GCC Code Coverage Report

Directory: src/
Date: 2022-04-03 02:20:16
Legend: low: >= 0% medium: >= 75.0% high: >= 90.0%

	Exec	Total	Coverage
Lines:	135	138	97.8%
Functions:	23	24	95.8%
Branches:	56	70	80.0%

List of functions

File	Lines			Functions		Branches	
i2c_utils.cpp		100.0%	46 / 46	100.0%	8 / 8	91.7%	11 / 12
restricted_base.cpp		0.0%	0 / 2	0.0%	0 / 1	-%	0 / 0
spi_utils.cpp		100.0%	29 / 29	100.0%	5 / 5	90.0%	18 / 20
timer_manager.cpp		100.0%	38 / 38	100.0%	6 / 6	77.3%	17 / 22
usart_utils.cpp		95.7%	22 / 23	100.0%	4 / 4	62.5%	10 / 16

Generated by: GCOVR (Version 5.1)

GCC Code Coverage Report

Directory: src/
Date: 2022-04-03 02:20:16
Legend: low: >= 0% medium: >= 75.0% high: >= 90.0%

	Exec	Total	Coverage
Lines:	135	138	97.8%
Functions:	23	24	95.8%
Branches:	56	70	80.0%

Function	File	Line	Call count
invalid_allocation_error_handler()	src/restricted_base.cpp	25	not called
stm32::TimerManager::delay_microsecond(unsigned int)	src/timer_manager.cpp	103	called 73 times
stm32::TimerManager::error_handler()	src/timer_manager.cpp	122	called 1 time
stm32::TimerManager::get_count()	src/timer_manager.cpp	117	called 1 time
stm32::TimerManager::initialise(TIM_TypeDef*)	src/timer_manager.cpp	53	called 22 times
stm32::TimerManager::reset()	src/timer_manager.cpp	77	called 94 times
stm32::delay_millisecond(unsigned int)	src/timer_manager.cpp	29	called 1 time
stm32::i2c::generate_start_condition(I2C_TypeDef*)	src/i2c_utils.cpp	122	called 6 times
stm32::i2c::generate_stop_condition(I2C_TypeDef*)	src/i2c_utils.cpp	117	called 1 time
stm32::i2c::initialise_slave_device(I2C_TypeDef*, unsigned char, stm32::i2c::StartType)	src/i2c_utils.cpp	30	called 6 times
stm32::i2c::receive_byte(I2C_TypeDef*, unsigned char&)	src/i2c_utils.cpp	90	called 1 time
stm32::i2c::send_ack(I2C_TypeDef*)	src/i2c_utils.cpp	133	called 1 time
stm32::i2c::send_byte(I2C_TypeDef*, unsigned char)	src/i2c_utils.cpp	99	called 2 times
stm32::i2c::send_nack(I2C_TypeDef*)	src/i2c_utils.cpp	138	called 1 time
stm32::i2c::set_numbytes(I2C_TypeDef*, unsigned int)	src/i2c_utils.cpp	127	called 1 time
stm32::spi::enable_spi(SPI_TypeDef*, bool)	src/spi_utils.cpp	28	called 3 times
stm32::spi::send_byte(SPI_TypeDef*, unsigned char)	src/spi_utils.cpp	38	called 2 times
stm32::spi::set_prescaler(SPI_TypeDef*, unsigned int)	src/spi_utils.cpp	84	called 7 times
stm32::spi::wait_for_bsy_flag(SPI_TypeDef*, unsigned int)	src/spi_utils.cpp	69	called 3 times
stm32::spi::wait_for_txe_flag(SPI_TypeDef*, unsigned int)	src/spi_utils.cpp	51	called 58 times
stm32::usart::enable_usart(USART_TypeDef*)	src/usart_utils.cpp	29	called 5 times
stm32::usart::transmit_byte(USART_TypeDef*, unsigned char)	src/usart_utils.cpp	36	called 1 time
stm32::usart::wait_for_bsy_flag(USART_TypeDef*, unsigned int)	src/usart_utils.cpp	62	called 2 times

stm32::usart::wait_for_tc_flag(USART_TypeDef*, unsigned int)	src/usart_utils.cpp	43	called 2 times

GCC Code Coverage Report

Directory: src/
Date: 2022-04-03 02:20:16
Legend: low: >= 0% medium: >= 75.0% high: >= 90.0%

	Exec	Total	Coverage
Lines:	135	138	97.8%
Functions:	23	24	95.8%
Branches:	56	70	80.0%

List of functions

File	Lines			Functions		Branches	
i2c_utils.cpp		100.0%	46 / 46	100.0%	8 / 8	91.7%	11 / 12
restricted_base.cpp		0.0%	0 / 2	0.0%	0 / 1	-%	0 / 0
spi_utils.cpp		100.0%	29 / 29	100.0%	5 / 5	90.0%	18 / 20
timer_manager.cpp		100.0%	38 / 38	100.0%	6 / 6	77.3%	17 / 22
usart_utils.cpp		95.7%	22 / 23	100.0%	4 / 4	62.5%	10 / 16

Generated by: GCOVR (Version 5.1)

GCC Code Coverage Report

Directory: src/
File: src/i2c_utils.cpp
Date: 2022-04-03 02:20:16

	Exec	Total	Coverage
Lines:	46	46	100.0%
Functions:	8	8	100.0%
Branches:	11	12	91.7%

▶ List of functions

Line	Branch	Exec	Source
1			`// MIT License`
2			
3			`// Copyright (c) 2022 Chris Sutton`
4			
5			`// Permission is hereby granted, free of charge, to any person obtaining a copy`
6			`// of this software and associated documentation files (the "Software"), to deal`
7			`// in the Software without restriction, including without limitation the rights`
8			`// to use, copy, modify, merge, publish, distribute, sublicense, and/or sell`
9			`// copies of the Software, and to permit persons to whom the Software is`
10			`// furnished to do so, subject to the following conditions:`
11			
12			`// The above copyright notice and this permission notice shall be included in all`
13			`// copies or substantial portions of the Software.`
14			
15			`// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR`
16			`// IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,`
17			`// FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE`
18			`// AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER`
19			`// LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,`
20			`// OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE`
21			`// SOFTWARE.`
22			
23			`#include <i2c_utils.hpp>`
24			`#include <timer_manager.hpp>`
25			
26			`namespace stm32::i2c`
27			`{`
28			
29			
30		6	`Status initialise_slave_device(I2C_TypeDef* i2c_handle, uint8_t addr, StartType start_type)`
31			`{`
32			`  // Set the master to operate in 7-bit addressing mode. Clear ADD10 bit[11]`
33		6	`  i2c_handle->CR2 = i2c_handle->CR2 & ~(I2C_CR2_ADD10);`
34			
35			`  // Set the address for the slave device. Set SADD bits[7:1].`
36			`  // The bits SADD[9],SADD[8] and SADD[0] are don't care.`
37		6	`  i2c_handle->CR2 = i2c_handle->CR2 & ~(I2C_CR2_SADD);`
38		6	`  i2c_handle->CR2 = i2c_handle->CR2 | (addr << 0);`
39			
40	▶ 2/2	6	`  if (start_type == StartType::PROBE) // generate START with AUTO-END enabled`
41			`  {`
42			`  // Master requests a write transfer`
43		2	`  i2c_handle->CR2 = i2c_handle->CR2 & ~(I2C_CR2_RD_WRN);`
44			
45			`  // Enable AUTOEND Mode. A STOP condition is automatically sent when NBYTES data are transferred.`
46		2	`  i2c_handle->CR2 = i2c_handle->CR2 | I2C_CR2_AUTOEND;`
47			
48			`  }`
49	▶ 2/2	4	`  else if (start_type == StartType::WRITE) // generate START with AUTO-END disabled`
50			`  {`
51			`  // Master requests a write transfer`
52		2	`  i2c_handle->CR2 = i2c_handle->CR2 & ~(I2C_CR2_RD_WRN);`
53			
54			`  // Disable RELOAD Mode. The transfer is completed after the NBYTES data transfer (STOP or RESTART follows).`
55		2	`  i2c_handle->CR2 = i2c_handle->CR2 & ~(I2C_CR2_RELOAD);`
56			
57			`  // Disable AUTOEND Mode. TC flag is set when NBYTES data are transferred, stretching SCL low.`
58		2	`  i2c_handle->CR2 = i2c_handle->CR2 & ~(I2C_CR2_AUTOEND);`
59			`  }`
60	▶ 1/2	2	`  else if (start_type == StartType::READ) // generate REPEATED START`
61			`  {`
62			
63			`  // Master requests a read transfer`
64		2	`  i2c_handle->CR2 = i2c_handle->CR2 | (I2C_CR2_RD_WRN);`
65			
66			`  // Disable RELOAD Mode. The transfer is completed after the NBYTES data transfer (STOP or RESTART follows).`
67		2	`  i2c_handle->CR2 = i2c_handle->CR2 & ~(I2C_CR2_RELOAD);`
68			
69			`  // Disable AUTOEND Mode. TC flag is set when NBYTES data are transferred, stretching SCL low.`

```
70                  2    i2c_handle->CR2 = i2c_handle->CR2 & ~(I2C_CR2_AUTOEND);
71                       }
72
73                       // Generate the restart/start condition
74                  6    generate_start_condition(i2c_handle);
75
76                       // give slave a chance to respond
77                  6    stm32::TimerManager::delay_microsecond(1000);
78
79                       // check if addr was not recognised by slave device
80      ▶ 2/2       6    if ( (i2c_handle->ISR & I2C_ISR_NACKF) == I2C_ISR_NACKF )
81                       {
82                  3     return Status::NACK;
83                       }
84                       // otherwise slave device is happy
85                  3    return Status::ACK;
86                       }
87
88
89
90                  1    Status receive_byte(I2C_TypeDef* i2c_handle, uint8_t &rx_byte)
91                       {
92                  1    rx_byte = i2c_handle->RXDR & I2C_RXDR_RXDATA;
93
94                  1    return Status::ACK;
95
96                       }
97
98
99                  2    Status send_byte(I2C_TypeDef* i2c_handle, uint8_t tx_byte)
100                      {
101                 2    i2c_handle->TXDR = tx_byte;
102
103                      // wait for I2C_ISR_TXE (Transmit data register empty) before continuing
104     ▶ 2/2       4    while (((i2c_handle->ISR & I2C_ISR_TXE) != I2C_ISR_TXE))
105                      {
106                       // do nothing
107                 2     stm32::TimerManager::delay_microsecond(10);
108                      }
109                      // check if slave device responded with NACK
110     ▶ 2/2       2    if (((i2c_handle->ISR & I2C_ISR_NACKF) == I2C_ISR_NACKF))
111                      {
112                 1     return Status::NACK;
113                      }
114                 1    return Status::ACK;
115                      }
116
117                 1    void generate_stop_condition(I2C_TypeDef* i2c_handle)
118                      {
119                 1    i2c_handle->CR2 = i2c_handle->CR2 | (I2C_CR2_STOP);
120                 1    }
121
122                 6    void generate_start_condition(I2C_TypeDef* i2c_handle)
123                      {
124                 6    i2c_handle->CR2 = i2c_handle->CR2 | (I2C_CR2_START);
125                 6    }
126
127                 1    void set_numbytes(I2C_TypeDef* i2c_handle, uint32_t nbytes)
128                      {
129                 1    i2c_handle->CR2 = i2c_handle->CR2 & ~(I2C_CR2_NBYTES);
130                 1    i2c_handle->CR2 = i2c_handle->CR2 | (nbytes << I2C_CR2_NBYTES_Pos);
131                 1    }
132
133                 1    void send_ack(I2C_TypeDef* i2c_handle)
134                      {
135                 1    i2c_handle->CR2 = i2c_handle->CR2 & ~(I2C_CR2_NACK);
136                 1    }
137
138                 1    void send_nack(I2C_TypeDef* i2c_handle)
139                      {
140                 1    i2c_handle->CR2 = i2c_handle->CR2 | (I2C_CR2_NACK);
141                 1    }
142
143                      } // namespace stm32::i2c
144
```

GCC Code Coverage Report

Directory: src/
File: src/restricted_base.cpp
Date: 2022-04-03 02:20:16

	Exec	Total	Coverage
Lines:	0	2	0.0%
Functions:	0	1	0.0%
Branches:	0	0	-%

▶ List of functions

Line	Branch	Exec	Source
1			`// MIT License`
2			
3			`// Copyright (c) 2022 Chris Sutton`
4			
5			`// Permission is hereby granted, free of charge, to any person obtaining a copy`
6			`// of this software and associated documentation files (the "Software"), to deal`
7			`// in the Software without restriction, including without limitation the rights`
8			`// to use, copy, modify, merge, publish, distribute, sublicense, and/or sell`
9			`// copies of the Software, and to permit persons to whom the Software is`
10			`// furnished to do so, subject to the following conditions:`
11			
12			`// The above copyright notice and this permission notice shall be included in all`
13			`// copies or substantial portions of the Software.`
14			
15			`// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR`
16			`// IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,`
17			`// FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE`
18			`// AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER`
19			`// LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,`
20			`// OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE`
21			`// SOFTWARE.`
22			
23			`#include <restricted_base.hpp>`
24			
25		✗	`void invalid_allocation_error_handler()`
26			`{`
27			
28		✗	`    while(true)`
29			`    {`
30			`        //`
31			`    }`
32			
33			`}`
34			
35			
36			
37			
38			`// void* RestrictedBase::operator new(size_t size [[maybe_unused]]) noexcept`
39			`// {`
40			`//     while(true)`
41			`//     {`
42			`//         // forbidden`
43			`//     }`
44			`//     // just to prevent compiler errors`
45			`//     void *p;`
46			`//     return p;`
47			
48			`// }`
49			
50			`// void RestrictedBase::operator delete(void* ptr) noexcept`
51			`// {`
52			`//     while(true)`
53			`//     {`
54			`//         // forbidden`
55			`//     }`
56			
57			`// }`

GCC Code Coverage Report

Directory: src/
File: src/spi_utils.cpp
Date: 2022-04-03 02:20:16

	Exec	Total	Coverage
Lines:	29	29	100.0%
Functions:	5	5	100.0%
Branches:	18	20	90.0%

▶ List of functions

Line	Branch	Exec	Source
1			`// MIT License`
2			
3			`// Copyright (c) 2022 Chris Sutton`
4			
5			`// Permission is hereby granted, free of charge, to any person obtaining a copy`
6			`// of this software and associated documentation files (the "Software"), to deal`
7			`// in the Software without restriction, including without limitation the rights`
8			`// to use, copy, modify, merge, publish, distribute, sublicense, and/or sell`
9			`// copies of the Software, and to permit persons to whom the Software is`
10			`// furnished to do so, subject to the following conditions:`
11			
12			`// The above copyright notice and this permission notice shall be included in all`
13			`// copies or substantial portions of the Software.`
14			
15			`// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR`
16			`// IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,`
17			`// FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE`
18			`// AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER`
19			`// LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,`
20			`// OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE`
21			`// SOFTWARE.`
22			
23			`#include <spi_utils.hpp>`
24			`#include <timer_manager.hpp>`
25			`namespace stm32::spi`
26			`{`
27			
28		3	`bool enable_spi(SPI_TypeDef *spi_handle, bool enable)`
29			`{`
30	▶ 2/2	3	`  if (spi_handle == nullptr) { return false; }`
31			
32	▶ 2/2	2	`  if (enable) { spi_handle->CR1 = spi_handle->CR1 | SPI_CR1_SPE; }`
33		1	`  else { spi_handle->CR1 = spi_handle->CR1 & ~SPI_CR1_SPE; }`
34			
35		2	`  return true;`
36			`}`
37			
38		2	`bool send_byte(SPI_TypeDef *spi_handle, uint8_t byte)`
39			`{`
40	▶ 2/2	2	`  if (spi_handle == nullptr) { return false; }`
41			
42		1	`  volatile uint8_t *spidr = ((volatile uint8_t *)&spi_handle->DR);`
43		1	`  *spidr = byte;`
44			`  // check the data has left the SPI FIFO`
45	▶ 1/2	1	`  while (!stm32::spi::wait_for_bsy_flag(spi_handle, 10));`
46	▶ 2/2	56	`  while (!stm32::spi::wait_for_txe_flag(spi_handle, 10));`
47			
48		1	`  return true;`
49			`}`
50			
51		58	`bool wait_for_txe_flag(SPI_TypeDef *spi_handle, uint32_t delay_us)`
52			`{`
53			
54			
55			`  // The TXE flag is set when transmission TXFIFO has enough space to store data to send.`
56	▶ 2/2	58	`  if ((spi_handle->SR & SPI_SR_TXE) != (SPI_SR_TXE))`
57			`  {`

```
58                          // give TX FIFO a chance to clear before checking again
59                 57       stm32::TimerManager::delay_microsecond(delay_us);
60     ▶ 2/2       57       if ((spi_handle->SR & SPI_SR_TXE) != (SPI_SR_TXE))
61                          {
62                 56           return false;
63                          }
64
65                      }
66                  2   return true;
67                  }
68
69                  3   bool wait_for_bsy_flag(SPI_TypeDef *spi_handle, uint32_t delay_us)
70                      {
71                          // When BSY is set, it indicates that a data transfer is in progress on the SPI
72     ▶ 2/2        3   if ((spi_handle->SR & SPI_SR_BSY) == SPI_SR_BSY)
73                          {
74                          // give SPI bus a chance to finish sending data before checking again
75                  1       stm32::TimerManager::delay_microsecond(delay_us);
76     ▶ 1/2        1       if ((spi_handle->SR & SPI_SR_BSY) == (SPI_SR_BSY))
77                          {
78                  1           return false;
79                          }
80                      }
81                  2   return true;
82                  }
83
84                  7   bool set_prescaler(SPI_TypeDef *spi_handle, uint32_t new_value)
85                      {
86     ▶ 2/2        7   if (spi_handle == nullptr) { return false; }
87
88                  6   spi_handle->CR1 = spi_handle->CR1 & ~(SPI_CR1_BR_2 | SPI_CR1_BR_1 | SPI_CR1_BR_0);
89                  6   spi_handle->CR1 = spi_handle->CR1 | (new_value);
90                  6   return true;
91                      }
92
93
94                      } // namespace stm32::spi
95
```

Generated by: GCOVR (Version 5.1)

GCC Code Coverage Report

Directory: src/
File: src/timer_manager.cpp
Date: 2022-04-03 02:20:16

	Exec	Total	Coverage
Lines:	38	38	100.0%
Functions:	6	6	100.0%
Branches:	17	22	77.3%

▶ List of functions

Line	Branch	Exec	Source
1			`// MIT License`
2			
3			`// Copyright (c) 2022 Chris Sutton`
4			
5			`// Permission is hereby granted, free of charge, to any person obtaining a copy`
6			`// of this software and associated documentation files (the "Software"), to deal`
7			`// in the Software without restriction, including without limitation the rights`
8			`// to use, copy, modify, merge, publish, distribute, sublicense, and/or sell`
9			`// copies of the Software, and to permit persons to whom the Software is`
10			`// furnished to do so, subject to the following conditions:`
11			
12			`// The above copyright notice and this permission notice shall be included in all`
13			`// copies or substantial portions of the Software.`
14			
15			`// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR`
16			`// IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,`
17			`// FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE`
18			`// AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER`
19			`// LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,`
20			`// OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE`
21			`// SOFTWARE.`
22			
23			`#include <timer_manager.hpp>`
24			
25			`namespace stm32`
26			`{`
27			
28			
29		1	`void delay_millisecond(uint32_t Delay)`
30			`{`
31		1	`  [[maybe_unused]] __IO uint32_t  tmp = SysTick->CTRL;  /* Clear the COUNTFLAG first */`
32			`  uint32_t tmpDelay; /* MISRAC2012-Rule-17.8 */`
33		1	`  tmpDelay  = Delay;`
34			`  /* Add a period to guaranty minimum wait */`
35	▶ 1/2	1	`  if (tmpDelay  < LL_MAX_DELAY)`
36			`  {`
37		1	`  tmpDelay ++;`
38			`  }`
39			
40	▶ 2/2	2570717	`  while (tmpDelay  != 0U)`
41			`  {`
42	▶ 2/2	2570716	`  if ((SysTick->CTRL & SysTick_CTRL_COUNTFLAG_Msk) != 0U)`
43			`  {`
44		11	`  tmpDelay --;`
45			`  }`
46			`  // simulate the "Clear on read by application or debugger."`
47			`  #ifdef X86_UNIT_TESTING_ONLY`
48		2570716	`  SysTick->CTRL = SysTick->CTRL & ~SysTick_CTRL_COUNTFLAG_Msk;`
49			`  #endif`
50			`  }`
51		1	`}`
52			
53		22	`bool TimerManager::initialise(TIM_TypeDef *timer)`
54			`{`
55			
56	▶ 2/2	22	`  if (timer == nullptr)`
57			`  {`
58	▶ 1/2	1	`  if (!error_handler())`
59			`  {`

```
                1            return false;
                         }

                     }

                     // stop the timer before re-assigning the pointer
▶ 2/2          21        if (m_timer != nullptr)
                         {
               20            m_timer->CR1 = m_timer->CR1 & ~(TIM_CR1_CEN);
                         }

               21        m_timer = timer;

               21        reset();
               21        return true;
                     }

               94    void TimerManager::reset()
                     {
                         // wait in limbo if not initialised
▶ 1/2          94        if (m_timer == nullptr) { error_handler(); }


                         // ensure the timer is disabled before setup
▶ 2/2          94        if ( (m_timer->CR1 & TIM_CR1_CEN) == TIM_CR1_CEN )
                         {
               73            m_timer->CR1 = m_timer->CR1 & ~(TIM_CR1_CEN);
                         }
                         // setup the timer to 1 us resolution (depending on the system clock frequency)
               94        m_timer->PSC = SystemCoreClock / 1000000UL;

                         // allow largest possible timeout
               94        m_timer->ARR = 0xFFFF-1;

                         // reset CNT
               94        m_timer->CNT = 0;

                         // start the timer and wait for the timeout
               94        m_timer->CR1 = m_timer->CR1 | (TIM_CR1_CEN);


               94    }

               73    bool TimerManager::delay_microsecond(uint32_t delay_us)
                     {
                         // wait in limbo if not initialised
▶ 1/2          73        if (m_timer == nullptr) { error_handler(); }

                         // @TODO change the prescaler to allow longer delays, clamp for now
▶ 1/2          73        if (delay_us > 0xFFFE) { delay_us = 0xFFFE; }

                         // setup the timer for timeout function
               73        reset();
▶ 2/2     959713970      while (m_timer->CNT < delay_us);
               73        return true;
                     }

                1    uint32_t TimerManager::get_count()
                     {
                1        return m_timer->CNT;
                     }

                1    bool TimerManager::error_handler()
                     {
                         #ifdef X86_UNIT_TESTING_ONLY
                1            return false;
                         #else
                             while(1)
                             {
                                 // stay here to allow stack trace to be shown in debugger...

                             }
                         #endif
                     }
```

```
134                 } // namespace stm32:
135
```

GCC Code Coverage Report

Directory: src/
File: src/usart_utils.cpp
Date: 2022-04-03 02:20:16

	Exec	Total	Coverage
Lines:	22	23	95.7%
Functions:	4	4	100.0%
Branches:	10	16	62.5%

▶ List of functions

Line	Branch	Exec	Source
1			`// MIT License`
2			
3			`// Copyright (c) 2022 Chris Sutton`
4			
5			`// Permission is hereby granted, free of charge, to any person obtaining a copy`
6			`// of this software and associated documentation files (the "Software"), to deal`
7			`// in the Software without restriction, including without limitation the rights`
8			`// to use, copy, modify, merge, publish, distribute, sublicense, and/or sell`
9			`// copies of the Software, and to permit persons to whom the Software is`
10			`// furnished to do so, subject to the following conditions:`
11			
12			`// The above copyright notice and this permission notice shall be included in all`
13			`// copies or substantial portions of the Software.`
14			
15			`// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR`
16			`// IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,`
17			`// FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE`
18			`// AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER`
19			`// LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,`
20			`// OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE`
21			`// SOFTWARE.`
22			
23			
24			`#include <usart_utils.hpp>`
25			`#include <timer_manager.hpp>`
26			`namespace stm32::usart`
27			`{`
28			
29		5	`bool enable_usart(USART_TypeDef *usart_handle)`
30			`{`
31	▶ 1/2	5	`    if (usart_handle == nullptr) { return false; }`
32		5	`    usart_handle->CR1 = usart_handle->CR1 | USART_CR1_UE;`
33		5	`    return true;`
34			`}`
35			
36		1	`bool transmit_byte(USART_TypeDef *usart_handle, uint8_t byte)`
37			`{`
38	▶ 1/2	1	`    if (usart_handle == nullptr) { return false; }`
39		1	`    usart_handle->TDR = byte;`
40		1	`    return true;`
41			`}`
42			
43		2	`bool wait_for_tc_flag(USART_TypeDef *usart_handle, uint32_t delay_us)`
44			`{`
45			
46	▶ 1/2	2	`    if (usart_handle == nullptr) { return false; }`
47			`    // Check the previous tranmission has completed`
48	▶ 2/2	2	`    if ((usart_handle->ISR & USART_ISR_TC) != (USART_ISR_TC))`
49			`    {`
50			`        // if not then wait before checking again`
51		1	`        stm32::TimerManager::delay_microsecond(delay_us);`
52	▶ 1/2	1	`        if ((usart_handle->ISR & USART_ISR_TC) != (USART_ISR_TC))`
53			`        {`
54		1	`            return false;`
55			`        }`
56			

```
57                  }
58
59          1       return true;
60                  }
61
62          2       bool wait_for_bsy_flag(USART_TypeDef *usart_handle, uint32_t delay_us)
63                  {
64  ▶ 1/2   2           if (usart_handle == nullptr)
65                      {
66          ✗               return false;
67                      }
68                      // When BSY is set, it indicates that a data transfer is in progress on the USART
69  ▶ 2/2   2           if ((usart_handle->ISR & USART_ISR_BUSY) == (USART_ISR_BUSY))
70                      {
71                          // give USART bus a chance to finish sending data before checking again
72          1               stm32::TimerManager::delay_microsecond(delay_us);
73  ▶ 1/2   1               if ((usart_handle->ISR & USART_ISR_BUSY) == (USART_ISR_BUSY))
74                          {
75          1                   return false;
76                          }
77                      }
78          1           return true;
79                  }
80
81                  } // namespace stm32::spi
82
```