USART

1.0

Generated by Doxygen 1.8.7

Sun Aug 21 2016 13:32:14

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

Chapter 3

Class Documentation

3.1 DataConverter Union Reference

define the union type used to convert between types.

`#include <common.h>`

Public Attributes

- long double d34_t
  64bit IEEE floating point number
- float f32_t [2]
  32bit IEEE float point number
- uint32_t ui32_t [2]
  unsigned 32bit.
- int32_t i32_t [2]
  signed 32bit.
- uint16_t ui16_t [4]
  unsigned 16bit.
- int16_t i16_t [4]
  signed 16bit.
- uint8_t ui8_t [8]
  unsigned 8bit.
- int8_t i8_t [8]
  singed 8bit.

3.1.1 Detailed Description

define the union type used to convert between types.

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

- /home/vagrant/Workspaces/Embedded/USART/usart/include/common.h

3.2 FIFOInterface Struct Reference

Define the interface for the FIFO list.

`#include <FiFoStructure.h>`

Public Attributes

- uint_fast8_t(∗ Initialize)(FIFOQueue ∗queue)

  Initialize the queue.
- uint_fast8_t(∗ Insert)(FIFOQueue ∗queue, const uint8_t byte)

  Insert an item at the end of the queue.
- uint_fast8_t(∗ Remove)(FIFOQueue ∗queue, uint8_t ∗dest)

  Remove an item from the front of the queue.

3.2.1 Detailed Description

Define the interface for the FIFO list.

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

- /home/vagrant/Workspaces/Embedded/USART/usart/include/LIST/FiFoStructure.h

3.3 FIFOQueue Struct Reference

The structure for the FIFO queue as an array of uint8_t.

`#include <FiFoStructure.h>`

Public Attributes

- uint8_t items [MAXQUEUESIZE]

  The FIFO buffer of uint8_t.
- uint8_t front

  The front or read/remove point of the queue.
- uint8_t rear

  The rear or insert point of the queue.

3.3.1 Detailed Description

The structure for the FIFO queue as an array of uint8_t.

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

- /home/vagrant/Workspaces/Embedded/USART/usart/include/LIST/FiFoStructure.h

3.4 SerialInterface Struct Reference

define the standard serial interface

`#include <SerialStructure.h>`

Public Attributes

- uint_fast8_t(∗ IsSerialOpen)(void)

  return the serial connection state
- uint_fast8_t(∗ Open)(const uint32_t baudrate)

opens the serial connection

- void(∗ Close)(void)

  closes serial connection

- uint_fast8_t(∗ SendByte)(const uint8_t source)

  send a single byte

- uint_fast8_t(∗ SendString)(const uint8_t ∗source)

  send a string that. The string should be terminated by null character

- uint_fast8_t(∗ SendArray)(const uint8_t ∗source, uint32_t length)

  send an array of data

- int_fast8_t(∗ DoesReceiveBufferHaveData)(void)

  return the state of the serial receive buffer

- int_fast8_t(∗ GetByte)(uint8_t ∗destination)

  get a single byte from the serial

3.4.1 Detailed Description

define the standard serial interface

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

- /home/vagrant/Workspaces/Embedded/USART/usart/include/MCU/SerialStructure.h

3.5 TickType Struct Reference

defines a non-blocking delay data type.

`#include <tick.h>`

Public Attributes

- uint32_t StartMs

  do not modify directly. Use Tick_DelayMs_NonBlocking

- uint32_t DelayMs

  Set the desire delay.

3.5.1 Detailed Description

defines a non-blocking delay data type.

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

- /home/vagrant/Workspaces/Embedded/USART/usart/include/MCU/tick.h

Chapter 4

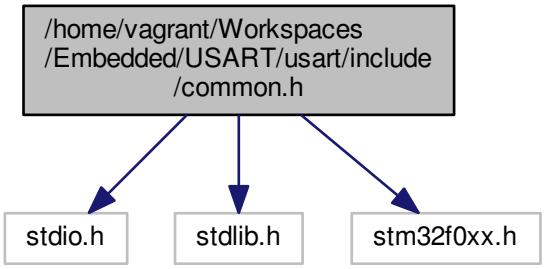
File Documentation

4.1 /home/vagrant/Workspaces/Embedded/USART/usart/include/common.h File Reference

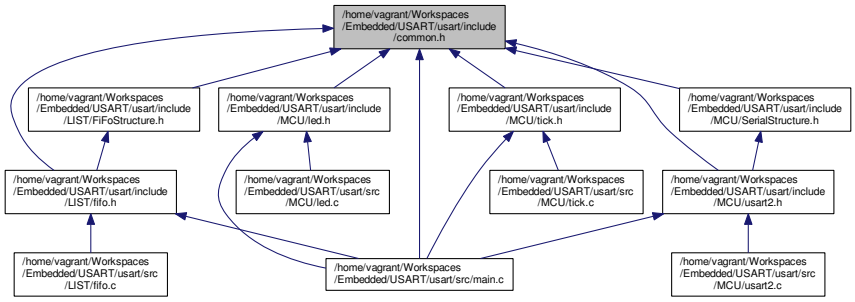
Holds all common code definitions.

```
#include <stdio.h>
#include <stdlib.h>
#include "stm32f0xx.h"
```

Include dependency graph for common.h:



This graph shows which files directly or indirectly include this file:



Classes

- union DataConverter

  define the union type used to convert between types.

Macros

- #define TRUE 1

  Defines the true state.
- #define FALSE 0

  Defines the false state.
- #define ERROR 2

  Defines the error state.
- #define FIRMWARE_VERSION "00.0001D"

  Firmware version D = development version of the firmware. Should only be used for testing purposes C = concession version. This version of the firmware is usual custome for a customer. see CONCESSION_NUMBER P = production version.
- #define HARDWARE_VERSION "00"

  Hardware version.
- #define COMPILED_DATA_TIME "[" __DATE__ " " __TIME__ "]"

  Hardware version.
- #define EN_DEBUG_INTERFACE

  Enables the debug interface and all debug message associated.

4.1.1 Detailed Description

Holds all common code definitions.

Author

: Ronald Sousa ()

4.1.2 Macro Definition Documentation

4.1.2.1 #define FIRMWARE_VERSION "00.0001D"

Firmware version D = development version of the firmware. Should only be used for testing purposes C = concession version. This version of the firmware is usual custome for a customer. see CONCESSION_NUMBER P = production version.

See also

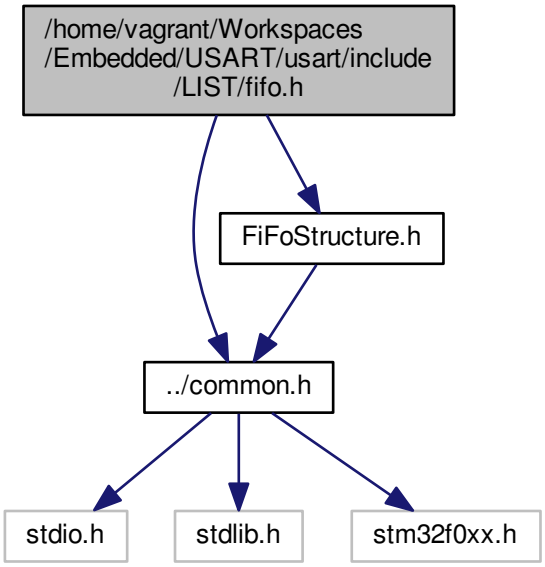
CONCESSION_NUMBER

4.2 /home/vagrant/Workspaces/Embedded/USART/usart/include/LIST/fifo.h File Reference

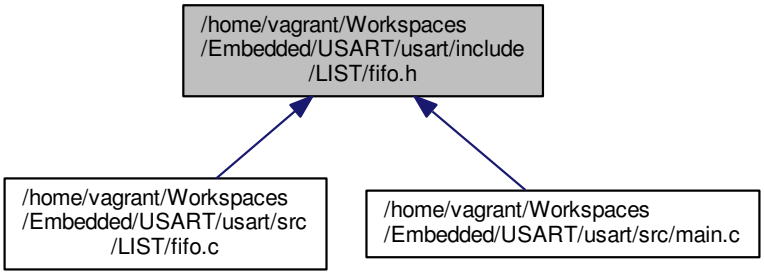
FIFO Library.

```
#include "../common.h"
#include "FiFoStructure.h"
```

Include dependency graph for fifo.h:



This graph shows which files directly or indirectly include this file:



Variables

- FIFOInterface FIFO

  Defines the standard implementation for the FIFO queue.

4.2.1 Detailed Description

FIFO Library.

Author

Steve Mayze

4.2.2 Variable Documentation

4.2.2.1 FIFOInterface FIFO

Defines the standard implementation for the FIFO queue.

See also

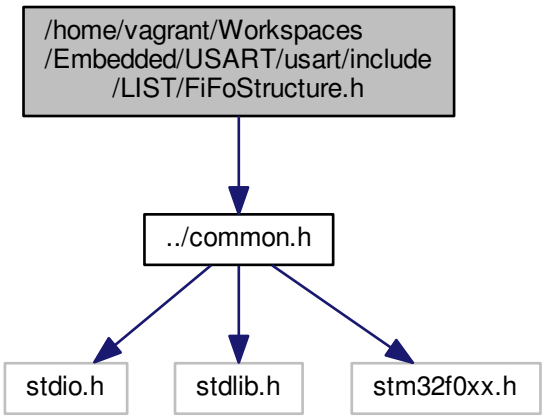
FiFoStructure.h

4.3 /home/vagrant/Workspaces/Embedded/USART/usart/include/LIST/FiFoStructure.h File Reference

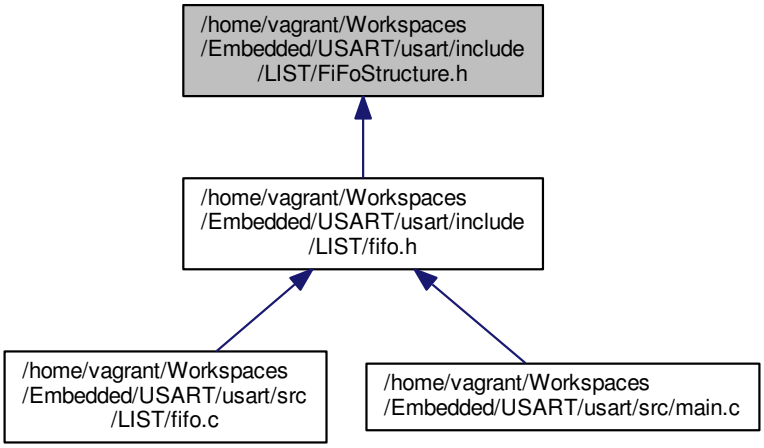
FIFO Library API structure.

```
#include "../common.h"
```

Include dependency graph for FiFoStructure.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct FIFOQueue

  The structure for the FIFO queue as an array of uint8_t.
- struct FIFOInterface

  Define the interface for the FIFO list.

Macros

- #define MAXQUEUESIZE 100

  defines the FIFO buffer maximum size

4.3.1 Detailed Description

FIFO Library API structure.

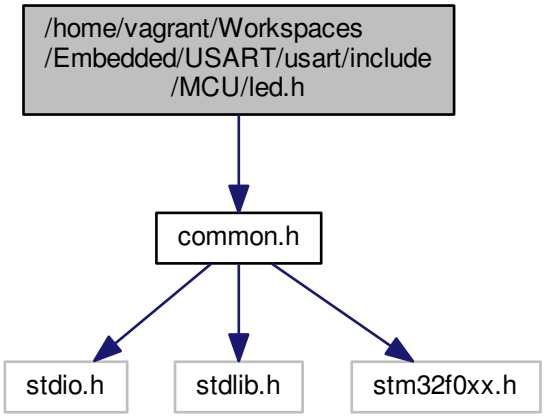
Author

Steve Mayze

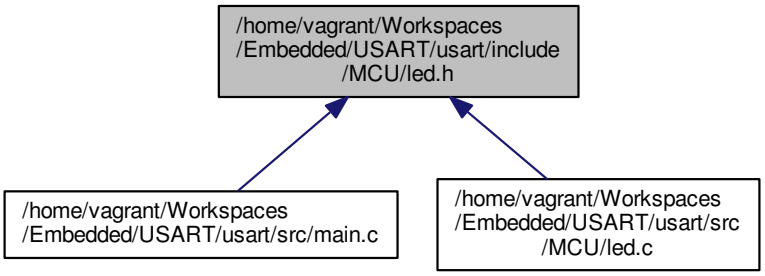
4.4 /home/vagrant/Workspaces/Embedded/USART/usart/include/MCU/led.h File Reference

```
#include "common.h"
```

Include dependency graph for led.h:



This graph shows which files directly or indirectly include this file:



Functions

- void Led_Init (void)

  Setup the LED IO.
- void Led_On (void)

  Turns on the LED.
- void Led_Off (void)

  Turns off the LED.
- void Led_Toggle (void)

  Toggle the LED state.

4.4.1 Detailed Description

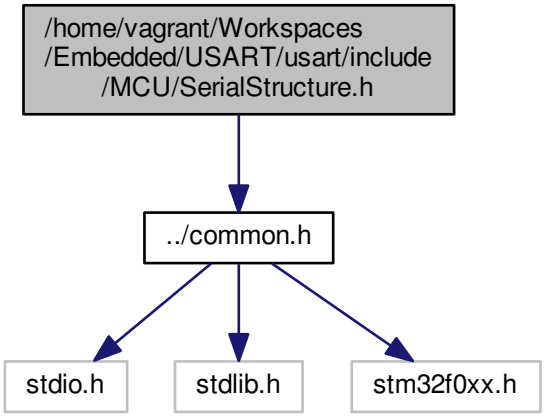
Author: Ronald Sousa ()

4.5 /home/vagrant/Workspaces/Embedded/USART/usart/include/MCU/SerialStructure.h File Reference

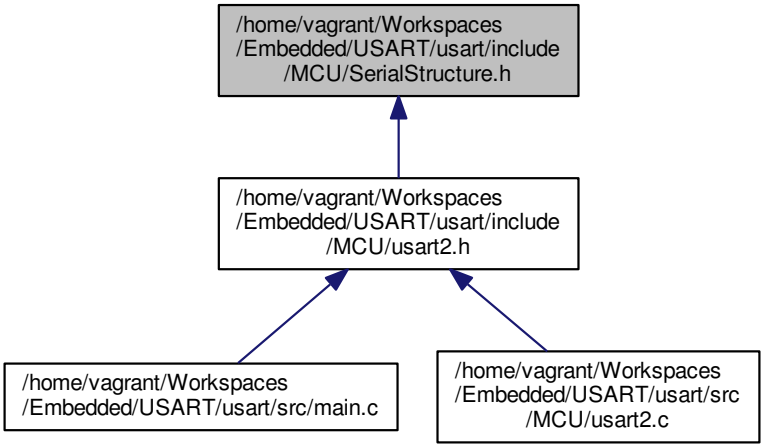
define the serial interface layer structure

```
#include "../common.h"
```

Include dependency graph for SerialStructure.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct SerialInterface

  define the standard serial interface

4.5.1 Detailed Description

define the serial interface layer structure

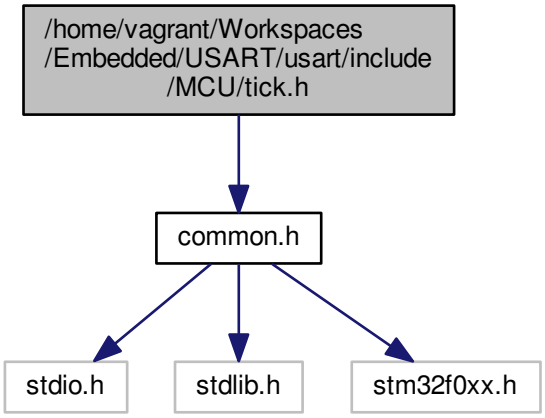
Author

Ronald Sousa www.HashDefineElectronics.com Hash Define Electronics Ltd

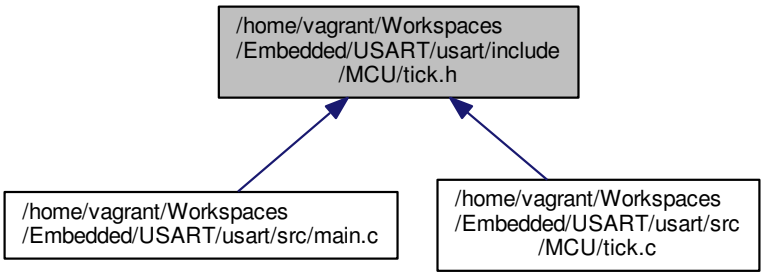
4.6 /home/vagrant/Workspaces/Embedded/USART/usart/include/MCU/tick.h File Reference

```
#include "common.h"
```

Include dependency graph for tick.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct TickType

defines a non-blocking delay data type.

Functions

- void Tick_init (void)

  setup the ARM M0 tick counter to trigger every 1ms
- uint32_t Tick_GetMs (void)

  return the number of mili-seconds since power up.
- int_fast8_t Tick_DelayMs_NonBlocking (uint_fast8_t reset, TickType *config)

  Non-blocking delay in ms.
- void Tick_DelayMs (uint32_t delayMs)

  this is a blocking delay.

4.6.1 Detailed Description

Author: Ronald Sousa ()

4.6.2 Function Documentation

4.6.2.1 void Tick_DelayMs (uint32_t *delayMs*)

this is a blocking delay.

```
#include "common.h"
#include "MCU/led.h"
#include "MCU/tick.h"

void main(void) {

    Led_Init();
    Tick_init();

    for( ;;) {
      Tick_DelayMs(1000); // delay 1s;
      Led_Toggle();
    }
}
```

Parameters

delayMs	how long to delay for.

4.6.2.2 int_fast8_t Tick_DelayMs_NonBlocking (uint_fast8_t *reset*, TickType * *config*)

Non-blocking delay in ms.

```
#include "common.h"
#include "MCU/led.h"
#include "MCU/tick.h"

void main(void) {
    TickType Delay;
    Delay.DelayMs = 1000; //set to 1s

    Led_Init();
    Tick_init();

    // reset the counter
    Tick_DelayMs_NonBlocking(TRUE, &Delay);

    for( ;;) {

        if(Tick_DelayMs_NonBlocking(TRUE, &Delay)) {
```

```
            // Delay has been reached

            Tick_DelayMs_NonBlocking(TRUE, &Delay);
            Led_Toggle();
            }
        else {
            // User code when the code delay hasn't passed
            }

    }
}
```

Parameters

reset	true = reset timer start value/ false = Check if time has lapsed
config	delay setting

Returns

1 = the desire delay has been reached. 0 = not reached the desire delay time. -1 = config point is null

Here is the call graph for this function:

4.6.2.3 uint32_t Tick_GetMs (void)

return the number of mili-seconds since power up.

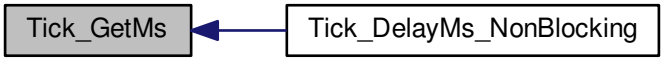
Returns

number of mili-seconds.

Note

the tick counter is expected to overflow and therefore code using the tick value should take this into account.

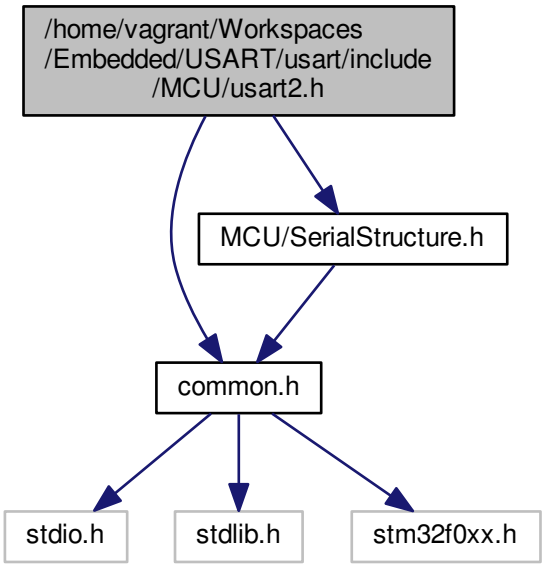
Here is the caller graph for this function:



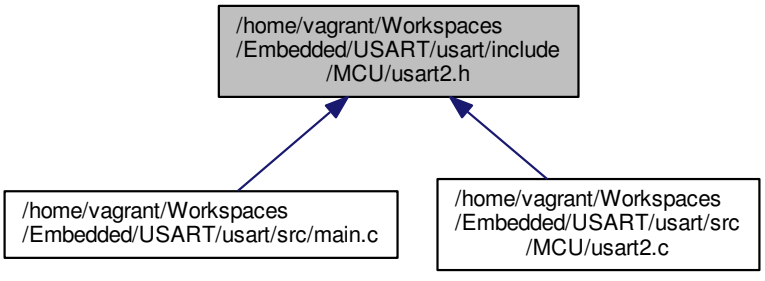
4.7 /home/vagrant/Workspaces/Embedded/USART/usart/include/MCU/usart2.h File Reference

```
#include "common.h"
#include "MCU/SerialStructure.h"
```

Include dependency graph for usart2.h:



This graph shows which files directly or indirectly include this file:



Variables

- SerialInterface SerialPort2

  Defines the standard serial functions for usart 2.

4.7.1 Detailed Description

Author

: Ronald Sousa ()

4.7.2 Variable Documentation

4.7.2.1 SerialInterface SerialPort2

Defines the standard serial functions for usart 2.

See also

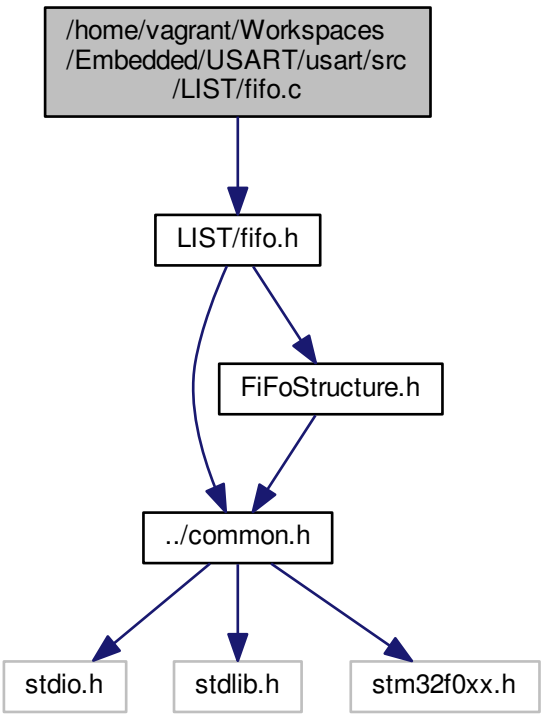
SerialInterface

4.8 /home/vagrant/Workspaces/Embedded/USART/usart/src/LIST/fifo.c File Reference

FIFO Library.

```
#include "LIST/fifo.h"
```

Include dependency graph for fifo.c:



Variables

- FIFOInterface FIFO

Defines the standard implementation for the FIFO queue.

4.8.1 Detailed Description

FIFO Library.

Author

Steve Mayze

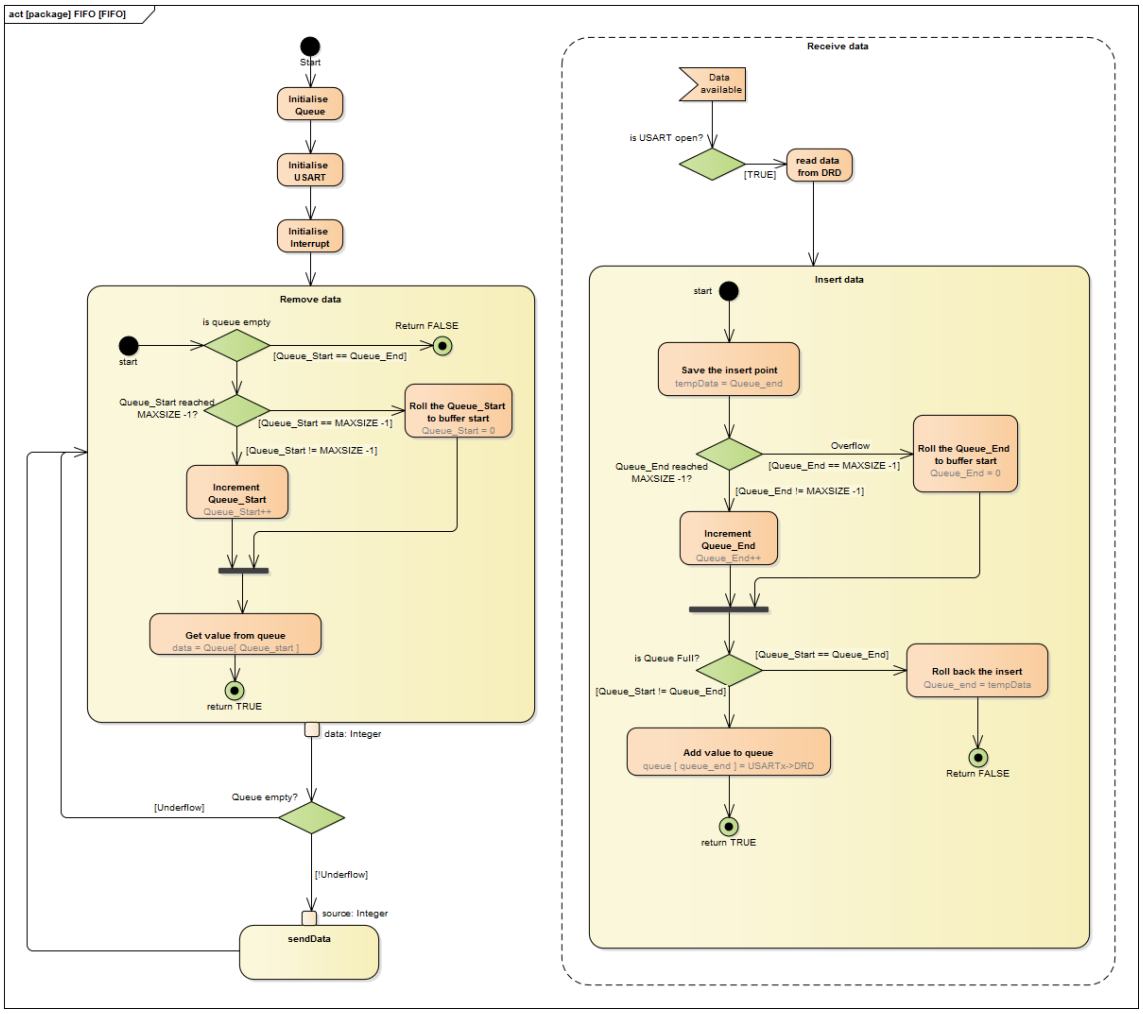


Figure 4.1: FIFO API Activity diagram

4.8.2 Variable Documentation

4.8.2.1 FIFOInterface FIFO

Initial value:

```
= {
        Initialize,
        Insert,
        Remove
}
```

Defines the standard implementation for the FIFO queue.

See also

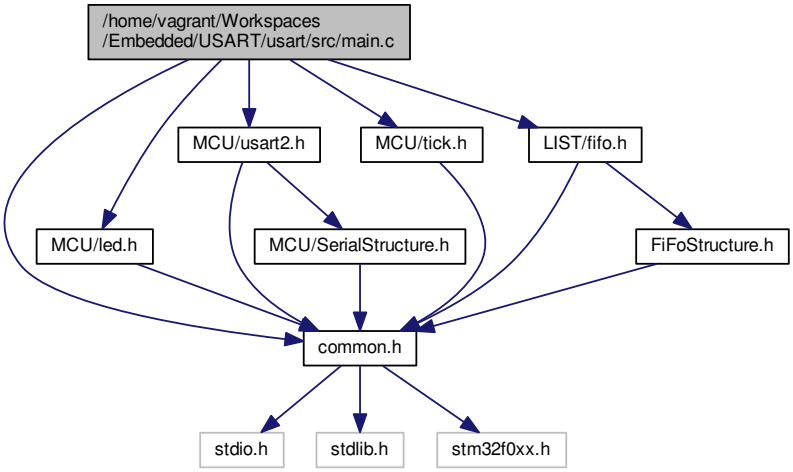
FiFoStructure.h

4.9 /home/vagrant/Workspaces/Embedded/USART/usart/src/main.c File Reference

This is the main program code.

```
#include "common.h"
#include "MCU/led.h"
#include "MCU/usart2.h"
#include "MCU/tick.h"
#include "LIST/fifo.h"
```

Include dependency graph for main.c:



Functions

- void main (void)

  the first user code function to be called after the ARM M0 has initial.

4.9.1 Detailed Description

This is the main program code.

Author

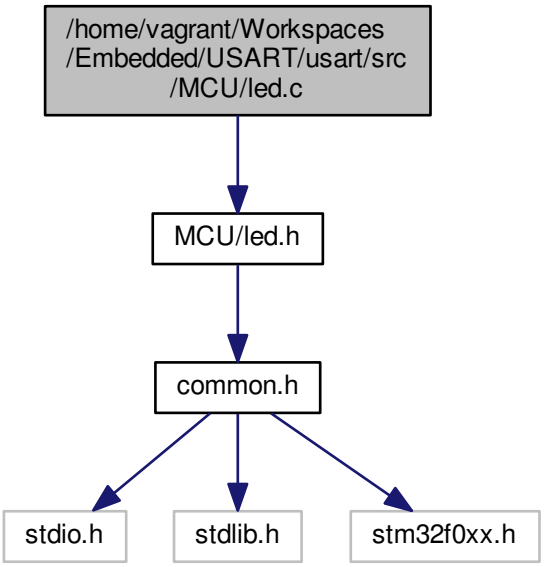
: Ronald Sousa (Opticalworm)

4.10 /home/vagrant/Workspaces/Embedded/USART/usart/src/MCU/led.c File Reference

this is the LED hardware interface layer.

```
#include "MCU/led.h"
```

Include dependency graph for led.c:



Macros

- #define LED_PIN 5

  defines the LED pin number

Functions

- void Led_On (void)

  Turns on the LED.
- void Led_Off (void)

  Turns off the LED.
- void Led_Toggle (void)

  Toggle the LED state.
- void Led_Init (void)

  Setup the LED IO.

4.10.1 Detailed Description

this is the LED hardware interface layer.

Author

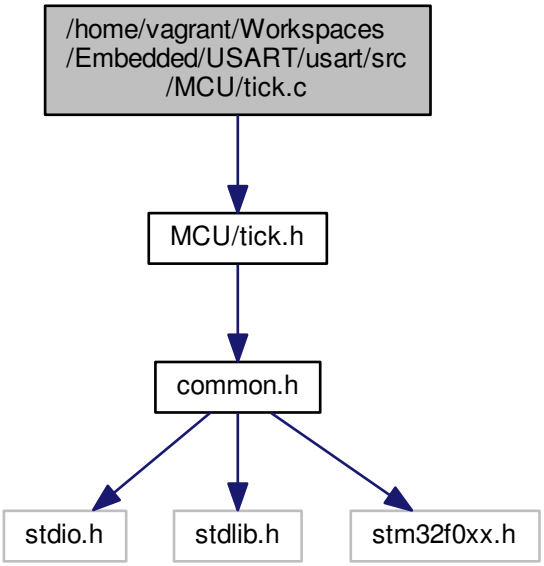
: Ronald Sousa ()

4.11 /home/vagrant/Workspaces/Embedded/USART/usart/src/MCU/tick.c File Reference

implements mili-second tick counter.

```
#include "MCU/tick.h"
```

Include dependency graph for tick.c:



Macros

- #define TIMER_FREQUENCY_HZ 1000

  defines the frequency we want the system tick to trigger. for 1ms = 1/1000hz

Functions

- void Tick_init (void)

  setup the ARM M0 tick counter to trigger every 1ms
- uint32_t Tick_GetMs (void)

  return the number of mili-seconds since power up.
- void Tick_DelayMs (uint32_t delayMs)

  this is a blocking delay.
- int_fast8_t Tick_DelayMs_NonBlocking (uint_fast8_t reset, TickType *config)

  Non-blocking delay in ms.
- void SysTick_Handler (void)

  ARM M0 hardware interrupt. This should trigger every 1 ms and update TickCounter.

4.11.1 Detailed Description

implements mili-second tick counter.

Author

: Ronald Sousa (Opticalworm)

4.11.2 Function Documentation

4.11.2.1 void SysTick_Handler (void)

ARM M0 hardware interrupt. This should trigger every 1 ms and update TickCounter.

See also

TickCounter

4.11.2.2 void Tick_DelayMs (uint32_t *delayMs*)

this is a blocking delay.

```
#include "common.h"
#include "MCU/led.h"
#include "MCU/tick.h"

void main(void) {

    Led_Init();
    Tick_init();

    for( ;;) {
      Tick_DelayMs(1000); // delay 1s;
      Led_Toggle();
    }
}
```

Parameters

delayMs	how long to delay for.

4.11.2.3 int_fast8_t Tick_DelayMs_NonBlocking (uint_fast8_t *reset,* TickType * *config*)

Non-blocking delay in ms.

```
#include "common.h"
#include "MCU/led.h"
#include "MCU/tick.h"

void main(void) {
    TickType Delay;
    Delay.DelayMs = 1000; //set to 1s

    Led_Init();
    Tick_init();

    // reset the counter
    Tick_DelayMs_NonBlocking(TRUE, &Delay);

    for( ;;) {

        if(Tick_DelayMs_NonBlocking(TRUE, &Delay)) {
            // Delay has been reached

            Tick_DelayMs_NonBlocking(TRUE, &Delay);
            Led_Toggle();
            }
        else {
            // User code when the code delay hasn't passed
            }

    }
}
```

Parameters

reset	true = reset timer start value/ false = Check if time has lapsed
config	delay setting

Returns

1 = the desire delay has been reached. 0 = not reached the desire delay time. -1 = config point is null

Here is the call graph for this function:

4.11.2.4 uint32_t Tick_GetMs (void)

return the number of mili-seconds since power up.

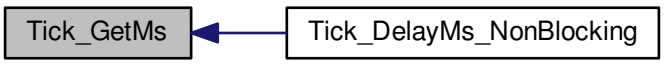
Returns

number of mili-seconds.

Note

the tick counter is expected to overflow and therefore code using the tick value should take this into account.

Here is the caller graph for this function:

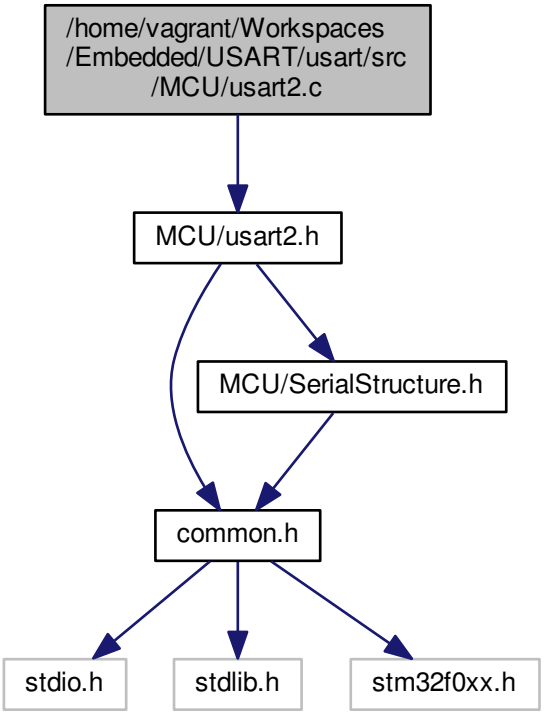


4.12 /home/vagrant/Workspaces/Embedded/USART/usart/src/MCU/usart2.c File Reference

STM32 serial2 MCU hardware interface layer. to maintain code portability, the hardware related code is split from the main logic.

```
#include "MCU/usart2.h"
```

Include dependency graph for usart2.c:



Macros

- #define GPIO_AFRL_AFR2_0 (uint32_t) 0x00000100

  Alternative function set bit 1 for AFR2.
- #define GPIO_AFRL_AFR3_0 (uint32_t) 0x00001000

  Alternative function set bit 1 for AFR3.

Variables

- SerialInterface SerialPort2

  Defines the standard serial functions for usart 2.

4.12.1 Detailed Description

STM32 serial2 MCU hardware interface layer. to maintain code portability, the hardware related code is split from the main logic.

Author

: Ronald Sousa (Opticalworm)

4.12.2 Variable Documentation

4.12.2.1 SerialInterface SerialPort2

Initial value:

```
= {
                                    IsSerialOpen,
                                    Open,
                                    Close,
                                    SendByte,
                                    SendString,
                                    SendArray,
                                    DoesReceiveBufferHaveData,
                                    GetByte
                                }
```

Defines the standard serial functions for usart 2.

See also

SerialInterface