_REG_CMD [2/2]

```
#define WRITE_STATUS_REG_CMD 0x01
```

&

6.15 CODEC

Audio codecs.

Audio codecs.

Driver for the TI PCM3060 Audio Codec.

Ak4556 Codec support.

Author

shensley

I don't see any real reason to have this be more than a function, but in case we want to add other functions down the road I wrapped the function in a class.

For now this is a limited interface that uses I2C to communicate with the PCM3060 The device can also be accessed with SPI, which is not yet supported.

For now all registers are set to their defaults, and the Init function will perform a MRST and SRST before setting the format to 24bit LJ, and disabling power save for both the ADC and DAC.

6.16 LED

LED driver devices.

LED driver devices.

6.17 SDRAM

SDRAM devices.

Classes

- struct dsy_sdram_handle

Macros

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

Enumerations

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

Functions

- uint8_t dsy_sdram_init (dsy_sdram_handle *dsy_hsdram)

6.17.1 Detailed Description

SDRAM devices.

6.17.2 Macro Definition Documentation

6.17.2.1 DSY_SDRAM_BSS

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

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

6.17.2.2 DSY_SDRAM_DATA

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

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

6.17.3 Enumeration Type Documentation

6.17.3.1 anonymous enum

```
anonymous enum
```

Enumerator

DSY_SDRAM_OK	&
DSY_SDRAM_ERR	&

6.17.3.2 dsy_sdram_pin

```
enum dsy_sdram_pin
```

This is PH5 on Daisy

Enumerator

DSY_SDRAM_PIN_SDNWE	&
DSY_SDRAM_PIN_LAST	&

6.17.3.3 dsy_sdram_state

```
enum dsy_sdram_state
```

Determines whether chip is initialized, and activated.

Enumerator

DSY_SDRAM_STATE_ENABLE	&
DSY_SDRAM_STATE_DISABLE	&
DSY_SDRAM_STATE_LAST	&

6.17.4 Function Documentation

6.17.4.1 dsy_sdram_init()

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

Initializes the SDRAM peripheral

6.18 BOARDS

Daisy devices. Pod, seed, etc.

Classes

- class daisy::DaisyField
- class daisy::DaisyPatch

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

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

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

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

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

6.18.1 Detailed Description

Daisy devices. Pod, seed, etc.

6.19 UTILITY

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

Classes

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

Macros

- #define DMA_BUFFER_MEM_SECTION __attribute__((section(".sram1_bss")))
- #define DTCM_MEM_SECTION __attribute__((section(".dtcmram_bss")))
- #define 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
- #define S322F_SCALE 4.6566129e-10f

- #define F2S32_SCALE 2147483647.f
- #define BSP_SD_CardInfo DSY_SD_CardInfoTypeDef
- #define MSD_OK ((uint8_t)0x00)
- #define MSD_ERROR ((uint8_t)0x01)
- #define MSD_ERROR_SD_NOT_PRESENT ((uint8_t)0x02)
- #define SD_TRANSFER_OK ((uint8_t)0x00)
- #define SD_TRANSFER_BUSY ((uint8_t)0x01)
- #define SD_PRESENT ((uint8_t)0x01)
- #define SD_NOT_PRESENT ((uint8_t)0x00)
- #define SD_DATATIMEOUT ((uint32_t)100000000)

Enumerations

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

Functions

- FORCE_INLINE float cube (float x)
- FORCE_INLINE float s162f (int16_t x)
- FORCE_INLINE int16_t f2s16 (float x)
- FORCE_INLINE float s242f (int32_t x)
- FORCE_INLINE int32_t f2s24 (float x)
- FORCE_INLINE float s322f (int32_t x)
- FORCE_INLINE int32_t f2s32 (float x)
- FORCE_INLINE dsy_gpio_pin dsy_pin (dsy_gpio_port port, uint8_t pin)
- FORCE_INLINE uint8_t dsy_pin_cmp (dsy_gpio_pin *a, dsy_gpio_pin *b)
- uint8_t BSP_SD_Init (void)
- uint8_t BSP_SD_ITConfig (void)
- uint8_t BSP_SD_ReadBlocks (uint32_t *pData, uint32_t ReadAddr, uint32_t NumOfBlocks, uint32_t Timeout)
- uint8_t BSP_SD_WriteBlocks (uint32_t *pData, uint32_t WriteAddr, uint32_t NumOfBlocks, uint32_t Timeout)
- uint8_t BSP_SD_ReadBlocks_DMA (uint32_t *pData, uint32_t ReadAddr, uint32_t NumOfBlocks)
- uint8_t BSP_SD_WriteBlocks_DMA (uint32_t *pData, uint32_t WriteAddr, uint32_t NumOfBlocks)
- uint8_t BSP_SD_Erase (uint32_t StartAddr, uint32_t EndAddr)
- uint8_t BSP_SD_GetCardState (void)
- void BSP_SD_GetCardInfo (DSY_SD_CardInfoTypeDef *CardInfo)
- uint8_t BSP_SD_IsDetected (void)
- void BSP_SD_AbortCallback (void)
- void BSP_SD_WriteCpltCallback (void)
- void BSP_SD_ReadCpltCallback (void)
- GPIO_TypeDef * dsy_hal_map_get_port (const dsy_gpio_pin *p)
- uint16_t dsy_hal_map_get_pin (const dsy_gpio_pin *p)
- void dsy_get_unique_id (uint32_t *w0, uint32_t *w1, uint32_t *w2)

Variables

- FontDef Font_6x8
- FontDef Font_7x10
- FontDef Font_11x18
- FontDef Font_16x26

6.19.1 Detailed Description

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

6.19.2 Macro Definition Documentation

6.19.2.1 BSP_SD_CardInfo

```
#define BSP_SD_CardInfo DSY_SD_CardInfoTypeDef
```

&

6.19.2.2 DMA_BUFFER_MEM_SECTION

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

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

6.19.2.3 DTCM_MEM_SECTION

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

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

6.19.2.4 F2S16_SCALE

```
#define F2S16_SCALE 32767.0f
```

(2 ** 15) - 1

6.19.2.5 F2S24_SCALE

```
#define F2S24_SCALE 8388608.0f
```

2 ** 23

6.19.2.6 F2S32_SCALE

```
#define F2S32_SCALE 2147483647.f
```

(2 ** 31) - 1

6.19.2.7 FBIPMAX

```
#define FBIPMAX 0.999985f
```

close to 1.0f-LSB at 16 bit

6.19.2.8 FBIPMIN

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

- (1 - LSB)

6.19.2.9 MSD_ERROR

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

&

6.19.2.10 MSD_ERROR_SD_NOT_PRESENT

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

&

6.19.2.11 MSD_OK

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

&

6.19.2.12 S162F_SCALE

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

1 / (2∗∗ 15)

6.19.2.13 S242F_SCALE

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

1 / (2 ∗∗ 23)

6.19.2.14 S24SIGN

```
#define S24SIGN 0x800000
```

2 ** 23

6.19.2.15 S322F_SCALE

```
#define S322F_SCALE 4.6566129e-10f
```

1 / (2** 31)

6.19.2.16 SD_DATATIMEOUT

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

&

6.19.2.17 SD_NOT_PRESENT

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

&

6.19.2.18 SD_PRESENT

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

&

6.19.2.19 SD_TRANSFER_BUSY

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

&

6.19.2.20 SD_TRANSFER_OK

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

&

6.19.3 Enumeration Type Documentation

6.19.3.1 dsy_gpio_port

```
enum dsy_gpio_port
```

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

Enumerator

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

6.19.4 Function Documentation

6.19.4.1 BSP_SD_AbortCallback()

```
void BSP_SD_AbortCallback (
             void  )
```

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

Abort the callback

6.19.4.2 BSP_SD_Erase()

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

Erase a section of memory

Parameters

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

Returns

card state, ERROR, etc.

6.19.4.3 BSP_SD_GetCardInfo()

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

Parameters

∗CardInfo	Pointer to write card info to

Parameters

CardInfo	&

6.19.4.4 BSP_SD_GetCardState()

```
uint8_t BSP_SD_GetCardState (
            void  )
```

Returns

card state, ERROR, etc.

6.19.4.5 BSP_SD_Init()

```
uint8_t BSP_SD_Init (
            void  )
```

Returns

card state, ERROR, etc.

6.19.4.6 BSP_SD_IsDetected()

```
uint8_t BSP_SD_IsDetected (
            void  )
```

Returns

Is card detected

6.19.4.7 BSP_SD_ITConfig()

```
uint8_t BSP_SD_ITConfig (
             void  )
```

Returns

card state, ERROR, etc.

6.19.4.8 BSP_SD_ReadBlocks()

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

Parameters

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

Returns

OK ERROR, etc.

6.19.4.9 BSP_SD_ReadBlocks_DMA()

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

No timeout

Parameters

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

Returns

card state, ERROR, etc.

6.19.4.10 BSP_SD_ReadCpltCallback()

```
void BSP_SD_ReadCpltCallback (
             void  )
```

Write complete callback

6.19.4.11 BSP_SD_WriteBlocks()

```
uint8_t BSP_SD_WriteBlocks (
             uint32_t * pData,
             uint32_t WriteAddr,
             uint32_t NumOfBlocks,
             uint32_t Timeout )
```

Parameters

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

Returns

card state, ERROR, etc.

6.19.4.12 BSP_SD_WriteBlocks_DMA()

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

No timeout

Parameters

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

Returns

card state, ERROR, etc.

6.19.4.13 BSP_SD_WriteCpltCallback()

```
void BSP_SD_WriteCpltCallback (
             void  )
```

Read complete callback

6.19.4.14 cube()

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

Computes cube.

Parameters

x	Number to be cubed

Returns

x ^ 3

6.19.4.15 dsy_get_unique_id()

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

Returns 96-bit Unique ID of the MCU

Author

shensley

Date

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

Parameters

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

6.19.4.16 dsy_hal_map_get_pin()

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

Parameters

**p*	Pin pin to get

Returns

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

6.19.4.17 dsy_hal_map_get_port()

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

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

Parameters

**p*	Pin pin to get

Returns

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

6.19.4.18 dsy_pin()

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

Helper for creating pins from port/pin combos easily

6.19.4.19 dsy_pin_cmp()

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

Helper for testing sameness of two dsy_gpio_pins

Returns

1 if same, 0 if different

6.19.4.20 f2s16()

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

Converts float to Signed 16-bit < close to 1.0f-LSB at 16 bit

< - (1 - LSB)

< close to 1.0f-LSB at 16 bit

< - (1 - LSB)

< close to 1.0f-LSB at 16 bit

< close to 1.0f-LSB at 16 bit

< (2 ** 15) - 1

6.19.4.21 f2s24()

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

Converts float to Signed 24-bit < close to 1.0f-LSB at 16 bit

< - (1 - LSB)

< close to 1.0f-LSB at 16 bit

< - (1 - LSB)

< close to 1.0f-LSB at 16 bit

< close to 1.0f-LSB at 16 bit

< 2 ** 23

6.19.4.22 f2s32()

```
FORCE_INLINE int32_t f2s32 (
            float x )
```

Converts float to Signed 24-bit < 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 ** 31) - 1

6.19.4.23 s162f()

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

Converts Signed 16-bit to float

Parameters

x	Number to be scaled.

Returns

Scaled number.

< 1 / (2** 15)

6.19.4.24 s242f()

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

Converts Signed 24-bit to float < 2 ** 23

< 2 ** 23

< 1 / (2 ** 23)

6.19.4.25 s322f()

```
FORCE_INLINE float s322f (
            int32_t x )
```

Converts Signed 32-bit to float < 1 / (2** 31)

6.19.5 Variable Documentation

6.19.5.1 Font_11x18

```
FontDef Font_11x18  [extern]
```

&

6.19.5.2 Font_16x26

```
FontDef Font_16x26  [extern]
```

&

6.19.5.3 Font_6x8

```
FontDef Font_6x8  [extern]
```

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

6.19.5.4 Font_7x10

```
FontDef Font_7x10  [extern]
```

&

6.20 USBD_CDC_IF

Usb VCP device module.

Modules

- USBD_CDC_IF_Exported_Defines
  Defines.
- USBD_CDC_IF_Exported_Types
  Types.
- USBD_CDC_IF_Exported_Macros
  Aliases.
- USBD_CDC_IF_Exported_Variables
  Public variables.
- USBD_CDC_IF_Exported_FunctionsPrototype
  Public functions declaration.

6.20.1 Detailed Description

Usb VCP device module.

6.21 USBD_CDC_IF_Exported_Defines

Defines.

Defines.

6.22 USBD_CDC_IF_Exported_Types

Types.

Typedefs

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

6.22.1 Detailed Description

Types.

6.22.2 Typedef Documentation

6.22.2.1 CDC_ReceiveCallback

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

Parameters

buf	buffer
size	buffer size

6.23 USBD_CDC_IF_Exported_Macros

Aliases.

Aliases.

6.24 USBD_CDC_IF_Exported_Variables

Public variables.

Variables

- USBD_CDC_ItfTypeDef USBD_Interface_fops_FS
- USBD_CDC_ItfTypeDef USBD_Interface_fops_HS

6.24.1 Detailed Description

Public variables.

6.24.2 Variable Documentation

6.24.2.1 USBD_Interface_fops_FS

```
USBD_CDC_ItfTypeDef USBD_Interface_fops_FS  [extern]
```

CDC Interface callback.

6.24.2.2 USBD_Interface_fops_HS

```
USBD_CDC_ItfTypeDef USBD_Interface_fops_HS  [extern]
```

CDC Interface callback.

6.25 USBD_CDC_IF_Exported_FunctionsPrototype

Public functions declaration.

Functions

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

6.25.1 Detailed Description

Public functions declaration.

6.25.2 Function Documentation

6.25.2.1 CDC_Set_Rx_Callback_FS()

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

&

6.25.2.2 CDC_Set_Rx_Callback_HS()

```
void CDC_Set_Rx_Callback_HS (
            CDC_ReceiveCallback cb )
```

&

6.25.2.3 CDC_Transmit_FS()

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

&

6.25.2.4 CDC_Transmit_HS()

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

&

6.26 USBD_CONF

Configuration file for Usb otg low level driver.

Modules

- USBD_CONF_Exported_Variables
  Public variables.
- USBD_CONF_Exported_Defines
  Defines for configuration of the Usb device.
- USBD_CONF_Exported_Macros
  Aliases.
- USBD_CONF_Exported_Types
  Types.
- USBD_CONF_Exported_FunctionsPrototype
  Declaration of public functions for Usb device.

6.26.1 Detailed Description

Configuration file for Usb otg low level driver.

6.27 USBD_CONF_Exported_Variables

Public variables.

Public variables.

6.28 USBD_CONF_Exported_Defines

Defines for configuration of the Usb device.

Macros

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

6.28.1 Detailed Description

Defines for configuration of the Usb device.

6.28.2 Macro Definition Documentation

6.28.2.1 DEVICE_FS

```
#define DEVICE_FS 0
```

FS and HS identification

6.28.2.2 DEVICE_HS

```
#define DEVICE_HS 1
```

&

6.28.2.3 USBD_DEBUG_LEVEL

```
#define USBD_DEBUG_LEVEL 3U
```

&

6.28.2.4 USBD_LPM_ENABLED

```
#define USBD_LPM_ENABLED 0U
```

&

6.28.2.5 USBD_MAX_NUM_CONFIGURATION

```
#define USBD_MAX_NUM_CONFIGURATION 1U
```

&

6.28.2.6 USBD_MAX_NUM_INTERFACES

```
#define USBD_MAX_NUM_INTERFACES 1U
```

&

6.28.2.7 USBD_MAX_STR_DESC_SIZ

```
#define USBD_MAX_STR_DESC_SIZ 512U
```

&

6.28.2.8 USBD_SELF_POWERED

```
#define USBD_SELF_POWERED 1U
```

&

6.28.2.9 USBD_SUPPORT_USER_STRING

```
#define USBD_SUPPORT_USER_STRING 0U
```

&

6.29 USBD_CONF_Exported_Macros

Aliases.

Macros

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

6.29.1 Detailed Description

Aliases.

6.29.2 Macro Definition Documentation

6.29.2.1 USBD_DbgLog

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

Value:

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

&

6.29.2.2 USBD_Delay

```
#define USBD_Delay HAL_Delay
```

Alias for delay.

6.29.2.3 USBD_ErrLog

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

Value:

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

&

6.29.2.4 USBD_free

```
#define USBD_free free
```

Alias for memory release.

6.29.2.5 USBD_malloc

```
#define USBD_malloc malloc
```

Alias for memory allocation.

6.29.2.6 USBD_memcpy

```
#define USBD_memcpy memcpy
```

Alias for memory copy.

6.29.2.7 USBD_memset

```
#define USBD_memset memset
```

Alias for memory set.

6.29.2.8 USBD_UsrLog

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

Value:

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

&

6.30 USBD_CONF_Exported_Types

Types.

Types.

6.31 USBD_CONF_Exported_FunctionsPrototype

Declaration of public functions for Usb device.

Declaration of public functions for Usb device.

6.32 USBD_DESC

Usb device descriptors module.

Modules

- USBD_DESC_Exported_Constants
  Constants.
- USBD_DESC_Exported_Defines
  Defines.
- USBD_DESC_Exported_TypesDefinitions
  Types.
- USBD_DESC_Exported_Macros
  Aliases.
- USBD_DESC_Exported_Variables
  Public variables.
- USBD_DESC_Exported_FunctionsPrototype
  Public functions declaration.

6.32.1 Detailed Description

Usb device descriptors module.

6.33 USBD_DESC_Exported_Constants

Constants.

Macros

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

6.33.1 Detailed Description

Constants.

6.33.2 Macro Definition Documentation

6.33.2.1 DEVICE_ID1

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

&

6.33.2.2 DEVICE_ID2

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

&

6.33.2.3 DEVICE_ID3

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

&

6.33.2.4 USB_SIZ_STRING_SERIAL

```
#define USB_SIZ_STRING_SERIAL 0x1A
```

&

6.34 USBD_DESC_Exported_Defines

Defines.

Defines.

6.35 USBD_DESC_Exported_TypesDefinitions

Types.

Types.

6.36 USBD_DESC_Exported_Macros

Aliases.

Aliases.

6.37 USBD_DESC_Exported_Variables

Public variables.

Variables

- USBD_DescriptorsTypeDef HS_Desc
- USBD_DescriptorsTypeDef FS_Desc

6.37.1 Detailed Description

Public variables.

6.37.2 Variable Documentation

6.37.2.1 FS_Desc

```
USBD_DescriptorsTypeDef FS_Desc  [extern]
```

Descriptor for the Usb device.

6.37.2.2 HS_Desc

```
USBD_DescriptorsTypeDef HS_Desc  [extern]
```

Descriptor for the Usb device.

6.38 USBD_DESC_Exported_FunctionsPrototype

Public functions declaration.

Public functions declaration.

6.39 Externals

6.40 STM32_USB_OTG_DEVICE_LIBRARY

For Usb device.

Modules

- USBD_CDC_IF

  Usb VCP device module.
- USBD_DESC

  Usb device descriptors module.

6.40.1 Detailed Description

For Usb device.

< Define to prevent recursive inclusion -------------------------------—

6.41 USBD_OTG_DRIVER

Modules

- USBD_CONF

  Configuration file for Usb otg low level driver.

6.41.1 Detailed Description

Chapter 7

Namespace Documentation

7.1 daisy Namespace Reference

Hardware defines and helpers for daisy field platform.

Classes

- class DaisyField
- 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 DaisyVersio

  Class that handles initializing all of the hardware specific to the Desmodus Versio hardware.
  Helper funtions are also in place to provide easy access to built-in controls and peripherals.
- class Ak4556
- class Pcm3060
- class Wm8731
- class LedDriverPca9685
- class ShiftRegister4021
- class AudioHandle
- class AnalogControl

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

  Interface for simple USB logging.
- class Logger< LOGGER_NONE >
- class LoggerImpl

  Logging I/O underlying implementation.
- class LoggerImpl< LOGGER_INTERNAL >

  Specialization for internal USB port.
- class LoggerImpl< LOGGER_EXTERNAL >

  Specialization for external USB port.
- class LoggerImpl< LOGGER_SEMIHOST >

  Specialization for semihosting (stdout)
- struct NoteOnEvent
- struct ControlChangeEvent
- 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.
- class OledDisplay
- class Parameter
- class RgbLed
- class Switch
- class Switch3
- struct WavFileInfo
- class WavPlayer
- struct AdcChannelConfig
- class AdcHandle
- class DacHandle
- class I2CHandle
- class SaiHandle
- struct SdmmcHandlerInit
- class SdmmcHandler
- class SpiHandle
- class TimerHandle
- class UartHandler
- class System
- class Color
- class RingBuffer
- class RingBuffer< T, 0 >
- class ScopedIrqBlocker

Enumerations

- enum LoggerDestination { LOGGER_NONE , LOGGER_INTERNAL , LOGGER_EXTERNAL , LOGGER_SEMIHOST }
- 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

- void dsy_i2c_global_init ()

7.1.1 Detailed Description

Hardware defines and helpers for daisy field platform.

7.1.2 Enumeration Type Documentation

7.1.2.1 LoggerDestination

```
enum daisy::LoggerDestination
```

Enumeration of destination ports for debug logging

Enumerator

LOGGER_NONE	mute logging
LOGGER_INTERNAL	internal USB port
LOGGER_EXTERNAL	external USB port
LOGGER_SEMIHOST	stdout

7.1.3 Function Documentation

7.1.3.1 dsy_i2c_global_init()

```
void daisy::dsy_i2c_global_init ( )
```

internal. Used for global init.

Chapter 8

Class Documentation

8.1 daisy::AdcChannelConfig Struct Reference

`#include <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, size_t mux_channels, dsy_gpio_pin mux_0, dsy_gpio_pin mux_↩ 1={DSY_GPIOX, 0}, dsy_gpio_pin mux_2={DSY_GPIOX, 0})

Public Attributes

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

8.1.1 Detailed Description

Configuration Structure for a given channel

8.1.2 Member Enumeration Documentation

8.1.2.1 MuxPin

`enum daisy::AdcChannelConfig::MuxPin`

Which pin to use for multiplexing

Enumerator

MUX_SEL_0	&
MUX_SEL_1	&
MUX_SEL_2	&
MUX_SEL_LAST	&

8.1.3 Member Function Documentation

8.1.3.1 InitMux()

```
void daisy::AdcChannelConfig::InitMux (
            dsy_gpio_pin adc_pin,
            size_t mux_channels,
            dsy_gpio_pin mux_0,
            dsy_gpio_pin mux_1 = {DSY_GPIOX, 0},
            dsy_gpio_pin mux_2 = {DSY_GPIOX, 0} )
```

Initializes a single ADC pin as a Multiplexed ADC. Requires a CD405X Multiplexer connected to the pin. You only need to supply the mux pins that are required, e.g. a 4052 mux would only require mux_0 and mux_1. Internal Callbacks handle the pin addressing.

Parameters

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

8.1.3.2 InitSingle()

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

Initializes a single ADC pin as an ADC.

Parameters

pin	Pin to init.

8.1.4 Member Data Documentation

8.1.4.1 mux_channels_

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

&

8.1.4.2 mux_pin_

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

&

8.1.4.3 pin_

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

&

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

- src/per/adc.h

8.2 daisy::AdcHandle Class Reference

```
#include <adc.h>
```

Public Types

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

Public Member Functions

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

8.2.1 Detailed Description

Handler for analog to digital conversion

8.2.2 Member Enumeration Documentation

8.2.2.1 OverSampling

```
enum daisy::AdcHandle::OverSampling
```

Supported oversampling amounts

Enumerator

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

8.2.3 Member Function Documentation

8.2.3.1 Get()

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

Single channel getter

Parameters

chn	channel to get

Returns

Converted value

8.2.3.2 GetFloat()

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

Get floating point from single channel

Parameters

chn	Channel to get from

Returns

Floating point converted value

8.2.3.3 GetMux()

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

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

Parameters

chn	Channel to get from
idx	&

Returns

data

8.2.3.4 GetMuxFloat()

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

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

Parameters

chn	Channel to get from
idx	&

Returns

Floating point data

8.2.3.5 GetMuxPtr()

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

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

Parameters

chn	Channel to get from
idx	&

Returns

Pointer to data

8.2.3.6 GetPtr()

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

Get pointer to a value from a single channel

Parameters

chn	

Returns

Pointer to converted value

8.2.3.7 Init()

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

Initializes the ADC with the pins passed in.

Parameters

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

8.2.3.8 Start()

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

Starts reading from the ADC

8.2.3.9 Stop()

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

Stops reading from the ADC

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

- src/per/adc.h

8.3 daisy::Ak4556 Class Reference

Static Public Member Functions

- static void Init (dsy_gpio_pin reset_pin)

8.3.1 Member Function Documentation

8.3.1.1 Init()

```
static void daisy::Ak4556::Init (
            dsy_gpio_pin reset_pin )  [static]
```

Initialization function for Ak4556 Can be called statically: Ak4556::Init(pin);

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

- src/dev/codec_ak4556.h

8.4 daisy::AnalogControl Class Reference

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

```
#include <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
- void SetCoeff (float val)
- uint16_t GetRawValue ()
- float GetRawFloat ()

8.4.1 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

8.4.2 Constructor & Destructor Documentation

8.4.2.1 AnalogControl()

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

Constructor

8.4.2.2 ~AnalogControl()

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

destructor

8.4.3 Member Function Documentation

8.4.3.1 GetRawFloat()

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

Returns a normalized float value representing the current ADC value.

8.4.3.2 GetRawValue()

```
uint16_t daisy::AnalogControl::GetRawValue ( )  [inline]
```

Returns the raw unsigned 16-bit value from the ADC

8.4.3.3 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.

8.4.3.4 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

8.4.3.5 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

8.4.3.6 SetCoeff()

```
void daisy::AnalogControl::SetCoeff (
            float val )  [inline]
```

Directly set the Coefficient of the one pole smoothing filter.

8.4.3.7 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

8.5 daisy::AudioHandle Class Reference

Classes

- struct Config

Public Types

- enum class **Result** { **OK** , **ERR** }
- typedef void(∗ AudioCallback) (float ∗∗in, float ∗∗out, size_t size)
- typedef void(∗ InterleavingAudioCallback) (float ∗in, float ∗out, size_t size)

Public Member Functions

- **AudioHandle** (const AudioHandle &other)=default
- AudioHandle & **operator=** (const AudioHandle &other)=default
- Result Init (const Config &config, SaiHandle sai)
- Result Init (const Config &config, SaiHandle sai1, SaiHandle sai2)
- const Config & GetConfig () const
- size_t GetChannels () const
- float GetSampleRate ()
- Result SetSampleRate (SaiHandle::Config::SampleRate samplerate)
- Result SetBlockSize (size_t size)
- Result SetPostGain (float val)
- Result Start (AudioCallback callback)
- Result Start (InterleavingAudioCallback callback)
- Result Stop ()
- Result ChangeCallback (AudioCallback callback)
- Result ChangeCallback (InterleavingAudioCallback callback)

8.5.1 Member Typedef Documentation

8.5.1.1 AudioCallback

```
typedef void(* daisy::AudioHandle::AudioCallback) (float **in, float **out, size_t size)
```

Non-Interleaving Callback format. Both arrays arranged by float[chn][sample]

8.5.1.2 InterleavingAudioCallback

```
typedef void(* daisy::AudioHandle::InterleavingAudioCallback) (float *in, float *out, size_↩
t size)
```

Non-Interleaving Callback format. audio is prepared as { L0, R0, L1, R1, . . . LN, RN }

8.5.2 Member Function Documentation

8.5.2.1 ChangeCallback() [1/2]

```
Result daisy::AudioHandle::ChangeCallback (
            AudioCallback callback )
```

Immediatley changes the audio callback to the non-interleaving callback passed in.

8.5.2.2 ChangeCallback() [2/2]

```
Result daisy::AudioHandle::ChangeCallback (
            InterleavingAudioCallback callback )
```

Immediatley changes the audio callback to the interleaving callback passed in.

8.5.2.3 GetChannels()

```
size_t daisy::AudioHandle::GetChannels ( ) const
```

Returns the number of channels of audio.

When using a single SAI this returns 2, when using two SAI it returns 4 If no SAI is initialized this returns 0

Eventually when we add non-standard I2S for each SAI this will be work differently

8.5.2.4 GetConfig()

```
const Config& daisy::AudioHandle::GetConfig ( ) const
```

Returns the Global Configuration struct for the Audio

8.5.2.5 GetSampleRate()

```
float daisy::AudioHandle::GetSampleRate ( )
```

Returns the Samplerate as a float

8.5.2.6 Init() [1/2]

```
Result daisy::AudioHandle::Init (
            const Config & config,
            SaiHandle sai )
```

Initializes audio to run using a single SAI configured in Stereo I2S mode.

8.5.2.7 Init() [2/2]

```
Result daisy::AudioHandle::Init (
            const Config & config,
            SaiHandle sai1,
            SaiHandle sai2 )
```

Initializes audio to run using two SAI, each configured in Stereo I2S mode.

8.5.2.8 SetBlockSize()

```
Result daisy::AudioHandle::SetBlockSize (
            size_t size )
```

Sets the block size after initialization, and updates the internal configuration struct. Get BlockSize and other details via the GetConfig

8.5.2.9 SetPostGain()

```
Result daisy::AudioHandle::SetPostGain (
            float val )
```

Sets the amount of gain adjustment to perform before and after callback. useful if the hardware has additional headroom, and the nominal value shouldn't be 1.0

Parameters

val	Gain adjustment amount. The hardware will clip at the reciprical of this value.

8.5.2.10 SetSampleRate()

```
Result daisy::AudioHandle::SetSampleRate (
            SaiHandle::Config::SampleRate samplerate )
```

Sets the samplerate, and reinitializes the sai as needed.

8.5.2.11 Start() [1/2]

```
Result daisy::AudioHandle::Start (
            AudioCallback callback )
```

Starts the Audio using the non-interleaving callback.

8.5.2.12 Start() [2/2]

```
Result daisy::AudioHandle::Start (
            InterleavingAudioCallback callback )
```

Starts the Audio using the interleaving callback. For now only two channels are supported via this method.

8.5.2.13 Stop()

```
Result daisy::AudioHandle::Stop ( )
```

Stop the Audio

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

- src/hid/audio.h

8.6 daisy::Color Class Reference

```
#include <color.h>
```

Public Types

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

Public Member Functions

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

8.6.1 Detailed Description

Class for handling simple colors

8.6.2 Member Enumeration Documentation

8.6.2.1 PresetColor

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

List of colors that have a preset RGB value

Enumerator

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

8.6.3 Member Function Documentation

8.6.3.1 Blue()

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

Returns the 0-1 value for Blue

8.6.3.2 Green()

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

Returns the 0-1 value for Green

8.6.3.3 Init() [1/2]

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

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

Parameters

red	Red value
green	Green value
blue	Blue value

8.6.3.4 Init() [2/2]

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

Initializes the Color with a given preset.

Parameters

c	Color to init to

8.6.3.5 Red()

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

Returns the 0-1 value for Red

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

- src/util/color.h

8.7 daisy::AudioHandle::Config Struct Reference

```
#include <audio.h>
```

Public Attributes

- size_t **blocksize**
- SaiHandle::Config::SampleRate **samplerate**
- float **postgain**

8.7.1 Detailed Description

Manually configurable details about the Audio Engine TODO: Figure out how to get samplerate in here.

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

- src/hid/audio.h

8.8 daisy::DacHandle::Config Struct Reference

```
#include <dac.h>
```

Public Attributes

- uint32_t target_samplerate
- Channel **chn**
- Mode **mode**
- BitDepth **bitdepth**
- BufferState **buff_state**

8.8.1 Detailed Description

Configuration structure for initializing the DAC structure.

8.8.2 Member Data Documentation

8.8.2.1 target_samplerate

```
uint32_t daisy::DacHandle::Config::target_samplerate
```

Target Samplerate in Hz used to configure the internal timebase for DMA mode. Does nothing in POLLING mode. If the value is 0 at Init time this will default to 48000Hz otherwise the driver will attempt meet the target.

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

- src/per/dac.h

8.9 daisy::I2CHandle::Config Struct Reference

```
#include <i2c.h>
```

Public Types

- enum class Peripheral { I2C_1 = 0 , I2C_2 , I2C_3 , I2C_4 }
- enum class Speed { I2C_100KHZ , I2C_400KHZ , I2C_1MHZ }

Public Attributes

- Peripheral periph
- struct {
  dsy_gpio_pin scl
  dsy_gpio_pin sda
  } pin_config
- Speed speed

8.9.1 Detailed Description

Contains settings for initialising an I2C interface.

8.9.2 Member Enumeration Documentation

8.9.2.1 Peripheral

enum daisy::I2CHandle::Config::Peripheral [strong]

Specifices the internal peripheral to use (these are mapped to different pins on the hardware).

Enumerator

I2C_1	&
I2C_2	&
I2C_3	&
I2C_4	&

8.9.2.2 Speed

enum daisy::I2CHandle::Config::Speed [strong]

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

Enumerator

I2C_100KHZ	&
I2C_400KHZ	&
I2C_1MHZ	&

8.9.3 Member Data Documentation

8.9.3.1 periph

```
Peripheral daisy::I2CHandle::Config::periph
```

&

8.9.3.2

```
struct { ... } daisy::I2CHandle::Config::pin_config
```

&

8.9.3.3 scl

```
dsy_gpio_pin daisy::I2CHandle::Config::scl
```

&

8.9.3.4 sda

```
dsy_gpio_pin daisy::I2CHandle::Config::sda
```

&

8.9.3.5 speed

```
Speed daisy::I2CHandle::Config::speed
```

&

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

- src/per/i2c.h

8.10 daisy::SaiHandle::Config Struct Reference

```
#include <sai.h>
```

Public Types

- enum class Peripheral { **SAI_1** , **SAI_2** }
- enum class SampleRate {
  SAI_8KHZ , **SAI_16KHZ** , **SAI_32KHZ** , **SAI_48KHZ** ,
  SAI_96KHZ }
- enum class BitDepth { **SAI_16BIT** , **SAI_24BIT** , **SAI_32BIT** }
- enum class Sync { **MASTER** , **SLAVE** }
- enum class Direction { **TRANSMIT** , **RECEIVE** }

Public Attributes

- Peripheral **periph**
- struct {
  dsy_gpio_pin **mclk**
  dsy_gpio_pin **fs**
  dsy_gpio_pin **sck**
  dsy_gpio_pin **sa**
  dsy_gpio_pin **sb**
  } **pin_config**
- SampleRate **sr**
- BitDepth **bit_depth**
- Sync **a_sync**
- Sync **b_sync**
- Direction **a_dir**
- Direction **b_dir**

8.10.1 Detailed Description

Contains settings for initialising an SAI Interface

8.10.2 Member Enumeration Documentation

8.10.2.1 BitDepth

`enum daisy::SaiHandle::Config::BitDepth [strong]`

Bit Depth that the hardware expects to be transferred to/from the device.

8.10.2.2 Direction

`enum daisy::SaiHandle::Config::Direction [strong]`

Specifies the direction for each peripheral block.

8.10.2.3 Peripheral

`enum daisy::SaiHandle::Config::Peripheral [strong]`

Specifies the internal peripheral to use (mapped to different hardware pins)

8.10.2.4 SampleRate

```
enum daisy::SaiHandle::Config::SampleRate  [strong]
```

Rate at which samples will be streaming to/from the device.

8.10.2.5 Sync

```
enum daisy::SaiHandle::Config::Sync  [strong]
```

Specifies whether a particular block is the master or the slave If both are set to slave, no MCLK signal will be used, and it is expected that the codec will have its own xtal.

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

- src/per/sai.h

8.11 daisy::ShiftRegister4021< num_daisychained, num_parallel >::Config Struct Reference

```
#include <sr_4021.h>
```

Public Attributes

- dsy_gpio_pin clk
- dsy_gpio_pin latch
- dsy_gpio_pin data [num_parallel]

8.11.1 Detailed Description

template<size_t num_daisychained = 1, size_t num_parallel = 1>
struct daisy::ShiftRegister4021< num_daisychained, num_parallel >::Config

Configuration Structure for handling the pin setting of the device

8.11.2 Member Data Documentation

8.11.2.1 clk

```
template<size_t num_daisychained = 1, size_t num_parallel = 1>
dsy_gpio_pin daisy::ShiftRegister4021< num_daisychained, num_parallel >::Config::clk
```

Clock pin to attach to pin 10 of device(s)

8.11.2.2 data

```
template<size_t num_daisychained = 1, size_t num_parallel = 1>
dsy_gpio_pin daisy::ShiftRegister4021< num_daisychained, num_parallel >::Config::data[num_↩
parallel]
```

Data Pin(s)

8.11.2.3 latch

```
template<size_t num_daisychained = 1, size_t num_parallel = 1>
dsy_gpio_pin daisy::ShiftRegister4021< num_daisychained, num_parallel >::Config::latch
```

Latch pin to attach to pin 9 of device(s)

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

- src/dev/sr_4021.h

8.12 daisy::System::Config Struct Reference

`#include <system.h>`

Public Types

- enum class SysClkFreq { **FREQ_400MHZ** , **FREQ_480MHZ** }

Public Member Functions

- void Defaults ()
- void Boost ()

Public Attributes

- SysClkFreq **cpu_freq**
- bool **use_dcache**
- bool **use_icache**

8.12.1 Detailed Description

Contains settings for initializing the System

8.12.2 Member Enumeration Documentation

8.12.2.1 SysClkFreq

`enum daisy::System::Config::SysClkFreq [strong]`

Specifies the system clock frequency that feeds APB/AHB clocks, etc.

8.12.3 Member Function Documentation

8.12.3.1 Boost()

`void daisy::System::Config::Boost ( ) [inline]`

Method to call on the struct to set to boost mode: CPU Freq set to 480MHz Cache Enabled

8.12.3.2 Defaults()

`void daisy::System::Config::Defaults ( ) [inline]`

Method to call on the struct to set to defaults CPU Freq set to 400MHz Cache Enabled

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

- src/sys/system.h

8.13 daisy::TimerHandle::Config Struct Reference

Public Types

- enum class Peripheral { TIM_2 = 0 , TIM_3 , TIM_4 , TIM_5 }
- enum class CounterDir { **UP** = 0 , **DOWN** }

Public Attributes

- Peripheral **periph**
- CounterDir **dir**

8.13.1 Member Enumeration Documentation

8.13.1.1 CounterDir

`enum daisy::TimerHandle::Config::CounterDir [strong]`

Direction of the auto-reload counter. TODO: Add support for the various versions of Up/Down counters.

8.13.1.2 Peripheral

`enum daisy::TimerHandle::Config::Peripheral [strong]`

Hardwaare Timer to configure, and use.

Enumerator

TIM↩_2	32-bit counter
TIM↩_3	16-bit counter
TIM↩_4	16-bit counter
TIM↩_5	32-bit counter

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

- src/per/tim.h

8.14 daisy::Wm8731::Config Struct Reference

`#include <codec_wm8731.h>`

Public Types

- enum class Format { **MSB_FIRST_RJ** = 0x00 , **MSB_FIRST_LJ** = 0x01 , **I2S** = 0x02 , **DSP** = 0x03 }
- enum class WordLength { **BITS_16** = (0x00 << 2) , **BITS_20** = (0x01 << 2) , **BITS_24** = (0x02 << 2) , **BITS_32** = (0x03 << 2) }

Public Member Functions

- void Defaults ()

Public Attributes

- bool mcu_is_master
- bool lr_swap
- bool csb_pin_state
- Format **fmt**
- WordLength **wl**

8.14.1 Detailed Description

Configuration struct for use in initializing the device. For now, only 48kHz is supported. USB Mode is also not yet supported.

8.14.2 Member Enumeration Documentation

8.14.2.1 Format

```
enum daisy::Wm8731::Config::Format  [strong]
```

Sets the communication format used

8.14.2.2 WordLength

```
enum daisy::Wm8731::Config::WordLength  [strong]
```

Defines the size of a sample in bits This is for communication only, the device processes audio at 24-bits, and the strips/pads bits to send to the processor.

8.14.3 Member Function Documentation

8.14.3.1 Defaults()

```
void daisy::Wm8731::Config::Defaults ( )  [inline]
```

Sets the following config: MCU is master = true L/R Swap = false CSB Pin state = false Format = MSB First LJ WordLength = 24-bit

8.14.4 Member Data Documentation

8.14.4.1 csb_pin_state

```
bool daisy::Wm8731::Config::csb_pin_state
```

Set true if tied high, and false if tied low. determines the I2C address for communicating with the device

8.14.4.2 lr_swap

```
bool daisy::Wm8731::Config::lr_swap
```

Sets whether the left/right channels are swapped or not.

8.14.4.3 mcu_is_master

```
bool daisy::Wm8731::Config::mcu_is_master
```

Sets the device to slave mode if true, and master mode if false.

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

- src/dev/codec_wm8731.h

8.15 daisy::ControlChangeEvent Struct Reference

```
#include <midi.h>
```

Public Attributes

- int channel
- uint8_t control_number
- uint8_t value

8.15.1 Detailed Description

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

8.15.2 Member Data Documentation

8.15.2.1 channel

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

&

8.15.2.2 control_number

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

&

8.15.2.3 value

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

&

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

- src/hid/midi.h

8.16 daisy::DacHandle Class Reference

```
#include <dac.h>
```

Classes

- struct Config

Public Types

- enum class Result { **OK** , **ERR** }
- enum class Channel { **ONE** , **TWO** , **BOTH** }
- enum class Mode { **POLLING** , **DMA** }
- enum class BitDepth { **BITS_8** , **BITS_12** }
- enum class BufferState { **ENABLED** , **DISABLED** }
- typedef void(∗ DacCallback) (uint16_t ∗∗out, size_t size)

Public Member Functions

- **DacHandle** (const DacHandle &other)=default
- DacHandle & **operator=** (const DacHandle &other)=default
- Result Init (const Config &config)
- const Config & **GetConfig** () const
- Result Start (uint16_t ∗buffer, size_t size, DacCallback cb)
- Result Start (uint16_t ∗buffer_1, uint16_t ∗buffer_2, size_t size, DacCallback cb)
- Result Stop ()
- Result WriteValue (Channel chn, uint16_t val)

8.16.1 Detailed Description

DAC handle for Built-in DAC Peripheral

For now only Normal Mode is supported, Sample and hold mode provides reduced power consumption, but requires a bit more setup.

For now connecting the DAC through other internal periphearls is also not supported.

Since the DAC channels have dedicated pins we don't need to pass in a pin config like with other modules. However, it is still important to not try to use the DAC pins for anything else. DAC Channel 1 is on PA4, and DAC Channel 2 is on PA5

8.16.2 Member Typedef Documentation

8.16.2.1 DacCallback

```
typedef void(* daisy::DacHandle::DacCallback) (uint16_t **out, size_t size)
```

Callback for DMA transfers. This is called every time half of the samples of the buffer are transmitted, and the buffer is readdy to be filled again.

The data is organized in arrays per channel, for example if both channels are in use: { {ch1-0, ch1-1, ch1-2 . . . ch1-N}, {ch2-0, ch2-1, ch2-2 . . . ch2–N} }

8.16.3 Member Enumeration Documentation

8.16.3.1 BitDepth

```
enum daisy::DacHandle::BitDepth  [strong]
```

Sets the number of bits per sample transmitted out of the DAC. The output range will be: 0V - VDDA The resolution will be roughly: bitdepth / (VDDA - 0V)

8.16.3.2 BufferState

```
enum daisy::DacHandle::BufferState  [strong]
```

Sets whether the DAC output is buffered for higher drive ability.

8.16.3.3 Channel

```
enum daisy::DacHandle::Channel  [strong]
```

Selects which channel(s) will be configured for use.

8.16.3.4 Mode

```
enum daisy::DacHandle::Mode  [strong]
```

Sets the Mode for the DAC channels.

Polling mode uses the blocking mode to transmit a single value at a time.

DMA mode uses a buffer, and periodically transmits it triggering a callback to fill the buffer when it is ready for more samples.

8.16.3.5 Result

```
enum daisy::DacHandle::Result  [strong]
```

Return Values for the DacHandle class

8.16.4 Member Function Documentation

8.16.4.1 Init()

```
Result daisy::DacHandle::Init (
            const Config & config )
```

Initialize the DAC Peripheral

8.16.4.2 Start() [1/2]

```
Result daisy::DacHandle::Start (
            uint16_t * buffer,
            size_t size,
            DacCallback cb )
```

Starts the DAC conversion on the DMA calling the user callback whenever new samples are ready to be filled.

This will return Result::ERR if used when configured to BOTH channels.

8.16.4.3 Start() [2/2]

```
Result daisy::DacHandle::Start (
            uint16_t * buffer_1,
            uint16_t * buffer_2,
            size_t size,
            DacCallback cb )
```

If using both channels, use this function to start the DMA transfer for both. The callback will provide an array per-channel to fill.

8.16.4.4 Stop()

```
Result daisy::DacHandle::Stop ( )
```

Stops the DAC channel(s).

8.16.4.5 WriteValue()

```
Result daisy::DacHandle::WriteValue (
            Channel chn,
            uint16_t val )
```

Sets and Writes value in Polling Mode Has no effect in DMA mode.

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

- src/per/dac.h

8.17 daisy::DaisyField Class Reference

Public Types

- enum { SW_1 , SW_2 , SW_LAST }
- enum {
  KNOB_1 , KNOB_2 , KNOB_3 , KNOB_4 ,
  KNOB_5 , KNOB_6 , KNOB_7 , KNOB_8 ,
  KNOB_LAST }
- enum {
  CV_1 , CV_2 , CV_3 , CV_4 ,
  CV_LAST }
- enum {
  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_KEY_A8 , LED_KEY_A7 , LED_KEY_A6 , LED_KEY_A5 ,
  LED_KEY_A4 , LED_KEY_A3 , LED_KEY_A2 , LED_KEY_A1 ,
  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 }

Public Member Functions

- void Init (bool boost=false)
- void DelayMs (size_t del)
- void StartAudio (AudioHandle::InterleavingAudioCallback cb)
- void StartAudio (AudioHandle::AudioCallback cb)
- void StopAudio ()
- void ChangeAudioCallback (AudioHandle::InterleavingAudioCallback cb)
- void ChangeAudioCallback (AudioHandle::AudioCallback cb)
- void SetAudioSampleRate (SaiHandle::Config::SampleRate samplerate)
- float AudioSampleRate ()
- void SetAudioBlockSize (size_t blocksize)
- size_t AudioBlockSize ()
- float AudioCallbackRate ()
- void StartAdc ()
- void StopAdc ()
- void StartDac ()
- void ProcessAnalogControls ()
- void ProcessDigitalControls ()

- void ProcessAllControls ()
- void SetCvOut1 (uint16_t val)
- void SetCvOut2 (uint16_t val)
- bool KeyboardState (size_t idx) const
- bool KeyboardRisingEdge (size_t idx) const
- bool KeyboardFallingEdge (size_t idx) const
- float GetKnobValue (size_t idx) const
- float GetCvValue (size_t idx) const
- Switch * GetSwitch (size_t idx)
- AnalogControl * GetKnob (size_t idx)
- AnalogControl * GetCv (size_t idx)
- void VegasMode ()

Public Attributes

- DaisySeed **seed**
- OledDisplay **display**
- dsy_gpio **gate_out**
- GateIn **gate_in**
- LedDriverPca9685< 2, true > **led_driver**
- Switch **sw** [SW_LAST]
- AnalogControl **knob** [KNOB_LAST]
- AnalogControl **cv** [CV_LAST]

8.17.1 Member Enumeration Documentation

8.17.1.1 anonymous enum

```
anonymous enum
```

enums for controls, etc.

Enumerator

SW_1	tactile switch
SW_2	tactile switch
SW_LAST	&

8.17.1.2 anonymous enum

```
anonymous enum
```

All knobs connect to Daisy Seed's ADC1 pin via CD4051 mux Knobs are in order that they are laid out on hardware.

Enumerator

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

8.17.1.3 anonymous enum

`anonymous enum`

Enumerator

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

8.17.1.4 anonymous enum

`anonymous enum`

Enumerator

LED_KEY_B1	&
LED_KEY_B2	&
LED_KEY_B3	&
LED_KEY_B4	&
LED_KEY_B5	&
LED_KEY_B6	&
LED_KEY_B7	&
LED_KEY_B8	&
LED_KEY_A8	&
LED_KEY_A7	&
LED_KEY_A6	&
LED_KEY_A5	&
LED_KEY_A4	&
LED_KEY_A3	&
LED_KEY_A2	&
LED_KEY_A1	&

Enumerator

LED_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	&

8.17.2 Member Function Documentation

8.17.2.1 AudioBlockSize()

```
size_t daisy::DaisyField::AudioBlockSize ( )
```

Returns the number of samples per channel in a block of audio.

8.17.2.2 AudioCallbackRate()

```
float daisy::DaisyField::AudioCallbackRate ( )
```

Returns the rate in Hz that the Audio callback is called

8.17.2.3 AudioSampleRate()

```
float daisy::DaisyField::AudioSampleRate ( )
```

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

8.17.2.4 ChangeAudioCallback() [1/2]

```
void daisy::DaisyField::ChangeAudioCallback (
            AudioHandle::AudioCallback cb )
```

Switch callback functions

Parameters

cb	New multichannel callback function.

8.17.2.5 ChangeAudioCallback() [2/2]

```
void daisy::DaisyField::ChangeAudioCallback (
            AudioHandle::InterleavingAudioCallback cb )
```

Switch callback functions

Parameters

cb	New interleaved callback function.

8.17.2.6 DelayMs()

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

Wait some ms before going on.

Parameters

del	Delay time in ms.

8.17.2.7 GetCv()

```
AnalogControl* daisy::DaisyField::GetCv (
            size_t idx )
```

Getter for CV objects.

Parameters

idx	The CV input of interest.

8.17.2.8 GetCvValue()

```
float daisy::DaisyField::GetCvValue (
            size_t idx ) const
```

Returns the CV input's value

Parameters

idx	The CV input of interest.

8.17.2.9 GetKnob()

```
AnalogControl* daisy::DaisyField::GetKnob (
            size_t idx )
```

Getter for knob objects

Parameters

idx	The knob input of interest.

8.17.2.10 GetKnobValue()

```
float daisy::DaisyField::GetKnobValue (
            size_t idx ) const
```

Returns the knob's value

Parameters

idx	The knob of interest.

8.17.2.11 GetSwitch()

```
Switch* daisy::DaisyField::GetSwitch (
            size_t idx )
```

Getter for switch objects

Parameters

idx	The switch of interest.

8.17.2.12 Init()

```
void daisy::DaisyField::Init (
            bool boost = false )
```

Initializes the Daisy Field, and all of its hardware.

8.17.2.13 KeyboardFallingEdge()

```
bool daisy::DaisyField::KeyboardFallingEdge (
            size_t idx ) const
```

Returns true if the key has just been released

Parameters

idx	the key of interest

8.17.2.14 KeyboardRisingEdge()

```
bool daisy::DaisyField::KeyboardRisingEdge (
            size_t idx ) const
```

Returns true if the key has just been pressed

Parameters

idx	the key of interest

8.17.2.15 KeyboardState()

```
bool daisy::DaisyField::KeyboardState (
            size_t idx ) const
```

Returns true if the key has not been pressed recently

Parameters

idx	the key of interest

8.17.2.16 ProcessAllControls()

```
void daisy::DaisyField::ProcessAllControls ( )  [inline]
```

Process Analog and Digital Controls

8.17.2.17 ProcessAnalogControls()

```
void daisy::DaisyField::ProcessAnalogControls ( )
```

Processes the ADC inputs, updating their values

8.17.2.18 ProcessDigitalControls()

```
void daisy::DaisyField::ProcessDigitalControls ( )
```

Process tactile switches and keyboard states

8.17.2.19 SetAudioBlockSize()

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

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

8.17.2.20 SetAudioSampleRate()

```
void daisy::DaisyField::SetAudioSampleRate (
            SaiHandle::Config::SampleRate samplerate )
```

Updates the Audio Sample Rate, and reinitializes. Audio must be stopped for this to work.

8.17.2.21 SetCvOut1()

```
void daisy::DaisyField::SetCvOut1 (
            uint16_t val )
```

Sets the output of CV out 1 to a value between 0-4095 that corresponds to 0-5V

8.17.2.22 SetCvOut2()

```
void daisy::DaisyField::SetCvOut2 (
            uint16_t val )
```

Sets the output of CV out 2 to a value between 0-4095 that corresponds to 0-5V

8.17.2.23 StartAdc()

```
void daisy::DaisyField::StartAdc ( )
```

Starts Transfering data from the ADC

8.17.2.24 StartAudio() [1/2]

```
void daisy::DaisyField::StartAudio (
            AudioHandle::AudioCallback cb )
```

Starts the callback \cb multichannel callback function

8.17.2.25 StartAudio() [2/2]

```
void daisy::DaisyField::StartAudio (
            AudioHandle::InterleavingAudioCallback cb )
```

Starts the callback \cb Interleaved callback function

8.17.2.26 StartDac()

```
void daisy::DaisyField::StartDac ( )
```

Turns on the built-in 12-bit DAC on the Daisy Seed **This is now deprecated and does nothing.** The polling use of the DACs now handles starting the tranmission.

8.17.2.27 StopAdc()

```
void daisy::DaisyField::StopAdc ( )
```

Stops Transfering data from the ADC

8.17.2.28 StopAudio()

```
void daisy::DaisyField::StopAudio ( )
```

Stops the audio if it is running.

8.17.2.29 VegasMode()

```
void daisy::DaisyField::VegasMode ( )
```

Light show, cycling through all LEDs, and OLED

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

- src/daisy_field.h

8.18 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 (bool boost=false)
- void DelayMs (size_t del)
- void StartAudio (AudioHandle::AudioCallback cb)
- void ChangeAudioCallback (AudioHandle::AudioCallback cb)
- void StopAudio ()
- void SetAudioSampleRate (SaiHandle::Config::SampleRate samplerate)
- float AudioSampleRate ()
- void SetAudioBlockSize (size_t size)
- size_t AudioBlockSize ()
- float AudioCallbackRate ()
- void StartAdc ()
- void StopAdc ()
- void ProcessAnalogControls ()
- void ProcessAllControls ()
- float GetKnobValue (Ctrl k)
- void ProcessDigitalControls ()
- 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

8.18.1 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

8.18.2 Member Enumeration Documentation

8.18.2.1 Ctrl

`enum daisy::DaisyPatch::Ctrl`

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

8.18.2.2 GateInput

`enum daisy::DaisyPatch::GateInput`

Daisy patch gate inputs

Enumerator

GATE_IN_LAST	<

8.18.3 Constructor & Destructor Documentation

8.18.3.1 DaisyPatch()

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

Constructor

8.18.3.2 ~DaisyPatch()

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

Destructor

8.18.4 Member Function Documentation

8.18.4.1 AudioBlockSize()

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

Returns the number of samples per channel in a block of audio.

8.18.4.2 AudioCallbackRate()

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

Returns the rate in Hz that the Audio callback is called

8.18.4.3 AudioSampleRate()

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

Get sample rate

8.18.4.4 ChangeAudioCallback()

```
void daisy::DaisyPatch::ChangeAudioCallback (
            AudioHandle::AudioCallback cb )
```

Switch callback functions

Parameters

cb	New multichannel callback function.

8.18.4.5 DelayMs()

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

Wait some ms before going on.

Parameters

del	Delay time in ms.

8.18.4.6 DisplayControls()

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

Control the display

8.18.4.7 GetKnobValue()

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

Get value for a particular control

Parameters

k	Which control to get

8.18.4.8 Init()

```
void daisy::DaisyPatch::Init (
            bool boost = false )
```

Initializes the daisy seed, and patch hardware.

8.18.4.9 ProcessAllControls()

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

Process Analog and Digital Controls

8.18.4.10 ProcessAnalogControls()

```
void daisy::DaisyPatch::ProcessAnalogControls ( )
```

Call at same rate as reading controls for good reads.

8.18.4.11 ProcessDigitalControls()

```
void daisy::DaisyPatch::ProcessDigitalControls ( )
```

Process the digital controls

8.18.4.12 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.

8.18.4.13 SetAudioSampleRate()

```
void daisy::DaisyPatch::SetAudioSampleRate (
            SaiHandle::Config::SampleRate samplerate )
```

Set the sample rate for the audio

8.18.4.14 StartAdc()

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

Start analog to digital conversion.

8.18.4.15 StartAudio()

```
void daisy::DaisyPatch::StartAudio (
            AudioHandle::AudioCallback cb )
```

Starts the callback \cb multichannel callback function

8.18.4.16 StopAdc()

```
void daisy::DaisyPatch::StopAdc ( )
```

Stops Transfering data from the ADC

8.18.4.17 StopAudio()

```
void daisy::DaisyPatch::StopAudio ( )
```

Stops the audio

8.18.5 Member Data Documentation

8.18.5.1 controls

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

Array of controls

8.18.5.2 display

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

&

8.18.5.3 encoder

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

Encoder object

8.18.5.4 gate_input

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

Gate inputs

8.18.5.5 gate_output

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

&

8.18.5.6 midi

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

Handles midi

8.18.5.7 seed

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

Seed object

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

- src/daisy_patch.h

8.19 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 (bool boost=false)
- void DelayMs (size_t del)
- void StartAudio (AudioHandle::InterleavingAudioCallback cb)
- void StartAudio (AudioHandle::AudioCallback cb)
- void ChangeAudioCallback (AudioHandle::InterleavingAudioCallback cb)
- void ChangeAudioCallback (AudioHandle::AudioCallback cb)
- void StopAudio ()
- void SetAudioSampleRate (SaiHandle::Config::SampleRate samplerate)
- float AudioSampleRate ()
- void SetAudioBlockSize (size_t size)
- size_t AudioBlockSize ()
- float AudioCallbackRate ()
- void StartAdc ()
- void StopAdc ()
- void ProcessAnalogControls ()
- void ProcessAllControls ()
- float GetKnobValue (Knob k)
- float GetExpression ()
- void ProcessDigitalControls ()
- 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]

8.19.1 Detailed Description

Helpers and hardware definitions for daisy petal.

8.19.2 Member Enumeration Documentation

8.19.2.1 FootswitchLed

`enum daisy::DaisyPetal::FootswitchLed`

footswitch leds

Enumerator

FOOTSWITCH_LED_1	&
FOOTSWITCH_LED_2	&
FOOTSWITCH_LED_3	&
FOOTSWITCH_LED_4	&
FOOTSWITCH_LED_LAST	&

8.19.2.2 Knob

`enum daisy::DaisyPetal::Knob`

Knobs

Enumerator

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

8.19.2.3 RingLed

`enum daisy::DaisyPetal::RingLed`

Leds in ringled

Enumerator

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

8.19.2.4 Sw

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

Switches

Enumerator

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

8.19.3 Constructor & Destructor Documentation

8.19.3.1 DaisyPetal()

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

Constructor

8.19.3.2 ~DaisyPetal()

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

Destructor

8.19.4 Member Function Documentation

8.19.4.1 AudioBlockSize()

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

Returns the number of samples per channel in a block of audio.

8.19.4.2 AudioCallbackRate()

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

Returns the rate in Hz that the Audio callback is called

8.19.4.3 AudioSampleRate()

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

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

8.19.4.4 ChangeAudioCallback() [1/2]

```
void daisy::DaisyPetal::ChangeAudioCallback (
            AudioHandle::AudioCallback cb )
```

Switch callback functions

Parameters

cb	New multichannel callback function.

8.19.4.5 ChangeAudioCallback() [2/2]

```
void daisy::DaisyPetal::ChangeAudioCallback (
            AudioHandle::InterleavingAudioCallback cb )
```

Switch callback functions

Parameters

cb	New interleaved callback function.

8.19.4.6 ClearLeds()

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

Turn all leds off

8.19.4.7 DelayMs()

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

Wait before moving on.

Parameters

del	Delay time in ms.

8.19.4.8 GetExpression()

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

&

8.19.4.9 GetKnobValue()

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

Get value per knob.

Parameters

k	Which knob to get

Returns

Floating point knob position.

8.19.4.10 Init()

```
void daisy::DaisyPetal::Init (
            bool boost = false )
```

Initialize daisy petal

8.19.4.11 ProcessAllControls()

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

Process Analog and Digital Controls

8.19.4.12 ProcessAnalogControls()

```
void daisy::DaisyPetal::ProcessAnalogControls ( )
```

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

8.19.4.13 ProcessDigitalControls()

```
void daisy::DaisyPetal::ProcessDigitalControls ( )
```

Process digital controls

8.19.4.14 SetAudioBlockSize()

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

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

Parameters

size	Audio block size

8.19.4.15 SetAudioSampleRate()

```
void daisy::DaisyPetal::SetAudioSampleRate (
            SaiHandle::Config::SampleRate samplerate )
```

Updates the Audio Sample Rate, and reinitializes. Audio must be stopped for this to work.

8.19.4.16 SetFootswitchLed()

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

Set footswitch LED

Parameters

idx	Led Index
bright	Brightness

8.19.4.17 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

8.19.4.18 StartAdc()

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

Start analog to digital conversion.

8.19.4.19 StartAudio() [1/2]

```
void daisy::DaisyPetal::StartAudio (
            AudioHandle::AudioCallback cb )
```

Starts the callback \cb multichannel callback function

8.19.4.20 StartAudio() [2/2]

```
void daisy::DaisyPetal::StartAudio (
            AudioHandle::InterleavingAudioCallback cb )
```

Starts the callback \cb Interleaved callback function

8.19.4.21 StopAdc()

```
void daisy::DaisyPetal::StopAdc ( )
```

Stops Transfering data from the ADC

8.19.4.22 StopAudio()

```
void daisy::DaisyPetal::StopAudio ( )
```

Stops the audio if it is running.

8.19.4.23 UpdateLeds()

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

Update Leds to values you had set.

8.19.5 Member Data Documentation

8.19.5.1 encoder

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

&

8.19.5.2 expression

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

&

8.19.5.3 footswitch_led

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

&

8.19.5.4 knob

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

&

8.19.5.5 ring_led

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

&

8.19.5.6 seed

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

&

8.19.5.7 switches

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

< &

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

- src/daisy_petal.h

8.20 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 (bool boost=false)
- void DelayMs (size_t del)
- void StartAudio (AudioHandle::InterleavingAudioCallback cb)
- void StartAudio (AudioHandle::AudioCallback cb)
- void ChangeAudioCallback (AudioHandle::InterleavingAudioCallback cb)
- void ChangeAudioCallback (AudioHandle::AudioCallback cb)
- void StopAudio ()
- void SetAudioSampleRate (SaiHandle::Config::SampleRate samplerate)
- float AudioSampleRate ()
- void SetAudioBlockSize (size_t blocksize)
- size_t AudioBlockSize ()
- float AudioCallbackRate ()
- void StartAdc ()
- void StopAdc ()
- void ProcessAnalogControls ()
- void ProcessAllControls ()
- float GetKnobValue (Knob k)
- void ProcessDigitalControls ()
- 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

8.20.1 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

8.20.2 Member Enumeration Documentation

8.20.2.1 Knob

`enum daisy::DaisyPod::Knob`

Knobs

Enumerator

KNOB_2	&
KNOB_LAST	&

8.20.2.2 Sw

`enum daisy::DaisyPod::Sw`

Switches

Enumerator

BUTTON_2	&
BUTTON_LAST	&

8.20.3 Member Function Documentation

8.20.3.1 AudioBlockSize()

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

Returns the number of samples per channel in a block of audio.

8.20.3.2 AudioCallbackRate()

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

Returns the rate in Hz that the Audio callback is called

8.20.3.3 AudioSampleRate()

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

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

8.20.3.4 ChangeAudioCallback() [1/2]

```
void daisy::DaisyPod::ChangeAudioCallback (
            AudioHandle::AudioCallback cb )
```

Switch callback functions

Parameters

cb	New multichannel callback function.

8.20.3.5 ChangeAudioCallback() [2/2]

```
void daisy::DaisyPod::ChangeAudioCallback (
            AudioHandle::InterleavingAudioCallback cb )
```

Switch callback functions

Parameters

cb	New interleaved callback function.

8.20.3.6 ClearLeds()

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

Reset Leds

8.20.3.7 DelayMs()

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

Wait for a bit

Parameters

del	Time to wait in ms.

8.20.3.8 GetKnobValue()

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

&

8.20.3.9 Init()

```
void daisy::DaisyPod::Init (
            bool boost = false )
```

Init related stuff.

8.20.3.10 ProcessAllControls()

```
void daisy::DaisyPod::ProcessAllControls ( )  [inline]
```

Process Analog and Digital Controls

8.20.3.11 ProcessAnalogControls()

```
void daisy::DaisyPod::ProcessAnalogControls ( )
```

Call at same rate as analog reads for smooth reading.

8.20.3.12 ProcessDigitalControls()

```
void daisy::DaisyPod::ProcessDigitalControls ( )
```

Process digital controls

8.20.3.13 SetAudioBlockSize()

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

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

8.20.3.14 SetAudioSampleRate()

```
void daisy::DaisyPod::SetAudioSampleRate (
            SaiHandle::Config::SampleRate samplerate )
```

Updates the Audio Sample Rate, and reinitializes. Audio must be stopped for this to work.

8.20.3.15 StartAdc()

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

Start analog to digital conversion.

8.20.3.16 StartAudio() [1/2]

```
void daisy::DaisyPod::StartAudio (
            AudioHandle::AudioCallback cb )
```

Starts the callback \cb multichannel callback function

8.20.3.17 StartAudio() [2/2]

```
void daisy::DaisyPod::StartAudio (
            AudioHandle::InterleavingAudioCallback cb )
```

Starts the callback \cb Interleaved callback function

8.20.3.18 StopAdc()

```
void daisy::DaisyPod::StopAdc ( )
```

Stops Transfering data from the ADC

8.20.3.19 StopAudio()

```
void daisy::DaisyPod::StopAudio ( )
```

Stops the audio if it is running.

8.20.3.20 UpdateLeds()

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

Update Leds to set colors

8.20.4 Member Data Documentation

8.20.4.1 button1

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

&

8.20.4.2 button2

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

&

8.20.4.3 buttons

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

&

8.20.4.4 encoder

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

&

8.20.4.5 knob1

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

&

8.20.4.6 knob2

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

&

8.20.4.7 knobs

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

&

8.20.4.8 led1

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

&

8.20.4.9 led2

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

&

8.20.4.10 seed

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

Public Members

8.20.5 autotoc_md10

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

- src/daisy_pod.h

8.21 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 (bool boost=false)
- void DelayMs (size_t del)
- void StartAudio (AudioHandle::InterleavingAudioCallback cb)
- void StartAudio (AudioHandle::AudioCallback cb)
- void ChangeAudioCallback (AudioHandle::InterleavingAudioCallback cb)
- void ChangeAudioCallback (AudioHandle::AudioCallback cb)
- void StopAudio ()
- void SetAudioSampleRate (SaiHandle::Config::SampleRate samplerate)
- float AudioSampleRate ()
- void SetAudioBlockSize (size_t blocksize)
- size_t AudioBlockSize ()
- float AudioCallbackRate () const
- void SetLed (bool state)
- void SetTestPoint (bool state)

Static Public Member Functions

- static dsy_gpio_pin GetPin (uint8_t pin_idx)
- template<typename... VA>
  static void Print (const char *format, VA... va)
- template<typename... VA>
  static void PrintLine (const char *format, VA... va)
- static void StartLog (bool wait_for_pc=false)

Public Attributes

- dsy_sdram_handle sdram_handle
- dsy_qspi_handle qspi_handle
- AudioHandle audio_handle
- AdcHandle adc
- DacHandle **dac**
- UsbHandle usb_handle
- dsy_gpio **led**
- dsy_gpio **testpoint**
- System **system**

8.21.1 Detailed Description

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

8.21.2 Member Function Documentation

8.21.2.1 AudioBlockSize()

```
size_t daisy::DaisySeed::AudioBlockSize ( )
```

Returns the number of samples per channel in a block of audio.

8.21.2.2 AudioCallbackRate()

```
float daisy::DaisySeed::AudioCallbackRate ( ) const
```

Returns the rate in Hz that the Audio callback is called

8.21.2.3 AudioSampleRate()

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

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

8.21.2.4 ChangeAudioCallback() [1/2]

```
void daisy::DaisySeed::ChangeAudioCallback (
            AudioHandle::AudioCallback cb )
```

Changes to a new multichannel callback

8.21.2.5 ChangeAudioCallback() [2/2]

```
void daisy::DaisySeed::ChangeAudioCallback (
            AudioHandle::InterleavingAudioCallback cb )
```

Changes to a new interleaved callback

8.21.2.6 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. &

8.21.2.7 DelayMs()

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

Wait some ms before going on.

Parameters

del	Delay time in ms.

8.21.2.8 GetPin()

```
static dsy_gpio_pin daisy::DaisySeed::GetPin (
            uint8_t pin_idx )  [static]
```

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

8.21.2.9 Init()

```
void daisy::DaisySeed::Init (
            bool boost = false )
```

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.

8.21.2.10 Print()

```
template<typename...  VA>
static void daisy::DaisySeed::Print (
            const char * format,
            VA...  va )  [inline], [static]
```

Print formatted debug log message

8.21.2.11 PrintLine()

```
template<typename...  VA>
static void daisy::DaisySeed::PrintLine (
            const char * format,
            VA...  va )  [inline], [static]
```

Print formatted debug log message with automatic line termination

8.21.2.12 SetAudioBlockSize()

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

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

8.21.2.13 SetAudioSampleRate()

```
void daisy::DaisySeed::SetAudioSampleRate (
            SaiHandle::Config::SampleRate samplerate )
```

Updates the Audio Sample Rate, and reinitializes. Audio must be stopped for this to work.

8.21.2.14 SetLed()

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

Sets the state of the built in LED

8.21.2.15 SetTestPoint()

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

Sets the state of the test point near pin 10

8.21.2.16 StartAudio() [1/2]

```
void daisy::DaisySeed::StartAudio (
            AudioHandle::AudioCallback cb )
```

Begins the audio for the seeds builtin audio. the specified callback will get called whenever new data is ready to be prepared. This will use the newer non-interleaved callback.

8.21.2.17 StartAudio() [2/2]

```
void daisy::DaisySeed::StartAudio (
            AudioHandle::InterleavingAudioCallback cb )
```

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

8.21.2.18 StartLog()

```
static void daisy::DaisySeed::StartLog (
            bool wait_for_pc = false )  [inline], [static]
```

Start the logging session. Optionally wait for terminal connection before proceeding.

8.21.2.19 StopAudio()

```
void daisy::DaisySeed::StopAudio ( )
```

Stops the audio if it is running.

8.21.3 Member Data Documentation

8.21.3.1 adc

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

&

8.21.3.2 audio_handle

```
AudioHandle daisy::DaisySeed::audio_handle
```

&

8.21.3.3 qspi_handle

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

&

8.21.3.4 sdram_handle

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

&

8.21.3.5 usb_handle

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

&

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

- src/daisy_seed.h

8.22 daisy::DaisyVersio Class Reference

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

```
#include <daisy_versio.h>
```

Public Types

- enum **AV_LEDS** {
  LED_0 , **LED_1** , **LED_2** , **LED_3** ,
  LED_LAST }
- enum **AV_KNOBS** {
  KNOB_0 , **KNOB_1** , **KNOB_2** , **KNOB_3** ,
  KNOB_4 , **KNOB_5** , **KNOB_6** , **KNOB_LAST** }
- enum **AV_TOGGLE3** { **SW_0** , **SW_1** , **SW_LAST** }

Public Member Functions

- void Init (bool boost=false)
- void DelayMs (size_t del)
- void StartAudio (AudioHandle::InterleavingAudioCallback cb)
- void StartAudio (AudioHandle::AudioCallback cb)
- void ChangeAudioCallback (AudioHandle::InterleavingAudioCallback cb)
- void ChangeAudioCallback (AudioHandle::AudioCallback cb)
- void StopAudio ()
- void SetAudioBlockSize (size_t size)
- size_t AudioBlockSize ()
- void SetAudioSampleRate (SaiHandle::Config::SampleRate samplerate)
- float AudioSampleRate ()
- float AudioCallbackRate ()
- void StartAdc ()
- void StopAdc ()
- void ProcessAnalogControls ()
- void ProcessAllControls ()
- bool SwitchPressed ()
- bool Gate ()
- void SetLed (size_t idx, float red, float green, float blue)
- float GetKnobValue (int idx)
- void UpdateLeds ()
- void **UpdateExample** ()

Public Attributes

- DaisySeed **seed**
- RgbLed **leds** [LED_LAST]
- AnalogControl **knobs** [KNOB_LAST]
- Switch **tap**
- GateIn **gate**
- Switch3 **sw** [SW_LAST]

8.22.1 Detailed Description

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

Author

Ankoor Apte, Noise Engineering

Date

October 2020

8.22.2 Member Function Documentation

8.22.2.1 AudioBlockSize()

```
size_t daisy::DaisyVersio::AudioBlockSize ( )
```

Returns the number of samples per channel in a block of audio.

8.22.2.2 AudioCallbackRate()

```
float daisy::DaisyVersio::AudioCallbackRate ( )
```

Returns the rate in Hz that the Audio callback is called

8.22.2.3 AudioSampleRate()

```
float daisy::DaisyVersio::AudioSampleRate ( )
```

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

8.22.2.4 ChangeAudioCallback() [1/2]

```
void daisy::DaisyVersio::ChangeAudioCallback (
            AudioHandle::AudioCallback cb )
```

Switch callback functions

Parameters

cb	New non-interleaved callback function.

8.22.2.5 ChangeAudioCallback() [2/2]

```
void daisy::DaisyVersio::ChangeAudioCallback (
            AudioHandle::InterleavingAudioCallback cb )
```

Switch callback functions

Parameters

cb	New interleaved callback function.

8.22.2.6 DelayMs()

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

Wait some ms before going on.

Parameters

del	Delay time in ms.

8.22.2.7 Gate()

```
bool daisy::DaisyVersio::Gate ( )
```

Returns true if gate in is HIGH

8.22.2.8 GetKnobValue()

```
float daisy::DaisyVersio::GetKnobValue (
            int idx )
```

Get Knob Value, float from 0.0f to 1.0f

8.22.2.9 Init()

```
void daisy::DaisyVersio::Init (
            bool boost = false )
```

Initializes the Versio, and all of its hardware.

8.22.2.10 ProcessAllControls()

```
void daisy::DaisyVersio::ProcessAllControls ( )  [inline]
```

Does what it says

8.22.2.11 ProcessAnalogControls()

```
void daisy::DaisyVersio::ProcessAnalogControls ( )
```

Normalize ADC CV input. Call this once per main loop update to normalize CV input to range (0.0f, 1.0f)

8.22.2.12 SetAudioBlockSize()

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

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

8.22.2.13 SetAudioSampleRate()

```
void daisy::DaisyVersio::SetAudioSampleRate (
            SaiHandle::Config::SampleRate samplerate )
```

Updates the Audio Sample Rate, and reinitializes. Audio must be stopped for this to work.

8.22.2.14 SetLed()

```
void daisy::DaisyVersio::SetLed (
            size_t idx,
            float red,
            float green,
            float blue )
```

Set an LED (idx $<$ 4) to a color

8.22.2.15 StartAdc()

```
void daisy::DaisyVersio::StartAdc ( )
```

Start analog to digital conversion.

8.22.2.16 StartAudio() [1/2]

```
void daisy::DaisyVersio::StartAudio (
            AudioHandle::AudioCallback cb )
```

Starts the callback \cb Non-interleaved callback function

8.22.2.17 StartAudio() [2/2]

```
void daisy::DaisyVersio::StartAudio (
            AudioHandle::InterleavingAudioCallback cb )
```

Starts the callback \cb Interleaved callback function

8.22.2.18 StopAdc()

```
void daisy::DaisyVersio::StopAdc ( )
```

Stop converting ADCs

8.22.2.19 StopAudio()

```
void daisy::DaisyVersio::StopAudio ( )
```

Stops the audio if it is running.

8.22.2.20 SwitchPressed()

```
bool daisy::DaisyVersio::SwitchPressed ( )
```

Returns true if momentary switch is pressed

8.22.2.21 UpdateLeds()

```
void daisy::DaisyVersio::UpdateLeds ( )
```

Update LED PWM state. Call this once per main loop update to correctly display led colors

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

- src/daisy_versio.h

8.23 dsy_gpio Struct Reference

```
#include <gpio.h>
```

Public Attributes

- dsy_gpio_pin pin
- dsy_gpio_mode mode
- dsy_gpio_pull pull

8.23.1 Detailed Description

Struct for holding the pin, and configuration

8.23.2 Member Data Documentation

8.23.2.1 mode

```
dsy_gpio_mode dsy_gpio::mode
```

&

8.23.2.2 pin

```
dsy_gpio_pin dsy_gpio::pin
```

&

8.23.2.3 pull

```
dsy_gpio_pull dsy_gpio::pull
```

&

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

- src/per/gpio.h

8.24 dsy_gpio_pin Struct Reference

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

Public Attributes

- dsy_gpio_port port
- uint8_t pin

8.24.1 Detailed Description

Hardware define pins

8.24.2 Member Data Documentation

8.24.2.1 pin

```
uint8_t dsy_gpio_pin::pin
```

number 0-15

8.24.2.2 port

```
dsy_gpio_port dsy_gpio_pin::port
```

&

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

- src/daisy_core.h

8.25 dsy_qspi_handle Struct Reference

```
#include <qspi.h>
```

Public Attributes

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

8.25.1 Detailed Description

Configuration structure for interfacing with QSPI Driver

8.25.2 Member Data Documentation

8.25.2.1 device

```
dsy_qspi_device dsy_qspi_handle::device
```

&

8.25.2.2 mode

```
dsy_qspi_mode dsy_qspi_handle::mode
```

&

8.25.2.3 pin_config

```
dsy_gpio_pin dsy_qspi_handle::pin_config[DSY_QSPI_PIN_LAST]
```

&

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

- src/per/qspi.h

8.26 DSY_SD_CardInfoTypeDef Struct Reference

```
#include <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

8.26.1 Detailed Description

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

8.26.2 Member Data Documentation

8.26.2.1 BlockNbr

```
uint32_t DSY_SD_CardInfoTypeDef::BlockNbr
```

Specifies the Card Capacity in blocks

8.26.2.2 BlockSize

```
uint32_t DSY_SD_CardInfoTypeDef::BlockSize
```

Specifies one block size in bytes

8.26.2.3 CardSpeed

`uint32_t DSY_SD_CardInfoTypeDef::CardSpeed`

Specifies the card Speed

8.26.2.4 CardType

`uint32_t DSY_SD_CardInfoTypeDef::CardType`

Specifies the card Type

8.26.2.5 CardVersion

`uint32_t DSY_SD_CardInfoTypeDef::CardVersion`

Specifies the card version

8.26.2.6 Class

`uint32_t DSY_SD_CardInfoTypeDef::Class`

Specifies the class of the card class

8.26.2.7 LogBlockNbr

`uint32_t DSY_SD_CardInfoTypeDef::LogBlockNbr`

Specifies the Card logical Capacity in blocks

8.26.2.8 LogBlockSize

`uint32_t DSY_SD_CardInfoTypeDef::LogBlockSize`

Specifies logical block size in bytes

8.26.2.9 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

8.27 dsy_sdram_handle Struct Reference

`#include <sdram.h>`

Public Attributes

- dsy_sdram_state state
- dsy_gpio_pin pin_config [DSY_SDRAM_PIN_LAST]

8.27.1 Detailed Description

Configuration struct for passing to initialization

8.27.2 Member Data Documentation

8.27.2.1 pin_config

`dsy_gpio_pin dsy_sdram_handle::pin_config[DSY_SDRAM_PIN_LAST]`

&

8.27.2.2 state

`dsy_sdram_state dsy_sdram_handle::state`

&

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

- src/dev/sdram.h

8.28 daisy::Encoder Class Reference

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

`#include <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

8.28.1 Detailed Description

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

Author

Stephen Hensley

Date

December 2019

8.28.2 Member Function Documentation

8.28.2.1 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.

8.28.2.2 FallingEdge()

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

Returns true if the encoder was just released.

8.28.2.3 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.

8.28.2.4 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.

8.28.2.5 Pressed()

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

Returns true while the encoder is held down.

8.28.2.6 RisingEdge()

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

Returns true if the encoder was just pressed.

8.28.2.7 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

8.29 FontDef Struct Reference

```
#include <oled_fonts.h>
```

Public Attributes

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

8.29.1 Detailed Description

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

Author

afiskon on github. Font struct

8.29.2 Member Data Documentation

8.29.2.1 data

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

Pointer to data font data array

8.29.2.2 FontHeight

```
uint8_t FontDef::FontHeight
```

Font height in pixels

8.29.2.3 FontWidth

```
const uint8_t FontDef::FontWidth
```

Font width in pixels

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

- src/util/oled_fonts.h

8.30 daisy::GateIn Class Reference

Generic Class for handling gate inputs through GPIO.

```
#include <gatein.h>
```

Public Member Functions

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

8.30.1 Detailed Description

Generic Class for handling gate inputs through GPIO.

Author

Stephen Hensley

Date

March 2020

8.30.2 Constructor & Destructor Documentation

8.30.2.1 GateIn()

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

GateIn Constructor

8.30.2.2 ~GateIn()

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

GateIn~ Destructor

8.30.3 Member Function Documentation

8.30.3.1 Init()

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

Init Initializes the gate input with specified hardware pin

8.30.3.2 State()

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

State Checks current state of gate input (no state required)

read function is inverted because of suggested BJT input circuit

8.30.3.3 Trig()

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

Trig Checks current state of gate input.

Returns

True if the GPIO just transitioned.

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

- src/hid/gatein.h

8.31 daisy::I2CHandle Class Reference

```
#include <i2c.h>
```

Classes

- struct Config

Public Types

- enum class Result { OK , ERR }
- typedef void(* CallbackFunctionPtr) (void *context, Result result)

Public Member Functions

- **I2CHandle** (const I2CHandle &other)=default
- I2CHandle & **operator=** (const I2CHandle &other)=default
- Result Init (const Config &config)
- const Config & GetConfig () const
- Result TransmitBlocking (uint16_t address, uint8_t *data, uint16_t size, uint32_t timeout)
- Result ReceiveBlocking (uint16_t address, uint8_t *data, uint16_t size, uint32_t timeout)
- Result TransmitDma (uint16_t address, uint8_t *data, uint16_t size, CallbackFunctionPtr callback, void *callback_context)
- Result ReadDataAtAddress (uint16_t address, uint16_t mem_address, uint16_t mem_address_size, uint8_t *data, uint16_t data_size, uint32_t timeout)
- Result WriteDataAtAddress (uint16_t address, uint16_t mem_address, uint16_t mem_address_size, uint8_t *data, uint16_t data_size, uint32_t timeout)

8.31.1 Detailed Description

A handle for interacting with an I2C peripheral. This is a dumb gateway that internally points to one of the four I2C peripherals after it was initialised. It can then be copied and passed around. Use an I2CHandle like this:

```
// setup the configuration
I2CHandle::Config i2c_conf;
i2c_conf.periph = I2CHandle::Config::Peripheral::I2C1;
i2c_conf.speed  = I2CHandle::Config::Speed::I2C_400KHZ;
i2c_conf.pin_config.scl  = {DSY_GPIOB, 8};
i2c_conf.pin_config.sda  = {DSY_GPIOB, 9};
// initialise the peripheral
I2CHandle i2c;
i2c.Init(i2c_conf);
// now i2c points to the corresponding peripheral and can be used.
i2c.TransmitBlocking( ... );
```

8.31.2 Member Typedef Documentation

8.31.2.1 CallbackFunctionPtr

```
typedef void(* daisy::I2CHandle::CallbackFunctionPtr) (void *context, Result result)
```

A callback to be executed when a dma transfer is complete.

8.31.3 Member Enumeration Documentation

8.31.3.1 Result

```
enum daisy::I2CHandle::Result  [strong]
```

Return values for I2C functions.

Enumerator

OK	&
ERR	&

8.31.4 Member Function Documentation

8.31.4.1 GetConfig()

```
const Config& daisy::I2CHandle::GetConfig ( ) const
```

Returns the current config.

8.31.4.2 Init()

```
Result daisy::I2CHandle::Init (
            const Config & config )
```

Initializes an I2C peripheral.

8.31.4.3 ReadDataAtAddress()

```
Result daisy::I2CHandle::ReadDataAtAddress (
            uint16_t address,
            uint16_t mem_address,
            uint16_t mem_address_size,
            uint8_t * data,
            uint16_t data_size,
            uint32_t timeout )
```

Reads an amount of data from a specific memory address.

Parameters

address	The slave device address.
mem_address	Pointer to data containing the address to read from device.
mem_address_size	Size of the memory address in bytes.
data	Pointer to buffer that will be filled with contents at mem_address
data_size	Size of the data to be read in bytes.

8.31.4.4 ReceiveBlocking()

```
Result daisy::I2CHandle::ReceiveBlocking (
            uint16_t address,
            uint8_t * data,
            uint16_t size,
            uint32_t timeout )
```

Receives data and blocks until the reception is complete. Use this for smaller transmissions of a few bytes.

Parameters

address	The slave device address.
data	A pointer to the data to be received.
size	The size of the data to be received, in bytes.
timeout	A timeout.

8.31.4.5 TransmitBlocking()

```
Result daisy::I2CHandle::TransmitBlocking (
            uint16_t address,
            uint8_t * data,
            uint16_t size,
            uint32_t timeout )
```

Transmits data and blocks until the transmission is complete. Use this for smaller transmissions of a few bytes.

Parameters

address	The slave device address.
data	A pointer to the data to be sent.
size	The size of the data to be sent, in bytes.
timeout	A timeout.

8.31.4.6 TransmitDma()

```
Result daisy::I2CHandle::TransmitDma (
            uint16_t address,
            uint8_t * data,
            uint16_t size,
            CallbackFunctionPtr callback,
            void * callback_context )
```

Transmits data with a DMA and returns immediately. Use this for larger transmissions. The pointer to data must be located in the D2 memory domain by adding the `DMA_BUFFER_MEM_SECTION` attribute like this: uint8_t DMA↩ _BUFFER_MEM_SECTION my_buffer[100]; If that is not possible for some reason, you MUST clear the cachelines spanning the size of the buffer, before initiating the dma transfer by calling `dsy_dma_clear_cache_for_↩ buffer(buffer, size);`

A single DMA is shared across I2C, I2C2 and I2C3. I2C4 has no DMA support (yet). If the DMA is busy with another transfer, the job will be queued and executed later. If there is a job waiting to be executed for this I2C peripheral, this function will block until the queue is free and the job can be queued.

Parameters

address	The slave device address.
data	A pointer to the data to be sent.
size	The size of the data to be sent, in bytes.
callback	A callback to execute when the transfer finishes, or NULL.
callback_context	A pointer that will be passed back to you in the callback.

8.31.4.7 WriteDataAtAddress()

```
Result daisy::I2CHandle::WriteDataAtAddress (
            uint16_t address,
            uint16_t mem_address,
            uint16_t mem_address_size,
            uint8_t * data,
            uint16_t data_size,
            uint32_t timeout )
```

Writes an amount of data from a specific memory address.

Parameters

address	The slave device address.
mem_address	Pointer to data containing the address to write to device.
mem_address_size	Size of the memory address in bytes.
data	Pointer to buffer that will be written to the mem_address
data_size	Size of the data to be written in bytes.

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

- src/per/i2c.h

8.32 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 <led.h>`

Public Member Functions

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

8.32.1 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

8.32.2 Member Function Documentation

8.32.2.1 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)

8.32.2.2 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.

8.32.2.3 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

8.33 daisy::LedDriverPca9685< numDrivers, persistentBufferContents > Class Template Reference

`#include <leddriver.h>`

Public Types

- using DmaBuffer = PCA9685TransmitBuffer[numDrivers]

Public Member Functions

- struct __attribute__ ((packed)) PCA9685TransmitBuffer
- void Init (I2CHandle i2c, const uint8_t(&addresses)[numDrivers], DmaBuffer dma_buffer_a, DmaBuffer dma_buffer_b, dsy_gpio_pin oe_pin={DSY_GPIOX, 0})
- constexpr int GetNumLeds () const
- void SetAllTo (float brightness)
- void SetAllTo (uint8_t brightness)
- void SetAllToRaw (uint16_t rawBrightness)
- void SetLed (int ledIndex, float brightness)
- void SetLed (int ledIndex, uint8_t brightness)
- void SetLedRaw (int ledIndex, uint16_t rawBrightness)
- void SwapBuffersAndTransmit ()

8.33.1 Detailed Description

template<int numDrivers, bool persistentBufferContents = true>
class daisy::LedDriverPca9685< numDrivers, persistentBufferContents >

LED driver for one or multiple PCA9685 12bit PWM chips connected to a single I2C peripheral. It includes gamma correction from 8bit brightness values but it can also be supplied with raw 12bit values. This driver uses two buffers - one for drawing, one for transmitting. Multiple LedDriverPca9685 instances can be used at the same time.

Parameters

numDrivers	The number of PCA9685 driver attached to the I2C peripheral.
persistentBufferContents	If set to true, the current draw buffer contents will be copied to the next draw buffer during SwapBuffersAndTransmit(). Use this, if you plan to write single leds at a time. If you will alway update all leds before calling SwapBuffersAndTransmit(), you can set this to false and safe some cycles.

8.33.2 Member Typedef Documentation

8.33.2.1 DmaBuffer

```
template<int numDrivers, bool persistentBufferContents = true>
using daisy::LedDriverPca9685< numDrivers, persistentBufferContents >::DmaBuffer = PCA9685↩
TransmitBuffer[numDrivers]
```

Buffer type for the entire DMA buffer.

8.33.3 Member Function Documentation

8.33.3.1 __attribute__()

```
template<int numDrivers, bool persistentBufferContents = true>
struct daisy::LedDriverPca9685< numDrivers, persistentBufferContents >::__attribute__ (
            (packed)  )  [inline]
```

Buffer Type for a single PCA9685 driver chip. register address

cycle at which to switch on the led

cycle at which to switch off the led

full size in bytes

8.33.3.2 GetNumLeds()

```
template<int numDrivers, bool persistentBufferContents = true>
constexpr int daisy::LedDriverPca9685< numDrivers, persistentBufferContents >::GetNumLeds ( )
const  [inline], [constexpr]
```

Returns the number of leds available from this driver.

8.33.3.3 Init()

```
template<int numDrivers, bool persistentBufferContents = true>
void daisy::LedDriverPca9685< numDrivers, persistentBufferContents >::Init (
            I2CHandle i2c,
            const uint8_t(&) addresses[numDrivers],
            DmaBuffer dma_buffer_a,
            DmaBuffer dma_buffer_b,
            dsy_gpio_pin oe_pin = {DSY_GPIOX, 0} )  [inline]
```

Initialises the driver.

Parameters

i2c	The I2C peripheral to use.
addresses	An array of addresses for each of the driver chips.
dma_buffer_a	The first buffer for the DMA. This must be placed in D2 memory by adding the DMA_BUFFER_MEM_SECTION attribute like this: `LedDriverPca9685<2>::DmaBuffer DMA_BUFFER_MEM_SECTION bufferA;`
dma_buffer	The second buffer for the DMA. This must be placed in D2 memory by adding the

8.33.3.4 SetAllTo() [1/2]

```
template<int numDrivers, bool persistentBufferContents = true>
void daisy::LedDriverPca9685< numDrivers, persistentBufferContents >::SetAllTo (
            float brightness )  [inline]
```

Sets all leds to a gamma corrected brightness between 0.0f and 1.0f.

8.33.3.5 SetAllTo() [2/2]

```
template<int numDrivers, bool persistentBufferContents = true>
void daisy::LedDriverPca9685< numDrivers, persistentBufferContents >::SetAllTo (
            uint8_t brightness )  [inline]
```

Sets all leds to a gamma corrected brightness between 0 and 255.

8.33.3.6 SetAllToRaw()

```
template<int numDrivers, bool persistentBufferContents = true>
void daisy::LedDriverPca9685< numDrivers, persistentBufferContents >::SetAllToRaw (
            uint16_t rawBrightness )  [inline]
```

Sets all leds to a raw 12bit brightness between 0 and 4095.

8.33.3.7 SetLed() [1/2]

```
template<int numDrivers, bool persistentBufferContents = true>
void daisy::LedDriverPca9685< numDrivers, persistentBufferContents >::SetLed (
            int ledIndex,
            float brightness )  [inline]
```

Sets a single led to a gamma corrected brightness between 0.0f and 1.0f.

8.33.3.8 SetLed() [2/2]

```
template<int numDrivers, bool persistentBufferContents = true>
void daisy::LedDriverPca9685< numDrivers, persistentBufferContents >::SetLed (
            int ledIndex,
            uint8_t brightness )  [inline]
```

Sets a single led to a gamma corrected brightness between 0 and 255.

8.33.3.9 SetLedRaw()

```
template<int numDrivers, bool persistentBufferContents = true>
void daisy::LedDriverPca9685< numDrivers, persistentBufferContents >::SetLedRaw (
            int ledIndex,
            uint16_t rawBrightness )  [inline]
```

Sets a single led to a raw 12bit brightness between 0 and 4095.

8.33.3.10 SwapBuffersAndTransmit()

```
template<int numDrivers, bool persistentBufferContents = true>
void daisy::LedDriverPca9685< numDrivers, persistentBufferContents >::SwapBuffersAndTransmit (
)  [inline]
```

Swaps the current draw buffer and the current transmit buffer and starts transmitting the values to all chips.

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

- src/dev/leddriver.h

8.34 daisy::Logger< dest > Class Template Reference

Interface for simple USB logging.

```
#include <logger.h>
```

Public Member Functions

- Logger ()

Static Public Member Functions

- static void Print (const char *format,...)
- static void PrintLine (const char *format,...)
- static void StartLog (bool wait_for_pc=false)
- static void PrintV (const char *format, va_list va)
- static void PrintLineV (const char *format, va_list va)

Protected Types

- enum LoggerConsts { **LOGGER_SYNC_OUT** = 0 , LOGGER_SYNC_IN = 2 }

Static Protected Member Functions

- static void TransmitSync (const void *buffer, size_t bytes)
- static void TransmitBuf ()
- static void AppendNewLine ()
- static constexpr size_t NewLineSeqLength ()

Static Protected Attributes

- static char tx_buff_ [128]
- static size_t tx_ptr_ = 0
- static size_t pc_sync_ = LOGGER_SYNC_OUT
- static LoggerImpl< dest > impl_

8.34.1 Detailed Description

template<LoggerDestination dest = LOGGER_INTERNAL>
class daisy::Logger< dest >

Interface for simple USB logging.

Author

Alexander Petrov-Savchenko (axp@soft-amp.com)

Date

November 2020

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

- src/hid/logger.h

8.35 daisy::Logger< LOGGER_NONE > Class Reference

`#include <logger.h>`

Static Public Member Functions

- static void **Print** (const char *format,...)
- static void **PrintLine** (const char *format,...)
- static void **StartLog** (bool wait_for_pc=false)
- static void **PrintV** (const char *format, va_list va)
- static void **PrintLineV** (const char *format, va_list va)

8.35.1 Detailed Description

Specialization for a muted log

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

- src/hid/logger.h

8.36 daisy::LoggerImpl< dest > Class Template Reference

Logging I/O underlying implementation.

`#include <logger_impl.h>`

Static Public Member Functions

- static void Init ()
- static bool Transmit (const void *buffer, size_t bytes)

8.36.1 Detailed Description

template<LoggerDestination dest>
class daisy::LoggerImpl< dest >

Logging I/O underlying implementation.

Author

Alexander Petrov-Savchenko (axp@soft-amp.com)

Date

November 2020

8.36.2 Member Function Documentation

8.36.2.1 Init()

```
template<LoggerDestination dest>
static void daisy::LoggerImpl< dest >::Init ( )  [inline], [static]
```

Initialize logging destination

8.36.2.2 Transmit()

```
template<LoggerDestination dest>
static bool daisy::LoggerImpl< dest >::Transmit (
            const void * buffer,
            size_t bytes )  [inline], [static]
```

Transmit a block of data

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

- src/hid/logger_impl.h

8.37 daisy::LoggerImpl< LOGGER_EXTERNAL > Class Reference

Specialization for external USB port.

`#include <logger_impl.h>`

Static Public Member Functions

- static void Init ()
- static bool Transmit (const void *buffer, size_t bytes)

Static Protected Attributes

- static UsbHandle usb_handle_

8.37.1 Detailed Description

Specialization for external USB port.

8.37.2 Member Function Documentation

8.37.2.1 Init()

```
static void daisy::LoggerImpl< LOGGER_EXTERNAL >::Init ( )  [inline], [static]
```

Initialize logging destination this implementation relies on the fact that UsbHandle class has no member variables and can be shared. assert this statement:

8.37.2.2 Transmit()

```
static bool daisy::LoggerImpl< LOGGER_EXTERNAL >::Transmit (
            const void * buffer,
            size_t bytes )  [inline], [static]
```

Transmit a block of data

8.37.3 Member Data Documentation

8.37.3.1 usb_handle_

```
UsbHandle daisy::LoggerImpl< LOGGER_EXTERNAL >::usb_handle_  [static], [protected]
```

USB Handle for CDC transfers

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

- src/hid/logger_impl.h

8.38 daisy::LoggerImpl< LOGGER_INTERNAL > Class Reference

Specialization for internal USB port.

```
#include <logger_impl.h>
```

Static Public Member Functions

- static void Init ()
- static bool Transmit (const void *buffer, size_t bytes)

Static Protected Attributes

- static UsbHandle usb_handle_

8.38.1 Detailed Description

Specialization for internal USB port.

8.38.2 Member Function Documentation

8.38.2.1 Init()

```
static void daisy::LoggerImpl< LOGGER_INTERNAL >::Init ( )  [inline], [static]
```

Initialize logging destination this implementation relies on the fact that UsbHandle class has no member variables and can be shared assert this statement:

8.38.2.2 Transmit()

```
static bool daisy::LoggerImpl< LOGGER_INTERNAL >::Transmit (
            const void * buffer,
            size_t bytes )  [inline], [static]
```

Transmit a block of data

8.38.3 Member Data Documentation

8.38.3.1 usb_handle_

`UsbHandle daisy::LoggerImpl< LOGGER_INTERNAL >::usb_handle_ [static], [protected]`

USB Handle for CDC transfers

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

- src/hid/logger_impl.h

8.39 daisy::LoggerImpl< LOGGER_SEMIHOST > Class Reference

Specialization for semihosting (stdout)

`#include <logger_impl.h>`

Static Public Member Functions

- static void Init ()
- static bool Transmit (const void *buffer, size_t bytes)

8.39.1 Detailed Description

Specialization for semihosting (stdout)

8.39.2 Member Function Documentation

8.39.2.1 Init()

`static void daisy::LoggerImpl< LOGGER_SEMIHOST >::Init ( ) [inline], [static]`

Initialize logging destination

8.39.2.2 Transmit()

```
static bool daisy::LoggerImpl< LOGGER_SEMIHOST >::Transmit (
            const void * buffer,
            size_t bytes )  [inline], [static]
```

Transmit a block of data

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

- src/hid/logger_impl.h

8.40 daisy::MidiEvent Struct Reference

```
#include <midi.h>
```

Public Member Functions

- NoteOnEvent AsNoteOn ()
- ControlChangeEvent AsControlChange ()

Public Attributes

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

8.40.1 Detailed Description

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

8.40.2 Member Function Documentation

8.40.2.1 AsControlChange()

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

Returns the data within the MidiEvent as a ControlChangeEvent struct.

8.40.2.2 AsNoteOn()

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

Returns the data within the MidiEvent as a NoteOnEvent struct

8.40.3 Member Data Documentation

8.40.3.1 channel

```
int daisy::MidiEvent::channel
```

&

8.40.3.2 data

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

&

8.40.3.3 type

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

&

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

- src/hid/midi.h

8.41 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 <midi.h>
```

Public Types

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

Public Member Functions

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

8.41.1 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

8.41.2 Member Enumeration Documentation

8.41.2.1 MidiInputMode

`enum daisy::MidiHandler::MidiInputMode`

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

Enumerator

INPUT_MODE_NONE	&
INPUT_MODE_UART1	&
INPUT_MODE_USB_INT	&
INPUT_MODE_USB_EXT	&

8.41.2.2 MidiOutputMode

`enum daisy::MidiHandler::MidiOutputMode`

Output mode

Enumerator

OUTPUT_MODE_NONE	&
OUTPUT_MODE_UART1	&
OUTPUT_MODE_USB_INT	&
OUTPUT_MODE_USB_EXT	&

8.41.3 Member Function Documentation

8.41.3.1 HasEvents()

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

Checks if there are unhandled messages in the queue

Returns

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

8.41.3.2 Init()

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

Initializes the MidiHandler

Parameters

in_mode	Input mode
out_mode	Output mode

8.41.3.3 Listen()

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

Start listening

8.41.3.4 Parse()

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

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

Parameters

byte	&

8.41.3.5 PopEvent()

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

Pops the oldest unhandled MidiEvent from the internal queue

Returns

The event to be handled

8.41.3.6 SendMessage()

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

SendMessage Send raw bytes as message

8.41.3.7 StartReceive()

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

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

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

- src/hid/midi.h

8.42 daisy::NoteOnEvent Struct Reference

```
#include <midi.h>
```

Public Attributes

- int channel
- uint8_t note
- uint8_t velocity

8.42.1 Detailed Description

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

8.42.2 Member Data Documentation

8.42.2.1 channel

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

&

8.42.2.2 note

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

&

8.42.2.3 velocity

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

&

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

- src/hid/midi.h

8.43 daisy::OledDisplay Class Reference

```
#include <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 (uint_fast8_t x, uint_fast8_t y, bool on)
- void DrawLine (uint_fast8_t x1, uint_fast8_t y1, uint_fast8_t x2, uint_fast8_t y2, bool on)
- void DrawRect (uint_fast8_t x1, uint_fast8_t y1, uint_fast8_t x2, uint_fast8_t y2, bool on, bool fill=false)
- void DrawArc (uint_fast8_t x, uint_fast8_t y, uint_fast8_t radius, int_fast16_t start_angle, int_fast16_t sweep, bool on)
- void DrawCircle (uint_fast8_t x, uint_fast8_t y, uint_fast8_t radius, bool on)
- char WriteChar (char ch, FontDef font, bool on)
- char WriteString (const char *str, FontDef font, bool on)
- void SetCursor (uint8_t x, uint8_t y)
- void Update ()

8.43.1 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

8.43.2 Member Enumeration Documentation

8.43.2.1 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

8.43.3 Member Function Documentation

8.43.3.1 DrawArc()

```
void daisy::OledDisplay::DrawArc (
            uint_fast8_t x,
            uint_fast8_t y,
            uint_fast8_t radius,
            int_fast16_t start_angle,
            int_fast16_t sweep,
            bool on )
```

Draws an arc around the specified coordinate

Parameters

x	x Coordinate of the center of the arc
y	y Coordinate of the center of the arc
radius	radius of the arc
start_angle	angle where to start the arc
sweep	total angle of the arc
on	on or off

8.43.3.2 DrawCircle()

```
void daisy::OledDisplay::DrawCircle (
            uint_fast8_t x,
            uint_fast8_t y,
            uint_fast8_t radius,
            bool on )  [inline]
```

Draws a circle around the specified coordinate

Parameters

x	x Coordinate of the center of the circle
y	y Coordinate of the center of the circle
radius	radius of the circle
on	on or off

8.43.3.3 DrawLine()

```
void daisy::OledDisplay::DrawLine (
            uint_fast8_t x1,
            uint_fast8_t y1,
            uint_fast8_t x2,
            uint_fast8_t y2,
            bool on )
```

Draws a line from (x1, y1) to (y1, y2)

Parameters

x1	x Coordinate of the starting point
y1	y Coordinate of the starting point
x2	x Coordinate of the ending point
y2	y Coordinate of the ending point
on	on or off

8.43.3.4 DrawPixel()

```
void daisy::OledDisplay::DrawPixel (
            uint_fast8_t x,
            uint_fast8_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

8.43.3.5 DrawRect()

```
void daisy::OledDisplay::DrawRect (
            uint_fast8_t x1,
            uint_fast8_t y1,
            uint_fast8_t x2,
            uint_fast8_t y2,
            bool on,
            bool fill = false )
```

Draws a rectangle based on two coordinates.

Parameters

x1	x Coordinate of the first point
y1	y Coordinate of the first point
x2	x Coordinate of the second point
y2	y Coordinate of the second point
on	on or off

8.43.3.6 Fill()

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

Fills the entire display with either on/off.

Parameters

on	Sets on or off.

8.43.3.7 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

8.43.3.8 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

8.43.3.9 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.

8.43.3.10 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

&

8.43.3.11 WriteString()

```
char daisy::OledDisplay::WriteString (
            const 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

8.44 daisy::Parameter Class Reference

```
#include <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 ()

8.44.1 Detailed Description

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

8.44.2 Member Enumeration Documentation

8.44.2.1 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.

8.44.3 Constructor & Destructor Documentation

8.44.3.1 Parameter()

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

Constructor

8.44.3.2 ~Parameter()

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

Destructor

8.44.4 Member Function Documentation

8.44.4.1 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.

8.44.4.2 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.

8.44.4.3 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

8.45 daisy::Pcm3060 Class Reference

Public Types

- enum class **Result** { **OK** , **ERR** }

Public Member Functions

- Result Init (I2CHandle i2c)

8.45.1 Member Function Documentation

8.45.1.1 Init()

```
Result daisy::Pcm3060::Init (
            I2CHandle i2c )
```

Initializes the PCM3060 in 24-bit MSB aligned I2S mode, and disables powersave

Parameters

i2c	Initialized I2CHandle configured at 400kHz or less

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

- src/dev/codec_pcm3060.h

8.46 daisy::RgbLed Class Reference

`#include <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 ()

8.46.1 Detailed Description

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

8.46.2 Member Function Documentation

8.46.2.1 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

8.46.2.2 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

8.46.2.3 SetColor()

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

Sets the RGB using a Color object.

Parameters

c	Color object to set.

8.46.2.4 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

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

```
#include <ringbuffer.h>
```

Public Member Functions

- void Init ()
- size_t capacity () const
- size_t writable () const
- size_t readable () const
- bool isEmpty () 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)
- void Advance (size_t num_elements)
- T * GetMutableBuffer ()

8.47.1 Detailed Description

template<typename T, size_t size>
class daisy::RingBuffer< T, size >

Utility Ring Buffer
imported from pichenettes/stmlib

8.47.2 Member Function Documentation

8.47.2.1 Advance()

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

Advances the write pointer, for when a peripheral is writing to the buffer.

8.47.2.2 capacity()

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

Returns

The total size of the ring buffer

8.47.2.3 Flush()

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

Flushes unread elements from the ring buffer

8.47.2.4 GetMutableBuffer()

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

Returns a pointer to the actual Ring Buffer Useful for when a peripheral needs direct access to the buffer.

8.47.2.5 ImmediateRead() [1/2]

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

Reads next element from ring buffer immediately

Returns

read value

8.47.2.6 ImmediateRead() [2/2]

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

Reads a number of elements into a buffer immediately

Parameters

destination	buffer to write to
num_elements	number of elements in buffer

8.47.2.7 Init()

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

Initializes the Ring Buffer

8.47.2.8 isEmpty()

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

Returns

True, if the buffer is empty.

8.47.2.9 Overwrite() [1/2]

```
template<typename T , size_t size>
void daisy::RingBuffer< T, size >::Overwrite (
            const T * source,
            size_t num_elements )  [inline]
```

Overwrites a number of elements using the source buffer as input.

Parameters

source	Input buffer
num_elements	Number of elements in source

8.47.2.10 Overwrite() [2/2]

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

Writes the new element to the ring buffer, overwriting unread data if necessary.

Parameters

v	Value to overwrite

8.47.2.11 Read()

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

Reads the first available element from the ring buffer

Returns

read value

8.47.2.12 readable()

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

Returns

number of unread elements in ring buffer

8.47.2.13 Swallow()

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

Read enough samples to make it possible to read 1 sample.

Parameters

n	Size of T ?

8.47.2.14 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.

8.47.2.15 Write()

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

Writes the value to the next available position in the ring buffer

Parameters

v	Value to write

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

- src/util/ringbuffer.h

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

`#include <ringbuffer.h>`

Public Member Functions

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

8.48.1 Detailed Description

template<typename T>
class daisy::RingBuffer< T, 0 >

Utility Ring Buffer imported from pichenettes/stmlib

8.48.2 Member Function Documentation

8.48.2.1 capacity()

```
template<typename T >
size_t daisy::RingBuffer< T, 0 >::capacity ( ) const  [inline]
```

Returns

0

8.48.2.2 Flush()

```
template<typename T >
void daisy::RingBuffer< T, 0 >::Flush ( )  [inline]
```

Flush the buffer

8.48.2.3 ImmediateRead() [1/2]

```
template<typename T >
T daisy::RingBuffer< T, 0 >::ImmediateRead ( )  [inline]
```

Returns

Read value

8.48.2.4 ImmediateRead() [2/2]

```
template<typename T >
void daisy::RingBuffer< T, 0 >::ImmediateRead (
            T * destination,
            size_t num_elements )  [inline]
```

Parameters

destination	&
num_elements	&

8.48.2.5 Init()

```
template<typename T >
void daisy::RingBuffer< T, 0 >::Init ( )  [inline]
```

Initialize ringbuffer

8.48.2.6 Overwrite() [1/2]

```
template<typename T >
void daisy::RingBuffer< T, 0 >::Overwrite (
            const T * source,
            size_t num_elements )  [inline]
```

Parameters

source	3
num_elements	&

8.48.2.7 Overwrite() [2/2]

```
template<typename T >
void daisy::RingBuffer< T, 0 >::Overwrite (
             T v )  [inline]
```

Parameters

v	Value to overwrite

8.48.2.8 Read()

```
template<typename T >
T daisy::RingBuffer< T, 0 >::Read ( )  [inline]
```

Returns

Read value

8.48.2.9 readable()

```
template<typename T >
size_t daisy::RingBuffer< T, 0 >::readable ( ) const  [inline]
```

Returns

0

8.48.2.10 writable()

```
template<typename T >
size_t daisy::RingBuffer< T, 0 >::writable ( ) const  [inline]
```

Returns

0

8.48.2.11 Write()

```
template<typename T >
void daisy::RingBuffer< T, 0 >::Write (
            T v )  [inline]
```

Parameters

v	Value to write

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

- src/util/ringbuffer.h

8.49 daisy::SaiHandle Class Reference

`#include <sai.h>`

Classes

- struct Config

Public Types

- enum class Result { **OK** , **ERR** }
- typedef void(∗ CallbackFunctionPtr) (int32_t ∗in, int32_t ∗out, size_t size)

Public Member Functions

- **SaiHandle** (const SaiHandle &other)=default
- SaiHandle & **operator=** (const SaiHandle &other)=default
- Result Init (const Config &config)
- const Config & GetConfig () const
- Result StartDma (int32_t ∗buffer_rx, int32_t ∗buffer_tx, size_t size, CallbackFunctionPtr callback)
- Result StopDma ()
- float GetSampleRate ()
- size_t GetBlockSize ()
- float GetBlockRate ()
- size_t GetOffset () const
- bool **IsInitialized** () const

8.49.1 Detailed Description

Support for I2S Audio Protocol with different bit-depth, samplerate options Allows for master or slave, as well as freedom of selecting direction, and other behavior for each peripheral, and block.

DMA Transfer commands must use buffers located within non-cached memory or use cache maintenance To declare an unitialized global element in the DMA memory section: int32_t DSY_DMA_BUFFER_SECTOR my_buffer[96];

Callback functions will be called once per half of the buffer. In the above example, the callback function would be called once for every 48 samples.

Use SAI Handle like this:

SaiHandle::Config sai_config; sai_config.periph = SaiHandle::Config::Peripheral::SAI_1; sai_config.sr = SaiHandle::Config::SampleRate::SAI_48KHZ; sai_config.bit_depth = SaiHandle::Config::BitDepth::SAI_24BIT; sai_config.a_sync = SaiHandle::Config::Sync::MASTER; sai_config.b_sync = SaiHandle::Config::Sync::SLAVE; sai_config.a_dir = SaiHandle::Config::Direction::RECEIVE; sai_config.b_dir = SaiHandle::Config::Direction::TRANSMIT; sai_config.pin_config.fs = {DSY_GPIOE, 4}; sai_config.pin_config.mclk = {DSY_GPIOE, 2}; sai_config.pin_config.sck = {DSY_GPIOE, 5}; sai_config.pin_config.sa = {DSY_GPIOE, 6}; sai_config.pin_config.sb = {DSY_GPIOE, 3}; // Then Initialize SaiHandle sai; sai.Init(sai_config); // Now you can use it: sai.StartDma(. . .);

8.49.2 Member Typedef Documentation

8.49.2.1 CallbackFunctionPtr

```
typedef void(* daisy::SaiHandle::CallbackFunctionPtr) (int32_t *in, int32_t *out, size_t size)
```

Callback Function to be called when DMA transfer is complete and half complete. This callback is prepared however the data is transmitted/received from the device. For example, using an AK4556 the data will be interleaved 24bit MSB Justified

The hid/audio class will be allow for type conversions, de-interleaving, etc.

8.49.3 Member Enumeration Documentation

8.49.3.1 Result

```
enum daisy::SaiHandle::Result  [strong]
```

Return values for SAI functions

8.49.4 Member Function Documentation

8.49.4.1 GetBlockRate()

```
float daisy::SaiHandle::GetBlockRate ( )
```

Returns the Block Rate of the current stream based on the size of the buffer passed in, and the current samplerate.

8.49.4.2 GetBlockSize()

```
size_t daisy::SaiHandle::GetBlockSize ( )
```

Returns the number of samples per audio block Calculated as Buffer Size / 2 / number of channels

8.49.4.3 GetConfig()

```
const Config& daisy::SaiHandle::GetConfig ( ) const
```

Returns the current configuration

8.49.4.4 GetOffset()

```
size_t daisy::SaiHandle::GetOffset ( ) const
```

Returns the current offset within the SAI buffer, will be either 0 or size/2

8.49.4.5 GetSampleRate()

```
float daisy::SaiHandle::GetSampleRate ( )
```

Returns the samplerate based on the current configuration

8.49.4.6 Init()

```
Result daisy::SaiHandle::Init (
            const Config & config )
```

Initializes an SAI peripheral

8.49.4.7 StartDma()

```
Result daisy::SaiHandle::StartDma (
            int32_t * buffer_rx,
            int32_t * buffer_tx,
            size_t size,
            CallbackFunctionPtr callback )
```

Starts Rx and Tx in Circular Buffer Mode The callback will be called when half of the buffer is ready, and will handle size/2 samples per callback.

8.49.4.8 StopDma()

```
Result daisy::SaiHandle::StopDma ( )
```

Stops the DMA stream for the SAI blocks in use.

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

- src/per/sai.h

8.50 daisy::ScopedIrqBlocker Class Reference

```
#include <scopedirqblocker.h>
```

8.50.1 Detailed Description

Tempoarily disables IRQ handlers with RAII techniques.

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

- src/util/scopedirqblocker.h

8.51 daisy::SdmmcHandler Class Reference

```
#include <sdmmc.h>
```

Public Member Functions

- void Init ()

8.51.1 Detailed Description

Configuration for interfacing with SD cards. Currently only supports operation using FatFS filesystem

8.51.2 Member Function Documentation

8.51.2.1 Init()

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

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

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

- src/per/sdmmc.h

8.52 daisy::SdmmcHandlerInit Struct Reference

```
#include <sdmmc.h>
```

Public Attributes

- SdmmcBitWidth bitdepth
- SdmmcSpeed speed

8.52.1 Detailed Description

Structure for setting the options above. Used to intiailize SdmmcHandler

8.52.2 Member Data Documentation

8.52.2.1 bitdepth

```
SdmmcBitWidth daisy::SdmmcHandlerInit::bitdepth
```

&

8.52.2.2 speed

```
SdmmcSpeed daisy::SdmmcHandlerInit::speed
```

&

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

- src/per/sdmmc.h

8.53 daisy::ShiftRegister4021< num_daisychained, num_parallel > Class Template Reference

Classes

- struct Config

Public Member Functions

- void Init (const Config &cfg)
- void Update ()
- const bool State (int index) const
- const Config & **GetConfig** () const

8.53.1 Member Function Documentation

8.53.1.1 Init()

```
template<size_t num_daisychained = 1, size_t num_parallel = 1>
void daisy::ShiftRegister4021< num_daisychained, num_parallel >::Init (
            const Config & cfg )  [inline]
```

Initializes the Device(s)

8.53.1.2 State()

```
template<size_t num_daisychained = 1, size_t num_parallel = 1>
const bool daisy::ShiftRegister4021< num_daisychained, num_parallel >::State (
            int index ) const  [inline]
```

returns the last read state of the input at the index. true indicates the pin is held HIGH.

See above for the layout of data when using multiple devices in series or parallel.

8.53.1.3 Update()

```
template<size_t num_daisychained = 1, size_t num_parallel = 1>
void daisy::ShiftRegister4021< num_daisychained, num_parallel >::Update ( )  [inline]
```

Reads the states of all pins on the connected device(s)

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

- src/dev/sr_4021.h

8.54 ShiftRegister595 Class Reference

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

`#include <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 ()

8.54.1 Detailed Description

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

Author

shensley

Date

May 2020

8.54.2 Member Enumeration Documentation

8.54.2.1 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"

8.54.3 Member Function Documentation

8.54.3.1 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.

8.54.3.2 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.

8.54.3.3 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

8.55 daisy::SpiHandle Class Reference

```
#include <spi.h>
```

Public Member Functions

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

8.55.1 Detailed Description

Handler for serial peripheral interface

8.55.2 Member Function Documentation

8.55.2.1 BlockingTransmit()

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

Blocking transmit

Parameters

*buff	input buffer
size	buffer size

8.55.2.2 Init()

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

Initializes handler

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

- src/per/spi.h

8.56 daisy::Switch Class Reference

```
#include <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
- bool RawState ()
- float TimeHeldMs () const

8.56.1 Detailed Description

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

Author

Stephen Hensley

Date

December 2019

8.56.2 Member Enumeration Documentation

8.56.2.1 Polarity

`enum daisy::Switch::Polarity`

Specifies whether the pressed is HIGH or LOW.

Enumerator

POLARITY_NORMAL	&
POLARITY_INVERTED	&

8.56.2.2 Pull

`enum daisy::Switch::Pull`

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

Enumerator

PULL_UP	&
PULL_DOWN	&
PULL_NONE	&

8.56.2.3 Type

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

Specifies the expected behavior of the switch

Enumerator

TYPE_TOGGLE	&
TYPE_MOMENTARY	&

8.56.3 Member Function Documentation

8.56.3.1 Debounce()

```
void daisy::Switch::Debounce ( )
```

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

8.56.3.2 FallingEdge()

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

Returns

true if the button was just released

8.56.3.3 Init() [1/2]

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

Simplified Init.

Parameters

pin	port/pin object to tell the switch which hardware pin to use.
update_rate	the rate at which the Debounce() function will be called. (used for timing).

8.56.3.4 Init() [2/2]

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

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

Parameters

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

8.56.3.5 Pressed()

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

Returns

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

8.56.3.6 RawState()

```
bool daisy::Switch::RawState ( )  [inline]
```

Returns

true if the button is held down, without debouncing

8.56.3.7 RisingEdge()

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

Returns

true if a button was just pressed.

8.56.3.8 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

8.57 daisy::Switch3 Class Reference

Public Types

- enum {
  POS_CENTER = 0 , **POS_LEFT** = 1 , **POS_UP** = 1 , **POS_RIGHT** = 2 ,
  POS_DOWN = 2 }

Public Member Functions

- void **Init** (dsy_gpio_pin pina, dsy_gpio_pin pinb)
- int **Read** ()

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

- src/hid/switch3.h

8.58 daisy::System Class Reference

Classes

- struct Config

Public Member Functions

- void Init ()
- void Init (const Config &config)
- void JumpToQspi ()
- const Config & GetConfig () const

Static Public Member Functions

- static uint32_t GetNow ()
- static uint32_t GetUs ()
- static uint32_t GetTick ()
- static void Delay (uint32_t delay_ms)
- static void DelayUs (uint32_t delay_us)
- static void DelayTicks (uint32_t delay_ticks)
- static uint32_t GetSysClkFreq ()
- static uint32_t GetHClkFreq ()
- static uint32_t GetPClk1Freq ()
- static uint32_t GetPClk2Freq ()

8.58.1 Member Function Documentation

8.58.1.1 Delay()

```
static void daisy::System::Delay (
            uint32_t delay_ms )  [static]
```

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

Parameters

delay_ms	Time to delay in ms

8.58.1.2 DelayTicks()

```
static void daisy::System::DelayTicks (
            uint32_t delay_ticks )  [static]
```

Blocking Delay using internal timer to wait

Parameters

delay_ticks	Time to ddelay in microseconds

8.58.1.3 DelayUs()

```
static void daisy::System::DelayUs (
            uint32_t delay_us )  [static]
```

Blocking Delay using internal timer to wait

Parameters

delay_us	Time to ddelay in microseconds

8.58.1.4 GetConfig()

```
const Config& daisy::System::GetConfig ( ) const  [inline]
```

Returns a const reference to the Systems Configuration struct

8.58.1.5 GetHClkFreq()

```
static uint32_t daisy::System::GetHClkFreq ( )  [static]
```

Returns the frequency of the HCLK (AHB) clock. This is derived from the System clock, and used to clock the CPU, memory, and peripherals mapped on the AHB, and APB Bus.

8.58.1.6 GetNow()

```
static uint32_t daisy::System::GetNow ( )  [static]
```

Returns

a uint32_t value of milliseconds since the SysTick started

8.58.1.7 GetPClk1Freq()

```
static uint32_t daisy::System::GetPClk1Freq ( )  [static]
```

Returns the frequency of the PCLK1 (APB1) clock This is used to clock various peripherals, and timers.

It's important to note that many timers run on a clock twice as fast as the peripheral clock for the timer.

8.58.1.8 GetPClk2Freq()

```
static uint32_t daisy::System::GetPClk2Freq ( )  [static]
```

Returns the frequency of the PCLK2 (APB2) clock This is used to clock various peripherals, and timers.

It's important to note that many timers run on a clock twice as fast as the peripheral clock for the timer.

8.58.1.9 GetSysClkFreq()

```
static uint32_t daisy::System::GetSysClkFreq ( )  [static]
```

Returns the Frequency of the system clock in Hz This is the primary system clock that is used to generate AXI Peripheral, APB, and AHB clocks.

8.58.1.10 GetTick()

```
static uint32_t daisy::System::GetTick ( )  [static]
```

Returns

a uint32_t of ticks at (PCLk1 $*$ 2)Hz Useful for measuring the number of CPU ticks something is taking.

8.58.1.11 GetUs()

```
static uint32_t daisy::System::GetUs ( )  [static]
```

Returns

a uint32_t of microseconds within the internal timer.

8.58.1.12 Init() [1/2]

```
void daisy::System::Init ( )
```

Default Initializer with no input will create an internal config, and set everything to Defaults

8.58.1.13 Init() [2/2]

```
void daisy::System::Init (
            const Config & config )
```

Configurable Initializer Initializes clock tree, DMA initializaiton and any necessary global inits.

8.58.1.14 JumpToQspi()

```
void daisy::System::JumpToQspi ( )
```

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

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

- src/sys/system.h

8.59 daisy::TimerHandle Class Reference

```
#include <tim.h>
```

Classes

- struct Config

Public Types

- enum class Result { **OK** , **ERR** }

Public Member Functions

- **TimerHandle** (const TimerHandle &other)=default
- TimerHandle & **operator=** (const TimerHandle &other)=default
- Result Init (const Config &config)
- const Config & GetConfig () const
- Result SetPeriod (uint32_t ticks)
- Result SetPrescaler (uint32_t val)
- Result Start ()
- Result Stop ()
- uint32_t GetFreq ()
- uint32_t GetTick ()
- uint32_t GetMs ()
- uint32_t GetUs ()
- void DelayTick (uint32_t del)
- void DelayMs (uint32_t del)
- void DelayUs (uint32_t del)

8.59.1 Detailed Description

Hardare timer peripheral support.

Supports general-function TIM peripherals:

- TIM2, TIM3, TIM4, TIM5

DaisySeed, and many internal peripherals utilize TIM2 for timing/delay purposes. It is configured to be at the maximum frequency: typically 200MHz or 240MHz (boost) for measuring/delaying for very short periods.

The GetUs/GetMs functions are available for convenience (and backwards compatibility), but to avoid wrapping errors on math when doing time-delta calculations, using ticks is recommended. The data can be converted to the final time-base after getting the difference in ticks. (Using GetFreq() can be used for these time-base calculations).

TODO:

- Fix issues with realtime getters, and wrapping of the timer(s).
  - This very noticeable with default settings for the 16-bit counters.
- Dispatch periodic callback(s)
- Other General purpose timers
- Non-internal clock sources
- Use of the four-tim channels per tim
  - PWM, etc.
  - InputCapture/OutputCompare, etc.
- HRTIM
- Advanced timers (TIM1/TIM8)

8.59.2 Member Enumeration Documentation

8.59.2.1 Result

```
enum daisy::TimerHandle::Result  [strong]
```

Return values for TIM funcitons.

8.59.3 Member Function Documentation

8.59.3.1 DelayMs()

```
void daisy::TimerHandle::DelayMs (
            uint32_t del )
```

Stay within this function for del milliseconds

8.59.3.2 DelayTick()

```
void daisy::TimerHandle::DelayTick (
            uint32_t del )
```

Stay within this function for del ticks

8.59.3.3 DelayUs()

```
void daisy::TimerHandle::DelayUs (
            uint32_t del )
```

Stay within this function for del microseconds

8.59.3.4 GetConfig()

```
const Config& daisy::TimerHandle::GetConfig ( ) const
```

Returns a const reference to the Config struct

8.59.3.5 GetFreq()

```
uint32_t daisy::TimerHandle::GetFreq ( )
```

Returns the frequency of each tick of the timer in Hz

8.59.3.6 GetMs()

```
uint32_t daisy::TimerHandle::GetMs ( )
```

Returns the ticks scaled as milliseconds

Use care when using for measurements and ensure that the TIM period can handle the maximum desired measurement.

8.59.3.7 GetTick()

```
uint32_t daisy::TimerHandle::GetTick ( )
```

Returns the number of counter position. This increments according to Config::CounterDir, and wraps around at the specified period (maxing out at 2^16 or 2^32 depending on the chosen TIM peripheral.

8.59.3.8 GetUs()

```
uint32_t daisy::TimerHandle::GetUs ( )
```

Returns the ticks scaled as microseconds

Use care when using for measurements and ensure that the TIM period can handle the maximum desired measurement.

8.59.3.9 Init()

```
Result daisy::TimerHandle::Init (
            const Config & config )
```

Initializes the timer according to the configuration

8.59.3.10 SetPeriod()

```
Result daisy::TimerHandle::SetPeriod (
            uint32_t ticks )
```

Sets the period of the Timer. This is the number of ticks it takes before it wraps back around. For self-managed timing, this can be left at the default. (0xffff for 16-bit and 0xffffffff for 32-bit timers). This can be changed "on-the-fly"

8.59.3.11 SetPrescaler()

```
Result daisy::TimerHandle::SetPrescaler (
            uint32_t val )
```

Sets the Prescalar applied to the TIM peripheral. This can be any number up to 0xffff This will adjust the rate of ticks: Calculated as APBN_Freq / prescalar per tick where APBN is APB1 for Most general purpose timers, and APB2 for HRTIM,a nd the advanced timers. This can be changed "on-the-fly"

8.59.3.12 Start()

```
Result daisy::TimerHandle::Start ( )
```

Starts the TIM peripheral specified by Config

8.59.3.13 Stop()

```
Result daisy::TimerHandle::Stop ( )
```

Stops the TIM peripheral specified by Config

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

- src/per/tim.h

8.60 daisy::UartHandler Class Reference

`#include <uart.h>`

Public Member Functions

- void Init ()
- int PollReceive (uint8_t *buff, size_t size, uint32_t timeout)
- int StartRx ()
- bool RxActive ()
- int FlushRx ()
- int PollTx (uint8_t *buff, size_t size)
- uint8_t PopRx ()
- size_t Readable ()
- int CheckError ()

8.60.1 Detailed Description

Uart Peripheral

Author

shensley

Date

March 2020

8.60.2 Member Function Documentation

8.60.2.1 CheckError()

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

Returns

the result of HAL_UART_GetError() to the user.

8.60.2.2 FlushRx()

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

Flushes the Receive Queue

Returns

OK or ERROR

8.60.2.3 Init()

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

Initializes the UART Peripheral

8.60.2.4 PollReceive()

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

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

Parameters

∗buff	Buffer to read to
size	Buff size
timeout	How long to timeout for (10ms?)

Returns

Data received

8.60.2.5 PollTx()

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

Sends an amount of data in blocking mode.

Parameters

∗buff	Buffer of data to send
size	Buffer size

Returns

OK or ERROR

8.60.2.6 PopRx()

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

Pops the oldest byte from the FIFO.

Returns

Popped byte

8.60.2.7 Readable()

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

Checks if there are any unread bytes in the FIFO

Returns

1 or 0 ??

8.60.2.8 RxActive()

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

Returns

whether Rx DMA is listening or not.

8.60.2.9 StartRx()

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

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. Size of the buffer is internally fixed to 256. Variable message lengths are transferred to the FIFO queue anytime there is 1 byte-period without incoming data

Returns

OK or ERROR

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

- src/per/uart.h

8.61 UsbHandle Class Reference

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

```
#include <usb.h>
```

Public Types

- enum class Result { **OK** , **ERR** , **OK** , **ERR** }
- enum UsbPeriph { FS_INTERNAL , FS_EXTERNAL , FS_BOTH , FS_INTERNAL , FS_EXTERNAL , FS_BOTH }
- enum class Result { **OK** , **ERR** , **OK** , **ERR** }
- enum UsbPeriph { FS_INTERNAL , FS_EXTERNAL , FS_BOTH , FS_INTERNAL , FS_EXTERNAL , FS_BOTH }
- typedef void(∗ ReceiveCallback) (uint8_t ∗buff, uint32_t ∗len)
- typedef void(∗ ReceiveCallback) (uint8_t ∗buff, uint32_t ∗len)

Public Member Functions

- void Init (UsbPeriph dev)
- Result TransmitInternal (uint8_t ∗buff, size_t size)
- Result TransmitExternal (uint8_t ∗buff, size_t size)
- void SetReceiveCallback (ReceiveCallback cb, UsbPeriph dev)
- void Init (UsbPeriph dev)
- Result TransmitInternal (uint8_t ∗buff, size_t size)
- Result TransmitExternal (uint8_t ∗buff, size_t size)
- void SetReceiveCallback (ReceiveCallback cb, UsbPeriph dev)

8.61.1 Detailed Description

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

Author

Stephen Hensley

Date

December 2019

8.61.2 Member Typedef Documentation

8.61.2.1 ReceiveCallback [1/2]

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

Function called upon reception of a buffer

8.61.2.2 ReceiveCallback [2/2]

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

Function called upon reception of a buffer

8.61.3 Member Enumeration Documentation

8.61.3.1 Result [1/2]

```
enum UsbHandle::Result  [strong]
```

Return values for USBHandle Functions

8.61.3.2 Result [2/2]

```
enum UsbHandle::Result  [strong]
```

Return values for USBHandle Functions

8.61.3.3 UsbPeriph [1/2]

```
enum UsbHandle::UsbPeriph
```

Specified which of the two USB Peripherals to initialize.

Enumerator

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

8.61.3.4 UsbPeriph [2/2]

```
enum UsbHandle::UsbPeriph
```

Specified which of the two USB Peripherals to initialize.

Enumerator

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

8.61.4 Member Function Documentation

8.61.4.1 Init() [1/2]

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

Initializes the specified peripheral(s) as USB CDC Devices

Parameters

dev	Device to initialize

8.61.4.2 Init() [2/2]

```
void UsbHandle::Init (
```

```
            UsbPeriph dev )
```

Initializes the specified peripheral(s) as USB CDC Devices

Parameters

dev	Device to initialize

8.61.4.3 SetReceiveCallback() [1/2]

```
void UsbHandle::SetReceiveCallback (
            ReceiveCallback cb,
            UsbPeriph dev )
```

sets the callback to be called upon reception of new data

Parameters

cb	Function to serve as callback
dev	Device to set callback for

8.61.4.4 SetReceiveCallback() [2/2]

```
void UsbHandle::SetReceiveCallback (
            ReceiveCallback cb,
            UsbPeriph dev )
```

sets the callback to be called upon reception of new data

Parameters

cb	Function to serve as callback
dev	Device to set callback for

8.61.4.5 TransmitExternal() [1/2]

```
Result 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

8.61.4.6 TransmitExternal() [2/2]

```
Result 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

8.61.4.7 TransmitInternal() [1/2]

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

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

Parameters

buff	Buffer to transmit
size	Buffer size

8.61.4.8 TransmitInternal() [2/2]

```
Result 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

8.62 WAV_FormatTypeDef Struct Reference

`#include <wav_format.h>`

Public Attributes

- uint32_t ChunkId
- uint32_t FileSize
- uint32_t FileFormat
- uint32_t SubChunk1ID
- uint32_t SubChunk1Size
- uint16_t AudioFormat
- uint16_t NbrChannels
- uint32_t SampleRate
- uint32_t ByteRate
- uint16_t BlockAlign
- uint16_t BitPerSample
- uint32_t SubChunk2ID
- uint32_t SubCHunk2Size

8.62.1 Detailed Description

Helper struct for handling the WAV file format

8.62.2 Member Data Documentation

8.62.2.1 AudioFormat

`uint16_t WAV_FormatTypeDef::AudioFormat`

&

8.62.2.2 BitPerSample

`uint16_t WAV_FormatTypeDef::BitPerSample`

&

8.62.2.3 BlockAlign

```
uint16_t WAV_FormatTypeDef::BlockAlign
```

&

8.62.2.4 ByteRate

```
uint32_t WAV_FormatTypeDef::ByteRate
```

&

8.62.2.5 ChunkId

```
uint32_t WAV_FormatTypeDef::ChunkId
```

&

8.62.2.6 FileFormat

```
uint32_t WAV_FormatTypeDef::FileFormat
```

&

8.62.2.7 FileSize

```
uint32_t WAV_FormatTypeDef::FileSize
```

&

8.62.2.8 NbrChannels

```
uint16_t WAV_FormatTypeDef::NbrChannels
```

&

8.62.2.9 SampleRate

```
uint32_t WAV_FormatTypeDef::SampleRate
```

&

8.62.2.10 SubChunk1ID

```
uint32_t WAV_FormatTypeDef::SubChunk1ID
```

&

8.62.2.11 SubChunk1Size

```
uint32_t WAV_FormatTypeDef::SubChunk1Size
```

&

8.62.2.12 SubChunk2ID

```
uint32_t WAV_FormatTypeDef::SubChunk2ID
```

&

8.62.2.13 SubCHunk2Size

```
uint32_t WAV_FormatTypeDef::SubCHunk2Size
```

&

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

- src/util/wav_format.h

8.63 daisy::WavFileInfo Struct Reference

```
#include <wavplayer.h>
```

Public Attributes

- WAV_FormatTypeDef raw_data
- char name [256]

8.63.1 Detailed Description

Struct containing details of Wav File.

8.63.2 Member Data Documentation

8.63.2.1 name

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

Wav filename

8.63.2.2 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

8.64 daisy::WavPlayer Class Reference

```
#include <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

8.64.1 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.

8.64.2 Member Function Documentation

8.64.2.1 Close()

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

Closes whatever file is currently open.

Returns

&

8.64.2.2 GetCurrentFile()

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

Returns

currently selected file.

8.64.2.3 GetLooping()

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

Returns

Whether the WavPlayer is looping or not.

8.64.2.4 GetNumberFiles()

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

Returns

The number of files loaded by the WavPlayer

8.64.2.5 Init()

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

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

8.64.2.6 Open()

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

Opens the file at index sel for reading.

Parameters

sel	File to open

8.64.2.7 Prepare()

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

Collects buffer for playback when needed.

8.64.2.8 Restart()

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

Resets the playback position to the beginning of the file immediately

8.64.2.9 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.

8.64.2.10 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

8.65 daisy::Wm8731 Class Reference

```
#include <codec_wm8731.h>
```

Classes

- struct Config

Public Types

- enum class Result { **OK** , **ERR** }

Public Member Functions

- Result Init (const Config &config, I2CHandle i2c)

8.65.1 Detailed Description

Device driver for Cirrus (Wolfsen) WM8731 Audio Codec

Currently only two-wire (I2C) interface format is supported, and only a limited set of features are configurable:

- Line inputs/outputs
- audio format/word length
- 48kHZ

Support for headphones, microphone, and full functionality still needs to be added.

Use the Driver like this (this will be compatible with the Daisy Seed audio/sai config): I2CHandle::Config i2c_config; I2CHandle i2c1_handle; Wm8731::Config codec_cfg; Wm8731 codec; i2c_config.periph = I2CHandle::Config::Peripheral::I2C_1; i2c_config.speed = I2CHandle::Config::Speed::I2C_400KHZ; i2c_config.↩ pin_config.scl = {DSY_GPIOB, 6}; i2c_config.pin_config.sda = {DSY_GPIOB, 9}; i2c1_handle.Init(i2c_config); codec_cfg.Defaults(); // MCU is master, 24-bit, MSB LJ codec.Init(codec_cfg, i2c1_handle);

8.65.2 Member Enumeration Documentation

8.65.2.1 Result

```
enum daisy::Wm8731::Result  [strong]
```

Return values for WM8731 Functions

8.65.3 Member Function Documentation

8.65.3.1 Init()

```
Result daisy::Wm8731::Init (
            const Config & config,
            I2CHandle i2c )
```

Initializes the WM8731 device

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

- src/dev/codec_wm8731.h

Chapter 9

File Documentation

9.1 src/sys/ffconf.h File Reference

```
#include "util/bsp_sd_diskio.h"
#include <stdlib.h>
```

Macros

- #define _FFCONF 68300
- #define _FS_READONLY 0
- #define _FS_MINIMIZE 0
- #define _USE_STRFUNC 2
- #define _USE_FIND 0
- #define _USE_MKFS 1
- #define _USE_FASTSEEK 1
- #define _USE_EXPAND 0
- #define _USE_CHMOD 0
- #define _USE_LABEL 0
- #define _USE_FORWARD 0
- #define _CODE_PAGE 850
- #define _USE_LFN 1
- #define _MAX_LFN 255
- #define _LFN_UNICODE 0
- #define _STRF_ENCODE 3
- #define _FS_RPATH 0
- #define _VOLUMES 1
- #define _STR_VOLUME_ID 0
- #define _VOLUME_STRS
- #define _MULTI_PARTITION 0
- #define _MIN_SS 512
- #define _MAX_SS 512
- #define _USE_TRIM 0
- #define _FS_NOFSINFO 0
- #define _FS_TINY 0
- #define _FS_EXFAT 0
- #define _FS_NORTC 0

- #define _NORTC_MON 6
- #define _NORTC_MDAY 4
- #define _NORTC_YEAR 2015
- #define _FS_LOCK 2
- #define _FS_REENTRANT 0
- #define _FS_TIMEOUT 1000
- #define _SYNC_t osSemaphoreId
- #define ff_malloc malloc
- #define ff_free free

9.1.1 Detailed Description

Further fatfs support.

9.1.2 Macro Definition Documentation

9.1.2.1 _CODE_PAGE

```
#define _CODE_PAGE  850
```

This option specifies the OEM code page to be used on the target system. / Incorrect setting of the code page can cause a file open failure. / / 1 - ASCII (No extended character. Non-LFN cfg. only) / 437 - U.S. / 720 - Arabic / 737 - Greek / 771 - KBL / 775 - Baltic / 850 - Latin 1 / 852 - Latin 2 / 855 - Cyrillic / 857 - Turkish / 860 - Portuguese / 861 - Icelandic / 862 - Hebrew / 863 - Canadian French / 864 - Arabic / 865 - Nordic / 866 - Russian / 869 - Greek 2 / 932 - Japanese (DBCS) / 936 - Simplified Chinese (DBCS) / 949 - Korean (DBCS) / 950 - Traditional Chinese (DBCS)

9.1.2.2 _FFCONF

```
#define _FFCONF 68300
```

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

Attention

9.1.2.3 _FS_EXFAT

```
#define _FS_EXFAT  0
```

This option switches support of exFAT file system. (0:Disable or 1:Enable) / When enable exFAT, also LFN needs to be enabled. (_USE_LFN >= 1) / Note that enabling exFAT discards C89 compatibility.

9.1.2.4 _FS_LOCK

```
#define _FS_LOCK  2
```

0:Disable or >=1:Enable The option _FS_LOCK switches file lock function to control duplicated file open / and illegal operation to open objects. This option must be 0 when _FS_READONLY / is 1. / / 0: Disable file lock function. To avoid volume corruption, application program / should avoid illegal open, remove and rename to the open objects. / >0: Enable file lock function. The value defines how many files/sub-directories / can be opened simultaneously under file lock control. Note that the file / lock control is independent of re-entrancy.

9.1.2.5 _FS_MINIMIZE

```
#define _FS_MINIMIZE  0
```

0 to 3 This option defines minimization level to remove some basic API functions. / / 0: All basic functions are enabled. / 1: f_stat(), f_getfree(), f_unlink(), f_mkdir(), f_truncate() and f_rename() / are removed. / 2: f_opendir(), f_readdir() and f_closedir() are removed in addition to 1. / 3: f_lseek() function is removed in addition to 2.

9.1.2.6 _FS_NOFSINFO

```
#define _FS_NOFSINFO  0
```

0,1,2 or 3 If you need to know correct free space on the FAT32 volume, set bit 0 of this / option, and f_getfree() function at first time after volume mount will force / a full FAT scan. Bit 1 controls the use of last allocated cluster number. / / bit0=0: Use free cluster count in the FSINFO if available. / bit0=1: Do not trust free cluster count in the FSINFO. / bit1=0: Use last allocated cluster number in the FSINFO if available. / bit1=1: Do not trust last allocated cluster number in the FSINFO.

9.1.2.7 _FS_NORTC

```
#define _FS_NORTC 0
```

&

9.1.2.8 _FS_READONLY

```
#define _FS_READONLY  0
```

0:Read/Write or 1:Read only This option switches read-only configuration. (0:Read/Write or 1:Read-only) / Read-only configuration removes writing API functions, f_write(), f_sync(), / f_unlink(), f_mkdir(), f_chmod(), f_rename(), f_truncate(), f_getfree() / and optional writing functions as well.

9.1.2.9 _FS_REENTRANT

```
#define _FS_REENTRANT 0
```

0:Disable or 1:Enable

9.1.2.10 _FS_RPATH

```
#define _FS_RPATH  0
```

0 to 2 This option configures support of relative path. / / 0: Disable relative path and remove related functions. / 1: Enable relative path. f_chdir() and f_chdrive() are available. / 2: f_getcwd() function is available in addition to 1.

9.1.2.11 _FS_TIMEOUT

```
#define _FS_TIMEOUT 1000
```

Timeout period in unit of time ticks

9.1.2.12 _FS_TINY

```
#define _FS_TINY  0
```

0:Normal or 1:Tiny This option switches tiny buffer configuration. (0:Normal or 1:Tiny) / At the tiny configuration, size of file object (FIL) is reduced _MAX_SS bytes. / Instead of private sector buffer eliminated from the file object, common sector / buffer in the file system object (FATFS) is used for the file data transfer.

9.1.2.13 _LFN_UNICODE

```
#define _LFN_UNICODE  0
```

0:ANSI/OEM or 1:Unicode This option switches character encoding on the API. (0:ANSI/OEM or 1:UTF-16) / To use Unicode string for the path name, enable LFN and set _LFN_UNICODE = 1. / This option also affects behavior of string I/O functions.

9.1.2.14 _MAX_LFN

```
#define _MAX_LFN  255
```

Maximum LFN length to handle (12 to 255) The _USE_LFN switches the support of long file name (LFN). / / 0: Disable support of LFN. _MAX_LFN has no effect. / 1: Enable LFN with static working buffer on the BSS. Always NOT thread-safe. / 2: Enable LFN with dynamic working buffer on the STACK. / 3: Enable LFN with dynamic working buffer on the HEAP. / / To enable the LFN, Unicode handling functions (option/unicode.c) must be added / to the project. The working buffer occupies (_MAX_LFN + 1) $*$ 2 bytes and / additional 608 bytes at exFAT enabled. _MAX_LFN can be in range from 12 to 255. / It should be set 255 to support full featured LFN operations. / When use stack for the working buffer, take care on stack overflow. When use heap / memory for the working buffer, memory management functions, ff_memalloc() and / ff_memfree(), must be added to the project.

9.1.2.15 _MAX_SS

```
#define _MAX_SS  512
```

512, 1024, 2048 or 4096 These options configure the range of sector size to be supported. (512, 1024, / 2048 or 4096) Always set both 512 for most systems, all type of memory cards and / harddisk. But a larger value may be required for on-board flash memory and some / type of optical media. When _MAX_SS is larger than _MIN_↩ SS, FatFs is configured / to variable sector size and GET_SECTOR_SIZE command must be implemented to the / disk_ioctl() function.

9.1.2.16 _MIN_SS

```
#define _MIN_SS 512
```

512, 1024, 2048 or 4096

9.1.2.17 _MULTI_PARTITION

```
#define _MULTI_PARTITION  0
```

0:Single partition, 1:Multiple partition This option switches support of multi-partition on a physical drive. / By default (0), each logical drive number is bound to the same physical drive / number and only an FAT volume found on the physical drive will be mounted. / When multi-partition is enabled (1), each logical drive number can be bound to / arbitrary physical drive and partition listed in the VolToPart[]. Also f_fdisk() / funciton will be available.

9.1.2.18 _NORTC_MDAY

```
#define _NORTC_MDAY 4
```

&

9.1.2.19 _NORTC_MON

```
#define _NORTC_MON 6
```

&

9.1.2.20 _NORTC_YEAR

```
#define _NORTC_YEAR  2015
```

The option _FS_NORTC switches timestamp functiton. If the system does not have / any RTC function or valid timestamp is not needed, set _FS_NORTC = 1 to disable / the timestamp function. All objects modified by FatFs will have a fixed timestamp / defined by _NORTC_MON, _NORTC_MDAY and _NORTC_YEAR in local time. / To enable timestamp function (_FS_NORTC = 0), get_fattime() function need to be / added to the project to get current time form real-time clock. _NORTC_MON, / _NORTC_MDAY and _NORTC_YEAR have no effect. / These options have no effect at read-only configuration (_FS_READONLY = 1).

9.1.2.21 _STR_VOLUME_ID

```
#define _STR_VOLUME_ID  0
```

0:Use only 0-9 for drive ID, 1:Use strings for drive ID

9.1.2.22 _STRF_ENCODE

```
#define _STRF_ENCODE  3
```

When _LFN_UNICODE == 1, this option selects the character encoding ON THE FILE to / be read/written via string I/O functions, f_gets(), f_putc(), f_puts and f_printf(). / / 0: ANSI/OEM / 1: UTF-16LE / 2: UTF-16BE / 3: UTF-8 / / This option has no effect when _LFN_UNICODE == 0.

9.1.2.23 _SYNC_t

```
#define _SYNC_t  osSemaphoreId
```

The option _FS_REENTRANT switches the re-entrancy (thread safe) of the FatFs / module itself. Note that regardless of this option, file access to different / volume is always re-entrant and volume control functions, f_mount(), f_mkfs() / and f_fdisk() function, are always not re-entrant. Only file/directory access / to the same volume is under control of this function. / / 0: Disable re-entrancy. _FS_TIMEOUT and _SYNC_t have no effect. / 1: Enable re-entrancy. Also user provided synchronization handlers, / ff_req_grant(), ff_rel_grant(), ff_del_syncobj() and ff_cre_syncobj() / function, must be added to the project. Samples are available in / option/syscall.c. / / The _FS↩ _TIMEOUT defines timeout period in unit of time tick. / The _SYNC_t defines O/S dependent sync object type. e.g. HANDLE, ID, OS_EVENT∗, / SemaphoreHandle_t and etc.. A header file for O/S definitions needs to be / included somewhere in the scope of ff.h.

9.1.2.24 _USE_CHMOD

```
#define _USE_CHMOD  0
```

This option switches attribute manipulation functions, f_chmod() and f_utime(). / (0:Disable or 1:Enable) Also _↩ FS_READONLY needs to be 0 to enable this option.

9.1.2.25 _USE_EXPAND

```
#define _USE_EXPAND  0
```

This option switches f_expand function. (0:Disable or 1:Enable)

9.1.2.26 _USE_FASTSEEK

```
#define _USE_FASTSEEK  1
```

This option switches fast seek feature. (0:Disable or 1:Enable)

9.1.2.27 _USE_FIND

```
#define _USE_FIND  0
```

This option switches filtered directory read functions, f_findfirst() and / f_findnext(). (0:Disable, 1:Enable 2:Enable with matching altname[] too)

9.1.2.28 _USE_FORWARD

```
#define _USE_FORWARD  0
```

This option switches f_forward() function. (0:Disable or 1:Enable)

9.1.2.29 _USE_LABEL

```
#define _USE_LABEL  0
```

This option switches volume label functions, f_getlabel() and f_setlabel(). / (0:Disable or 1:Enable)

9.1.2.30 _USE_LFN

```
#define _USE_LFN 1
```

0 to 3

9.1.2.31 _USE_MKFS

```
#define _USE_MKFS  1
```

This option switches f_mkfs() function. (0:Disable or 1:Enable)

9.1.2.32 _USE_STRFUNC

```
#define _USE_STRFUNC  2
```

0:Disable or 1-2:Enable This option switches string functions, f_gets(), f_putc(), f_puts() and / f_printf(). / / 0: Disable string functions. / 1: Enable without LF-CRLF conversion. / 2: Enable with LF-CRLF conversion.

9.1.2.33 _USE_TRIM

```
#define _USE_TRIM  0
```

This option switches support of ATA-TRIM. (0:Disable or 1:Enable) / To enable Trim function, also CTRL_TRIM command should be implemented to the / disk_ioctl() function.

9.1.2.34 _VOLUME_STRS

```
#define _VOLUME_STRS
```

Value:
```
    "RAM", "NAND", "CF", "SD1", "SD2", "USB1", "USB2", \
        "USB3"
```

_STR_VOLUME_ID switches string support of volume ID. / When _STR_VOLUME_ID is set to 1, also pre-defined strings can be used as drive / number in the path name. _VOLUME_STRS defines the drive ID strings for each / logical drives. Number of items must be equal to _VOLUMES. Valid characters for / the drive ID strings are: A-Z and 0-9.

9.1.2.35 _VOLUMES

```
#define _VOLUMES 1
```

Number of volumes (logical drives) to be used.

9.1.2.36 ff_free

```
#define ff_free  free
```

define the ff_malloc ff_free macros as standard malloc free

9.1.2.37 ff_malloc

```
#define ff_malloc  malloc
```

define the ff_malloc ff_free macros as standard malloc free

9.2 src/usbd/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)
- void CDC_Set_Rx_Callback_HS (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

9.2.1 Detailed Description

: Header for usbd_cdc_if.c file.

Version

: v1.0_Cube

Attention

9.3 src/usbd/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(...)

9.3.1 Detailed Description

: Header for usbd_conf.c file.

Version

: v1.0_Cube

Attention

9.4 src/usbd/usbd_desc.h File Reference

: Header for usbd_conf.c file.

```
#include "usbd_def.h"
```

Macros

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

Variables

- USBD_DescriptorsTypeDef HS_Desc
- USBD_DescriptorsTypeDef FS_Desc

9.4.1 Detailed Description

: Header for usbd_conf.c file.

Version

: v1.0_Cube

Attention

Index