GCC Code Coverage Report

			Exec	Total	Coverage
Directory:	./				
Date:	2021-12-18 03:55:13	**Lines:**	121	137	88.3 %
Legend:	low: < 75.0 % medium: >= 75.0 % high: >= 90.0 %	**Branches:**	81	156	51.9 %

File	Lines			Branches	
cpp_ssd1306/inc/font.hpp	🟩	100.0 %	8 / 8	100.0 %	2 / 2
cpp_ssd1306/inc/ssd1306.hpp	🟨	86.0 %	43 / 50	51.6 %	32 / 62
cpp_ssd1306/src/ssd1306.cpp	🟩	100.0 %	70 / 70	58.8 %	47 / 80
main_app/src/mainapp.cpp		0.0 %	0 / 9	0.0 %	0 / 12

Generated by: GCOVR (Version 4.2)

GCC Code Coverage Report

			Exec	Total	Coverage
Directory:	./				
File:	cpp_ssd1306/inc/font.hpp	**Lines:**	8	8	100.0 %
Date:	2021-12-18 03:55:13	**Branches:**	2	2	100.0 %

```
Line Branch  Exec  Source
   1               // MIT License
   2
   3               // Copyright (c) 2021 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 __FONT_HPP__
  24               #define __FONT_HPP__
  25
  26               #include <stdint.h>
  27               #include <array>
  28               //#include <variant>
  29               //#include <fontdata.hpp>
  30
  31               #define _Font5x5_
  32               #define _Font5x7_
  33               #define _Font7x10_
  34               #define _Font11x18_
  35               #define _Font16x26_
  36
  37               namespace ssd1306
  38               {
  39
  40               template<std::size_t FONT_SIZE>
  41               class Font
  42               {
  43
  44               public:
  45
  46                // @brief Construct a new Font object
  47                Font() = default;
  48
  49                // @brief function to get a font pixel (16bit half-word).
  50                // @param idx The position in the font data array to retrieve data
  51                // @return uint16_t The halfword of data we retrieve
  52              557  bool get_pixel(size_t idx, uint32_t &bit_line)
  53                {
  54       ✓✓     557   if (idx > data.size())
  55                 {
  56                2    return false;
  57                 }
  58                 else
  59                 {
  60              555    bit_line = static_cast<uint32_t>(data.at(idx));
  61              555    return true;
  62                 }
  63                }
  64
  65                // @brief get the width member variable
  66                // @return uint8_t the width value
  67             9084  uint8_t width() { return m_width; }
```

```
           
            // @brief get tte height member variable
            // @return uint8_t the height value
      1836  uint8_t height() { return m_height; }
           
            // @brief helper function to get the size of the private font data array.
            // @return size_t the array size
        10  size_t size() { return data.size(); }
           
            std::array<char, 95> character_map {
             ' ', '!', '"', '#', '$', '%', '&', '\'','(', ')',
             '*', '+', ',', '-', '.', '/', '0', '1', '2', '3',
             '4', '5', '6', '7', '8', '9', ':', ';', '<', '=',
             '>', '?', '@', 'A', 'B', 'C', 'D', 'E', 'F', 'G',
             'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q',
             'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', '[',
             '\\',']', '^', '_', '`', 'a', 'b', 'c', 'd', 'e',
             'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o',
             'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y',
             'z', '{', '|', '}', '~'
            };
           
           private:
           
            // @brief The width of the font in pixels
            static uint8_t m_width;
           
            // @brief The height of the font in pixels
            static uint8_t m_height;
           
            // @brief the font data
            static std::array<uint16_t, FONT_SIZE> data;
           
           };
           
           // specializations
           #ifdef _Font5x5_
            typedef Font<475> Font5x5;
           #endif
           
           #ifdef _Font5x7_
            typedef Font<680> Font5x7;
           #endif
           
           #ifdef _Font7x10_
            typedef Font<950> Font7x10;
           #endif
           
           #ifdef _Font11x18_
            typedef Font<1710> Font11x18;
           #endif
           
           #ifdef _Font16x26_
            typedef Font<2470> Font16x26;
           #endif
           
           
           
           
           
           
           
           } // namespace ssd1306
           
           #endif // __FONT_HPP__
```

GCC Code Coverage Report

			Exec	Total	Coverage
Directory:	./				
File:	cpp_ssd1306/inc/ssd1306.hpp	**Lines:**	43	50	86.0 %
Date:	2021-12-18 03:55:13	**Branches:**	32	62	51.6 %

```
Line Branch Exec  Source
   1              // MIT License
   2
   3              // Copyright (c) 2021 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              // @note See datasheet
  24              // https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf
  25
  26              #ifndef Display_HPP_
  27              #define Display_HPP_
  28
  29              #include <variant>
  30              #include <font.hpp>
  31              #include <sstream>
  32              #include <iostream>
  33              #include <array>
  34              #include <utility>
  35
  36
  37
  38              #if defined(USE_SSD1306_HAL_DRIVER) || defined(USE_SSD1306_LL_DRIVER)
  39               // Required when using GCC 10.3.1 arm-none-eabi
  40               // warning: compound assignment with 'volatile'-qualified left operand is deprecated [-Wvolatile]
  41               #pragma GCC diagnostic push
  42               #pragma GCC diagnostic ignored "-Wvolatile"
  43                #include "main.h"
  44                #include "spi.h"
  45               #pragma GCC diagnostic pop
  46              #endif
  47              // this macro is defined in HAL but we still need it when using LL or no stm32 framework
  48              #if !defined(USE_SSD1306_HAL_DRIVER)
  49                  #define UNUSED(X) (void)X
  50              #endif
  51
  52
  53              namespace ssd1306
  54              {
  55
  56              // @brief
  57              enum class Colour: uint16_t
  58              {
  59                  Black = 0x00,
  60                  White = 0x01
  61              };
  62
  63              // @brief
  64              class Display
  65              {
  66              public:
  67             6   Display() = default;
  68
  69            12   virtual ~Display() = default;
  70
  71               // @brief
  72               bool init();
  73
  74
  75               // @brief
  76               // @tparam FONT_SIZE
  77               // @param msg
  78               // @param font
  79               // @param x
  80               // @param y
  81               // @param bg
  82               // @param fg
  83               // @param padding
  84               // @param update
  85               // @return char
  86               template<std::size_t FONT_SIZE>
  87               char write(std::stringstream &msg, Font<FONT_SIZE> &font, uint8_t x, uint8_t y, Colour bg, Colour fg, bool padding, bool update);
  88
  89
  90               // @brief Get the display width. Can be used to create a std::array
  91               // @return constexpr uint16_t
  92               static constexpr uint16_t get_display_width() { return m_width; }
  93
  94               // @brief Get the display height. Can be used to create a std::array
```

```
     // @return constexpr uint16_t
     static constexpr uint16_t get_display_height() { return m_height; }

    private:

     // @brief
     // @param x
     // @param y
     // @param colour
     bool draw_pixel(uint8_t x, uint8_t y, Colour colour);

     // @brief
     // @param colour
     void fill(Colour colour);

     // @brief
     bool update_screen();

     // @brief
     void reset();

     // @brief Set the cursor object
     // @param x
     // @param y
     bool set_cursor(uint8_t x, uint8_t y);


     // @brief
     // @param cmd_byte
     bool write_command(uint8_t cmd_byte);

     // @brief
     // @param data_buffer
     // @param data_buffer_size
     bool write_data(uint16_t page_idx_within_buffer);

     // @brief
        uint16_t m_currentx {0};

     // @brief
        uint16_t m_currenty {0};

     // @brief
        uint8_t m_inverted {0};

     // @brief
        uint8_t m_initialized {0};

     // @brief The display width in bytes. Also the size of each GDDRAM page
        static const uint16_t m_width {128};

     // @brief The display height, in bytes. Also the number of pages (8) multiplied by the bits per page column (8)
        static const uint16_t m_height {64};

    #ifdef USE_SSD1306_HAL_DRIVER

     // @brief
     SPI_HandleTypeDef m_spi_port {hspi1};
     // @brief
     uint16_t m_cs_port {0};
     // @brief
     uint16_t m_cs_pin {0};
     // @brief
     GPIO_TypeDef* m_dc_port {SPI1_DC_GPIO_Port};
     // @brief
     uint16_t m_dc_pin {SPI1_DC_Pin};
     // @brief
     GPIO_TypeDef* m_reset_port {SPI1_RESET_GPIO_Port};
     // @brief
     uint16_t m_reset_pin {SPI1_RESET_Pin};

    #endif

    protected:

     // @brief byte buffer for ssd1306. Access to derived classes like ssd1306_tester is permitted.
        std::array<uint8_t, (m_width*m_height)/8> m_buffer;

     // @brief
     // @tparam FONT_SIZE
     // @param ss
     // @param font
     // @param colour
     // @param padding
     // @return char
     template<std::size_t FONT_SIZE>
     char write_string(std::stringstream &ss, Font<FONT_SIZE> &font, Colour colour, bool padding);

     // @brief
     // @tparam FONT_SIZE
     // @param ch
     // @param font
     // @param colour
     // @param padding
     // @return char
     template<std::size_t FONT_SIZE>
     char write_char(char ch, Font<FONT_SIZE> &font, Colour colour, bool padding);


     // @brief Get the buffer object. Used for testing only.
     // @notes use
     // @param buffer
```

```
             //void get_buffer(std::array<uint8_t, (m_width*m_height)/8> &buffer) { buffer = m_buffer; }

            };

            // Out-of-class definitions of member function templates

            template<std::size_t FONT_SIZE>
         13 char Display::write(std::stringstream &msg, Font<FONT_SIZE> &font, uint8_t x, uint8_t y, Colour bg, Colour fg, bool padding, bool update)
            {

         13     fill(bg);
   ✓✓    13     if (!set_cursor(x, y))
             {
          4   return 0;
             }
          9     char res = write_string(msg, font, fg, padding);
   ✓✗     9     if (update)
                {
          9         update_screen();
                }
          9     return res;
            }

            template<std::size_t FONT_SIZE>
         36 char Display::write_string(std::stringstream &ss, Font<FONT_SIZE> &font, Colour color, bool padding)
            {
                // Write until null-byte
             char ch;
 ✓✗✓✗
   ✓✓    36     while (ss.get(ch))
                {
 ✓✗✗✓    25         if (write_char(ch, font, color, padding) != ch)
                    {
                        // Char could not be written
                        return ch;
                    }
                }

                // Everything ok
         11     return ch;
            }

            template<std::size_t FONT_SIZE>
         16 char Display::write_char(char ch, Font<FONT_SIZE> &font, Colour colour, bool padding)
            {

                // Check remaining space on current line
 ✓✗✗✓    32     if (m_width <= (m_currentx + font.height()) ||
   ✗✓    16         m_width <= (m_currenty + font.height()))
                {
                    // Not enough space on current line
                    return 0;
                }

                // add extra leading horizontal space
   ✓✗    16     if (padding)
                {
   ✓✓   337      for(size_t n = 0; n < font.height(); n++)
              {
 ✓✗✗✓   321    if (!draw_pixel(m_currentx, (m_currenty + n), Colour::Black))
               {
                return false;
               }
              }
         16      m_currentx += 1;
                }

                // Use the font to write
                uint32_t font_data_word;
   ✓✓   311     for(size_t font_height_idx = 0; font_height_idx < font.height(); font_height_idx++)
             {
 ✓✗✗✓   296         if (!font.get_pixel( (ch - 32) * font.height() + font_height_idx, font_data_word )) { return false; }

            #ifdef ENABLE_SSD1306_TEST_STDOUT
              // separator for the font
                    std::cout << std::endl;
            #endif

   ✓✓  4630         for(size_t font_width_idx = 0; font_width_idx < font.width(); font_width_idx++)
              {
   ✓✓  4335             if((font_data_word << font_width_idx) & 0x8000)
                        {
  ✓✗✗  1676              switch (colour)
                {
       1183      case Colour::White:
 ✓✗✗✓  1183       if (!draw_pixel(m_currentx + font_width_idx, m_currenty + font_height_idx, Colour::White))
                  {
                   return false;
                  }
       1183       break;

        493      case Colour::Black:
 ✗✗✗✗   493       if (!draw_pixel(m_currentx + font_width_idx, m_currenty + font_height_idx, Colour::Black))
                  {
                   return false;
                  }
        493       break;
                }
                        }
                        else
                        {
```

```
297   ✓✗✗ 2659                switch (colour)

298                      {
299         1488          case Colour::White:
300  ✓✗✗✓   1488           if (!draw_pixel(m_currentx + font_width_idx, m_currenty + font_height_idx, Colour::Black))
301                         {
302                          return false;
303                         }
304         1488           break;
305
306         1171          case Colour::Black:
307  ✗✗✗✗   1171           if (!draw_pixel(m_currentx + font_width_idx, m_currenty + font_height_idx, Colour::White))
308                         {
309                          return false;
310                         }
311         1171           break;
312                      }
313                              }
314                          }
315                      }
316
317                      // The current space is now taken
318           15         m_currentx += font.width();
319
320                      // add extra leading horizontal space
321    ✓✗     15         if (padding)
322                   {
323           15          m_currentx += 1;
324                   }
325
326                      // Return written char for validation
327           15         return ch;
328                  }
329
330
331
332                  } // namespace ssd1306
333
334                  #endif /* Display_HPP_ */
```

GCC Code Coverage Report

			Exec	Total	Coverage
Directory:	./				
File:	cpp_ssd1306/src/ssd1306.cpp	**Lines:**	70	70	100.0 %
Date:	2021-12-18 03:55:13	**Branches:**	47	80	58.8 %

```
Line Branch  Exec  Source
   1                // MIT License
   2
   3                // Copyright (c) 2021 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                // @note See datasheet
  24                // https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf
  25
  26                #include "ssd1306.hpp"
  27                #include <iomanip>
  28                #include <bitset>
  29                #include <type_traits>
  30
  31                namespace ssd1306
  32                {
  33
  34               8 bool Display::init()
  35                {
  36                    #if defined(USE_SSD1306_LL_DRIVER)
  37                        LL_SPI_Enable(SPI1);
  38                    #endif
  39
  40               8     bool res = true;
  41
  42               8  reset();
  43
  44                    // Wait for the screen to boot
  45                    #if defined(USE_SSD1306_HAL_DRIVER)
  46                        HAL_Delay(100);
  47                    #elif defined(USE_SSD1306_LL_DRIVER)
  48                        LL_mDelay(100);
  49                    #endif
  50
  51    ✗✓         8     if (!write_command(0xAE)) { return false; } // display off
  52    ✗✓         8     if (!write_command(0x20)) { return false; } // Set Memory Addressing Mode
  53    ✗✓         8     if (!write_command(0x10)) { return false; } // 00,Horizontal Addressing Mode; 01,Vertical Addressing Mode; 10,Page Addressing Mode (RESET); 11,Invalid
  54    ✗✓         8     if (!write_command(0xB0)) { return false; } // Set Page Start Address for Page Addressing Mode,0-7
  55    ✗✓         8     if (!write_command(0xC8)) { return false; } // Set COM Output Scan Direction - 0xC0/0xC8 (mirror vertically)
  56    ✗✓         8     if (!write_command(0x00)) { return false; } // set low column address
  57    ✗✓         8     if (!write_command(0x10)) { return false; } // set high column address
  58    ✗✓         8     if (!write_command(0x40)) { return false; } // set start line address
  59    ✗✓         8     if (!write_command(0x81)) { return false; } // set contrast control register
  60    ✗✓         8     if (!write_command(0xFF)) { return false; } // set constrast value
  61    ✗✓         8     if (!write_command(0xA0)) { return false; } // set segment re-map 0 to 127 - 0xA0/0xA1 (mirror horizontally)
  62    ✗✓         8     if (!write_command(0xA6)) { return false; } // set normal color - 0xA6/0xA7 (colour inverse)
  63    ✗✓         8     if (!write_command(0xA8)) { return false; } // set multiplex ratio(1 to 64)
  64    ✗✓         8     if (!write_command(0x3F)) { return false; } //
  65    ✗✓         8     if (!write_command(0xA4)) { return false; } // 0xA4 output follows RAM content, 0xA5 output ignores RAM content
  66    ✗✓         8     if (!write_command(0xD3)) { return false; } // set display offset
  67    ✗✓         8     if (!write_command(0x00)) { return false; } // not offset

  68    ✗✓         8     if (!write_command(0xD5)) { return false; } // set display clock divide ratio/oscillator frequency
  69    ✗✓         8     if (!write_command(0xF0)) { return false; } // set divide ratio
  70    ✗✓         8     if (!write_command(0xD9)) { return false; } // set pre-charge period
  71    ✗✓         8     if (!write_command(0x22)) { return false; } //
  72    ✗✓         8     if (!write_command(0xDA)) { return false; } // set com pins hardware configuration
  73    ✗✓         8     if (!write_command(0x12)) { return false; }
  74    ✗✓         8     if (!write_command(0xDB)) { return false; } // set vcomh
  75    ✗✓         8     if (!write_command(0x20)) { return false; } // 0x20,0.77xVcc
  76    ✗✓         8     if (!write_command(0x8D)) { return false; } // set DC-DC enable
  77    ✗✓         8     if (!write_command(0x14)) { return false; } //
  78    ✗✓         8     if (!write_command(0xAF)) { return false; } // turn on Display panel
  79
  80                    // Clear screen
  81               8     fill(Colour::Black);
  82
  83                    // Flush buffer to screen
  84               8     update_screen();
  85
  86                    // Set default values for screen object
  87               8     m_currentx = 0;
  88               8     m_currenty = 0;
  89
  90               8     m_initialized = 1;
  91
  92               8     return res;
  93                }
  94
  95
  96              15 void Display::fill(Colour color)
  97                {
  98    ✓✓     15375     for(auto &pixel : m_buffer)
  99                    {
 100    ✓✓     15360         pixel = (color == Colour::Black) ? 0x00 : 0xFF;
 101                    }
 102              15 }
 103
 104              13 bool Display::update_screen()
```

```
            {
  ✓✓   117      for(uint8_t i = 0; i < 8; i++)
                {
                    // Set Page Start Address: 0-7
  ✗✓   104          if (!write_command(0xB0 + i)) { return false; }

                    // set low column address
  ✗✓   104          if (!write_command(0x00)) { return false; }

                    // set high column address
  ✗✓   104          if (!write_command(0x10)) { return false; }

                    // the index of the page within the buffer
       104          uint16_t page_idx {static_cast<uint16_t>(m_width * i)};

  ✗✓   104          if (!write_data(page_idx)) { return false; }
                }

        13      return true;
            }

      4656  bool Display::draw_pixel(uint8_t x, uint8_t y, Colour color)
            {
                // Draw in the right color
  ✓✓  4656      if(color == Colour::White)
                {
      2354          m_buffer[x + (y / 8) * m_width] |= 1 << (y % 8);
                    #ifdef ENABLE_SSD1306_TEST_STDOUT
                            std::cout << "1";
                    #endif
                }
                else
                {
      2302          m_buffer[x + (y / 8) * m_width] &= ~(1 << (y % 8));
                    #ifdef ENABLE_SSD1306_TEST_STDOUT
                            std::cout << "_";
                    #endif
                }

      4656      return true;
            }

         7  bool Display::set_cursor(uint8_t x, uint8_t y)
            {
✓✓✓✓     7      if(x >= m_width || y >= m_height)
                {
         2          return false;
                }
                else
                {
         5          m_currentx = x;
         5          m_currenty = y;
                }
         5      return true;
            }


         8  void Display::reset()
            {
             // CS = High (not selected)
             //HAL_GPIO_WritePin(Display_CS_Port, Display_CS_Pin, GPIO_PIN_SET);

             // Reset the Display
                #ifdef USE_SSD1306_HAL_DRIVER
                    HAL_GPIO_WritePin(m_reset_port, m_reset_pin, GPIO_PIN_RESET);
                    HAL_Delay(10);
                    HAL_GPIO_WritePin(m_reset_port, m_reset_pin, GPIO_PIN_SET);
                    HAL_Delay(10);
                #elif defined(USE_SSD1306_LL_DRIVER)
                    LL_GPIO_ResetOutputPin(SPI1_RESET_GPIO_Port, SPI1_RESET_Pin);
                    LL_mDelay(10);
                    LL_GPIO_SetOutputPin(SPI1_RESET_GPIO_Port, SPI1_RESET_Pin);
                    LL_mDelay(10);
                #endif
         8  }

       536  bool Display::write_command(uint8_t cmd_byte)
            {
                #if defined(USE_SSD1306_HAL_DRIVER)
                    HAL_StatusTypeDef res = HAL_OK;
                    //HAL_GPIO_WritePin(m_cs_port, m_cs_pin, GPIO_PIN_RESET); // select Display
                    HAL_GPIO_WritePin(m_dc_port, m_dc_pin, GPIO_PIN_RESET); // command
                    res = HAL_SPI_Transmit(&m_spi_port, (uint8_t *) &cmd_byte, 1, HAL_MAX_DELAY);
                    if (res != HAL_OK)
                    {
                        return false;
                    }
                    return true;
                    //HAL_GPIO_WritePin(m_cs_port, m_cs_pin, GPIO_PIN_SET); // un-select Display
                #elif defined(USE_SSD1306_LL_DRIVER)
                    LL_GPIO_ResetOutputPin(SPI1_DC_GPIO_Port, SPI1_DC_Pin);
                    LL_SPI_TransmitData8(SPI1, cmd_byte);
                    return true;
                #else
                    UNUSED (cmd_byte);
       536          return true;
                #endif

            }

       104  bool Display::write_data(uint16_t page_idx_within_buffer)
            {
                #if defined(USE_SSD1306_HAL_DRIVER)
                    HAL_StatusTypeDef res = HAL_OK;
                    //HAL_GPIO_WritePin(m_cs_port, m_cs_pin, GPIO_PIN_RESET); // select Display
                    HAL_GPIO_WritePin(m_dc_port, m_dc_pin, GPIO_PIN_SET); // data
                    res = HAL_SPI_Transmit(&m_spi_port, data_buffer[page_idx_within_buffer], m_width, HAL_MAX_DELAY);
                    if (res != HAL_OK)
                    {
                        return false;
                    }
                    return true;
                    //HAL_GPIO_WritePin(m_cs_port, m_cs_pin, GPIO_PIN_SET); // un-select Display
                #else
                    #if defined(USE_SSD1306_LL_DRIVER)
220                      LL_GPIO_SetOutputPin(SPI1_DC_GPIO_Port, SPI1_DC_Pin);
221                  #endif
222          104     uint8_t byte_len {8};
223     ✓✓  1768     for (uint16_t idx = page_idx_within_buffer; idx < page_idx_within_buffer + m_width; idx += byte_len)
224                  {
225     ✓✗  1664         if (idx < m_buffer.size())
226                      {
227                          #if defined(USE_SSD1306_LL_DRIVER)
228                              LL_SPI_TransmitData8(SPI1, m_buffer[page_idx_within_buffer]);
229                          #else
230         1664                 std::cout << "0x" << std::hex << std::setfill ('0') <<  std::setw(2) << +m_buffer[page_idx_within_buffer] << " ";
231                          #endif
232                      }
233                  }
234          104     return true;
235              #endif
236          }
237
238          } // namespace ssd1306
```

GCC Code Coverage Report

			Exec	Total	Coverage
Directory:	./				
Date:	2021-12-18 03:55:13	**Lines:**	121	137	88.3 %
Legend:	low: < 75.0 % medium: >= 75.0 % high: >= 90.0 %	**Branches:**	81	156	51.9 %

File	Lines			Branches	
cpp_ssd1306/inc/font.hpp	🟩	100.0 %	8 / 8	100.0 %	2 / 2
cpp_ssd1306/inc/ssd1306.hpp	🟨	86.0 %	43 / 50	51.6 %	32 / 62
cpp_ssd1306/src/ssd1306.cpp	🟩	100.0 %	70 / 70	58.8 %	47 / 80
main_app/src/mainapp.cpp		0.0 %	0 / 9	0.0 %	0 / 12

Generated by: GCOVR (Version 4.2)

GCC Code Coverage Report

			Exec	Total	Coverage
Directory:	./				
File:	main_app/src/mainapp.cpp	Lines:	0	9	0.0 %
Date:	2021-12-18 03:55:13	Branches:	0	12	0.0 %

```
Branch  Exec  Source
              /*
               * mainapp.cpp
               *
               *  Created on: 7 Nov 2021
               *      Author: chris
               */

              #include "mainapp.hpp"
              #include <ssd1306.hpp>
              // #include <tlc5955.hpp>
              #include <chrono>
              #include <thread>

              #include <sstream>

              #ifdef __cplusplus
              extern "C"
              {
              #endif

              bool tlc5955_callback {false};

              #ifdef USE_TLC5955_HAL_DRIVER
               #include <stm32g0xx_hal.h>
               void HAL_SPI_TxCpltCallback(SPI_HandleTypeDef *hspi)
               {
                UNUSED(hspi);
                //HAL_SPI_DMAPause(&hspi2);

                // disable the gsclk
                HAL_GPIO_WritePin(TLC5955_SPI2_GSCLK_GPIO_Port, TLC5955_SPI2_GSCLK_Pin, GPIO_PIN_RESET);

                // toggle the latch pin
                //HAL_Delay(1);
                HAL_GPIO_WritePin(TLC5955_SPI2_LAT_GPIO_Port, TLC5955_SPI2_LAT_Pin, GPIO_PIN_SET);
                //HAL_Delay(1);
                HAL_GPIO_WritePin(TLC5955_SPI2_LAT_GPIO_Port, TLC5955_SPI2_LAT_Pin, GPIO_PIN_RESET);
                //HAL_Delay(1);

                // enable the gsclk
                HAL_GPIO_WritePin(TLC5955_SPI2_GSCLK_GPIO_Port, TLC5955_SPI2_GSCLK_Pin, GPIO_PIN_SET);

                //HAL_SPI_DMAResume(&hspi2);
               }
              #endif

              void mainapp()
              {

               // tlc5955::Driver leds;
               // leds.reset();
               // leds.set_latch_cmd(true);
               // leds.set_function_cmd(false, true, false, true, false);
               // leds.set_global_brightness_cmd(0x7F, 0x7F, 0x7F);
               // leds.set_max_current_cmd(0x4, 0x4, 0x4);
               // leds.set_dot_correction_cmd_all(0x7F);
               // leds.process_register();

               // enable the gsclk
               // #ifdef USE_TLC5955_HAL_DRIVER
                  //  HAL_GPIO_WritePin(TLC5955_SPI2_GSCLK_GPIO_Port, TLC5955_SPI2_GSCLK_Pin, GPIO_PIN_SET);

               //  leds.start_dma_transmit();
               // #endif


               static ssd1306::Font5x7 font;
```

```
    static ssd1306::Display oled;

    oled.init();

    uint8_t count = 0;
    //uint32_t delay_ms {0};
    while(true)
    {
     // run oled animation
     std::stringstream msg;
     msg << font.character_map[count];
     oled.write(msg, font, 2, 2, ssd1306::Colour::Black, ssd1306::Colour::White, 3, true);
     if (count < font.character_map.size() - 1) { count++; }
     else { count = 0; }

     // // turn all LEDs off
     // leds.reset();
     // leds.set_latch_cmd(false);
     // leds.set_greyscale_cmd_all(0x0000);
     // leds.process_register();

     // #ifdef USE_TLC5955_HAL_DRIVER
     //  HAL_Delay(100);
     // #endif

     // // turn all LEDs on
     // leds.reset();
     // leds.set_latch_cmd(false);
     // leds.set_greyscale_cmd_all(0x7FFF);
     // leds.process_register();

     // #ifdef USE_TLC5955_HAL_DRIVER
     //  HAL_Delay(100);
     // #endif

    }
   }



   #ifdef __cplusplus
   }
   #endif
```