GCC Code Coverage Report

			Exec	Total	Coverage
Directory:	./				
Date:	2022-03-26 22:51:46	**Lines:**	158	281	56.2 %
Legend:	low: < 75.0 % medium: >= 75.0 % high: >= 90.0 %	**Branches:**	258	734	35.1 %

File	Lines			Branches	
include/bitset_utils.hpp		100.0 %	21 / 21	100.0 %	6 / 6
include/byte_utils.hpp		100.0 %	9 / 9	- %	0 / 0
include/static_map.hpp		100.0 %	5 / 5	100.0 %	4 / 4
src/i2c_utils.cpp		0.0 %	0 / 41	0.0 %	0 / 12
src/restricted_base.cpp		0.0 %	0 / 2	- %	0 / 0
src/spi_utils.cpp		0.0 %	0 / 28	0.0 %	0 / 18
src/timer_manager.cpp		0.0 %	0 / 32	0.0 %	0 / 20
src/usart_utils.cpp		0.0 %	0 / 20	0.0 %	0 / 12
tests/catch_bitset_utils.cpp		100.0 %	86 / 86	37.5 %	156 / 416
tests/catch_byte_utils.cpp		100.0 %	8 / 8	37.5 %	12 / 32
tests/catch_static_map.cpp		100.0 %	29 / 29	37.4 %	80 / 214

GCC Code Coverage Report

			Exec	Total	Coverage
Directory:	./				
Date:	2022-03-26 22:51:46	**Lines:**	158	281	56.2 %
Legend:	low: < 75.0 % medium: >= 75.0 % high: >= 90.0 %	**Branches:**	258	734	35.1 %

File	Lines			Branches	
include/bitset_utils.hpp		100.0 %	21 / 21	100.0 %	6 / 6
include/byte_utils.hpp		100.0 %	9 / 9	- %	0 / 0
include/static_map.hpp		100.0 %	5 / 5	100.0 %	4 / 4
src/i2c_utils.cpp		0.0 %	0 / 41	0.0 %	0 / 12
src/restricted_base.cpp		0.0 %	0 / 2	- %	0 / 0
src/spi_utils.cpp		0.0 %	0 / 28	0.0 %	0 / 18
src/timer_manager.cpp		0.0 %	0 / 32	0.0 %	0 / 20
src/usart_utils.cpp		0.0 %	0 / 20	0.0 %	0 / 12
tests/catch_bitset_utils.cpp		100.0 %	86 / 86	37.5 %	156 / 416
tests/catch_byte_utils.cpp		100.0 %	8 / 8	37.5 %	12 / 32
tests/catch_static_map.cpp		100.0 %	29 / 29	37.4 %	80 / 214

GCC Code Coverage Report

			Exec	Total	Coverage
Directory:	./				
File:	include/bitset_utils.hpp	Lines:	21	21	100.0 %
Date:	2022-03-26 22:51:46	Branches:	6	6	100.0 %

```
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                #ifndef __BITSET_UTILS_HPP__
  24                #define __BITSET_UTILS_HPP__
  25
  26                // used for x86-based testing
  27                #ifdef X86_UNIT_TESTING_ONLY
  28                    #include <iostream>
  29                #endif
  30
  31                // used for arm target debug mode only.
  32                #ifdef USE_RTT
  33                  #include <SEGGER_RTT.h>
  34                #endif
  35
  36                #include <stdint.h>
  37                #include <bitset>
  38                #include <array>
  39
  40                namespace noarch::bit_manip
  41                {
  42
  43
  44                // @brief Adds source std::bitset to target std::bitset with msb_offset
  45                // @tparam TARGET_SIZE The size of the source bitset container
  46                // @tparam SOURCE_SIZE The size of the target bitset container
  47                // @param target The target bitset container to copy into
  48                // @param source The source bitset container to copy from
  49                // @param msb_offset insertion index starting from the right-most position
  50                template<std::size_t TARGET_SIZE, std::size_t SOURCE_SIZE>
  51          16    bool insert_bitset_at_offset(std::bitset<TARGET_SIZE> &target,  const std::bitset<SOURCE_SIZE> &source, const uint16_t &msb_offset)
  52                {
  53                    // protect against oversized msb_offset or SOURCE params
  54        ✓✓ 16     if (msb_offset + SOURCE_SIZE > TARGET_SIZE)
  55                    {
  56           8            return false;
  57                    }
  58                    // iterate over the source bitset pattern
  59        ✓✓ 40     for (uint16_t idx = 0; idx < source.size(); idx++)
  60                    {
  61                        // start from the common register msb and work backwards towards lsb,
  62                        // ...minus the offset
  63        ✓✓ 32         if (source.test(idx))
  64                        {
  65           16               target.set(msb_offset + (source.size() - 1)  - idx, true);
  66                        }
  67                        else
  68                        {
  69           16               target.set(msb_offset + (source.size() - 1) - idx , false);
  70                        }
  71                    }
  72           8        return true;
  73                }
  74
  75                // @brief Converts bits to same sized byte array LSB first. 0101 becomes 1010.
  76                // Oversized bitsets are truncated, undersized bitsets are zero-padded
  77                // @tparam TARGET_SIZE The size of the target_array std::array
  78                // @tparam SOURCE_SIZE The size of the source_bitset std::bitset
  79                // @param target_array The std::array object copied to. Caution, all pre-existing contents is destroyed.
  80                // @param source_bitset The std::bitset object copied from.
  81                template<std::size_t TARGET_SIZE, std::size_t SOURCE_SIZE>
  82           3    bool bitset_to_bytearray(std::array<uint8_t, TARGET_SIZE> &target_array, const std::bitset<SOURCE_SIZE> &source_bitset)
  83                {
  84
  85                    // 8-bit byte
  86           3        const uint8_t word_size_bits = 8;
  87
  88                    // clear the array before starting
  89           3        target_array.fill(0);
  90
  91                    // iterate each byte in the array and fill it
  92           8        for (uint16_t byte_array_idx = 0; byte_array_idx < target_array.size(); byte_array_idx++)
```

```
93                  {
94                      // This is the current position within the bitset, relative to the current byte
95          5           uint16_t bit_offset_within_pattern = byte_array_idx * word_size_bits;
96
97                      // used to bitshift the individual bits into the current byte
98          5           int8_t bit_offset_within_byte = word_size_bits - 1;
99
100                     // iterate the bitset position [n -> n + 8)
101         45          for (uint16_t pattern_idx = bit_offset_within_pattern; pattern_idx < bit_offset_within_pattern + word_size_bits; pattern_idx++)
102                     {
103                         // double check we haven't overshot the input bitset length
104         40              if (pattern_idx < source_bitset.size())
105                         {
106
107         32                  target_array[byte_array_idx] |= (source_bitset.test(pattern_idx) << bit_offset_within_byte);
108         32                  bit_offset_within_byte--;
109                         }
110                         else
111                         {
112         8                   target_array[byte_array_idx] |= 0;
113         8                   bit_offset_within_byte--;
114                         }
115                     }
116                 }
117         3       return true;
118             }
119
120             // @brief Print out the provided bitset as bytes
121             // @param pattern The bitset to print
122             template<std::size_t BITSET_SIZE>
123             void print_bits(std::bitset<BITSET_SIZE> &pattern [[maybe_unused]])
124             {
125
126                 #ifdef USE_RTT
127                     for (uint16_t idx = 0; idx < pattern.size(); idx++)
128                     {
129                         if (idx % 8 == 0)
130                             SEGGER_RTT_printf(0, " ");
131                         if (idx % 64 == 0)
132                             SEGGER_RTT_printf(0, "\n");
133                         SEGGER_RTT_printf(0, "%u ", +pattern.test(idx));
134                     }
135                     SEGGER_RTT_printf(0, "\n");
136                 #endif
137                 #ifdef X86_UNIT_TESTING_ONLY
138                     for (uint16_t idx = 0; idx < pattern.size(); idx++)
139                     {
140                         if (idx % 8 == 0)
141                         {
142                             std::cout << " ";
143                         }
144                         if (idx % 64 == 0)
145                         {
146                             std::cout << std::endl;
147                         }
148                         std::cout << std::noboolalpha << pattern.test(idx);
149                     }
150                     std::cout << std::endl;
151                 #endif
152             }
153
154             } // namespace noarch::bit_manip
155
156
157             #endif // __BITSET_UTILS_HPP__
```

GCC Code Coverage Report

		Exec	Total	Coverage
Directory:	./			
File:	include/byte_utils.hpp	Lines:	9	9	100.0 %
Date:	2022-03-26 22:51:46	Branches:	0	0	- %

```
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                #ifndef __BYTE_UTILS_HPP__
  24                #define __BYTE_UTILS_HPP__
  25
  26                #include <stdint.h>
  27                #ifdef X86_UNIT_TESTING_ONLY
  28                    #include <iostream>
  29                    #include <iomanip>
  30                #endif
  31
  32                namespace noarch::byte_manip
  33                {
  34
  35                template<std::size_t BYTE_ARRAY_SIZE>
  36        2       bool print_bytes(std::array<uint8_t, BYTE_ARRAY_SIZE> &bytes [[maybe_unused]])
  37                {
  38        2           if (bytes.empty())
  39                    {
  40        1               return false;
  41                    }
  42       65           for (uint16_t idx = 0; idx < bytes.size(); idx++)
  43                    {
  44
  45       64               if (idx % 16 == 0)
  46                        {
  47                            #if defined(USE_RTT)
  48                                SEGGER_RTT_printf(0, "\n");
  49                            #elif defined(X86_UNIT_TESTING_ONLY)
  50        4                       std::cout << std::endl;
  51                            #endif
  52                        }
  53                        #if defined(USE_RTT)
  54                            SEGGER_RTT_printf(0, "0x%02x ", +bytes[idx]);
  55                        #elif defined(X86_UNIT_TESTING_ONLY)
  56       64                   std::cout << " 0x" << std::setfill('0') << std::setw(2) << std::hex << +bytes[idx];
  57                        #endif
  58                    }
  59                    #if defined(USE_RTT)
  60                        SEGGER_RTT_printf(0, "\n");
  61                    #elif defined(X86_UNIT_TESTING_ONLY)
  62        1               std::cout << std::endl;
  63                    #endif
  64        1           return true;
  65                }
  66
  67                }   // namespace noarch::byte_manip
  68
```

```
69          #endif // __BYTE_UTILS_HPP__
```

GCC Code Coverage Report

			Exec	Total	Coverage
Directory:	./				
File:	include/static_map.hpp	**Lines:**	5	5	100.0 %
Date:	2022-03-26 22:51:46	**Branches:**	4	4	100.0 %

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			#ifndef __STATIC_MAP_HPP__
25			#define __STATIC_MAP_HPP__
26			
27			// #include <algorithm>
28			#include <array>
29			
30			// @brief For working example see https://godbolt.org/z/deza1Ecnn
31			
32			namespace noarch::containers
33			{
34			
35			// @brief Associative container with contains key-value pairs that is allocated at compile-time
36			// @tparam Key The Key
37			// @tparam Value The Value
38			// @tparam Size The size of the map/number of Key/Value pairs. Must be constant.
39			template <typename Key, typename Value, std::size_t Size>
40			struct StaticMap {
41			
42			// @brief The dictionary
43			std::array<std::pair<Key, Value>, Size> data;
44			
45			// @brief access specified element
46			// @param key The key element to match
47			// @return Value* Pointer to the value element, or nullptr if not found
48		4	Value* find_key(const Key &key) {
49			
50	✓✓	10	for (std::pair<Key, Value> &pair : data)
51			{
52	✓✓	9	if (pair.first == key)
53			{
54		3	return &pair.second;
55			}
56			}
57			// or return nullptr as the search completed without match
58		1	return nullptr;
59			}
60			};
61			
62			} // namespace oarch::containers
63			
64			#endif // __STATIC_MAP_HPP__

GCC Code Coverage Report

		Exec	Total	Coverage
Directory:	./			
File:	src/i2c_utils.cpp	Lines: 0	41	0.0 %
Date:	2022-03-26 22:51:46	Branches: 0	12	0.0 %

```
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                // we can't unit test this without mocking
  30              Status send_addr(I2C_TypeDef* i2c_handle [[maybe_unused]], uint8_t addr [[maybe_unused]], MsgType type [[maybe_unused]])
  31              {
  32                 // Set the master to operate in 7-bit addressing mode. Clear ADD10 bit[11]
  33                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                i2c_handle->CR2 = i2c_handle->CR2 & ~(I2C_CR2_SADD);
  38                i2c_handle->CR2 = i2c_handle->CR2 | (addr << 0);
  39
  40                if (type == MsgType::PROBE) // generate START with AUTO-END enabled
  41                {
  42                 // Master requests a write transfer
  43                 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                 i2c_handle->CR2 = i2c_handle->CR2 | I2C_CR2_AUTOEND;
  47
  48                }
  49                else if (type == MsgType::WRITE) // generate START with AUTO-END disabled
  50                {
  51                 // Master requests a write transfer
  52                 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                 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                 i2c_handle->CR2 = i2c_handle->CR2 & ~(I2C_CR2_AUTOEND);
  59                }
  60                else if (type == MsgType::READ) // generate REPEATED START
  61                {
  62
  63                 // Master requests a read transfer
  64                 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                 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                 i2c_handle->CR2 = i2c_handle->CR2 & ~(I2C_CR2_AUTOEND);
  71                }
  72
  73                // Generate the restart/start condition
  74                generate_start_condition(i2c_handle);
  75
  76                // give slave a chance to respond
  77                stm32::TimerManager::delay_microsecond(1000);
  78
  79                // check if addr was not recognised by slave device
  80                if ( (i2c_handle->ISR & I2C_ISR_NACKF) == I2C_ISR_NACKF )
  81                {
  82                 return Status::NACK;
```

```
 }
 // otherwise slave device is happy
 return Status::ACK;

}



Status receive_byte(I2C_TypeDef* i2c_handle [[maybe_unused]], uint8_t &rx_byte [[maybe_unused]])
{
 rx_byte = i2c_handle->RXDR & I2C_RXDR_RXDATA;

 return Status::ACK;

}


Status send_byte(I2C_TypeDef* i2c_handle [[maybe_unused]], uint8_t tx_byte [[maybe_unused]])
{
 i2c_handle->TXDR = tx_byte;

 // wait for TX FIFO to be transmitted before continuing
 while (((i2c_handle->ISR & I2C_ISR_TXE) == I2C_ISR_TXE) == false)
 {
  // do nothing
  stm32::TimerManager::delay_microsecond(10);
 }
 // check if slave device responded with NACK
 if (((i2c_handle->ISR & I2C_ISR_NACKF) == I2C_ISR_NACKF) == true)
 {
  return Status::NACK;
 }
 return Status::ACK;
}

void generate_stop_condition(I2C_TypeDef* i2c_handle)
{
 i2c_handle->CR2 = i2c_handle->CR2 | (I2C_CR2_STOP);
}

void generate_start_condition(I2C_TypeDef* i2c_handle)
{
 i2c_handle->CR2 = i2c_handle->CR2 | (I2C_CR2_START);
}

void set_numbytes(I2C_TypeDef* i2c_handle, uint32_t nbytes)
{
 i2c_handle->CR2 = i2c_handle->CR2 & ~(I2C_CR2_NBYTES);
 i2c_handle->CR2 = i2c_handle->CR2 | (nbytes << I2C_CR2_NBYTES_Pos);
}

void send_ack(I2C_TypeDef* i2c_handle)
{
 i2c_handle->CR2 = i2c_handle->CR2 & ~(I2C_CR2_NACK);
}

void send_nack(I2C_TypeDef* i2c_handle)
{
 i2c_handle->CR2 = i2c_handle->CR2 | (I2C_CR2_NACK);
}

} // namespace stm32::i2c
```

GCC Code Coverage Report

		Exec	Total	Coverage
Directory:	./			
File:	src/restricted_base.cpp	Lines: 0	2	0.0 %
Date:	2022-03-26 22:51:46	Branches: 0	0	- %

```
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            0   void invalid_allocation_error_handler()
  26                {
  27            0       while(true)
  28                    {
  29                        //
  30                    }
  31                }
  32
  33
  34
  35
  36                // void* RestrictedBase::operator new(size_t size [[maybe_unused]]) noexcept
  37                // {
  38                //     while(true)
  39                //     {
  40                //         // forbidden
  41                //     }
  42                //     // just to prevent compiler errors
  43                //     void *p;
  44                //     return p;
  45
  46                // }
  47
  48                // void RestrictedBase::operator delete(void* ptr) noexcept
  49                // {
  50                //     while(true)
  51                //     {
  52                //         // forbidden
  53                //     }
  54
  55                // }
```

GCC Code Coverage Report

			Exec	Total	Coverage
Directory:	./				
File:	src/spi_utils.cpp	**Lines:**	0	28	0.0 %
Date:	2022-03-26 22:51:46	**Branches:**	0	18	0.0 %

```
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 <spi_utils.hpp>
  25                #include <timer_manager.hpp>
  26                namespace stm32::spi
  27                {
  28
  29        #####   void enable_spi(SPI_TypeDef *spi_handle, bool enable)
  30                {
  31        #####       if (enable)
  32                    {
  33        #####           spi_handle->CR1 = spi_handle->CR1 | SPI_CR1_SPE;
  34                    }
  35                    else
  36                    {
  37        #####           spi_handle->CR1 = spi_handle->CR1 & ~SPI_CR1_SPE;
  38                    }
  39                }
  40
  41        #####   void send_byte(SPI_TypeDef *spi_handle, uint8_t byte)
  42                {
  43        #####       volatile uint8_t *spidr = ((volatile uint8_t *)&spi_handle->DR);
  44        #####       *spidr = byte;
  45                    // check the data has left the SPI FIFO
  46        #####       while (!stm32::spi::wait_for_txe_flag(spi_handle, 10));
  47        #####       while (!stm32::spi::wait_for_bsy_flag(spi_handle, 10));
  48                }
  49
  50        #####   bool wait_for_txe_flag(SPI_TypeDef *spi_handle, uint32_t delay_us)
  51                {
  52
  53        #####       if (spi_handle == nullptr)
  54                    {
  55        #####           return false;
  56                    }
  57                    // The TXE flag is set when transmission TXFIFO has enough space to store data to send.
  58        #####       if ((spi_handle->SR & SPI_SR_TXE) != (SPI_SR_TXE))
  59                    {
  60                        // give TX FIFO a chance to clear before checking again
  61        #####           stm32::TimerManager::delay_microsecond(delay_us);
  62        #####           if ((spi_handle->SR & SPI_SR_TXE) != (SPI_SR_TXE))
  63                        {
  64        #####               return false;
  65                        }
  66
  67                    }
```

```
68                  return true;
69              }
70
71              bool wait_for_bsy_flag(SPI_TypeDef *spi_handle, uint32_t delay_us)
72              {
73                  if (spi_handle == nullptr)
74                  {
75                      return false;
76                  }
77                  // When BSY is set, it indicates that a data transfer is in progress on the SPI
78                  if ((spi_handle->SR & SPI_SR_BSY) == (SPI_SR_BSY))
79                  {
80                      // give SPI bus a chance to finish sending data before checking again
81                      stm32::TimerManager::delay_microsecond(delay_us);
82                      if ((spi_handle->SR & SPI_SR_BSY) == (SPI_SR_BSY))
83                      {
84                          return false;
85                      }
86                  }
87                  return true;
88              }
89
90              void set_prescaler(SPI_TypeDef *spi_handle, uint32_t new_value)
91              {
92                  spi_handle->CR1 = spi_handle->CR1 & ~(SPI_CR1_BR_2 | SPI_CR1_BR_1 | SPI_CR1_BR_0);
93                  spi_handle->CR1 = spi_handle->CR1 | (new_value);
94              }
95
96
97              } // namespace stm32::spi
```

GCC Code Coverage Report

		Exec	Total	Coverage
Directory: ./				
File: src/timer_manager.cpp	Lines:	0	32	0.0 %
Date: 2022-03-26 22:51:46	Branches:	0	20	0.0 %

```
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            0   void delay_millisecond(uint32_t Delay)
  30                {
  31            0     __IO uint32_t  tmp = SysTick->CTRL;  /* Clear the COUNTFLAG first */
  32                    uint32_t tmpDelay; /* MISRAC2012-Rule-17.8 */
  33                  /* Add this code to indicate that local variable is not used */
  34            0     ((void)tmp);
  35            0     tmpDelay  = Delay;
  36                  /* Add a period to guaranty minimum wait */
  37            0     if (tmpDelay  < LL_MAX_DELAY)
  38                  {
  39            0       tmpDelay ++;
  40                  }
  41
  42            0     while (tmpDelay  != 0U)
  43                  {
  44            0       if ((SysTick->CTRL & SysTick_CTRL_COUNTFLAG_Msk) != 0U)
  45                    {
  46            0         tmpDelay --;
  47                    }
  48                  }
  49                }
  50
  51            0   void TimerManager::initialise(TIM_TypeDef *timer)
  52                {
  53                    // if (m_timer == nullptr) { m_timer = std::unique_ptr<TIM_TypeDef>(timer); }
  54            0       if (m_timer == nullptr) { m_timer = timer; }
  55            0       else { error_handler(); }
  56            0       reset();
  57                }
  58
  59            0   void TimerManager::reset()
  60                {
  61                    // wait in limbo if not initialised
  62            0       if (m_timer == nullptr) { error_handler(); }
  63
  64
  65                    // ensure the timer is disabled before setup
  66            0       if ( (m_timer->CR1 & TIM_CR1_CEN) == TIM_CR1_CEN )
  67                    {
```

```
68                      m_timer->CR1 = m_timer->CR1 & ~(TIM_CR1_CEN);
69                  }
70                  // setup the timer to 1 us resolution (depending on the system clock frequency)
71                  m_timer->PSC = SystemCoreClock / 1000000UL;
72
73                  // allow largest possible timeout
74                  m_timer->ARR = 0xFFFF-1;
75
76                  // reset CNT
77                  m_timer->CNT = 0;
78
79                  // start the timer and wait for the timeout
80                  m_timer->CR1 = m_timer->CR1 | (TIM_CR1_CEN);
81
82
83              }
84
85              void TimerManager::delay_microsecond(uint32_t delay_us)
86              {
87                  // wait in limbo if not initialised
88                  if (m_timer == nullptr) { error_handler(); }
89
90                  // @TODO change the prescaler to allow longer delays, clamp for now
91                  if (delay_us > 0xFFFE) { delay_us = 0xFFFE; }
92
93                  // setup the timer for timeout function
94                  reset();
95                  while (m_timer->CNT < delay_us);
96              }
97
98              uint32_t TimerManager::get_count()
99              {
100                 // make sure timer is running
101                 if ( (m_timer->CR1 & TIM_CR1_CEN) == TIM_CR1_CEN )
102                 {
103                     m_timer->CR1 = m_timer->CR1 | (TIM_CR1_CEN);
104                 }
105                 return m_timer->CNT;
106
107             }
108
109             void TimerManager::error_handler()
110             {
111                 while(1)
112                 {
113                     // stay here to allow stack trace to be shown in debugger...
114                 }
115             }
116
117             } // namespace stm32:
```

GCC Code Coverage Report

		Exec	Total	Coverage
Directory: ./				
File: src/usart_utils.cpp	Lines:	0	20	0.0 %
Date: 2022-03-26 22:51:46	Branches:	0	12	0.0 %

```
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        #####   void enable_usart(USART_TypeDef *usart_handle)
  30                {
  31        #####       usart_handle->CR1 = usart_handle->CR1 | USART_CR1_UE;
  32                }
  33
  34        #####   void transmit_byte(USART_TypeDef *usart_handle, uint8_t byte)
  35                {
  36        #####       usart_handle->TDR = byte;
  37                }
  38
  39        #####   bool wait_for_tc_flag(USART_TypeDef *usart_handle, uint32_t delay_us)
  40                {
  41
  42        #####       if (usart_handle == nullptr)
  43                    {
  44        #####           return false;
  45                    }
  46                    // Check the previous tranmission has completed
  47        #####       if ((usart_handle->ISR & USART_ISR_TC) != (USART_ISR_TC))
  48                    {
  49                        // if not then wait before checking again
  50        #####           stm32::TimerManager::delay_microsecond(delay_us);
  51        #####           if ((usart_handle->ISR & USART_ISR_TC) != (USART_ISR_TC))
  52                        {
  53        #####               return false;
  54                        }
  55
  56                    }
  57
  58        #####       return true;
  59                }
  60
  61        #####   bool wait_for_bsy_flag(USART_TypeDef *usart_handle, uint32_t delay_us)
  62                {
  63        #####       if (usart_handle == nullptr)
  64                    {
  65        #####           return false;
  66                    }
  67                    // When BSY is set, it indicates that a data transfer is in progress on the USART
```

```
68                      if ((usart_handle->ISR & USART_ISR_BUSY) == (USART_ISR_BUSY))
69                      {
70                          // give USART bus a chance to finish sending data before checking again
71                          stm32::TimerManager::delay_microsecond(delay_us);
72                          if ((usart_handle->ISR & USART_ISR_BUSY) == (USART_ISR_BUSY))
73                          {
74                              return false;
75                          }
76                      }
77                      return true;
78                  }
79
80                  } // namespace stm32::spi
```

GCC Code Coverage Report

			Exec	Total	Coverage
Directory:	./				
File:	tests/catch_bitset_utils.cpp	Lines:	86	86	100.0 %
Date:	2022-03-26 22:51:46	Branches:	156	416	37.5 %

```
Line Branch Exec  Source
   1                #include <catch2/catch_all.hpp>
   2                #include <bitset_utils.hpp>
   3                #include <byte_utils.hpp>
   4
   5                /// @brief insert bit pattern starting from zero msb_offset argument
   6              1 TEST_CASE("insert_bitset_at_offset - zero msb_offset", "[bitset_utils]")
   7                {
   8              1     const size_t target_size{8};
   9              1     const size_t source_size{4};
  10
  11     ✓✗       1     std::bitset<target_size> target("00000000");
  12     ✓✗       1     std::bitset<source_size> source("1111");
  13
  14     ✓✗       1     std::bitset<target_size> expected_output("00001111");
  15
     ✓✗✓✗
     ✓✗✓✗
  16     ✓✗✗✗     1     REQUIRE(noarch::bit_manip::insert_bitset_at_offset(target, source, 0));
       ✗✗
     ✓✗✓✗
     ✓✗✓✗
  17     ✓✗✗✗     1     REQUIRE(target == expected_output);
       ✗✗
  18
  19                    // reset back to 00000000 and check
     ✓✗✓✗
     ✓✗✓✗
  20     ✓✗✗✗     1     REQUIRE(noarch::bit_manip::insert_bitset_at_offset(target, source.flip(), 0));
       ✗✗
     ✓✗✓✗
     ✓✗✓✗
  21     ✓✗✗✗     1     REQUIRE(target == 0);
       ✗✗
  22
  23              1 }
  24
  25                /// @brief Insert at -1 offset (wraps around to 65535, which is handled by input checks)
  26              1 TEST_CASE("insert_bitset_at_offset - msb_offset wraparound", "[bitset_utils]")
  27                {
  28              1     const size_t target_size{8};
  29              1     const size_t source_size{4};
  30
  31     ✓✗       1     std::bitset<target_size> target("00000000");
  32     ✓✗       1     std::bitset<source_size> source("1111");
  33
  34     ✓✗       1     std::bitset<target_size> expected_output("00001111");
  35              1     std::bitset<target_size> original_target(target);
  36
  37                    // operation fails, target is not undated
     ✓✗✓✗
     ✓✗✓✗
  38     ✓✗✓✗     1     REQUIRE_FALSE(noarch::bit_manip::insert_bitset_at_offset(target, source, -1));
     ✗✗✗✗
     ✓✗✓✗
     ✓✗✓✗
  39     ✓✗✓✗     1     REQUIRE_FALSE(target == expected_output);
     ✗✗✗✗
     ✓✗✓✗
     ✓✗✓✗
  40     ✓✗✗✗     1     REQUIRE(target == original_target);
       ✗✗
  41              1 }
  42
  43                /// @brief Insert at offset larger than target bitset
  44              1 TEST_CASE("insert_bitset_at_offset - oversized msb_offset", "[bitset_utils]")
  45                {
  46              1     const size_t target_size{8};
  47              1     const size_t source_size{4};
  48
  49     ✓✗       1     std::bitset<target_size> target("00000000");
  50     ✓✗       1     std::bitset<source_size> source("1111");
  51
  52     ✓✗       1     std::bitset<target_size> expected_output("00001111");
  53              1     std::bitset<target_size> original_target(target);
  54
  55                    // operation fails, target is not undated
     ✓✗✓✗
     ✓✗✓✗
  56     ✓✗✓✗     1     REQUIRE_FALSE(noarch::bit_manip::insert_bitset_at_offset(target, source, target_size + 1));
     ✗✗✗✗
```

```
✓✗✓✗
✓✗✓✗  1      REQUIRE_FALSE(target == expected_output);
✓✗✓✗
✗✗✗✗
✓✗✓✗
✓✗✓✗  1      REQUIRE(target == original_target);
✓✗✗✗
 ✗✗
          1  }
             /// @brief Insert at offset within tolerances for SOURCE to fit within TARGET
          1  TEST_CASE("insert_bitset_at_offset - offset_index_at_limit_for_source_size", "[bitset_utils]")
             {
          1      const size_t target_size{8};
          1      const size_t source_size{4};
          1      const size_t offset_index_at_limit_for_source_size = target_size - source_size;

   ✓✗     1      std::bitset<target_size> target("00000000");
   ✓✗     1      std::bitset<source_size> source("1111");

   ✓✗     1      std::bitset<target_size> expected_output("11110000");

✓✗✓✗
✓✗✓✗  1      REQUIRE(noarch::bit_manip::insert_bitset_at_offset(target, source, offset_index_at_limit_for_source_size));
✓✗✗✗
 ✗✗
✓✗✓✗
✓✗✓✗  1      REQUIRE(target == expected_output);
✓✗✗✗
 ✗✗

             // reset back to 00000000 and check
✓✗✓✗
✓✗✓✗  1      REQUIRE(noarch::bit_manip::insert_bitset_at_offset(target, source.flip(), offset_index_at_limit_for_source_size));
✓✗✗✗
 ✗✗
✓✗✓✗
✓✗✓✗  1      REQUIRE(target == 0);
✓✗✗✗
 ✗✗

          1  }

             /// @brief Insert at offset that surpasses tolerances for SOURCE to fit within TARGET
          1  TEST_CASE("insert_bitset_at_offset - offset_index_too_large_for_source_size", "[bitset_utils]")
             {
          1      const size_t target_size{8};
          1      const size_t source_size{4};
          1      const size_t offset_index_too_large_for_source_size = target_size-(source_size - 1);

   ✓✗     1      std::bitset<target_size> target("00000000");
   ✓✗     1      std::bitset<source_size> source("1111");

   ✓✗     1      std::bitset<target_size> expected_output("00001111");
          1      std::bitset<target_size> original_target(target);

             // operation fails, target is not undated
✓✗✓✗
✓✗✓✗  1      REQUIRE_FALSE(noarch::bit_manip::insert_bitset_at_offset(target, source, offset_index_too_large_for_source_size));
✓✗✓✗
✗✗✗✗
✓✗✓✗
✓✗✓✗  1      REQUIRE_FALSE(target == expected_output);
✓✗✓✗
✗✗✗✗
✓✗✓✗
✓✗✓✗
✓✗✗✗  1      REQUIRE(target == original_target);
 ✗✗

          1  }

             /// @brief SOURCE is too large to fit within TARGET
          1  TEST_CASE("insert_bitset_at_offset - ", "[bitset_utils]")
             {
          1      const size_t target_size{8};
          1      const size_t source_size{9};

   ✓✗     1      std::bitset<target_size> target("00000000");
   ✓✗     1      std::bitset<source_size> source("1111");

   ✓✗     1      std::bitset<target_size> expected_output("00001111");
          1      std::bitset<target_size> original_target(target);

             // operation fails, target is not undated
✓✗✓✗
✓✗✓✗  1      REQUIRE_FALSE(noarch::bit_manip::insert_bitset_at_offset(target, source, 0));
✓✗✓✗
✗✗✗✗
```

```
                ✓✗✓✗
                ✓✗✓✗
115             ✓✗✓✗    1       REQUIRE_FALSE(target == expected_output);
                ✓✗✓✗
                ✗✗✗✗
                ✓✗✓✗
                ✓✗✓✗
116             ✓✗✗✗    1       REQUIRE(target == original_target);
                  ✗✗
117                             1   }
118
119
120                             /// @brief check order is reversed as expected
121                         1   TEST_CASE("bitset_to_bytearray - check MSB/LSB integrity", "[bitset_utils]")
122                             {
123                         1       std::bitset<16> input_16bits(0xAAAA);                   //  10101010 10101010
124                         1       std::array<uint8_t, 2> expected_2byte{0x55, 0x55};      //  01010101 01010101
125                             std::array<uint8_t, 2> output_2byte;
                ✓✗✓✗
                ✓✗✓✗
126             ✓✗✗✗    1       REQUIRE(noarch::bit_manip::bitset_to_bytearray(output_2byte, input_16bits));
                  ✗✗
127                             // noarch::byte_manip::print_bytes(output_2byte);
128                             // noarch::byte_manip::print_bytes(expected_2byte);
                ✓✗✓✗
                ✓✗✓✗
129             ✓✗✗✗    1       REQUIRE(output_2byte == expected_2byte);
                  ✗✗
130                             1   }
131
132                             /// @brief Check input bits are truncated if output byte array is too small
133                         1   TEST_CASE("bitset_to_bytearray - param size mismatch: bits > bytes TRUNCATION", "[bitset_utils]")
134                             {
135                         1       std::bitset<8> input_16bits(0xAAAA);                   //  10101010 10101010
136                         1       std::array<uint8_t, 1> expected_2byte{0x55};      //  01010101 01010101
137                             std::array<uint8_t, 1> output_2byte;
                ✓✗✓✗
                ✓✗✓✗
138             ✓✗✗✗    1       REQUIRE(noarch::bit_manip::bitset_to_bytearray(output_2byte, input_16bits));
                  ✗✗
139                             // noarch::byte_manip::print_bytes(output_2byte);
140                             // noarch::byte_manip::print_bytes(expected_2byte);
                ✓✗✓✗
                ✓✗✓✗
141             ✓✗✗✗    1       REQUIRE(output_2byte == expected_2byte);
                  ✗✗
142                             1   }
143
144                             /// @brief Check input bits are xero-padded if output byte array is too large
145                         1   TEST_CASE("bitset_to_bytearray - param size mismatch: bytes > bits ZERO PADDING", "[bitset_utils]")
146                             {
147                         1       std::bitset<8> input_16bits(0xAA);                     //  10101010 10101010
148                         1       std::array<uint8_t, 2> expected_2byte{0x55, 0x00};      //  01010101 00000000
149                             std::array<uint8_t, 2> output_2byte;
                ✓✗✓✗
                ✓✗✓✗
150             ✓✗✗✗    1       REQUIRE(noarch::bit_manip::bitset_to_bytearray(output_2byte, input_16bits));
                  ✗✗
151                             // noarch::byte_manip::print_bytes(output_2byte);
152                             // noarch::byte_manip::print_bytes(expected_2byte);
                ✓✗✓✗
                ✓✗✓✗
153             ✓✗✗✗    1       REQUIRE(output_2byte == expected_2byte);
                  ✗✗
154                             1   }
```

GCC Code Coverage Report

			Exec	Total	Coverage
Directory:	./				
File:	tests/catch_byte_utils.cpp	**Lines:**	8	8	100.0 %
Date:	2022-03-26 22:51:46	**Branches:**	12	32	37.5 %

Line	Branch	Exec	Source
1			#include <catch2/catch_all.hpp>
2			
3			#include <byte_utils.hpp>
4			#include <algorithm>
5			
6		1	TEST_CASE("Empty byte array", "[byte_utils]")
7			{
8			std::array<uint8_t, 0> bytes;
9	✓✗✓✗ ✓✗✓✗ ✓✗✓✗ ✗✗✗✗	1	REQUIRE_FALSE(noarch::byte_manip::print_bytes(bytes));
10		1	}
11			
12		1	TEST_CASE("Initialised byte array", "[byte_utils]")
13			{
14		1	const size_t array_size{64};
15			std::array<uint8_t, array_size> input_bytes;
16	✓✗ ✓✗✓✗	1	std::fill(input_bytes.begin(), input_bytes.end(), 10);
17	✓✗✓✗ ✓✗✗✗ ✗✗	1	REQUIRE(noarch::byte_manip::print_bytes(input_bytes));
18		1	}

GCC Code Coverage Report

			Exec	Total	Coverage
Directory:	./				
File:	tests/catch_static_map.cpp	Lines:	29	29	100.0 %
Date:	2022-03-26 22:51:46	Branches:	80	214	37.4 %

```
Line Branch Exec  Source
   1                #include <catch2/catch_all.hpp>
   2                #include <static_map.hpp>
   3                #include <static_string.hpp>
   4
   5                using namespace noarch::containers;
   6
   7
   8                /// @brief lookup a valid "Key"; must check for nullptr
   9              1 TEST_CASE("Static_map - valid lookup", "[static_map]")
  10                {
  11
  12                    // an example enum for map "Keys"
  13                    enum class ExampleKeyType
  14                    {
  15                        ONE,
  16                        TWO,
  17                        THREE,
  18                        FOUR
  19                    };
  20
  21                    // an example type for map "Values"
  22                    struct ExampleValueType
  23                    {
  24                        // constructor/initialiser
  25              3         ExampleValueType(int int_value, std::string string_value) : m_int_value(int_value), m_string_value(string_value)
  26                        {
  27                            // no action
  28              3         }
  29                        // some integer data member
  30                        int m_int_value;
  31                        // some string data member
  32                        std::string m_string_value;
  33                    };
  34
  35                    // declare our data pair for the map at compile time
  36              1     const int value_for_key_one{100};
  37              1     const int value_for_key_two{200};
  38              1     const int value_for_key_three{300};
  39                    static std::array<std::pair<ExampleKeyType, ExampleValueType>, 3 > data_set
  40                    {{
  41 ✓✗✓✗          2         { ExampleKeyType::ONE, ExampleValueType(value_for_key_one, "100") },
  42 ✓✗✓✗          2         { ExampleKeyType::TWO, ExampleValueType(value_for_key_two, "200") },
  43 ✓✗✓✗          2         { ExampleKeyType::THREE, ExampleValueType(value_for_key_three, "300") }
  44 ✓✗✓✗
     ✗✗           7     }};
  45
  46
  47                    // initialise the StaticMap container with our static allocated data set
  48                    StaticMap<ExampleKeyType, ExampleValueType, data_set.size()> the_map =
  49    ✓✗              2         StaticMap<ExampleKeyType, ExampleValueType, data_set.size()>{{data_set}};
  50
  51                    // lookup valid key ONE
  52              1     ExampleValueType* res_one = the_map.find_key(ExampleKeyType::ONE);
     ✓✗✓✗
  53 ✓✗✓✗          1     REQUIRE(res_one != nullptr);
     ✓✗✗✗
       ✗✗
     ✓✗✓✗
  54 ✓✗✓✗          1     REQUIRE(res_one->m_int_value == value_for_key_one);
     ✓✗✗✗
       ✗✗
     ✓✗✓✗
  55 ✓✗✓✗          1     REQUIRE_FALSE(res_one->m_int_value == value_for_key_two);
     ✓✗✓✗
     ✗✗✗✗
     ✓✗✓✗
  56 ✓✗✓✗          1     REQUIRE_FALSE(res_one->m_int_value == value_for_key_three);
     ✓✗✓✗
     ✗✗✗✗
  57
  58                    // lookup valid key TWO
  59              1     ExampleValueType* res_two = the_map.find_key(ExampleKeyType::TWO);
     ✓✗✓✗
  60 ✓✗✓✗          1     REQUIRE(res_two != nullptr);
     ✓✗✗✗
       ✗✗
     ✓✗✓✗
  61 ✓✗✓✗          1     REQUIRE(res_two->m_int_value == value_for_key_two);
     ✓✗✗✗
       ✗✗
```

```
                    ✓✗✓✗
62                  ✓✗✓✗  1      REQUIRE_FALSE(res_two->m_int_value == value_for_key_one);
                    ✓✗✓✗
                    ✗✗✗✗
                    ✓✗✓✗
63                  ✓✗✓✗  1      REQUIRE_FALSE(res_two->m_int_value == value_for_key_three);
                    ✓✗✓✗
                    ✗✗✗✗
64
65                          1      // lookup valid key THREE
66                  ✓✗✓✗  1      ExampleValueType* res_three = the_map.find_key(ExampleKeyType::THREE);
                    ✓✗✓✗
67                  ✓✗✓✗  1      REQUIRE(res_three != nullptr);
                    ✓✗✗✗
                      ✗✗
                    ✓✗✓✗
68                  ✓✗✓✗  1      REQUIRE(res_three->m_int_value == value_for_key_three);
                    ✓✗✗✗
                      ✗✗
                    ✓✗✓✗
69                  ✓✗✓✗  1      REQUIRE_FALSE(res_three->m_int_value == value_for_key_one);
                    ✓✗✓✗
                    ✗✗✗✗
                    ✓✗✓✗
70                  ✓✗✓✗  1      REQUIRE_FALSE(res_three->m_int_value == value_for_key_two);
                    ✓✗✓✗
                    ✗✗✗✗
71
72                          1      // lookup invalid key FOUR
73                  ✓✗✓✗  1      ExampleValueType* res_four = the_map.find_key(ExampleKeyType::FOUR);
                    ✓✗✓✗
74                  ✓✗✓✗  1      REQUIRE(res_four == nullptr);
                    ✓✗✗✗
                      ✗✗
75
76
77                          1  }
78
```