

StreamingTx Libraries

Generated by Doxygen 1.8.5

Mon Nov 27 2017 19:34:49

Contents

1	Module Index	1
1.1	Modules	1
2	Data Structure Index	3
2.1	Data Structures	3
3	Module Documentation	5
3.1	Analog to Digital Conversion	5
3.1.1	Detailed Description	5
3.1.2	Enumeration Type Documentation	5
3.1.2.1	adc_channel	5
3.1.3	Function Documentation	5
3.1.3.1	adc_value	5
3.2	General Purpose Input/Output	7
3.2.1	Detailed Description	7
3.2.2	Enumeration Type Documentation	7
3.2.2.1	gpio_config	7
3.2.2.2	gpio_pins	8
3.2.3	Function Documentation	8
3.2.3.1	gpio_clear	8
3.2.3.2	gpio_config	8
3.2.3.3	gpio_get	9
3.2.3.4	gpio_set	9
3.2.3.5	gpio_toggle	9
3.3	Cyclic Redundancy Check	10
3.3.1	Detailed Description	10
3.4	UART input/output	11
3.4.1	Detailed Description	11
3.5	Utility functions	12
3.5.1	Detailed Description	12
3.5.2	Function Documentation	12
3.5.2.1	chip_init	12

3.5.2.2	delay_ms	12
3.5.2.3	delay_us	12
4	Data Structure Documentation	13
4.1	gpio_regs Struct Reference	13
4.1.1	Detailed Description	13
4.2	telem_firmware Struct Reference	13
4.2.1	Detailed Description	13
4.3	telem_packet Struct Reference	14
4.3.1	Detailed Description	14
4.4	telem_play Struct Reference	14
4.4.1	Detailed Description	14
4.5	telem_status Struct Reference	14
4.5.1	Detailed Description	15
4.6	telem_tx_status Struct Reference	15
4.6.1	Detailed Description	15
Index		16

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Analog to Digital Conversion	5
General Purpose Input/Output	7
Cyclic Redundancy Check	10
UART input/output	11
Utility functions	12

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

gpio_regs	Declaration of how the hardware is laid out on STM8 processors (e.g	13
telem_firmware	Telemetry packet for the command to write to new firmware	13
telem_packet	Telemetry packet from RX to TX	14
telem_play	Telemetry packet for the command to play a tune	14
telem_status	Telemetry status packet	14
telem_tx_status	Tx_status structure sent one byte at a time to RX	15

Chapter 3

Module Documentation

3.1 Analog to Digital Conversion

Enumerations

- enum `adc_channel` { `STICK_ROLL` = 1, `STICK_PITCH` = 0, `STICK_THROTTLE` = 3, `STICK_YAW` = 2 }
- The meaning of each analog channel, assuming mode2 stick mapping.*

Functions

- void `adc_init` (void)
This function initialises the ADC module.
- uint16_t `adc_value` (uint8_t chan)
This function returns the most recently converted data from a specified channel.
- void `adc_irq` (void)
This is the interrupt routine for supporting ADC conversions.

3.1.1 Detailed Description

3.1.2 Enumeration Type Documentation

3.1.2.1 enum `adc_channel`

The meaning of each analog channel, assuming mode2 stick mapping.

Enumerator

- `STICK_ROLL`** Right joystick horizontal axis.
- `STICK_PITCH`** Right joystick vertical axis.
- `STICK_THROTTLE`** Left joystick vertical axis.
- `STICK_YAW`** Left joystick horizontal axis.

3.1.3 Function Documentation

3.1.3.1 uint16_t `adc_value` (uint8_t *chan*)

This function returns the most recently converted data from a specified channel.

Returns

Returns the raw input value (not normalised).

Parameters

<i>chan</i>	Which channel are we interested in now. See adc_channel
-------------	---

3.2 General Purpose Input/Output

Support raw GPIO access.

Data Structures

- struct `gpio_regs`

Declaration of how the hardware is laid out on STM8 processors (e.g.

Enumerations

- enum `gpio_pins` {
`GPIO_PORTA` = 0x000, `GPIO_PORTB` = 0x100, `GPIO_PORTC` = 0x200, `GPIO_PORTD` = 0x300,
`GPIO_PORTE` = 0x400, `GPIO_PORTF` = 0x500, `GPIO_PORTG` = 0x600, `GPIO_PORTH` = 0x700,
`GPIO_PORTI` = 0x800, `GPIO_PIN0` = (1 << 0), `GPIO_PIN1` = (1 << 1), `GPIO_PIN2` = (1 << 2),
`GPIO_PIN3` = (1 << 3), `GPIO_PIN4` = (1 << 4), `GPIO_PIN5` = (1 << 5), `GPIO_PIN6` = (1 << 6),
`GPIO_PIN7` = (1 << 7) }

Definition of ports; one of these can be ored with one or more pin bits to refer to a collection of pins on a single port.

- enum `gpio_config` {
`GPIO_INPUT_FLOAT` = 0x0, `GPIO_INPUT_PULLUP` = 0x2, `GPIO_INPUT_FLOAT_IRQ` = 0x1, `GPIO_INPU-`
`T_PULLUP_IRQ` = 0x3,
`GPIO_OUTPUT_OPEN_DRAIN` = 0x0, `GPIO_OUTPUT_PUSHPULL` = 0x6, `GPIO_OUTPUT_OPEN_DRAIN-`
`_FAST` = 0x5, `GPIO_OUTPUT_PUSHPULL_FAST` = 0x7,
`GPIO_SET` = 0x10, `GPIO_CLEAR` = 0x20 }

Configuration values, for `gpio_config`.

Functions

- void `gpio_config` (uint16_t pins, enum `gpio_config` config)
Configure one or more pins on a port.
- void `gpio_set` (uint16_t pins)
Set one or more pins on a port high.
- void `gpio_clear` (uint16_t pins)
Set one or more pins on a port low.
- void `gpio_toggle` (uint16_t pins)
Toggle one or more pins on a port between high and low.
- bool `gpio_get` (uint16_t pin)
Get the current state of an input pin.

3.2.1 Detailed Description

Support raw GPIO access. This module is for configuring and using GPIO pins directly within the project.

3.2.2 Enumeration Type Documentation

3.2.2.1 enum `gpio_config`

Configuration values, for `gpio_config`.

Enumerator

`GPIO_INPUT_FLOAT` Input pin with no pullup.

GPIO_INPUT_PULLUP Input pin with internal pullup resistor active.

GPIO_INPUT_FLOAT_IRQ Input pin with no pullup; generates IRQ.

GPIO_INPUT_PULLUP_IRQ Input pin with internal pullup resistor active; generates IRQ.

GPIO_OUTPUT_OPEN_DRAIN Output pin as open drain.

GPIO_OUTPUT_PUSHPULL Output pin as push pull.

GPIO_OUTPUT_OPEN_DRAIN_FAST Output pin as open drain with fast response.

GPIO_OUTPUT_PUSHPULL_FAST Output pin as push pull with fast response.

GPIO_SET Flag to set a GPIO.

GPIO_CLEAR Flag to clear a GPIO.

3.2.2.2 enum gpio_pins

Definition of ports; one of these can be ored with one or more pin bits to refer to a collection of pins on a single port.

Enumerator

GPIO_PORTA Port A.

GPIO_PORTB Port B.

GPIO_PORTC Port C.

GPIO_PORTD Port D.

GPIO_PORTE Port E.

GPIO_PORTF Port F.

GPIO_PORTG Port G.

GPIO_PORTH Port H.

GPIO_PORTI Port I.

GPIO_PIN0 Pin 0 of a port.

GPIO_PIN1 Pin 1 of a port.

GPIO_PIN2 Pin 2 of a port.

GPIO_PIN3 Pin 3 of a port.

GPIO_PIN4 Pin 4 of a port.

GPIO_PIN5 Pin 5 of a port.

GPIO_PIN6 Pin 6 of a port.

GPIO_PIN7 Pin 7 of a port.

3.2.3 Function Documentation

3.2.3.1 void gpio_clear (uint16_t pins)

Set one or more pins on a port low.

Assumes the port is configured for output.

Parameters

<i>pins</i>	One or more pins to set low on a single specified GPIO port. See gpio_pins
-------------	--

3.2.3.2 void gpio_config (uint16_t pins, enum gpio_config config)

Configure one or more pins on a port.

Parameters

<i>pins</i>	One or more pins to configure on a single specified GPIO port. See gpio_pins
<i>config</i>	The configuration format wanted for the specified pin(s)

3.2.3.3 `bool gpio_get (uint16_t pin)`

Get the current state of an input pin.

Assumes the port is configured for digital input.

Returns

Returns true if at least one specified GPIO pin is high (false if all are low).

Parameters

<i>pin</i>	One or more pins to test on a single specified GPIO port. See gpio_pins
------------	---

3.2.3.4 `void gpio_set (uint16_t pins)`

Set one or more pins on a port high.

Assumes the port is configured for output.

Parameters

<i>pins</i>	One or more pins to set high on a single specified GPIO port. See gpio_pins
-------------	---

3.2.3.5 `void gpio_toggle (uint16_t pins)`

Toggle one or more pins on a port between high and low.

Assumes the port is configured for output.

Parameters

<i>pins</i>	One or more pins to toggle between high and low on a single specified GPIO port. See gpio_pins
-------------	--

3.3 Cyclic Redundancy Check

Support calculating CRCs.

Functions

- `uint8_t crc_crc8` (`const uint8_t *p`, `uint16_t len`)
8-bit crc
- `uint32_t crc_crc32` (`const uint8_t *p`, `uint16_t len`)
a poor-mans crc32, re-using the crc16 table

3.3.1 Detailed Description

Support calculating CRCs.

3.4 UART input/output

Functions

- void `uart2_init` (void)
Initialise UART2 for output debugging.
- void `uart2_putchar` (char c)
Output a single character to UART2.
- void `uart2_write` (const char *str)
Output a nul-terminated string to UART2.

3.4.1 Detailed Description

3.5 Utility functions

Support utility functions such as chip setup, LED, timing and maths.

Functions

- void `chip_init` (void)
Initialise the chip and PCB.
- void `led_init` (void)
Initialise the LEDs.
- void `led_green_set` (bool set)
Turn the green LED on or off as specified.
- void `led_yellow_set` (bool set)
Turn the yellow LED on or off as specified.
- void `led_green_toggle` (void)
Toggle the green LED on or off.
- void `led_yellow_toggle` (void)
Toggle the yellow LED on or off.
- void `delay_ms` (uint16_t d)
Busy loop to wait a number of milliseconds.
- void `delay_us` (uint16_t d)
Busy loop to wait a number of microseconds.
- uint16_t `get_random16` (void)
Simple 16 bit random number generator.

3.5.1 Detailed Description

Support utility functions such as chip setup, LED, timing and maths.

3.5.2 Function Documentation

3.5.2.1 void `chip_init` (void)

Initialise the chip and PCB.

This function is specific to the hardware layout

3.5.2.2 void `delay_ms` (uint16_t d)

Busy loop to wait a number of milliseconds.

3.5.2.3 void `delay_us` (uint16_t d)

Busy loop to wait a number of microseconds.

Chapter 4

Data Structure Documentation

4.1 gpio_regs Struct Reference

Declaration of how the hardware is laid out on STM8 processors (e.g.

Data Fields

- uint8_t [ODR](#)
Output data register.
- uint8_t [IDR](#)
Input data register.
- uint8_t [DDR](#)
Data direction register.
- uint8_t [CR1](#)
Control register one.
- uint8_t [CR2](#)
Control register two.

4.1.1 Detailed Description

Declaration of how the hardware is laid out on STM8 processors (e.g. STM85105)

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

- E:/ArduPilot/StreamingGPSTransmitter/lib/gpio.c

4.2 telem_firmware Struct Reference

Telemetry packet for the command to write to new firmware.

```
#include <telem_structure.h>
```

4.2.1 Detailed Description

Telemetry packet for the command to write to new firmware.

This is also used to play a tune.

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

- E:/ArduPilot/StreamingGPSTransmitter/include/telem_structure.h

4.3 telem_packet Struct Reference

telemetry packet from RX to TX

```
#include <telem_structure.h>
```

Data Fields

- uint8_t [crc](#)
simple CRC

4.3.1 Detailed Description

telemetry packet from RX to TX

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

- E:/ArduPilot/StreamingGPSTransmitter/include/telem_structure.h

4.4 telem_play Struct Reference

Telemetry packet for the command to play a tune.

```
#include <telem_structure.h>
```

4.4.1 Detailed Description

Telemetry packet for the command to play a tune.

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

- E:/ArduPilot/StreamingGPSTransmitter/include/telem_structure.h

4.5 telem_status Struct Reference

Telemetry status packet.

```
#include <telem_structure.h>
```

Data Fields

- uint8_t [pps](#)
packets per second received
- uint8_t [rssi](#)
lowpass rssi
- uint8_t [flags](#)

- TELEM_FLAG_**
 - `uint8_t flight_mode`
flight mode

4.5.1 Detailed Description

Telemetry status packet.

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

- `E:/ArduPilot/StreamingGPSTransmitter/include/telem_structure.h`

4.6 telem_tx_status Struct Reference

tx_status structure sent one byte at a time to RX.

```
#include <telem_structure.h>
```

Data Fields

- `uint8_t crc`
Simple crc.

4.6.1 Detailed Description

tx_status structure sent one byte at a time to RX.

This is packed into channels 8, 9 and 10 (using 32 bits of a possible 33)

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

- `E:/ArduPilot/StreamingGPSTransmitter/include/telem_structure.h`

Index

- adc_channel
 - Analog to Digital Conversion, [5](#)
- adc_value
 - Analog to Digital Conversion, [5](#)
- Analog to Digital Conversion, [5](#)
 - adc_channel, [5](#)
 - adc_value, [5](#)
 - STICK_PITCH, [5](#)
 - STICK_ROLL, [5](#)
 - STICK_THROTTLE, [5](#)
 - STICK_YAW, [5](#)
- chip_init
 - Utility functions, [12](#)
- Cyclic Redundancy Check, [10](#)
- delay_ms
 - Utility functions, [12](#)
- delay_us
 - Utility functions, [12](#)
- GPIO_CLEAR
 - General Purpose Input/Output, [8](#)
- GPIO_INPUT_FLOAT
 - General Purpose Input/Output, [7](#)
- GPIO_INPUT_FLOAT_IRQ
 - General Purpose Input/Output, [8](#)
- GPIO_INPUT_PULLUP
 - General Purpose Input/Output, [7](#)
- GPIO_INPUT_PULLUP_IRQ
 - General Purpose Input/Output, [8](#)
- GPIO_OUTPUT_OPEN_DRAIN
 - General Purpose Input/Output, [8](#)
- GPIO_OUTPUT_OPEN_DRAIN_FAST
 - General Purpose Input/Output, [8](#)
- GPIO_OUTPUT_PUSHPULL
 - General Purpose Input/Output, [8](#)
- GPIO_OUTPUT_PUSHPULL_FAST
 - General Purpose Input/Output, [8](#)
- GPIO_PIN0
 - General Purpose Input/Output, [8](#)
- GPIO_PIN1
 - General Purpose Input/Output, [8](#)
- GPIO_PIN2
 - General Purpose Input/Output, [8](#)
- GPIO_PIN3
 - General Purpose Input/Output, [8](#)
- GPIO_PIN4
 - General Purpose Input/Output, [8](#)
- GPIO_PIN5
 - General Purpose Input/Output, [8](#)
- GPIO_PIN6
 - General Purpose Input/Output, [8](#)
- GPIO_PIN7
 - General Purpose Input/Output, [8](#)
- GPIO_PORTA
 - General Purpose Input/Output, [8](#)
- GPIO_PORTB
 - General Purpose Input/Output, [8](#)
- GPIO_PORTC
 - General Purpose Input/Output, [8](#)
- GPIO_PORTD
 - General Purpose Input/Output, [8](#)
- GPIO_PORTE
 - General Purpose Input/Output, [8](#)
- GPIO_PORTF
 - General Purpose Input/Output, [8](#)
- GPIO_PORTG
 - General Purpose Input/Output, [8](#)
- GPIO_PORTH
 - General Purpose Input/Output, [8](#)
- GPIO_PORTI
 - General Purpose Input/Output, [8](#)
- GPIO_SET
 - General Purpose Input/Output, [8](#)
- General Purpose Input/Output
 - GPIO_CLEAR, [8](#)
 - GPIO_INPUT_FLOAT, [7](#)
 - GPIO_INPUT_FLOAT_IRQ, [8](#)
 - GPIO_INPUT_PULLUP, [7](#)
 - GPIO_INPUT_PULLUP_IRQ, [8](#)
 - GPIO_OUTPUT_OPEN_DRAIN, [8](#)
 - GPIO_OUTPUT_OPEN_DRAIN_FAST, [8](#)
 - GPIO_OUTPUT_PUSHPULL, [8](#)
 - GPIO_OUTPUT_PUSHPULL_FAST, [8](#)
 - GPIO_PIN0, [8](#)
 - GPIO_PIN1, [8](#)
 - GPIO_PIN2, [8](#)
 - GPIO_PIN3, [8](#)
 - GPIO_PIN4, [8](#)
 - GPIO_PIN5, [8](#)
 - GPIO_PIN6, [8](#)
 - GPIO_PIN7, [8](#)
 - GPIO_PORTA, [8](#)
 - GPIO_PORTB, [8](#)
 - GPIO_PORTC, [8](#)
 - GPIO_PORTD, [8](#)
 - GPIO_PORTE, [8](#)
 - GPIO_PORTF, [8](#)

- GPIO_PORTG, [8](#)
- GPIO_PORTH, [8](#)
- GPIO_PORTI, [8](#)
- GPIO_SET, [8](#)
- General Purpose Input/Output, [7](#)
 - gpio_clear, [8](#)
 - gpio_config, [7](#), [8](#)
 - gpio_get, [9](#)
 - gpio_pins, [8](#)
 - gpio_set, [9](#)
 - gpio_toggle, [9](#)
- gpio_clear
 - General Purpose Input/Output, [8](#)
- gpio_config
 - General Purpose Input/Output, [7](#), [8](#)
- gpio_get
 - General Purpose Input/Output, [9](#)
- gpio_pins
 - General Purpose Input/Output, [8](#)
- gpio_regs, [13](#)
- gpio_set
 - General Purpose Input/Output, [9](#)
- gpio_toggle
 - General Purpose Input/Output, [9](#)
- STICK_PITCH
 - Analog to Digital Conversion, [5](#)
- STICK_ROLL
 - Analog to Digital Conversion, [5](#)
- STICK_THROTTLE
 - Analog to Digital Conversion, [5](#)
- STICK_YAW
 - Analog to Digital Conversion, [5](#)
- telem_firmware, [13](#)
- telem_packet, [14](#)
- telem_play, [14](#)
- telem_status, [14](#)
- telem_tx_status, [15](#)
- UART input/output, [11](#)
- Utility functions, [12](#)
 - chip_init, [12](#)
 - delay_ms, [12](#)
 - delay_us, [12](#)