CSSE2010/7201
AVR Project
Base Code Documentation

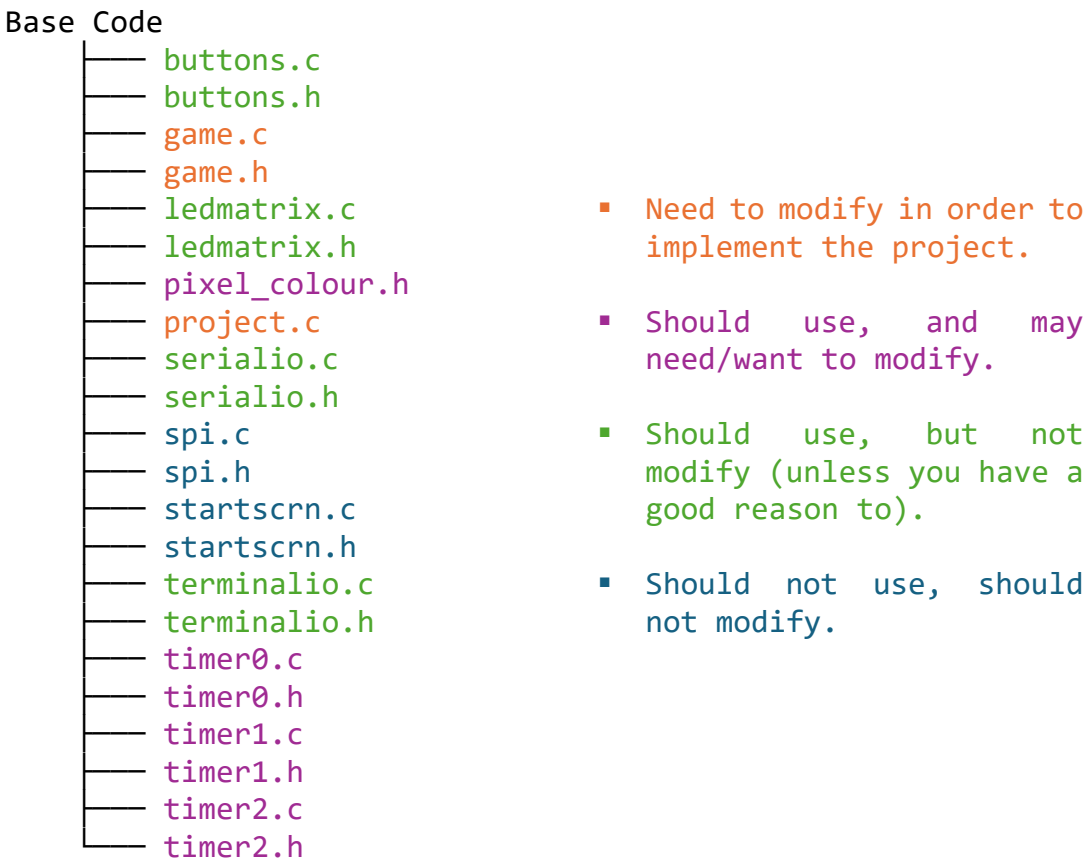
Revision 1.00 - Semester 2 2024

Last Updated: 21/09/2024

PDF Generated: 29/09/2024 12:07:42 PM

Overview

You have been provided with some base code to serve as a starting point for this AVR project. The base code contains several `.C` and `.H` files, and various modules to provide hardware abstraction. Below is a list of files which you have been provided with:



The main files which you will be modifying are `project.c` and `game.c`/`.h`. `project.c` is the entry point of the project which contains the game event loop and examples of how time-based events are implemented. You should read and understand this file. `game.c`/`.h` contains the implementation of the game components and is used to store the state of the game. You should read this file and understand what representation is used for the game state and the board representations. You may also need to modify and/or create other files, depending on how you choose to implement your project.

Modules

Public

These modules are publicly available to provide hardware abstraction and to help you with implementing the project. You are encouraged to use functionalities provided by these modules whenever possible, and you may modify them upon need (you will have to modify some of them in order to complete certain tasks).

Private

These modules are private, which means you should not attempt to use or modify them. They are only used by the base code, and by modifying them or using them in the code you write, you run the risk of breaking the project.

Buttons

Description:	Functions and definitions for interacting with the push buttons. It is assumed that buttons B0 – B3 are connected to pins B0 – B3.
Source File:	buttons.c
Include File:	buttons.h
Updated:	2024-07-23
Access:	Public
Should Modify:	No

Definitions

NUM_BUTTONS

Type:
Preprocessor Definition

Value:
4

ButtonState

Type:
Enumeration

Members:

Name	Value	Description
NO_BUTTON_PUSHED	-1	No button pushes.
BUTTON0_PUSHED	0	Button 0 pushed.
BUTTON1_PUSHED	1	Button 1 pushed.
BUTTON2_PUSHED	2	Button 2 pushed.
BUTTON3_PUSHED	3	Button 3 pushed.

Functions

```
void init_buttons(void)

  Description:
    Sets up pin change interrupts on pins B0 to B3. It is assumed that
    global interrupts are off when this function is called and are
    enabled sometime after this function is called. This function
    should only be called once.

  Parameters:
    None

  Returns:
    None
```

```
ButtonState button_pushed(void)

  Description:
    Gets the last button pushed. A small queue of button pushes is
    kept. This function should be called frequently enough to ensure
    the queue does not overflow. Excess button pushes are discarded.

  Parameters:
    None

  Returns:
    The last button pushed (BUTTONx_PUSHED), or NO_BUTTON_PUSHED if
    there are no button pushes to return.
```

```
void clear_button_presses(void)

  Description:
    Clears all buffered button presses.

  Parameters:
    None

  Returns:
    None
```

Pixel Colour

Description: Definitions for LED matrix colours.

Source File: N/A
Include File: pixel_colour.h

Updated: 2024-07-17

Access: Public
Should Modify: Maybe

Definitions

PixelColour

Type:
Data Type

Definition:
uint8_t

COLOUR_BLACK

Type:
Preprocessor Definition

Value:
0x00

COLOUR_RED

Type:
Preprocessor Definition

Value:
0x0F

COLOUR_LIGHT_GREEN

Type:
Preprocessor Definition

Value:
0x11

COLOUR_GREEN

Type:
Preprocessor Definition

Value:
0xF0

COLOUR_DARK_GREEN

Type:
Preprocessor Definition

Value:
0x10

COLOUR_LIGHT_YELLOW

Type:
Preprocessor Definition

Value:
0x11

COLOUR_YELLOW

Type:
Preprocessor Definition

Value:
0xFF

COLOUR_LIGHT_ORANGE

Type:
Preprocessor Definition

Value:
0x13

COLOUR_ORANGE

Type:
Preprocessor Definition

Value:
0x3C

LED Matrix

Description:	Functions and definitions for interacting with the LED matrix via SPI. These should be used to encapsulate all sending of SPI commands.
Source File:	ledmatrix.c
Include File:	ledmatrix.h
Updated:	2024-07-17
Access:	Public
Should Modify:	No

Definitions

MATRIX_NUM_ROWS

Type:
Preprocessor Definition

Value:
8

MATRIX_NUM_COLUMNS

Type:
Preprocessor Definition

Value:
16

MatrixData

Type:
Data Type

Definition:
```
PixelColour[MATRIX_NUM_ROWS][MATRIX_NUM_COLUMNS]
```

```
MatrixRow

  Type:
    Data Type

  Definition:
    PixelColour[MATRIX_NUM_COLUMNS]
```

```
MatrixColumn

  Type:
    Data Type

  Definition:
    PixelColour[MATRIX_NUM_ROWS]
```

Functions

```
void init_ledmatrix(void)

  Description:
    Sets up the LED matrix. This function must be called before any of
    the other LED matrix functions can be used. This function should
    only be called once.

  Parameters:
    None

  Returns:
    None
```

```
void ledmatrix_update_all(MatrixData data)

  Description:
    Updates all pixels of the LED matrix.

  Parameters:
    data           New colours for all pixels of the LED matrix.
```

```
  Returns:
    None
```

```
void ledmatrix_update_pixel(uint8_t row, uint8_t col, PixelColour pixel)

  Description:
    Updates a specific pixel of the LED matrix.

  Parameters:
    row             The row number of the pixel.
    col             The column number of the pixel.
    pixel           New colour of the pixel.

  Returns:
    None
```

```
void ledmatrix_update_row(uint8_t row, MatrixRow data)

  Description:
    Updates a row of the LED matrix.

  Parameters:
    row             The row to update.
    data            New colours for the row.

  Returns:
    None
```

```
void ledmatrix_update_column(uint8_t col, MatrixColumn data)

  Description:
    Updates a column of the LED matrix.

  Parameters:
    col             The column to update.
    data            New colours for the column.

  Returns:
    None
```

```
void ledmatrix_shift_display_left(void)
```

```
  Description:
    Shifts the entire LED matrix to the left by one column.

  Parameters:
    None

  Returns:
    None
```

```
void ledmatrix_shift_display_right(void)

  Description:
    Shifts the entire LED matrix to the right by one column.

  Parameters:
    None

  Returns:
    None
```

```
void ledmatrix_shift_display_up(void)

  Description:
    Shifts the entire LED matrix up by one row.

  Parameters:
    None

  Returns:
    None
```

```
void ledmatrix_shift_display_down(void)

  Description:
    Shifts the entire LED matrix down by one row.

  Parameters:
    None

  Returns:
    None
```

```
void ledmatrix_clear(void)

  Description:
    Clears the entire LED matrix.

  Parameters:
    None

  Returns:
    None
```

```
void copy_matrix_column(MatrixColumn from, MatrixColumn to)

  Description:
    Copies pixel colours from one MatrixColumn to another.

  Parameters:
    from            The source MatrixColumn.
    to              The destination MatrixColumn.

  Returns:
    None
```

```
void copy_matrix_row(MatrixRow from, MatrixRow to)

  Description:
    Copies pixel colours from one MatrixRow to another.

  Parameters:
    from            The source MatrixRow.
    to              The destination MatrixRow.

  Returns:
    None
```

```
void set_matrix_column_to_colour(MatrixColumn matrix_column, PixelColour
colour)

  Description:
    Sets the pixel colours of a MatrixColumn.

  Parameters:
    matrix_column   The MatrixColumn.
    colour          The colour.
```

```
  Returns:
    None
```

```
void set_matrix_row_to_colour(MatrixRow matrix_row, PixelColour colour)

  Description:
    Sets the pixel colours of a MatrixRow.

  Parameters:
    matrix_row     The MatrixRow.
    colour         The colour.

  Returns:
    None
```

Serial I/O

Description:	Module to allow standard input/output routines to be used via serial port 0 and functions for interacting with the input buffer.
Source File:	serialio.c
Include File:	serialio.h
Updated:	2024-07-17
Access:	Public
Should Modify:	No

Functions

```
void init_serial_stdio(long baudrate, bool echo)

  Description:
    Initialises serial I/O using the UART. This function must be called
    before any of the standard I/O functions. This function should only
    be called once.

  Parameters:
    baudrate   The baud rate (e.g., 19200).
    echo       Whether inputs are echoed back.

  Returns:
    None
```

```
bool serial_input_available(void)

  Description:
    Tests if input is available from the serial port. If there is input
    available, then it can be read with a suitable standard I/O library
    function, e.g., fgetc().

  Parameters:
    None

  Returns:
```

```
    Whether inputs are available.
```

```
void clear_serial_input_buffer(void)

  Description:
    Discards any input waiting to be read from the serial port. Useful
    for when characters may have been typed when we didn't want them.

  Parameters:
    None

  Returns:
    None
```

Terminal I/O

Description:	Functions and definitions for interacting with the terminal. These should be used to encapsulate all sending of escape sequences.
Source File:	terminalio.c
Include File:	terminalio.h
Updated:	2024-07-21
Access:	Public
Should Modify:	No

Definitions

DisplayParameter

Type:
Enumeration

Members:

Name	Value	Description
TERM_RESET	0	Resets terminal display attributes.
TERM_BRIGHT	1	Renders text brighter.
TERM_DIM	2	Renders text dimmer.
TERM_UNDERSCORE	4	Underscores text.
TERM_BLINK	5	Makes text blink.
TERM_REVERSE	7	Reverse video.
TERM_HIDDEN	8	Hide characters.
FG_BLACK	30	Black foreground.
FG_RED	31	Red foreground.
FG_GREEN	32	Green foreground.
FG_YELLOW	33	Yellow foreground.
FG_BLUE	34	Blue foreground.
FG_MAGENTA	35	Magenta foreground.
FG_CYAN	36	Cyan foreground.
FG_WHITE	37	White foreground.
BG_BLACK	40	Black background.
BG_RED	41	Red background.
BG_GREEN	42	Green background.
BG_YELLOW	43	Yellow background.
BG_BLUE	44	Blue background.

```
    BG_MAGENTA          45      Magenta background.
    BG_CYAN             46      Cyan background.
    BG_WHITE            47      White background.
```

Functions

```
void move_terminal_cursor(int row, int col)

  Description:
    Moves the terminal cursor to a new location. Row and column numbers
    use 0-based indexing.

  Parameters:
    row     The new row number of the terminal cursor.
    col     The new column number of the terminal cursor.

  Returns:
    None
```

```
void normal_display_mode(void)

  Description:
    Resets the terminal display mode.

  Parameters:
    None

  Returns:
    None
```

```
void clear_terminal(void)

  Description:
    Clears the terminal.

  Parameters:
    None

  Returns:
    None
```

```
void clear_to_end_of_line(void)

  Description:
    Clears to the end of the row the cursor is on.

  Parameters:
    None

  Returns:
    None
```

```
void set_display_attribute(DisplayParameter parameter)

  Description:
    Clears to the end of the row the cursor is on.

  Parameters:
    parameter      The display attribute to set.

  Returns:
    None
```

```
void hide_cursor(void)

  Description:
    Hides the blinking terminal cursor from the user.

  Parameters:
    None

  Returns:
    None
```

```
void show_cursor(void)

  Description:
    Shows the blinking terminal cursor to the user.

  Parameters:
    None
```

```
  Returns:
    None
```

```
void enable_scrolling_for_whole_display(void)

  Description:
    Enables scrolling for the entire terminal.

  Parameters:
    None

  Returns:
    None
```

```
void set_scroll_region(int row1, int row2)

  Description:
    Sets a custom scroll region.

  Parameters:
    row1    The top row of the region, inclusive.
    row2    The bottom row of the region, inclusive.

  Returns:
    None
```

```
void scroll_down(void)

  Description:
    Scrolls the scroll region of the terminal down. If the cursor is in
    the first (top) row of the scroll region then scroll the scroll
    region down by one row. The bottom row of the scroll region will be
    lost. The top row of the scroll region will be blank. If the cursor
    is not in the first row of the scroll region, then the cursor will
    just be moved up by one row.

  Parameters:
    None

  Returns:
    None
```

```
void scroll_up(void)

  Description:
    Scrolls the scroll region of the terminal up. If the cursor is in
    the last (bottom) row of the scroll region then scroll the scroll
    region up by one row. The top row of the scroll region will be
    lost. The bottom row of the scroll region will be blank. If the
    cursor is not in the last row of the scroll region, then cursor
    will just be moved down by one row.

  Parameters:
    None

  Returns:
    None
```

```
void draw_horizontal_line(int row, int start_col, int end_col)

  Description:
    Draws a white horizontal line on the terminal. Row and column
    numbers use 0-based indexing.

  Parameters:
    row             The row to draw the line on.
    start_col       The start column of the line, inclusive.
    end_col         The end column of the line, inclusive.

  Returns:
    None
```

```
void draw_vertical_line(int col, int start_row, int end_row)

  Description:
    Draws a white vertical line on the terminal. Row and column numbers
    use 0-based indexing.

  Parameters:
    col             The column to draw the line on.
    start_row       The start row of the line, inclusive.
    end_row         The end row of the line, inclusive.

  Returns:
    None
```

Timer 0

Description:	Module for the system clock, and function(s) for getting the current time. Timer 0 is setup to generate an interrupt every millisecond. Tasks that have to occur regularly (every millisecond or few) can be added to the interrupt handler (in timer0.c) or can be added to the main event loop that checks the clock tick value. Any tasks undertaken in the interrupt handler should be kept short so that we don't run the risk of missing an interrupt in future.
Source File:	timer0.c
Include File:	timer0.h
Updated:	2024-07-18
Access:	Public
Should Modify:	Maybe

Functions

```
void init_timer0(void)

  Description:
    Initialises timer 0 for system clock. An interrupt will be
    generated every millisecond to update the time reference. This
    function must be called before any of the other timer 0 functions
    can be used. This function should only be called once.

  Parameters:
    None

  Returns:
    None
```

```
uint32_t get_current_time(void)

  Description:
    Gets the current time (milliseconds since the timer was
    initialised).
```

```
  Parameters:
    None

  Returns:
    Milliseconds since timer 0 was initialised.
```

Timer 1

Description:	Skeletal timer 1 module. To be implemented by the programmer upon need.
Source File:	timer1.c
Include File:	timer1.h
Updated:	2024-07-18
Access:	Public
Should Modify:	Yes

Functions

```
void init_timer1(void)

  Description:
    Skeletal timer 1 initialisation function.

  Parameters:
    None

  Returns:
    None
```

Timer 2

Description:	Skeletal timer 2 module. To be implemented by the programmer upon need.
Source File:	timer2.c
Include File:	timer2.h
Updated:	2024-07-18
Access:	Public
Should Modify:	Yes

Functions

```
void init_timer2(void)

  Description:
    Skeletal timer 2 initialisation function.

  Parameters:
    None

  Returns:
    None
```

Serial Peripheral Interface

Description:	Functions for sending and receiving data via SPI. This module is only used by the base code, and you should not attempt to modify or use this module in the code you write yourself.
Source File:	spi.c
Include File:	spi.h
Updated:	2024-07-19
Access:	Private
Should Modify:	No

Functions

```
void spi_setup_master(uint8_t clockdivider)

  Description:
    Sets up SPI communication as a master. This function must be called
    before any of the SPI functions can be used. This function should
    only be called once.

  Parameters:
    clockdivider   The clock divider, should be one of 2, 4, 8, 16, 32,
                   64, 128.

  Returns:
    None
```

```
uint8_t spi_send_byte(uint8_t byte)

  Description:
    Sends and receives an SPI byte. This function will take at least 8
    cycles of the divided clock (i.e. will busy wait).

  Parameters:
    byte     The byte to send.

  Returns:
```

```
    The byte received.
```

Start Screen

Description:	Functions for displaying the start screen animation on the LED matrix and the title ASCII art on the terminal. This module is only used by the base code, and you should not attempt to modify or use this module in the code you write yourself.
Source File:	startscrn.c
Include File:	startscrn.h
Updated:	2024-07-18
Access:	Private
Should Modify:	No

Functions

```
void setup_start_screen(void)

  Description:
    Sets up the start screen on the LED matrix. This function must be
    called before the start screen may be updated.

  Parameters:
    None

  Returns:
    None
```

```
void update_start_screen(void)

  Description:
    Updates the start screen on the LED matrix.

  Parameters:
    None

  Returns:
    None
```

```
void display_terminal_title(uint8_t row, uint8_t col)

  Description:
    Draws the terminal title ASCII art.

  Parameters:
    row     The start row of the ASCII art.
    col     The start column of the ASCII art.

  Returns:
    None
```