My Project

Generated by Doxygen 1.10.0

Chapter 1

760 Pizza Press firwmare

Make sure you are building Release. Contact Austin Brown (austinb@bostonprecisionmotion.com) for support.

Chapter 2

Topic Index

2.1 Topics

Here is a list of all topics with brief descriptions:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

Chapter 5

Topic Documentation

5.1 CMSIS

Topics

- Stm32f0xx_system

5.1.1 Detailed Description

5.1.2 Stm32f0xx_system

Topics

- STM32F0xx_System_Private_Includes
- STM32F0xx_System_Private_TypesDefinitions
- STM32F0xx_System_Private_Defines
- STM32F0xx_System_Private_Macros
- STM32F0xx_System_Private_Variables
- STM32F0xx_System_Private_FunctionPrototypes
- STM32F0xx_System_Private_Functions

5.1.2.1 Detailed Description

5.1.2.2 STM32F0xx_System_Private_Includes

5.1.2.3 STM32F0xx_System_Private_TypesDefinitions

5.1.2.4 STM32F0xx_System_Private_Defines

Macros

- #define HSE_VALUE ((uint32_t)8000000)
- #define HSI_VALUE ((uint32_t)8000000)
- #define HSI48_VALUE ((uint32_t)48000000)

5.1.2.4.1 Detailed Description

5.1.2.4.2 Macro Definition Documentation

5.1.2.4.2.1 HSE_VALUE

```
#define HSE_VALUE ((uint32_t)8000000)
```

Default value of the External oscillator in Hz. This value can be provided and adapted by the user application.

5.1.2.4.2.2 HSI48_VALUE

```
#define HSI48_VALUE ((uint32_t)48000000)
```

Default value of the HSI48 Internal oscillator in Hz. This value can be provided and adapted by the user application.

5.1.2.4.2.3 HSI_VALUE

```
#define HSI_VALUE ((uint32_t)8000000)
```

Default value of the Internal oscillator in Hz. This value can be provided and adapted by the user application.

5.1.2.5 STM32F0xx_System_Private_Macros

5.1.2.6 STM32F0xx_System_Private_Variables

Variables

- uint32_t **SystemCoreClock** = 8000000
- const uint8_t **AHBPrescTable** [16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
- const uint8_t **APBPrescTable** [8] = {0, 0, 0, 0, 1, 2, 3, 4}

5.1.2.6.1 Detailed Description

5.1.2.7 STM32F0xx_System_Private_FunctionPrototypes

5.1.2.8 STM32F0xx_System_Private_Functions

Functions

- void SystemInit (void)

  Setup the microcontroller system.
- void SystemCoreClockUpdate (void)

  Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

5.1.2.8.1 Detailed Description

5.1.2.8.2 Function Documentation

5.1.2.8.2.1 SystemCoreClockUpdate()

```
void SystemCoreClockUpdate (
            void  )
```

Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

Note

Each time the core clock (HCLK) changes, this function must be called to update SystemCoreClock variable value. Otherwise, any configuration based on this variable will be incorrect.

- The system frequency computed by this function is not the real frequency in the chip. It is calculated based on the predefined constant and the selected clock source:

- If SYSCLK source is HSI, SystemCoreClock will contain the HSI_VALUE(∗)
- If SYSCLK source is HSE, SystemCoreClock will contain the HSE_VALUE(∗∗)
- If SYSCLK source is PLL, SystemCoreClock will contain the HSE_VALUE(∗∗) or HSI_VALUE(∗) multiplied/divided by the PLL factors.

(∗) HSI_VALUE is a constant defined in stm32f0xx_hal_conf.h file (default value 8 MHz) but the real value may vary depending on the variations in voltage and temperature.

(∗∗) HSE_VALUE is a constant defined in stm32f0xx_hal_conf.h file (its value depends on the application requirements), user has to ensure that HSE_VALUE is same as the real frequency of the crystal used. Otherwise, this function may have wrong result.

- The result of this function could be not correct when using fractional value for HSE crystal.

Parameters

None	

Return values

None	

5.1.2.8.2.2 SystemInit()

```
void SystemInit (
            void  )
```

Setup the microcontroller system.

Parameters

None	

Return values

None	

Chapter 6

Class Documentation

6.1 __MenuItem Struct Reference

Menu item structure–contains all data needed to display a menu item.

`#include <structs.h>`

Public Attributes

- MenuType type
- uint16_t **length**
  menu length
- int16_t **upper**
  numerical upper bound for value
- uint16_t **index**
  menu index
- int16_t **lower**
  numerical lower bound for value
- int16_t **step**
  numerical entry step
- int16_t **flag**
  yes/no entry flag mask
- int16_t **value**
  if applicable, target value modified by menu action
- int16_t * **target**
  pointer to target value to edit
- const char * **name**
  menu item name
- const char * **titlename**
  menu item title (usually shorter)
- struct __MenuItem * **parent**
  Null ptr if top level menu.
- struct __MenuItem * **items** [16]
  Child menu entries are NULL if not defined.
- HAL_StatusTypeDef(* **display**)(struct __MenuItem *)
  Write to the display.

6.1.1 Detailed Description

Menu item structure–contains all data needed to display a menu item.

6.1.2 Member Data Documentation

6.1.2.1 type

```
MenuType __MenuItem::type
```

0: menu 1: numerical entry 2: yes/no entry

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

- Core/Inc/structs.h

6.2 Button Struct Reference

Debounced button state.

```
#include <structs.h>
```

Public Attributes

- int **ctr**
- int **repeat_ctr**
- bool **state**
- bool **rising_edge_flag**
- bool **falling_edge_flag**

6.2.1 Detailed Description

Debounced button state.

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

- Core/Inc/structs.h

6.3 Config Union Reference

```
#include <structs.h>
```

Public Attributes

- uint32_t **regs** [5]
- struct {
  uint16_t **flags**
  int16_t **top_temp**
  int16_t **bottom_temp**
  int16_t **press_time1**
  int16_t **press_time2**
  int16_t **burps**
  uint32_t **ctr**
  };

6.3.1 Detailed Description

Press configuration data stored in RTC registers Functions in config.h

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

- Core/Inc/structs.h

6.4 MotorPI Struct Reference

PI controller parameters and state.

`#include <structs.h>`

Public Attributes

- float **KP**
- float **TI**
- float **accum**
- float **max_accum**

6.4.1 Detailed Description

PI controller parameters and state.

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

- Core/Inc/structs.h

6.5 Press Struct Reference

Container for all press mechanical and thermal setpoint/state.

`#include <structs.h>`

Public Attributes

- PressSetpoint **press_setpoint**
- PressState **press_state**
- ThermalSetpoint **thermal_setpoint**
- ThermalState **thermal_state**
- Config **config**

6.5.1 Detailed Description

Container for all press mechanical and thermal setpoint/state.

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

- Core/Inc/structs.h

6.6 PressSetpoint Struct Reference

Mechanical press setpoint data.

`#include <structs.h>`

Public Attributes

- int16_t **burps**
- int16_t **press_ticks1**
- int16_t **press_ticks2**
- bool **auto_mode**
- bool **enable**

6.6.1 Detailed Description

Mechanical press setpoint data.

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

- Core/Inc/structs.h

6.7 PressState Struct Reference

Mechanical press state data.

`#include <structs.h>`

Public Attributes

- PressMode **mode**
- PressCycleMode **cycle**
- bool **overload_flag**
- int16_t **burp_ctr**
- int16_t **ticks_until_next**
- float **motor_setpoint**
- float **motor_slew_limited_setpoint**
- float **current_limit**
- uint32_t **error_code**

6.7.1 Detailed Description

Mechanical press state data.

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

- Core/Inc/structs.h

6.8 ThermalSetpoint Struct Reference

Thermal press setpoint data.

```
#include <structs.h>
```

Public Attributes

- float **top_temp**
- float **bottom_temp**
- bool **enable**

6.8.1 Detailed Description

Thermal press setpoint data.

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

- Core/Inc/structs.h

6.9 ThermalState Struct Reference

Thermal press state data.

```
#include <structs.h>
```

Public Attributes

- union {
  float **temp_buf** [4]
  struct {
  float **top1**
  float **bottom1**
  float **top2**
  float **bottom2**
  }
  };

- float **top_temp**
- float **bottom_temp**
- float **top_threshold**
- float **bottom_threshold**
- bool **top_ready**
- bool **bottom_ready**
- uint16_t **bad_read_countdown** [4]
- uint8_t **error**
- uint32_t **error_code**
- bool **top_ssr_on**
- bool **bottom_ssr_on**

6.9.1 Detailed Description

Thermal press state data.

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

- Core/Inc/structs.h

Chapter 7

File Documentation

7.1 Core/Inc/config.h File Reference

760 Pizza Press configuration constants, backup, and restore

```
#include "main.h"
#include "structs.h"
```

Macros

- #define **CONFIG_MODE_FLAG** (1u << 1)
- #define **CONFIG_UNITS_FLAG** (1u << 2)
- #define **CONFIG_BUZZER_FLAG** (1u << 3)
- #define **CONFIG_ECO_FLAG** (1u << 4)
- #define **DEFAULT_CONFIG_FLAGS** 1u
- #define **DEFAULT_TOP_TEMP** -40
- #define **DEFAULT_BOTTOM_TEMP** -40
- #define **DEFAULT_PRESS_TIME** 1000
- #define **DEFAULT_BURPS** 1
- #define **BURPS_LOWER_LIM** 0
- #define **BURPS_UPPER_LIM** 5
- #define **PRESS_TIME_LOWER_LIM** 500
- #define **PRESS_TIME_UPPER_LIM** 10000
- #define **TEMP_LOWER_LIM_F** 120
- #define **TEMP_UPPER_LIM_F** 325
- #define **TEMP_LOWER_LIM_C** 50
- #define **TEMP_UPPER_LIM_C** 163
- #define **THERMO_SCALING_FACTOR** 0.983f

Functions

- void **backup_settings** (Config *)

  Write config settings to RTC backup registers.
- void restore_settings (Config *)
- void reset_defaults (Config *)
- void **config_to_setpoints** (Press *)

  Process config flags and set press temperature setpoints.

7.1.1 Detailed Description

760 Pizza Press configuration constants, backup, and restore

Author

Aaron Yeiser

Date

2022-08-10

Copyright

7.1.2 Function Documentation

7.1.2.1 reset_defaults()

```
void reset_defaults (
            Config * config )
```

Reset configuration to default. restore_settings() must be called to write to RTC registers

7.1.2.2 restore_settings()

```
void restore_settings (
            Config * config )
```

Read settings from RTC backup registers If RTC settings are invalid, we restore to default

7.2 config.h

Go to the documentation of this file.

```
00001
00011 #ifndef INC_CONFIG_H_
00012 #define INC_CONFIG_H_
00013
00014 #include "main.h"
00015 #include "structs.h"
00016
00017 // 0 (1)
00018 #define CONFIG_MODE_FLAG (1u << 1)       // Manual (Auto)
00019 #define CONFIG_UNITS_FLAG (1u << 2)      // Fahrenheit (Celsius)
00020 #define CONFIG_BUZZER_FLAG (1u << 3)     // Off (On)
00021 #define CONFIG_ECO_FLAG (1u << 4)        // Off (On)
00022
00023 // if this bit is not 1, the backup registers were reset
00024 #define DEFAULT_CONFIG_FLAGS 1u
00025 #define DEFAULT_TOP_TEMP -40
00026 #define DEFAULT_BOTTOM_TEMP -40
00027 #define DEFAULT_PRESS_TIME 1000          // milliseconds
00028 #define DEFAULT_BURPS 1
00029
00030 #define BURPS_LOWER_LIM 0
00031 #define BURPS_UPPER_LIM 5
```

```
00032
00033 #define PRESS_TIME_LOWER_LIM 500
00034 #define PRESS_TIME_UPPER_LIM 10000
00035
00036 #define TEMP_LOWER_LIM_F 120
00037 #define TEMP_UPPER_LIM_F 325 // 400
00038
00039 #define TEMP_LOWER_LIM_C 50
00040 #define TEMP_UPPER_LIM_C 163
00041
00042 // fudge factor for thermocouple measurement
00043 #define THERMO_SCALING_FACTOR 0.983f
00044 //#define THERMO_SCALING_FACTOR 1.0f
00045
00047 void backup_settings(Config*);
00048
00051 void restore_settings(Config*);
00052
00055 void reset_defaults(Config*);
00056
00058 void config_to_setpoints(Press*);
00059
00060
00061 #endif /* INC_CONFIG_H_ */
```

7.3 Core/Inc/control.h File Reference

760 Pizza Press mechanical and thermo controls

```
#include "main.h"
#include "debounce.h"
#include "config.h"
#include "structs.h"
```

Macros

- #define **ECO_TIMEOUT** 900000ul
- #define **MAX_SLEW_RATE** 0.01f
- #define **DUTY_CYCLE_FAST** 0.99f
- #define **DUTY_CYCLE_SLOW** 0.5f
- #define **DUTY_CYCLE_JOG** 0.3f
- #define **PRESS_TIME_FASTDROP** 1000
- #define **PRESS_TIME_TAP_UP** 700
- #define **PRESS_TIME_TAP_DOWN** 2000
- #define **MOTOR_CURRENT_MAX** 14.0f
- #define **MOTOR_CURRENT_HIGH** 8.0f
- #define **MOTOR_CURRENT_LOW** 4.0f
- #define **MAX_CURRENT_SQR** (MOTOR_CURRENT_MAX∗MOTOR_CURRENT_MAX)
- #define **CURRENT_FILT** 0.05f
- #define **R_SHUNT** 0.002f
- #define **ADC_VOLTAGE** 3.3f
- #define **SHUNT_GAIN** 43.0f
- #define **ADC_CONV_FACTOR** (ADC_VOLTAGE / (SHUNT_GAIN ∗ R_SHUNT ∗ 4096.0f))
- #define **THERM_FILTER_COEFF** 0.01f
- #define **THERM_DEADBAND** 0.5f
- #define **PRESS_OK** 0u
- #define **ERR_INTERLOCK** 1ul
- #define **ERR_OVERCURRENT** (1ul << 1)
- #define **ERR_BAD_MOTOR** (1ul << 2)
- #define **ERR_BAD_SWITCH** (1ul << 3)

- #define **ERR_OVERSHOOT** (1ul << 4)
- #define **ERR_BAD_TOP_THERMO1** (1ul << 5)
- #define **ERR_BAD_BOTTOM_THERMO1** (1ul << 6)
- #define **ERR_BAD_TOP_THERMO2** (1ul << 7)
- #define **ERR_BAD_BOTTOM_THERMO2** (1ul << 8)
- #define **ERR_TOP_THERMO_MISMATCH** (1ul << 9)
- #define **ERR_BOTTOM_THERMO_MISMATCH** (1ul << 10)
- #define **ERR_TOP_HEATER** (1ul << 11)
- #define **ERR_BOTTOM_HEATER** (1ul << 12)

Functions

- uint32_t check_interlocks (Press *press)
  Check press safety interlocks.
- void motor_state_machine (TIM_HandleTypeDef *htim, Press *press)
  Press mechanical state machine.
- float get_shunt_current (ADC_HandleTypeDef *hadc)
  Read the ADC current in Amps.
- float motor_pi_update (MotorPI *state, float err)
  Run PI controller for motor control.
- void motor_pwm_update (TIM_HandleTypeDef *htim, Press *press, float current)
  Set motor duty cycle with slew rate limit and current limit.
- HAL_StatusTypeDef read_thermocouples (SPI_HandleTypeDef *hspi, Press *press)
  Read SPI thermocouple readers.
- void thermal_control_loop (SPI_HandleTypeDef *hspi, Press *press)
  run thermal bang-bang control loop
- int getTopTempDisplay (Press *press)
  Top temperature rounded to int and in the correct units.
- int getBottomTempDisplay (Press *press)
  Bottom temperature rounded to int and in the correct units.

7.3.1 Detailed Description

760 Pizza Press mechanical and thermo controls

Author

Aaron Yeiser

Date

2022-08-05

Copyright

7.3.2 Function Documentation

7.3.2.1 check_interlocks()

```
uint32_t check_interlocks (
            Press * press )
```

Check press safety interlocks.

Parameters

press	Press state

ERR_INTERLOCK flag is set if the tray is open (interlock = 1) and the press is not homed to the top of travel ERR↩ _INTERLOCK flag is set if the tray is open and the press state is not READY or DONE ERR_BAD_SWITCH flag is set if the top and bottom switches are both tripped

Returns

error state flags

7.3.2.2 get_shunt_current()

```
float get_shunt_current (
            ADC_HandleTypeDef * hadc )
```

Read the ADC current in Amps.

Parameters

hadc	the ADC handle to read

Returns

float the current

7.3.2.3 getBottomTempDisplay()

```
int getBottomTempDisplay (
            Press * press )
```

Bottom temperature rounded to int and in the correct units.

Parameters

press	Press state

Returns

int Temperature to display

7.3.2.4 getTopTempDisplay()

```
int getTopTempDisplay (
            Press * press )
```

Top temperature rounded to int and in the correct units.

Parameters

press	Press state

Returns

int Temperature to display

7.3.2.5 motor_pi_update()

```
float motor_pi_update (
            MotorPI * state,
            float err )
```

Run PI controller for motor control.

Parameters

state	PI controller state
err	setpoint - measured error

Returns

controller effort

Note

this is not currently used

7.3.2.6 motor_pwm_update()

```
void motor_pwm_update (
            TIM_HandleTypeDef * htim,
            Press * press,
            float current )
```

Set motor duty cycle with slew rate limit and current limit.

Parameters

htim	PWM timer handle
press	Press state
current	Measured press current

7.3.2.7 motor_state_machine()

```
void motor_state_machine (
```

```
            TIM_HandleTypeDef * htim,
            Press * press )
```

Press mechanical state machine.

Parameters

htim	PWM timer for motor control
press	Press state

Note

press->press_state.ticks_until_next is used to execute an action after some time delay

PRESS_READY: The press is sitting at top stroke. It will start descending when buttons are pressed All relevant state variables are reset

PRESS_ERROR: Something bad happened. Slowly jog the press up to top of stroke at low current PRESS_DONE state is entered once the press is homed

PRESS_DOWN: Move the press down If cycle mode is PRESS_FASTDROP (exiting READY mode) move down quickly for PRESS_TIME_FASTDROP ticks Once fastdrop is done we limit the drop speed to avoid crashing the press

In manual mode we continue pressing until the bottom limit switch is tripped or the buttons released In auto mode we continue pressing until a timeout is tripped or bottom limit switch is tripped

PRESS_DWELL: Press is all the way down Handle dough tapping in auto mode (count number of taps remaining) Start upward motion if buttons released (manual mode) or tapping timeout (auto mode)

PRESS_UP: Move the press up (slow if tapping dough, fast otherwise) Move the press down if in manual mode or buttons pressed Move the press down if tapping dough and tap counter is nonzero

PRESS_DONE: Immediately goes to PRESS_READY after buttons released This prevents the press from immediately cycling in manual mode

PRESS_JOG: Jog mode, used to move the platen up and down with menu buttons

7.3.2.8 read_thermocouples()

```
HAL_StatusTypeDef read_thermocouples (
            SPI_HandleTypeDef * hspi,
            Press * press )
```

Read SPI thermocouple readers.

Parameters

hspi	SPI handle
press	Press state

Returns

HAL_StatusTypeDef SPI read status

Note

we cycle through thermocouples, only one of four TCs is read per tick

7.3.2.9 thermal_control_loop()

```
void thermal_control_loop (
            SPI_HandleTypeDef * hspi,
            Press * press )
```

run thermal bang-bang control loop

Parameters

hspi	SPI handle for reading thermocouples
press	Press state

7.4 control.h

Go to the documentation of this file.

```
00001
00011 #ifndef INC_CONTROL_H_
00012 #define INC_CONTROL_H_
00013
00014 #include "main.h"
00015 #include "debounce.h"
00016 #include "config.h"
00017 #include "structs.h"
00018
00019 #define ECO_TIMEOUT 900000ul // 15 minutes = 900 seconds
00020
00021 // duty cycle
00022 #define MAX_SLEW_RATE 0.01f
00023
00024 #define DUTY_CYCLE_FAST 0.99f
00025 #define DUTY_CYCLE_SLOW 0.5f
00026 #define DUTY_CYCLE_JOG 0.3f
00027
00028 // time units in milliseconds
00029 #define PRESS_TIME_FASTDROP 1000
00030 #define PRESS_TIME_TAP_UP 700
00031 #define PRESS_TIME_TAP_DOWN 2000
00032
00033
00034 // overcurrent protection (amps)
00035 #define MOTOR_CURRENT_MAX 14.0f
00036 #define MOTOR_CURRENT_HIGH 8.0f
00037 #define MOTOR_CURRENT_LOW 4.0f
00038
00039 #define MAX_CURRENT_SQR (MOTOR_CURRENT_MAX*MOTOR_CURRENT_MAX)
00040 #define CURRENT_FILT 0.05f
00041
00042
00043 // current measurement
00044 #define R_SHUNT 0.002f
00045 #define ADC_VOLTAGE 3.3f
00046 #define SHUNT_GAIN 43.0f
00047 #define ADC_CONV_FACTOR (ADC_VOLTAGE / (SHUNT_GAIN * R_SHUNT * 4096.0f))
00048
00049 // thermal control
00050 #define THERM_FILTER_COEFF 0.01f    // update rate of 250 Hz --> 400 ms time constant
```

```
00051 #define THERM_DEADBAND 0.5f
00052
00053 // error code flags
00054 #define PRESS_OK 0u
00055
00056 #define ERR_INTERLOCK 1ul
00057 #define ERR_OVERCURRENT (1ul << 1)
00058 #define ERR_BAD_MOTOR (1ul << 2)
00059 #define ERR_BAD_SWITCH (1ul << 3)
00060 #define ERR_OVERSHOOT (1ul << 4)
00061
00062 #define ERR_BAD_TOP_THERMO1 (1ul << 5)
00063 #define ERR_BAD_BOTTOM_THERMO1 (1ul << 6)
00064 #define ERR_BAD_TOP_THERMO2 (1ul << 7)
00065 #define ERR_BAD_BOTTOM_THERMO2 (1ul << 8)
00066
00067 #define ERR_TOP_THERMO_MISMATCH (1ul << 9)
00068 #define ERR_BOTTOM_THERMO_MISMATCH (1ul << 10)
00069
00070 #define ERR_TOP_HEATER (1ul << 11)
00071 #define ERR_BOTTOM_HEATER (1ul << 12)
00072
00083 uint32_t check_interlocks(Press* press);
00084
00119 void motor_state_machine(TIM_HandleTypeDef* htim, Press* press);
00120
00126 float get_shunt_current(ADC_HandleTypeDef* hadc);
00127
00136 float motor_pi_update(MotorPI* state, float err);
00137
00144 void motor_pwm_update(TIM_HandleTypeDef* htim, Press* press, float current);
00145
00154 HAL_StatusTypeDef read_thermocouples(SPI_HandleTypeDef* hspi, Press* press);
00155
00163 void thermal_control_loop(SPI_HandleTypeDef* hspi, Press* press);
00164
00170 int getTopTempDisplay(Press* press);
00171
00177 int getBottomTempDisplay(Press* press);
00178
00179 #endif /* INC_CONTROL_H_ */
```

7.5 Core/Inc/debounce.h File Reference

760 Pizza Press debounced buttons

```
#include "main.h"
#include "structs.h"
```

Macros

- #define **SETTLING_TIME** 10
- #define **REPEAT_TIME** 500
- #define **REPEAT_INTERVAL** 50

Functions

- bool debounce (Button *button, bool state)

  Debounce a button. This function must be called frequently (at least 1kHz)
- void debounce_menu_buttons (void)
- void **debounce_activate_buttons** (void)

  Debounce the activate buttons (on press sides)
- void **debounce_interlock** (void)

  Debounce the press interlock.

7.5.1 Detailed Description

760 Pizza Press debounced buttons

Author

Aaron Yeiser

Date

2022-08-05

Copyright

7.5.2 Function Documentation

7.5.2.1 debounce()

```
bool debounce (
            Button * button,
            bool state )
```

Debounce a button. This function must be called frequently (at least 1kHz)

If the button is held down this will also generate a rising edge flag every REPEAT_INTERVAL after a delay of REPEAT_TIME

Parameters

button	The button debounced state
state	Measured state of the physical button

Returns

bool Debounced button state

7.5.2.2 debounce_menu_buttons()

```
void debounce_menu_buttons (
            void  )
```

Debounce all of the menu up/down/enter buttons Note that menu buttons are active low but the debounced states are active high

7.6 debounce.h

Go to the documentation of this file.

```
00001
00011 #ifndef INC_DEBOUNCE_H_
00012 #define INC_DEBOUNCE_H_
00013
00014 #include "main.h"
00015 #include "structs.h"
00016
00017 // Debounce parameters
00018 #define SETTLING_TIME 10
00019 #define REPEAT_TIME 500
00020 #define REPEAT_INTERVAL 50
00021
00032 bool debounce(Button* button, bool state);
00033
00037 void debounce_menu_buttons(void);
00038
00040 void debounce_activate_buttons(void);
00041
00043 void debounce_interlock(void);
00044
00045 #endif /* INC_DEBOUNCE_H_ */
```

7.7 Core/Inc/font_16x12.h File Reference

760 Pizza Press 16x12 px large display font

```
#include <stdint.h>
```

Macros

- #define **FONT8x8_START** 0x20
- #define **FONT16x12_WIDTH** 12
- #define **FONT16x12_HEIGHT** 16
- #define **FONT16x12_BYTES** 1

7.7.1 Detailed Description

760 Pizza Press 16x12 px large display font

Author

Austin Brown

Date

2022-08-05

Copyright

7.8 font_16x12.h

Go to the documentation of this file.

```
00001
00011 #ifndef _FONT_16x12_H_
00012 #define _FONT_16x12_H_
00013
00014 #include <stdint.h>
00015
00016 //----- DEFINES -----
00017 #define FONT8x8_START                   0x20
00018 //#define FONT8x8_END                     0x44
00019 #define FONT16x12_WIDTH                 12
00020 #define FONT16x12_HEIGHT                16
00021 #define FONT16x12_BYTES                 1
00022
00023 //=======================
00024
00025
00026 static const uint8_t font_16x12_0[][12] = {
00027 { 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
00028 { 0x0, 0x0, 0x0, 0x0, 0x0,0xfe,0xfe, 0x0, 0x0, 0x0, 0x0, 0x0},
00029 { 0x0, 0x0, 0x0, 0x0,0x1e, 0x0, 0x0,0x1e, 0x0, 0x0, 0x0, 0x0},
00030 { 0x0, 0x0,0x40,0x40,0xf8,0x40,0x40,0xf8,0x40,0x40, 0x0, 0x0},
00031 { 0x0, 0x0,0x30,0x48,0x84,0xfe,0x84,0x84, 0x4, 0x8, 0x0, 0x0},
00032 { 0x0, 0x0,0x18,0x24,0x18, 0x0,0xc0,0x30, 0xc, 0x0, 0x0, 0x0},
00033 { 0x0, 0x0, 0x0,0xbc,0x42,0x82,0x42,0x42,0x3c, 0x0, 0x0, 0x0},
00034 { 0x0, 0x0, 0x0, 0x0, 0x0,0x1e, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
00035 { 0x0, 0x0, 0x0, 0x0,0xe0,0x18, 0x6, 0x0, 0x0, 0x0, 0x0, 0x0},
00036 { 0x0, 0x0, 0x0, 0x0, 0x0, 0x6,0x18,0xe0, 0x0, 0x0, 0x0, 0x0},
00037 //{ 0x0, 0x0, 0x0,0x60,0x80,0x80,0x60, 0x0, 0x0, 0x0, 0x0, 0x0},
00038 {0x00,0x00,0x80,0xc0,0xc0,0xc0,0xc0,0x80,0x00,0x00,0x00,0x00}, // modified asterisk
00039 { 0x0, 0x0, 0x0, 0x0, 0x0,0xe0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
00040 { 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
00041 { 0x0, 0x0, 0x0,0x80,0x80,0x80,0x80,0x80,0x80, 0x0, 0x0, 0x0},
00042 { 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
00043 { 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,0xc0,0x30, 0xc, 0x0, 0x0, 0x0},
00044 { 0x0, 0x0,0xf8, 0x4, 0x2,0x82,0x82, 0x2, 0x4,0xf8, 0x0, 0x0},
00045 { 0x0, 0x0, 0x0, 0x0, 0x8, 0xc,0xfe, 0x0, 0x0, 0x0, 0x0, 0x0},
00046 { 0x0, 0x0,0x18, 0x4, 0x2, 0x2, 0x2,0x82,0x44,0x38, 0x0, 0x0},
00047 { 0x0, 0x0, 0x8, 0x4, 0x2,0x42,0x42,0x42,0xbc, 0x0, 0x0, 0x0},
00048 { 0x0, 0x0,0x80,0x40,0x20,0x10, 0x8, 0x4,0xfe, 0x0, 0x0, 0x0},
00049 { 0x0, 0x0,0x7e,0x42,0x22,0x22,0x22,0x22,0x42,0x82, 0x0, 0x0},
00050 { 0x0, 0x0,0xf8,0xc4,0x42,0x42,0x42,0x42,0x84, 0x8, 0x0, 0x0},
00051 { 0x0, 0x0, 0x2, 0x2, 0x2, 0x2, 0x2,0xc2,0x32, 0xe, 0x0, 0x0},
00052 { 0x0, 0x0, 0x0,0xbc,0x42,0x42,0x42,0x42,0xbc, 0x0, 0x0, 0x0},
00053 { 0x0, 0x0,0x78,0x84, 0x2, 0x2, 0x2, 0x2,0x84,0xf8, 0x0, 0x0},
00054 { 0x0, 0x0, 0x0, 0x0, 0x0,0x60,0x60, 0x0, 0x0, 0x0, 0x0, 0x0},
00055 { 0x0, 0x0, 0x0, 0x0, 0x0,0x60,0x60, 0x0, 0x0, 0x0, 0x0, 0x0},
00056 { 0x0, 0x0,0x80,0x80,0x40,0x40,0x20,0x20,0x10,0x10, 0x0, 0x0},
00057 { 0x0, 0x0, 0x0,0x40,0x40,0x40,0x40,0x40,0x40,0x40, 0x0, 0x0},
00058 { 0x0, 0x0, 0x0,0x10,0x10,0x20,0x20,0x40,0x40,0x80,0x80, 0x0},
00059 { 0x0, 0x0,0x18, 0x4, 0x2, 0x2, 0x2, 0x2,0x84,0x78, 0x0, 0x0},
00060 { 0x0, 0x0,0xf8, 0x4,0xe2,0x12,0x12,0xf2, 0x4,0xf8, 0x0, 0x0},
00061 { 0x0, 0x0,0xfc, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2,0xfc, 0x0, 0x0},
00062 { 0x0, 0x0,0xfe,0x42,0x42,0x42,0x42,0x42,0xa4,0x18, 0x0, 0x0},
00063 { 0x0, 0x0,0xf8, 0x4, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x0, 0x0},
00064 { 0x0, 0x0,0xfe, 0x2, 0x2, 0x2, 0x2, 0x2, 0x4,0xf8, 0x0, 0x0},
00065 { 0x0, 0x0,0xfe,0x42,0x42,0x42,0x42, 0x2, 0x2, 0x2, 0x0, 0x0},
00066 { 0x0, 0x0,0xfe,0x42,0x42,0x42,0x42, 0x2, 0x2, 0x2, 0x0, 0x0},
00067 { 0x0, 0x0,0xf8, 0x4, 0x2, 0x2,0x82,0x82,0x82,0x84, 0x0, 0x0},
00068 { 0x0, 0x0,0xfe,0x80,0x80,0x80,0x80,0x80,0x80,0xfe, 0x0, 0x0},
00069 { 0x0, 0x0, 0x0, 0x2, 0x2,0xfe, 0x2, 0x2, 0x0, 0x0, 0x0, 0x0},
00070 { 0x0, 0x0, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2,0xfe, 0x0, 0x0},
00071 { 0x0, 0x0,0xfe,0x80,0x40,0x20,0x10, 0x8, 0x4, 0x2, 0x0, 0x0},
00072 { 0x0, 0x0,0xfe, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
00073 { 0x0, 0x0,0xfe, 0xc,0x30,0x40,0x40,0x30, 0xc,0xfe, 0x0, 0x0},
00074 { 0x0, 0x0,0xfe, 0x6,0x18,0x60,0x80, 0x0, 0x0,0xfe, 0x0, 0x0},
00075 { 0x0, 0x0,0xf8, 0x4, 0x2, 0x2, 0x2, 0x2, 0x4,0xf8, 0x0, 0x0},
00076 { 0x0, 0x0,0xfe, 0x2, 0x2, 0x2, 0x2, 0x2,0x84,0x78, 0x0, 0x0},
00077 { 0x0, 0x0,0xf8, 0x4, 0x2, 0x2, 0x2, 0x2, 0x4,0xf8, 0x0, 0x0},
00078 { 0x0, 0x0,0xfe, 0x2, 0x2, 0x2, 0x2, 0x2,0x84,0x78, 0x0, 0x0},
00079 { 0x0, 0x0,0x38,0x44,0x82,0x82,0x82,0x82, 0x4, 0x0, 0x0, 0x0},
00080 { 0x0, 0x0, 0x2, 0x2, 0x2,0xfe, 0x2, 0x2, 0x2, 0x2, 0x0, 0x0},
00081 { 0x0, 0x0,0xfe, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,0xfe, 0x0, 0x0},
00082 { 0x0, 0x0,0x1e,0xe0, 0x0, 0x0, 0x0, 0x0,0xe0,0x1e, 0x0, 0x0},
00083 { 0x0, 0x0,0xfe, 0x0, 0x0,0xe0,0xe0, 0x0, 0x0,0xfe, 0x0, 0x0},
00084 { 0x0, 0x0, 0x6,0x18,0x60,0x80,0x80,0x60,0x18, 0x6, 0x0, 0x0},
00085 { 0x0, 0x0, 0x6,0x18,0x60,0x80,0x80,0x60,0x18, 0x6, 0x0, 0x0},
00086 { 0x0, 0x0, 0x2, 0x2, 0x2,0x82,0x82,0x62,0x1a, 0x6, 0x0, 0x0},
00087 { 0x0, 0x0, 0x0, 0x0,0xfe, 0x2, 0x2, 0x2, 0x0, 0x0, 0x0, 0x0},
00088 { 0x0, 0x0, 0x0,0xc,0x30,0xc0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
00089 { 0x0, 0x0, 0x0, 0x0, 0x2, 0x2, 0x2,0xfe, 0x0, 0x0, 0x0, 0x0},
00090 { 0x0, 0x0,0x10, 0x8, 0x4, 0x2, 0x2, 0x4, 0x8,0x10, 0x0, 0x0},
00091 { 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
```

```
00092 { 0x0, 0x0, 0x0, 0x0, 0x0, 0x6,0x18, 0x0, 0x0, 0x0, 0x0, 0x0},
00093 { 0x0, 0x0, 0x0,0x80,0x40,0x40,0x40,0x40,0x40,0x80,0xc0, 0x0, 0x0},
00094 { 0x0, 0x0,0xfe,0x40,0x40,0x40,0x40,0x40,0x80, 0x0, 0x0, 0x0},
00095 { 0x0, 0x0, 0x0,0x80,0x40,0x40,0x40,0x40,0x40,0x80, 0x0, 0x0},
00096 { 0x0, 0x0, 0x0,0x80,0x40,0x40,0x40,0x40,0x40,0xfe, 0x0, 0x0},
00097 { 0x0, 0x0, 0x0,0x80,0x40,0x40,0x40,0x40,0x40,0x80, 0x0, 0x0},
00098 { 0x0, 0x0, 0x0,0xf8, 0x4, 0x2, 0x2, 0x2, 0x2, 0x0, 0x0, 0x0},
00099 { 0x0, 0x0,0xc0,0x20,0x10,0x10,0x10,0x10,0x20,0xc0, 0x0, 0x0},
00100 { 0x0, 0x0,0xfe,0x80,0x80,0x80,0x80,0x80, 0x0, 0x0, 0x0, 0x0},
00101 { 0x0, 0x0, 0x0, 0x0, 0x0,0xc0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
00102 { 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,0x60, 0x0, 0x0, 0x0, 0x0, 0x0},
00103 { 0x0, 0x0,0xfe, 0x0, 0x0, 0x0, 0x0, 0x0,0x80,0x80, 0x0, 0x0},
00104 { 0x0, 0x0, 0x0, 0x0,0xfe, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
00105 { 0x0, 0x0,0x80,0x40,0x80, 0x0, 0x0,0x80,0x40,0x80, 0x0, 0x0},
00106 { 0x0, 0x0,0x80,0x40,0x40,0x40,0x40,0x40,0x40,0x80, 0x0, 0x0},
00107 { 0x0, 0x0, 0x0,0x80,0x40,0x40,0x40,0x40,0x80, 0x0, 0x0, 0x0},
00108 { 0x0, 0x0,0xf0,0x10,0x10,0x10,0x10,0x10,0x20,0xc0, 0x0, 0x0},
00109 { 0x0, 0x0,0xc0,0x20,0x10,0x10,0x10,0x10,0x10,0xf0, 0x0, 0x0},
00110 { 0x0, 0x0, 0x0,0xc0,0x80,0x80,0x40,0x40,0x40, 0x0, 0x0, 0x0},
00111 { 0x0, 0x0,0x80,0x40,0x20,0x20,0x20,0x20,0x40,0x80, 0x0, 0x0},
00112 { 0x0, 0x0, 0x0,0x10,0x10,0xfe,0x10,0x10, 0x0, 0x0, 0x0, 0x0},
00113 { 0x0, 0x0,0xc0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,0xc0, 0x0, 0x0},
00114 { 0x0, 0x0,0xc0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,0xc0, 0x0, 0x0},
00115 { 0x0, 0x0,0xc0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,0xc0, 0x0, 0x0},
00116 { 0x0, 0x0,0x40,0x80, 0x0, 0x0, 0x0, 0x0,0x80,0x40, 0x0, 0x0},
00117 { 0x0, 0x0,0x40,0x80, 0x0, 0x0, 0x0, 0x0,0x80,0x40, 0x0, 0x0},
00118 { 0x0, 0x0,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0xc0, 0x0, 0x0},
00119 { 0x0, 0x0, 0x0,0x80,0x80,0x78, 0x4, 0x2, 0x2, 0x0, 0x0, 0x0},
00120 { 0x0, 0x0, 0x0, 0x0, 0x0,0xfe, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
00121 { 0x0, 0x0, 0x0, 0x2, 0x2, 0x4,0x78,0x80,0x80, 0x0, 0x0, 0x0},
00122 { 0x0, 0x0,0x80,0x40,0x40,0x80, 0x0, 0x0, 0x0,0x80, 0x0, 0x0},
00123 { 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}
00124 };
00125
00126
00127
00128 static const uint8_t font_16x12_1[][12] = {
00129 { 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
00130 { 0x0, 0x0, 0x0, 0x0, 0x0,0x33,0x33, 0x0, 0x0, 0x0, 0x0, 0x0},
00131 { 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
00132 { 0x0, 0x0, 0x2, 0x2,0x1f, 0x2, 0x2,0x1f, 0x2, 0x2, 0x0, 0x0},
00133 { 0x0, 0x0, 0x8,0x10,0x10,0x3f,0x10,0x10, 0x9, 0x6, 0x0, 0x0},
00134 { 0x0, 0x0, 0x0,0x30, 0xc, 0x3,0x18,0x24,0x18, 0x0, 0x0, 0x0},
00135 { 0x0, 0x0, 0xf,0x10,0x20,0x10, 0xd,0x12,0x22, 0x2, 0x0, 0x0},
00136 { 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
00137 { 0x0, 0x0, 0x0, 0x0, 0x3, 0xc,0x30, 0x0, 0x0, 0x0, 0x0, 0x0},
00138 { 0x0, 0x0, 0x0, 0x0, 0x0,0x30, 0xc, 0x3, 0x0, 0x0, 0x0, 0x0},
00139 //{ 0x0, 0x0, 0x1, 0xd, 0x3, 0x3, 0xd, 0x1, 0x0, 0x0, 0x0, 0x0},
00140 {0x00,0x00,0x01,0x03,0x03,0x03,0x03,0x01,0x00,0x00,0x00,0x00}, // modified asterisk
00141 { 0x0, 0x0, 0x1, 0x1, 0x1, 0xf, 0x1, 0x1, 0x1, 0x0, 0x0, 0x0},
00142 { 0x0, 0x0, 0x0, 0x0, 0x0,0x30, 0xc, 0x0, 0x0, 0x0, 0x0, 0x0},
00143 { 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
00144 { 0x0, 0x0, 0x0, 0x0,0x30,0x30, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
00145 { 0x0, 0x0, 0x0,0x30, 0xc, 0x3, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
00146 { 0x0, 0x0, 0xf,0x10,0x20,0x21,0x21,0x20,0x10, 0xf, 0x0, 0x0},
00147 { 0x0, 0x0, 0x0,0x20,0x20,0x3f,0x20,0x20, 0x0, 0x0, 0x0, 0x0},
00148 { 0x0, 0x0,0x30,0x28,0x24,0x22,0x21,0x20,0x20,0x20, 0x0, 0x0},
00149 { 0x0, 0x0, 0x8,0x10,0x20,0x20,0x20,0x20,0x10, 0xf, 0x0, 0x0},
00150 { 0x0, 0x0, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1,0x3f, 0x1, 0x0, 0x0},
00151 { 0x0, 0x0,0x20,0x20,0x20,0x20,0x20,0x20,0x10, 0xf, 0x0, 0x0},
00152 { 0x0, 0x0, 0xf,0x10,0x20,0x20,0x20,0x20,0x10, 0xf, 0x0, 0x0},
00153 { 0x0, 0x0, 0x0, 0x0,0x30, 0xc, 0x3, 0x0, 0x0, 0x0, 0x0, 0x0},
00154 { 0x0, 0x0, 0xf,0x10,0x20,0x20,0x20,0x20,0x10, 0xf, 0x0, 0x0},
00155 { 0x0, 0x0, 0x8,0x10,0x21,0x21,0x21,0x21,0x10, 0xf, 0x0, 0x0},
00156 { 0x0, 0x0, 0x0, 0x0, 0x0, 0x6, 0x6, 0x0, 0x0, 0x0, 0x0, 0x0},
00157 { 0x0, 0x0, 0x0, 0x0, 0x0,0x18, 0x6, 0x0, 0x0, 0x0, 0x0, 0x0},
00158 { 0x0, 0x0, 0x1, 0x1, 0x2, 0x2, 0x4, 0x4, 0x8, 0x8, 0x0, 0x0},
00159 { 0x0, 0x0, 0x0, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x0, 0x0},
00160 { 0x0, 0x0, 0x0, 0x8, 0x8, 0x4, 0x4, 0x2, 0x2, 0x1, 0x1, 0x0},
00161 { 0x0, 0x0, 0x0, 0x0, 0x0,0x32,0x32, 0x1, 0x0, 0x0, 0x0, 0x0},
00162 { 0x0, 0x0, 0xf,0x10,0x23,0x24,0x24,0x27,0x12, 0x9, 0x0, 0x0},
00163 { 0x0, 0x0,0x3f, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1,0x3f, 0x0, 0x0},
00164 { 0x0, 0x0,0x3f,0x20,0x20,0x20,0x20,0x20,0x10, 0xf, 0x0, 0x0},
00165 { 0x0, 0x0, 0xf,0x10,0x20,0x20,0x20,0x20,0x20,0x20, 0x0, 0x0},
00166 { 0x0, 0x0,0x3f,0x20,0x20,0x20,0x20,0x20,0x10, 0xf, 0x0, 0x0},
00167 { 0x0, 0x0,0x3f,0x20,0x20,0x20,0x20,0x20,0x20,0x20, 0x0, 0x0},
00168 { 0x0, 0x0,0x3f, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
00169 { 0x0, 0x0, 0xf,0x10,0x20,0x20,0x20,0x20,0x10, 0xf, 0x0, 0x0},
00170 { 0x0, 0x0,0x3f, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,0x3f, 0x0, 0x0},
00171 { 0x0, 0x0, 0x0,0x20,0x20,0x3f,0x20,0x20, 0x0, 0x0, 0x0, 0x0},
00172 { 0x0, 0x0,0x20,0x20,0x20,0x20,0x20,0x20,0x10, 0xf, 0x0, 0x0},
00173 { 0x0, 0x0,0x3f, 0x0, 0x1, 0x2, 0x4, 0x8,0x10,0x20, 0x0, 0x0},
00174 { 0x0, 0x0,0x3f,0x20,0x20,0x20,0x20,0x20,0x20,0x20, 0x0, 0x0},
00175 { 0x0, 0x0,0x3f, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,0x3f, 0x0, 0x0},
00176 { 0x0, 0x0,0x3f, 0x0, 0x0, 0x0, 0x1, 0x6,0x38,0x3f, 0x0, 0x0},
00177 { 0x0, 0x0, 0xf,0x10,0x20,0x20,0x20,0x10, 0xf, 0x0, 0x0, 0x0},
00178 { 0x0, 0x0,0x3f, 0x1, 0x1, 0x1, 0x1, 0x1, 0x0, 0x0, 0x0, 0x0},
```

```
00179 { 0x0, 0x0, 0xf,0x10,0x20,0x20,0x26,0x3c,0x18,0x37, 0x0, 0x0},
00180 { 0x0, 0x0,0x3f, 0x1, 0x1, 0x3, 0x5, 0x9,0x10,0x20, 0x0, 0x0},
00181 { 0x0, 0x0, 0x0,0x10,0x20,0x20,0x20,0x20,0x11, 0xe, 0x0, 0x0},
00182 { 0x0, 0x0, 0x0, 0x0, 0x0,0x3f, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
00183 { 0x0, 0x0, 0xf,0x10,0x20,0x20,0x20,0x20,0x10, 0xf, 0x0, 0x0},
00184 { 0x0, 0x0, 0x0, 0x0, 0x7,0x38,0x38, 0x7, 0x0, 0x0, 0x0, 0x0},
00185 { 0x0, 0x0, 0x1,0x3e,0x18, 0x7, 0x7,0x18,0x3e, 0x1, 0x0, 0x0},
00186 { 0x0, 0x0,0x30, 0xc, 0x3, 0x0, 0x0, 0x3, 0xc,0x30, 0x0, 0x0},
00187 { 0x0, 0x0, 0x0, 0x0, 0x0,0x3f,0x3f, 0x0, 0x0, 0x0, 0x0, 0x0},
00188 { 0x0, 0x0,0x30,0x2c,0x23,0x20,0x20,0x20,0x20,0x20, 0x0, 0x0},
00189 { 0x0, 0x0, 0x0, 0x0,0x3f,0x20,0x20,0x20, 0x0, 0x0, 0x0, 0x0},
00190 { 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x3, 0xc,0x30, 0x0, 0x0, 0x0},
00191 { 0x0, 0x0, 0x0, 0x0,0x20,0x20,0x20,0x3f, 0x0, 0x0, 0x0, 0x0},
00192 { 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
00193 { 0x0, 0x0,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20, 0x0, 0x0},
00194 { 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
00195 { 0x0, 0x0, 0xf,0x10,0x20,0x20,0x20,0x20,0x10,0x3f, 0x0, 0x0},
00196 { 0x0, 0x0,0x3f,0x20,0x20,0x20,0x20,0x20,0x10, 0xf, 0x0, 0x0},
00197 { 0x0, 0x0, 0xf,0x10,0x20,0x20,0x20,0x20,0x20,0x10, 0x0, 0x0},
00198 { 0x0, 0x0, 0xf,0x10,0x20,0x20,0x20,0x20,0x20,0x3f, 0x0, 0x0},
00199 { 0x0, 0x0, 0xf,0x12,0x22,0x22,0x22,0x22,0x22,0x13, 0x0, 0x0},
00200 { 0x0, 0x0, 0x1,0x3f, 0x1, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
00201 { 0x0, 0x0,0x10,0x21,0x22,0x22,0x22,0x22,0x22,0x12, 0xf, 0x0, 0x0},
00202 { 0x0, 0x0,0x3f, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1,0x3e, 0x0, 0x0},
00203 { 0x0, 0x0, 0x0, 0x0, 0x0,0x3e, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
00204 { 0x0, 0x0, 0x0, 0x0,0x20,0x10, 0xf, 0x0, 0x0, 0x0, 0x0, 0x0},
00205 { 0x0, 0x0,0x3f, 0x4, 0xa, 0xa,0x11,0x11,0x20,0x20, 0x0, 0x0},
00206 { 0x0, 0x0, 0x0, 0x0,0x1f,0x20,0x20,0x20, 0x0, 0x0, 0x0, 0x0},
00207 { 0x0, 0x0,0x3f, 0x0, 0x0, 0x3, 0x3, 0x0, 0x0,0x3f, 0x0, 0x0},
00208 { 0x0, 0x0,0x3f, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,0x3f, 0x0, 0x0},
00209 { 0x0, 0x0, 0xf,0x10,0x20,0x20,0x20,0x20,0x10, 0xf, 0x0, 0x0},
00210 { 0x0, 0x0,0x3f, 0x2, 0x2, 0x2, 0x2, 0x2, 0x1, 0x0, 0x0, 0x0},
00211 { 0x0, 0x0, 0x0, 0x1, 0x2, 0x2, 0x2, 0x2, 0x2,0x3f, 0x0, 0x0},
00212 { 0x0, 0x0, 0x0,0x3f, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
00213 { 0x0, 0x0, 0x8,0x11,0x22,0x22,0x22,0x22,0x14, 0x8, 0x0, 0x0},
00214 { 0x0, 0x0, 0x0, 0x0, 0x0,0x3f, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
00215 { 0x0, 0x0, 0xf,0x10,0x20,0x20,0x20,0x20,0x10,0x3f, 0x0, 0x0},
00216 { 0x0, 0x0, 0x0, 0x3, 0xc,0x30,0x30, 0xc, 0x3, 0x0, 0x0, 0x0},
00217 { 0x0, 0x0, 0x3,0x3c, 0x8, 0xe, 0xe, 0x8,0x3c, 0x3, 0x0, 0x0},
00218 { 0x0, 0x0,0x20,0x10, 0x9, 0x6, 0x6, 0x9,0x10,0x20, 0x0, 0x0},
00219 { 0x0, 0x0, 0x0,0x20,0x21,0x22,0x1e, 0x1, 0x0, 0x0, 0x0, 0x0},
00220 { 0x0, 0x0,0x30,0x28,0x24,0x24,0x22,0x22,0x21,0x20, 0x0, 0x0},
00221 { 0x0, 0x0, 0x0, 0x0, 0x0, 0xf,0x10,0x20,0x20, 0x0, 0x0, 0x0},
00222 { 0x0, 0x0, 0x0, 0x0, 0x0,0x3f, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
00223 { 0x0, 0x0, 0x0,0x20,0x20,0x10, 0xf, 0x0, 0x0, 0x0, 0x0, 0x0},
00224 { 0x0, 0x0, 0x1, 0x0, 0x0, 0x0, 0x1, 0x2, 0x2, 0x1, 0x0, 0x0},
00225 { 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}
00226 };
00227
00228 #endif // include
```

7.9 Core/Inc/font_8x8.h File Reference

760 Pizza Press 8x8 px small display font

```
#include <stdint.h>
```

Macros

- #define **FONT8x8_START** 0x20
- #define **FONT8x8_WIDTH** 7
- #define **FONT8x8_HEIGHT** 8
- #define **FONT8x8_BYTES** 1

7.9.1 Detailed Description

760 Pizza Press 8x8 px small display font

Author

Austin Brown

Date

2022-08-05

Copyright

7.10 font_8x8.h

Go to the documentation of this file.

```
00001
00011 #ifndef _FONT_8x8_H_
00012 #define _FONT_8x8_H_
00013
00014 #include <stdint.h>
00015
00016 //----- DEFINES -----
00017 #define FONT8x8_START                   0x20
00018 //#define FONT8x8_END                     0x44
00019 #define FONT8x8_WIDTH                   7
00020 #define FONT8x8_HEIGHT                  8
00021 #define FONT8x8_BYTES                   1
00022
00023 //=======================
00024 static const uint8_t font_8x8[][7] = {
00025 {0x00,0x00,0x00,0x00,0x00,0x00,0x00},
00026 {0x00,0x00,0x5F,0x00,0x00,0x00,0x00},
00027 {0x00,0x00,0x07,0x00,0x07,0x00,0x00},
00028 {0x00,0x14,0x7F,0x14,0x7F,0x14,0x00},
00029 {0x00,0x24,0x2A,0x7F,0x2A,0x12,0x00},
00030 {0x00,0x23,0x13,0x08,0x64,0x62,0x00},
00031 {0x00,0x36,0x49,0x55,0x22,0x50,0x00},
00032 {0x00,0x00,0x05,0x03,0x00,0x00,0x00},
00033 {0x00,0x1C,0x22,0x41,0x00,0x00,0x00},
00034 {0x00,0x41,0x22,0x1C,0x00,0x00,0x00},
00035 {0x00,0x08,0x2A,0x1C,0x2A,0x08,0x00},
00036 {0x00,0x08,0x08,0x3E,0x08,0x08,0x00},
00037 {0x00,0xA0,0x60,0x00,0x00,0x00,0x00},
00038 {0x00,0x08,0x08,0x08,0x08,0x08,0x00},
00039 {0x00,0x60,0x60,0x00,0x00,0x00,0x00},
00040 {0x00,0x20,0x10,0x08,0x04,0x02,0x00},
00041 {0x00,0x3E,0x51,0x49,0x45,0x3E,0x00},
00042 {0x00,0x00,0x42,0x7F,0x40,0x00,0x00},
00043 {0x00,0x62,0x51,0x49,0x49,0x46,0x00},
00044 {0x00,0x22,0x41,0x49,0x49,0x36,0x00},
00045 {0x00,0x18,0x14,0x12,0x7F,0x10,0x00},
00046 {0x00,0x27,0x45,0x45,0x45,0x39,0x00},
00047 {0x00,0x3C,0x4A,0x49,0x49,0x30,0x00},
00048 {0x00,0x01,0x71,0x09,0x05,0x03,0x00},
00049 {0x00,0x36,0x49,0x49,0x49,0x36,0x00},
00050 {0x00,0x06,0x49,0x49,0x29,0x1E,0x00},
00051 {0x00,0x00,0x36,0x36,0x00,0x00,0x00},
00052 {0x00,0x00,0xAC,0x6C,0x00,0x00,0x00},
00053 {0x00,0x08,0x14,0x22,0x41,0x00,0x00},
00054 {0x00,0x14,0x14,0x14,0x14,0x14,0x00},
00055 {0x00,0x41,0x22,0x14,0x08,0x00,0x00},
00056 {0x00,0x02,0x01,0x51,0x09,0x06,0x00},
00057 {0x00,0x32,0x49,0x79,0x41,0x3E,0x00},
00058 {0x00,0x7E,0x09,0x09,0x09,0x7E,0x00},
00059 {0x00,0x7F,0x49,0x49,0x49,0x36,0x00},
00060 {0x00,0x3E,0x41,0x41,0x41,0x22,0x00},
00061 {0x00,0x7F,0x41,0x41,0x22,0x1C,0x00},
00062 {0x00,0x7F,0x49,0x49,0x49,0x41,0x00},
00063 {0x00,0x7F,0x09,0x09,0x09,0x01,0x00},
00064 {0x00,0x3E,0x41,0x41,0x51,0x72,0x00},
00065 {0x00,0x7F,0x08,0x08,0x08,0x7F,0x00},
00066 {0x00,0x41,0x7F,0x41,0x00,0x00,0x00},
00067 {0x00,0x20,0x40,0x41,0x3F,0x01,0x00},
00068 {0x00,0x7F,0x08,0x14,0x22,0x41,0x00},
00069 {0x00,0x7F,0x40,0x40,0x40,0x40,0x00},
```

```
00070 {0x00,0x7F,0x02,0x0C,0x02,0x7F,0x00},
00071 {0x00,0x7F,0x04,0x08,0x10,0x7F,0x00},
00072 {0x00,0x3E,0x41,0x41,0x41,0x3E,0x00},
00073 {0x00,0x7F,0x09,0x09,0x09,0x06,0x00},
00074 {0x00,0x3E,0x41,0x51,0x21,0x5E,0x00},
00075 {0x00,0x7F,0x09,0x19,0x29,0x46,0x00},
00076 {0x00,0x26,0x49,0x49,0x49,0x32,0x00},
00077 {0x00,0x01,0x01,0x7F,0x01,0x01,0x00},
00078 {0x00,0x3F,0x40,0x40,0x40,0x3F,0x00},
00079 {0x00,0x1F,0x20,0x40,0x20,0x1F,0x00},
00080 {0x00,0x3F,0x40,0x38,0x40,0x3F,0x00},
00081 {0x00,0x63,0x14,0x08,0x14,0x63,0x00},
00082 {0x00,0x03,0x04,0x78,0x04,0x03,0x00},
00083 {0x00,0x61,0x51,0x49,0x45,0x43,0x00},
00084 {0x00,0x7F,0x41,0x41,0x00,0x00,0x00},
00085 {0x00,0x02,0x04,0x08,0x10,0x20,0x00},
00086 {0x00,0x41,0x41,0x7F,0x00,0x00,0x00},
00087 {0x00,0x04,0x02,0x01,0x02,0x04,0x00},
00088 {0x00,0x80,0x80,0x80,0x80,0x80,0x00},
00089 {0x00,0x01,0x02,0x04,0x00,0x00,0x00},
00090 {0x00,0x20,0x54,0x54,0x54,0x78,0x00},
00091 {0x00,0x7F,0x48,0x44,0x44,0x38,0x00},
00092 {0x00,0x38,0x44,0x44,0x28,0x00,0x00},
00093 {0x00,0x38,0x44,0x44,0x48,0x7F,0x00},
00094 {0x00,0x38,0x54,0x54,0x54,0x18,0x00},
00095 {0x00,0x08,0x7E,0x09,0x02,0x00,0x00},
00096 {0x00,0x18,0xA4,0xA4,0xA4,0x7C,0x00},
00097 {0x00,0x7F,0x08,0x04,0x04,0x78,0x00},
00098 {0x00,0x00,0x7D,0x00,0x00,0x00,0x00},
00099 {0x00,0x80,0x84,0x7D,0x00,0x00,0x00},
00100 {0x00,0x7F,0x10,0x28,0x44,0x00,0x00},
00101 {0x00,0x41,0x7F,0x40,0x00,0x00,0x00},
00102 {0x00,0x7C,0x04,0x18,0x04,0x78,0x00},
00103 {0x00,0x7C,0x08,0x04,0x7C,0x00,0x00},
00104 {0x00,0x38,0x44,0x44,0x38,0x00,0x00},
00105 {0x00,0xFC,0x24,0x24,0x18,0x00,0x00},
00106 {0x00,0x18,0x24,0x24,0xFC,0x00,0x00},
00107 {0x00,0x00,0x7C,0x08,0x04,0x00,0x00},
00108 {0x00,0x48,0x54,0x54,0x24,0x00,0x00},
00109 {0x00,0x04,0x7F,0x44,0x00,0x00,0x00},
00110 {0x00,0x3C,0x40,0x40,0x7C,0x00,0x00},
00111 {0x00,0x1C,0x20,0x40,0x20,0x1C,0x00},
00112 {0x00,0x3C,0x40,0x30,0x40,0x3C,0x00},
00113 {0x00,0x44,0x28,0x10,0x28,0x44,0x00},
00114 {0x00,0x1C,0xA0,0xA0,0x7C,0x00,0x00},
00115 {0x00,0x44,0x64,0x54,0x4C,0x44,0x00},
00116 {0x00,0x08,0x36,0x41,0x00,0x00,0x00},
00117 {0x00,0x00,0x7F,0x00,0x00,0x00,0x00},
00118 {0x00,0x41,0x36,0x08,0x00,0x00,0x00},
00119 {0x00,0x02,0x01,0x01,0x02,0x01,0x00},
00120 {0x00,0x02,0x05,0x05,0x02,0x00,0x00}
00121 };
00122 /*
00123 const uint8_t font_8x8[][8] = {
00124 {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},
00125 {0x00,0x00,0x5F,0x00,0x00,0x00,0x00,0x00},
00126 {0x00,0x00,0x07,0x00,0x07,0x00,0x00,0x00},
00127 {0x00,0x14,0x7F,0x14,0x7F,0x14,0x00,0x00},
00128 {0x00,0x24,0x2A,0x7F,0x2A,0x12,0x00,0x00},
00129 {0x00,0x23,0x13,0x08,0x64,0x62,0x00,0x00},
00130 {0x00,0x36,0x49,0x55,0x22,0x50,0x00,0x00},
00131 {0x00,0x00,0x05,0x03,0x00,0x00,0x00,0x00},
00132 {0x00,0x1C,0x22,0x41,0x00,0x00,0x00,0x00},
00133 {0x00,0x41,0x22,0x1C,0x00,0x00,0x00,0x00},
00134 {0x00,0x08,0x2A,0x1C,0x2A,0x08,0x00,0x00},
00135 {0x00,0x08,0x08,0x3E,0x08,0x08,0x00,0x00},
00136 {0x00,0xA0,0x60,0x00,0x00,0x00,0x00,0x00},
00137 {0x00,0x08,0x08,0x08,0x08,0x08,0x00,0x00},
00138 {0x00,0x60,0x60,0x00,0x00,0x00,0x00,0x00},
00139 {0x00,0x20,0x10,0x08,0x04,0x02,0x00,0x00},
00140 {0x00,0x3E,0x51,0x49,0x45,0x3E,0x00,0x00},
00141 {0x00,0x00,0x42,0x7F,0x40,0x00,0x00,0x00},
00142 {0x00,0x62,0x51,0x49,0x49,0x46,0x00,0x00},
00143 {0x00,0x22,0x41,0x49,0x49,0x36,0x00,0x00},
00144 {0x00,0x18,0x14,0x12,0x7F,0x10,0x00,0x00},
00145 {0x00,0x27,0x45,0x45,0x45,0x39,0x00,0x00},
00146 {0x00,0x3C,0x4A,0x49,0x49,0x30,0x00,0x00},
00147 {0x00,0x01,0x71,0x09,0x05,0x03,0x00,0x00},
00148 {0x00,0x36,0x49,0x49,0x49,0x36,0x00,0x00},
00149 {0x00,0x06,0x49,0x49,0x29,0x1E,0x00,0x00},
00150 {0x00,0x00,0x36,0x36,0x00,0x00,0x00,0x00},
00151 {0x00,0x00,0xAC,0x6C,0x00,0x00,0x00,0x00},
00152 {0x00,0x08,0x14,0x22,0x41,0x00,0x00,0x00},
00153 {0x00,0x14,0x14,0x14,0x14,0x14,0x00,0x00},
00154 {0x00,0x41,0x22,0x14,0x08,0x00,0x00,0x00},
00155 {0x00,0x02,0x01,0x51,0x09,0x06,0x00,0x00},
00156 {0x00,0x32,0x49,0x79,0x41,0x3E,0x00,0x00},
```

```
00157 {0x00,0x7E,0x09,0x09,0x09,0x7E,0x00,0x00},
00158 {0x00,0x7F,0x49,0x49,0x49,0x36,0x00,0x00},
00159 {0x00,0x3E,0x41,0x41,0x41,0x22,0x00,0x00},
00160 {0x00,0x7F,0x41,0x41,0x22,0x1C,0x00,0x00},
00161 {0x00,0x7F,0x49,0x49,0x49,0x41,0x00,0x00},
00162 {0x00,0x7F,0x09,0x09,0x09,0x01,0x00,0x00},
00163 {0x00,0x3E,0x41,0x41,0x51,0x72,0x00,0x00},
00164 {0x00,0x7F,0x08,0x08,0x08,0x7F,0x00,0x00},
00165 {0x00,0x41,0x7F,0x41,0x00,0x00,0x00,0x00},
00166 {0x00,0x20,0x40,0x41,0x3F,0x01,0x00,0x00},
00167 {0x00,0x7F,0x08,0x14,0x22,0x41,0x00,0x00},
00168 {0x00,0x7F,0x40,0x40,0x40,0x40,0x00,0x00},
00169 {0x00,0x7F,0x02,0x0C,0x02,0x7F,0x00,0x00},
00170 {0x00,0x7F,0x04,0x08,0x10,0x7F,0x00,0x00},
00171 {0x00,0x3E,0x41,0x41,0x41,0x3E,0x00,0x00},
00172 {0x00,0x7F,0x09,0x09,0x09,0x06,0x00,0x00},
00173 {0x00,0x3E,0x41,0x51,0x21,0x5E,0x00,0x00},
00174 {0x00,0x7F,0x09,0x19,0x29,0x46,0x00,0x00},
00175 {0x00,0x26,0x49,0x49,0x49,0x32,0x00,0x00},
00176 {0x00,0x01,0x01,0x7F,0x01,0x01,0x00,0x00},
00177 {0x00,0x3F,0x40,0x40,0x40,0x3F,0x00,0x00},
00178 {0x00,0x1F,0x20,0x40,0x20,0x1F,0x00,0x00},
00179 {0x00,0x3F,0x40,0x38,0x40,0x3F,0x00,0x00},
00180 {0x00,0x63,0x14,0x08,0x14,0x63,0x00,0x00},
00181 {0x00,0x03,0x04,0x78,0x04,0x03,0x00,0x00},
00182 {0x00,0x61,0x51,0x49,0x45,0x43,0x00,0x00},
00183 {0x00,0x7F,0x41,0x41,0x00,0x00,0x00,0x00},
00184 {0x00,0x02,0x04,0x08,0x10,0x20,0x00,0x00},
00185 {0x00,0x41,0x41,0x7F,0x00,0x00,0x00,0x00},
00186 {0x00,0x04,0x02,0x01,0x02,0x04,0x00,0x00},
00187 {0x00,0x80,0x80,0x80,0x80,0x80,0x00,0x00},
00188 {0x00,0x01,0x02,0x04,0x00,0x00,0x00,0x00},
00189 {0x00,0x20,0x54,0x54,0x54,0x78,0x00,0x00},
00190 {0x00,0x7F,0x48,0x44,0x44,0x38,0x00,0x00},
00191 {0x00,0x38,0x44,0x44,0x28,0x00,0x00,0x00},
00192 {0x00,0x38,0x44,0x44,0x48,0x7F,0x00,0x00},
00193 {0x00,0x38,0x54,0x54,0x54,0x18,0x00,0x00},
00194 {0x00,0x08,0x7E,0x09,0x02,0x00,0x00,0x00},
00195 {0x00,0x18,0xA4,0xA4,0xA4,0x7C,0x00,0x00},
00196 {0x00,0x7F,0x08,0x04,0x04,0x78,0x00,0x00},
00197 {0x00,0x00,0x7D,0x00,0x00,0x00,0x00,0x00},
00198 {0x00,0x80,0x84,0x7D,0x00,0x00,0x00,0x00},
00199 {0x00,0x7F,0x10,0x28,0x44,0x00,0x00,0x00},
00200 {0x00,0x41,0x7F,0x40,0x00,0x00,0x00,0x00},
00201 {0x00,0x7C,0x04,0x18,0x04,0x78,0x00,0x00},
00202 {0x00,0x7C,0x08,0x04,0x7C,0x00,0x00,0x00},
00203 {0x00,0x38,0x44,0x44,0x38,0x00,0x00,0x00},
00204 {0x00,0xFC,0x24,0x24,0x18,0x00,0x00,0x00},
00205 {0x00,0x18,0x24,0x24,0xFC,0x00,0x00,0x00},
00206 {0x00,0x00,0x7C,0x08,0x04,0x00,0x00,0x00},
00207 {0x00,0x48,0x54,0x54,0x24,0x00,0x00,0x00},
00208 {0x00,0x04,0x7F,0x44,0x00,0x00,0x00,0x00},
00209 {0x00,0x3C,0x40,0x40,0x7C,0x00,0x00,0x00},
00210 {0x00,0x1C,0x20,0x40,0x20,0x1C,0x00,0x00},
00211 {0x00,0x3C,0x40,0x30,0x40,0x3C,0x00,0x00},
00212 {0x00,0x44,0x28,0x10,0x28,0x44,0x00,0x00},
00213 {0x00,0x1C,0xA0,0xA0,0x7C,0x00,0x00,0x00},
00214 {0x00,0x44,0x64,0x54,0x4C,0x44,0x00,0x00},
00215 {0x00,0x08,0x36,0x41,0x00,0x00,0x00,0x00},
00216 {0x00,0x00,0x7F,0x00,0x00,0x00,0x00,0x00},
00217 {0x00,0x41,0x36,0x08,0x00,0x00,0x00,0x00},
00218 {0x00,0x02,0x01,0x01,0x02,0x01,0x00,0x00},
00219 {0x00,0x02,0x05,0x05,0x02,0x00,0x00,0x00}
00220 };*/
00221
00222 #endif
```

7.11 Core/Inc/main.h File Reference

: Header for main.c file. This file contains the common defines of the application.

```
#include "stm32f0xx_hal.h"
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <ctype.h>
```

```
#include <math.h>
#include "structs.h"
```

Macros

- #define **max**(a, b) ((a)>(b) ? (a) : (b))
- #define **min**(a, b) ((a)<(b) ? (a) : (b))
- #define **clip**(a, b, c) (max(min((a), (b)), (c)))
- #define **__F_TO_C**(f) ((5∗((f)-32))/9)
- #define **__C_TO_F**(c) (32+(((c)∗9)/5))
- #define **__ROUND5**(x) ((((x)+2)/5)∗5)
- #define **__F_TO_C_FLOAT**(f) (((f)-32.0f)/1.8f)
- #define **__C_TO_F_FLOAT**(c) (32.0f+((c)∗1.8f))
- #define **__READ_UP_SW**() (HAL_GPIO_ReadPin(UP_BUTTON_GPIO_Port, UP_BUTTON_Pin))
- #define **__READ_DOWN_SW**() (HAL_GPIO_ReadPin(DOWN_BUTTON_GPIO_Port, DOWN_BUTTON_↩ Pin))
- #define **__READ_ENTER_SW**() (HAL_GPIO_ReadPin(ENTER_BUTTON_GPIO_Port, ENTER_BUTTON↩ _Pin))
- #define **__READ_LEFT_ACTIVATE_SW**() (HAL_GPIO_ReadPin(LEFT_ACTIVATE_BUTTON_GPIO_Port, LEFT_ACTIVATE_BUTTON_Pin))
- #define **__READ_RIGHT_ACTIVATE_SW**() (HAL_GPIO_ReadPin(RIGHT_ACTIVATE_BUTTON_GPIO_↩ Port, RIGHT_ACTIVATE_BUTTON_Pin))
- #define **__READ_BOTTOM_TRAVEL_SW**() (HAL_GPIO_ReadPin(BOTTOM_TRAVEL_SWITCH_GPIO_↩ Port, BOTTOM_TRAVEL_SWITCH_Pin))
- #define **__READ_TOP_TRAVEL_SW**() (HAL_GPIO_ReadPin(TOP_TRAVEL_SWITCH_GPIO_Port, TOP↩ _TRAVEL_SWITCH_Pin))
- #define **__READ_PLATTER_SW**() (HAL_GPIO_ReadPin(PLATTER_SWITCH_GPIO_Port, PLATTER_↩ SWITCH_Pin))
- #define **__WRITE_THERMO_TOP1_CS**(x) (HAL_GPIO_WritePin(CS_THERMOCOUPLE_TOP1_GPIO_↩ Port, CS_THERMOCOUPLE_TOP1_Pin, (x)))
- #define **__WRITE_THERMO_TOP2_CS**(x) (HAL_GPIO_WritePin(CS_THERMOCOUPLE_TOP2_GPIO_↩ Port, CS_THERMOCOUPLE_TOP2_Pin, (x)))
- #define **__WRITE_THERMO_BOTTOM1_CS**(x) (HAL_GPIO_WritePin(CS_THERMOCOUPLE_BOTTOM1↩ _GPIO_Port, CS_THERMOCOUPLE_BOTTOM1_Pin, (x)))
- #define **__WRITE_THERMO_BOTTOM2_CS**(x) (HAL_GPIO_WritePin(CS_THERMOCOUPLE_BOTTOM2↩ _GPIO_Port, CS_THERMOCOUPLE_BOTTOM2_Pin, (x)))
- #define **__WRITE_SCREEN_CS**(x) (HAL_GPIO_WritePin(SCREEN_CS_GPIO_Port, SCREEN_CS_Pin, (x)))
- #define **__WRITE_SCREEN_DC**(x) (HAL_GPIO_WritePin(SCREEN_DATASEL_GPIO_Port, SCREEN_↩ DATASEL_Pin, (x)))
- #define **__WRITE_SCREEN_RESET**(x) (HAL_GPIO_WritePin(SCREEN_RESET_GPIO_Port, SCREEN_↩ RESET_Pin, (x)))
- #define **__WRITE_BLUE_LED**(x) (HAL_GPIO_WritePin(BLUE_LED_PIN_GPIO_Port, BLUE_LED_PIN_↩ Pin, (x)))
- #define **__WRITE_WHITE_LED**(x) (HAL_GPIO_WritePin(WHITE_LED_PIN_GPIO_Port, WHITE_LED_↩ PIN_Pin, (x)))
- #define **__TOGGLE_BLUE_LED**() (HAL_GPIO_TogglePin(BLUE_LED_PIN_GPIO_Port, BLUE_LED_PIN↩ _Pin))
- #define **__TOGGLE_WHITE_LED**() (HAL_GPIO_TogglePin(WHITE_LED_PIN_GPIO_Port, WHITE_LED↩ _PIN_Pin))
- #define **__WRITE_TOP_PLATTER_HEAT**(x) (HAL_GPIO_WritePin(TOP_PLATTER_HEAT_GPIO_Port, TOP_PLATTER_HEAT_Pin, (x)))
- #define **__WRITE_BOTTOM_PLATTER_HEAT**(x) (HAL_GPIO_WritePin(BOTTOM_PLATTER_HEAT_↩ GPIO_Port, BOTTOM_PLATTER_HEAT_Pin, (x)))

- #define **__READ_TOP_SSR**() (HAL_GPIO_ReadPin(TOP_PLATTER_HEAT_GPIO_Port, TOP_PLATTER↩ _HEAT_Pin))
- #define **__READ_BOTTOM_SSR**() (HAL_GPIO_ReadPin(BOTTOM_PLATTER_HEAT_GPIO_Port, BOTTOM↩ _PLATTER_HEAT_Pin))
- #define **BUZZER_PERIOD** (CLOCK_FREQ / BUZZER_FREQ)
- #define **CLOCK_FREQ** 48000000
- #define **DEAD_TIME** 25
- #define **PWM_FREQ** 1000
- #define **PWM_PERIOD** (CLOCK_FREQ / (2∗PWM_FREQ))
- #define **BUZZER_FREQ** 2400
- #define **DOWN_BUTTON_Pin** GPIO_PIN_13
- #define **DOWN_BUTTON_GPIO_Port** GPIOC
- #define **ENTER_BUTTON_Pin** GPIO_PIN_14
- #define **ENTER_BUTTON_GPIO_Port** GPIOC
- #define **UP_BUTTON_Pin** GPIO_PIN_15
- #define **UP_BUTTON_GPIO_Port** GPIOC
- #define **VBUS_SENSE_Pin** GPIO_PIN_0
- #define **VBUS_SENSE_GPIO_Port** GPIOA
- #define **OPAMP_VOUT_Pin** GPIO_PIN_1
- #define **OPAMP_VOUT_GPIO_Port** GPIOA
- #define **SCREEN_RESET_Pin** GPIO_PIN_5
- #define **SCREEN_RESET_GPIO_Port** GPIOA
- #define **BLUE_LED_PIN_Pin** GPIO_PIN_6
- #define **BLUE_LED_PIN_GPIO_Port** GPIOA
- #define **RIGHT_ACTIVATE_BUTTON_Pin** GPIO_PIN_1
- #define **RIGHT_ACTIVATE_BUTTON_GPIO_Port** GPIOB
- #define **WHITE_LED_PIN_Pin** GPIO_PIN_2
- #define **WHITE_LED_PIN_GPIO_Port** GPIOB
- #define **LEFT_ACTIVATE_BUTTON_Pin** GPIO_PIN_10
- #define **LEFT_ACTIVATE_BUTTON_GPIO_Port** GPIOB
- #define **SCREEN_CS_Pin** GPIO_PIN_12
- #define **SCREEN_CS_GPIO_Port** GPIOB
- #define **SCREEN_DATASEL_Pin** GPIO_PIN_14
- #define **SCREEN_DATASEL_GPIO_Port** GPIOB
- #define **PLATTER_SWITCH_Pin** GPIO_PIN_10
- #define **PLATTER_SWITCH_GPIO_Port** GPIOA
- #define **TOP_PLATTER_HEAT_Pin** GPIO_PIN_11
- #define **TOP_PLATTER_HEAT_GPIO_Port** GPIOA
- #define **BOTTOM_PLATTER_HEAT_Pin** GPIO_PIN_12
- #define **BOTTOM_PLATTER_HEAT_GPIO_Port** GPIOA
- #define **TOP_TRAVEL_SWITCH_Pin** GPIO_PIN_6
- #define **TOP_TRAVEL_SWITCH_GPIO_Port** GPIOF
- #define **BOTTOM_TRAVEL_SWITCH_Pin** GPIO_PIN_7
- #define **BOTTOM_TRAVEL_SWITCH_GPIO_Port** GPIOF
- #define **CS_THERMOCOUPLE_BOTTOM2_Pin** GPIO_PIN_6
- #define **CS_THERMOCOUPLE_BOTTOM2_GPIO_Port** GPIOB
- #define **CS_THERMOCOUPLE_BOTTOM1_Pin** GPIO_PIN_7
- #define **CS_THERMOCOUPLE_BOTTOM1_GPIO_Port** GPIOB
- #define **CS_THERMOCOUPLE_TOP2_Pin** GPIO_PIN_8
- #define **CS_THERMOCOUPLE_TOP2_GPIO_Port** GPIOB
- #define **CS_THERMOCOUPLE_TOP1_Pin** GPIO_PIN_9
- #define **CS_THERMOCOUPLE_TOP1_GPIO_Port** GPIOB

Functions

- void HAL_TIM_MspPostInit (TIM_HandleTypeDef *htim)
- void Error_Handler (void)

  This function is executed in case of error occurrence.

Variables

- ADC_HandleTypeDef **hadc**
- SPI_HandleTypeDef **hspi1**
- I2C_HandleTypeDef **hi2c2**
- TIM_HandleTypeDef **htim1**
- TIM_HandleTypeDef **htim2**
- RTC_HandleTypeDef **hrtc**
- UART_HandleTypeDef **huart2**
- MenuItem * **current_menu**
- MenuItem **status_menu**
- MenuItem **debug_menu**
- MenuItem **lifetime_menu**
- MenuItem **jog_menu**
- Press **press**
- uint32_t **ticks**
- Button **menu_up_button**
- Button **menu_down_button**
- Button **menu_enter_button**
- Button **activate_left_button**
- Button **activate_right_button**
- Button **press_bottom_limit**
- Button **press_top_limit**
- Button **tray_interlock**

7.11.1 Detailed Description

: Header for main.c file. This file contains the common defines of the application.

Attention

7.11.2 Function Documentation

7.11.2.1 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

Return values

None	

7.11.2.2 HAL_TIM_MspPostInit()

```
void HAL_TIM_MspPostInit (
            TIM_HandleTypeDef * htim )
```

TIM1 GPIO Configuration PA7 ------> TIM1_CH1N PB0 ------> TIM1_CH2N PA8 ------> TIM1_CH1 PA9 ------> TIM1_CH2

TIM2 GPIO Configuration PB11 ------> TIM2_CH4

TIM1 GPIO Configuration PA7 ------> TIM1_CH1N PB0 ------> TIM1_CH2N PA8 ------> TIM1_CH1 PA9 ------> TIM1_CH2

TIM2 GPIO Configuration PB11 ------> TIM2_CH4

7.12 main.h

Go to the documentation of this file.

```
00001 /* USER CODE BEGIN Header */
00019 /* USER CODE END Header */
00020
00021 /* Define to prevent recursive inclusion -------------------------------------*/
00022 #ifndef __MAIN_H
00023 #define __MAIN_H
00024
00025 #ifdef __cplusplus
00026 extern "C" {
00027 #endif
00028
00029 /* Includes ------------------------------------------------------------------*/
00030 #include "stm32f0xx_hal.h"
00031
00032 /* Private includes ----------------------------------------------------------*/
00033 /* USER CODE BEGIN Includes */
00034 #include <string.h>
00035 #include <stdio.h>
00036 #include <stdlib.h>
00037 #include <stdbool.h>
00038 #include <ctype.h>
00039 #include <math.h>
00040
00041 #include "structs.h"
00042
00043 //#define CYCLE_MODE
00044
00045 /* USER CODE END Includes */
00046
00047 /* Exported types ------------------------------------------------------------*/
00048 /* USER CODE BEGIN ET */
00049
00050 /* USER CODE END ET */
00051
00052 /* Exported constants --------------------------------------------------------*/
00053 /* USER CODE BEGIN EC */
00054
00055 extern ADC_HandleTypeDef hadc;
00056 extern SPI_HandleTypeDef hspi1;
00057 extern I2C_HandleTypeDef hi2c2;
00058 extern TIM_HandleTypeDef htim1;
00059 extern TIM_HandleTypeDef htim2;
00060 extern RTC_HandleTypeDef hrtc;
00061 extern UART_HandleTypeDef huart2;
00062
00063 extern MenuItem* current_menu;
```

```
00064 extern MenuItem status_menu;
00065 extern MenuItem debug_menu;
00066 extern MenuItem lifetime_menu;
00067
00068 extern MenuItem jog_menu;
00069
00070 extern Press press;
00071 extern uint32_t ticks;
00072
00073 extern Button menu_up_button;
00074 extern Button menu_down_button;
00075 extern Button menu_enter_button;
00076
00077 extern Button activate_left_button;
00078 extern Button activate_right_button;
00079
00080 extern Button press_bottom_limit;
00081 extern Button press_top_limit;
00082 extern Button tray_interlock;
00083
00084 /* USER CODE END EC */
00085
00086 /* Exported macro ------------------------------------------------------------*/
00087 /* USER CODE BEGIN EM */
00088
00089 #define max(a,b) ((a)>(b) ? (a) : (b))
00090 #define min(a,b) ((a)<(b) ? (a) : (b))
00091 #define clip(a,b,c) (max(min((a), (b)), (c))) // value, max, min
00092
00093 #define __F_TO_C(f) ((5*((f)-32))/9)
00094 #define __C_TO_F(c) (32+(((c)*9)/5))
00095 #define __ROUND5(x) ((((x)+2)/5)*5)
00096
00097 #define __F_TO_C_FLOAT(f) (((f)-32.0f)/1.8f)
00098 #define __C_TO_F_FLOAT(c) (32.0f+((c)*1.8f))
00099
00100 #define __READ_UP_SW() (HAL_GPIO_ReadPin(UP_BUTTON_GPIO_Port, UP_BUTTON_Pin))
00101 #define __READ_DOWN_SW() (HAL_GPIO_ReadPin(DOWN_BUTTON_GPIO_Port, DOWN_BUTTON_Pin))
00102 #define __READ_ENTER_SW() (HAL_GPIO_ReadPin(ENTER_BUTTON_GPIO_Port, ENTER_BUTTON_Pin))
00103
00104 #define __READ_LEFT_ACTIVATE_SW() (HAL_GPIO_ReadPin(LEFT_ACTIVATE_BUTTON_GPIO_Port,
      LEFT_ACTIVATE_BUTTON_Pin))
00105 #define __READ_RIGHT_ACTIVATE_SW() (HAL_GPIO_ReadPin(RIGHT_ACTIVATE_BUTTON_GPIO_Port,
      RIGHT_ACTIVATE_BUTTON_Pin))
00106
00107 #define __READ_BOTTOM_TRAVEL_SW() (HAL_GPIO_ReadPin(BOTTOM_TRAVEL_SWITCH_GPIO_Port,
      BOTTOM_TRAVEL_SWITCH_Pin))
00108 #define __READ_TOP_TRAVEL_SW() (HAL_GPIO_ReadPin(TOP_TRAVEL_SWITCH_GPIO_Port, TOP_TRAVEL_SWITCH_Pin))
00109 #define __READ_PLATTER_SW() (HAL_GPIO_ReadPin(PLATTER_SWITCH_GPIO_Port, PLATTER_SWITCH_Pin))
00110
00111 #define __WRITE_THERMO_TOP1_CS(x) (HAL_GPIO_WritePin(CS_THERMOCOUPLE_TOP1_GPIO_Port,
      CS_THERMOCOUPLE_TOP1_Pin, (x)))
00112 #define __WRITE_THERMO_TOP2_CS(x) (HAL_GPIO_WritePin(CS_THERMOCOUPLE_TOP2_GPIO_Port,
      CS_THERMOCOUPLE_TOP2_Pin, (x)))
00113 #define __WRITE_THERMO_BOTTOM1_CS(x) (HAL_GPIO_WritePin(CS_THERMOCOUPLE_BOTTOM1_GPIO_Port,
      CS_THERMOCOUPLE_BOTTOM1_Pin, (x)))
00114 #define __WRITE_THERMO_BOTTOM2_CS(x) (HAL_GPIO_WritePin(CS_THERMOCOUPLE_BOTTOM2_GPIO_Port,
      CS_THERMOCOUPLE_BOTTOM2_Pin, (x)))
00115
00116 #define __WRITE_SCREEN_CS(x) (HAL_GPIO_WritePin(SCREEN_CS_GPIO_Port, SCREEN_CS_Pin, (x)))
00117 #define __WRITE_SCREEN_DC(x) (HAL_GPIO_WritePin(SCREEN_DATASEL_GPIO_Port, SCREEN_DATASEL_Pin, (x)))
00118 #define __WRITE_SCREEN_RESET(x) (HAL_GPIO_WritePin(SCREEN_RESET_GPIO_Port, SCREEN_RESET_Pin, (x)))
00119
00120 #define __WRITE_BLUE_LED(x) (HAL_GPIO_WritePin(BLUE_LED_PIN_GPIO_Port, BLUE_LED_PIN_Pin, (x)))
00121 #define __WRITE_WHITE_LED(x) (HAL_GPIO_WritePin(WHITE_LED_PIN_GPIO_Port, WHITE_LED_PIN_Pin, (x)))
00122 #define __TOGGLE_BLUE_LED() (HAL_GPIO_TogglePin(BLUE_LED_PIN_GPIO_Port, BLUE_LED_PIN_Pin))
00123 #define __TOGGLE_WHITE_LED() (HAL_GPIO_TogglePin(WHITE_LED_PIN_GPIO_Port, WHITE_LED_PIN_Pin))
00124
00125
00126 #define __WRITE_TOP_PLATTER_HEAT(x) (HAL_GPIO_WritePin(TOP_PLATTER_HEAT_GPIO_Port,
      TOP_PLATTER_HEAT_Pin, (x)))
00127 #define __WRITE_BOTTOM_PLATTER_HEAT(x) (HAL_GPIO_WritePin(BOTTOM_PLATTER_HEAT_GPIO_Port,
      BOTTOM_PLATTER_HEAT_Pin, (x)))
00128
00129 #define __READ_TOP_SSR() (HAL_GPIO_ReadPin(TOP_PLATTER_HEAT_GPIO_Port, TOP_PLATTER_HEAT_Pin))
00130 #define __READ_BOTTOM_SSR() (HAL_GPIO_ReadPin(BOTTOM_PLATTER_HEAT_GPIO_Port, BOTTOM_PLATTER_HEAT_Pin))
00131
00132 /* USER CODE END EM */
00133
00134 void HAL_TIM_MspPostInit(TIM_HandleTypeDef *htim);
00135
00136 /* Exported functions prototypes ---------------------------------------------*/
00137 void Error_Handler(void);
00138
00139 /* USER CODE BEGIN EFP */
00140
00141 /* USER CODE END EFP */
```

```
00142
00143 /* Private defines -----------------------------------------------------------*/
00144 #define BUZZER_PERIOD (CLOCK_FREQ / BUZZER_FREQ)
00145 #define CLOCK_FREQ 48000000
00146 #define DEAD_TIME 25
00147 #define PWM_FREQ 1000
00148 #define PWM_PERIOD (CLOCK_FREQ / (2*PWM_FREQ))
00149 #define BUZZER_FREQ 2400
00150 #define DOWN_BUTTON_Pin GPIO_PIN_13
00151 #define DOWN_BUTTON_GPIO_Port GPIOC
00152 #define ENTER_BUTTON_Pin GPIO_PIN_14
00153 #define ENTER_BUTTON_GPIO_Port GPIOC
00154 #define UP_BUTTON_Pin GPIO_PIN_15
00155 #define UP_BUTTON_GPIO_Port GPIOC
00156 #define VBUS_SENSE_Pin GPIO_PIN_0
00157 #define VBUS_SENSE_GPIO_Port GPIOA
00158 #define OPAMP_VOUT_Pin GPIO_PIN_1
00159 #define OPAMP_VOUT_GPIO_Port GPIOA
00160 #define SCREEN_RESET_Pin GPIO_PIN_5
00161 #define SCREEN_RESET_GPIO_Port GPIOA
00162 #define BLUE_LED_PIN_Pin GPIO_PIN_6
00163 #define BLUE_LED_PIN_GPIO_Port GPIOA
00164 #define RIGHT_ACTIVATE_BUTTON_Pin GPIO_PIN_1
00165 #define RIGHT_ACTIVATE_BUTTON_GPIO_Port GPIOB
00166 #define WHITE_LED_PIN_Pin GPIO_PIN_2
00167 #define WHITE_LED_PIN_GPIO_Port GPIOB
00168 #define LEFT_ACTIVATE_BUTTON_Pin GPIO_PIN_10
00169 #define LEFT_ACTIVATE_BUTTON_GPIO_Port GPIOB
00170 #define SCREEN_CS_Pin GPIO_PIN_12
00171 #define SCREEN_CS_GPIO_Port GPIOB
00172 #define SCREEN_DATASEL_Pin GPIO_PIN_14
00173 #define SCREEN_DATASEL_GPIO_Port GPIOB
00174 #define PLATTER_SWITCH_Pin GPIO_PIN_10
00175 #define PLATTER_SWITCH_GPIO_Port GPIOA
00176 #define TOP_PLATTER_HEAT_Pin GPIO_PIN_11
00177 #define TOP_PLATTER_HEAT_GPIO_Port GPIOA
00178 #define BOTTOM_PLATTER_HEAT_Pin GPIO_PIN_12
00179 #define BOTTOM_PLATTER_HEAT_GPIO_Port GPIOA
00180 #define TOP_TRAVEL_SWITCH_Pin GPIO_PIN_6
00181 #define TOP_TRAVEL_SWITCH_GPIO_Port GPIOF
00182 #define BOTTOM_TRAVEL_SWITCH_Pin GPIO_PIN_7
00183 #define BOTTOM_TRAVEL_SWITCH_GPIO_Port GPIOF
00184 #define CS_THERMOCOUPLE_BOTTOM2_Pin GPIO_PIN_6
00185 #define CS_THERMOCOUPLE_BOTTOM2_GPIO_Port GPIOB
00186 #define CS_THERMOCOUPLE_BOTTOM1_Pin GPIO_PIN_7
00187 #define CS_THERMOCOUPLE_BOTTOM1_GPIO_Port GPIOB
00188 #define CS_THERMOCOUPLE_TOP2_Pin GPIO_PIN_8
00189 #define CS_THERMOCOUPLE_TOP2_GPIO_Port GPIOB
00190 #define CS_THERMOCOUPLE_TOP1_Pin GPIO_PIN_9
00191 #define CS_THERMOCOUPLE_TOP1_GPIO_Port GPIOB
00192
00193 /* USER CODE BEGIN Private defines */
00194
00195 /* USER CODE END Private defines */
00196
00197 #ifdef __cplusplus
00198 }
00199 #endif
00200
00201 #endif /* __MAIN_H */
```

7.13 Core/Inc/menu.h File Reference

760 Pizza Press screen menus

```
#include "main.h"
#include "config.h"
#include "structs.h"
#include "SSD1306.h"
```

Macros

- #define **MENU_TIMEOUT** 7000ul

Functions

- void **init_menus** (void)

  Initialize menu linkage.
- void **link_menus** (MenuItem *parent, MenuItem *child)

  operation to link parent and child menu items
- void set_row (const char *str, uint8_t rownum, uint8_t font)

  set RAM text buffer for screen
- HAL_StatusTypeDef write_row (uint8_t rownum)

  write RAM text buffer row to SSD1306
- MenuItem * menu_up (MenuItem *)

  Called when menu up button is pressed.
- MenuItem * menu_down (MenuItem *)

  Called when menu down button is pressed.
- MenuItem * menu_enter (MenuItem *)

  Called when menu enter button is pressed.
- HAL_StatusTypeDef **menu_return_home** (void)
- HAL_StatusTypeDef **debug_display** (MenuItem *)
- HAL_StatusTypeDef **lifetime_display** (MenuItem *)
- HAL_StatusTypeDef **status_display** (MenuItem *)
- HAL_StatusTypeDef **thermocouple_readout** (void)
- HAL_StatusTypeDef **generic_display** (MenuItem *)
- HAL_StatusTypeDef **temperature_display** (MenuItem *)
- HAL_StatusTypeDef **press_time_display** (MenuItem *)
- HAL_StatusTypeDef **manual_mode_display** (MenuItem *)
- HAL_StatusTypeDef **reset_display** (MenuItem *)
- HAL_StatusTypeDef **units_display** (MenuItem *)
- HAL_StatusTypeDef **jog_display** (MenuItem *)

7.13.1 Detailed Description

760 Pizza Press screen menus

Author

Aaron Yeiser

Date

2022-08-05

Copyright

7.13.2 Function Documentation

7.13.2.1 menu_down()

```
MenuItem * menu_down (
            MenuItem * item )
```

Called when menu down button is pressed.

Parameters

the	current menu item

Returns

the next menu item to go to

7.13.2.2 menu_enter()

```
MenuItem * menu_enter (
            MenuItem * item )
```

Called when menu enter button is pressed.

Parameters

the	current menu item

Returns

the next menu item to go to

7.13.2.3 menu_up()

```
MenuItem * menu_up (
            MenuItem * item )
```

Called when menu up button is pressed.

Parameters

the	current menu item

Returns

the next menu item to go to

7.13.2.4 set_row()

```
void set_row (
            const char * str,
            uint8_t rownum,
            uint8_t font )
```

set RAM text buffer for screen

Parameters

str	Text to write to the row
rownum	Row (0-7) to write to
font	0 is small, 1 is big

7.13.2.5 write_row()

```
HAL_StatusTypeDef write_row (
            uint8_t rownum )
```

write RAM text buffer row to SSD1306

Parameters

rownum	the row to write to (0-7)

Returns

HAL_StatusTypeDef the status of writing to the row

Note

this is a blocking function

7.14 menu.h

Go to the documentation of this file.

```
00001
00011 #ifndef INC_MENU_H_
00012 #define INC_MENU_H_
00013
00014 #include "main.h"
00015 #include "config.h"
00016 #include "structs.h"
00017 #include "SSD1306.h"
00018
00019 #define MENU_TIMEOUT 7000ul // 7 seconds
00020
00022 void init_menus(void);
00023
00025 void link_menus(MenuItem* parent, MenuItem* child);
00026
00033 void set_row(const char* str, uint8_t rownum, uint8_t font);
00034
00042 HAL_StatusTypeDef write_row(uint8_t rownum);
00043
00049 MenuItem* menu_up(MenuItem*);
00050
00056 MenuItem* menu_down(MenuItem*);
00057
00063 MenuItem* menu_enter(MenuItem*);
00064
00065 HAL_StatusTypeDef menu_return_home(void);
00066
00067 HAL_StatusTypeDef debug_display(MenuItem*);
00068 HAL_StatusTypeDef lifetime_display(MenuItem*);
00069 HAL_StatusTypeDef status_display(MenuItem*);
00070 HAL_StatusTypeDef thermocouple_readout(void);
00071
00072 // function to write data to the screen
```

```
00073 HAL_StatusTypeDef generic_display(MenuItem*);
00074 HAL_StatusTypeDef temperature_display(MenuItem*);
00075 HAL_StatusTypeDef press_time_display(MenuItem*);
00076 HAL_StatusTypeDef manual_mode_display(MenuItem*);
00077 HAL_StatusTypeDef reset_display(MenuItem*);
00078 HAL_StatusTypeDef units_display(MenuItem*);
00079 HAL_StatusTypeDef jog_display(MenuItem*);
00080
00081 #endif /* INC_MENU_H_ */
```

7.15 Core/Inc/SSD1306.h File Reference

760 Pizza Press SSD1306 display driver

```
#include "font_8x8.h"
#include "font_16x12.h"
#include <stdio.h>
#include <string.h>
#include "main.h"
```

Macros

- #define **SSD1306_128_64**
- #define **SSD1306_SLAVE_ADDR** 0b01111000
- #define **SSD1306_LCDWIDTH** 128
- #define **SSD1306_LCDHEIGHT** 64
- #define **SSD1306_SETCONTRAST** 0x81
- #define **SSD1306_DISPLAYALLON_RESUME** 0xA4
- #define **SSD1306_DISPLAYALLON** 0xA5
- #define **SSD1306_NORMALDISPLAY** 0xA6
- #define **SSD1306_INVERTDISPLAY** 0xA7
- #define **SSD1306_DISPLAYOFF** 0xAE
- #define **SSD1306_DISPLAYON** 0xAF
- #define **SSD1306_SETDISPLAYOFFSET** 0xD3
- #define **SSD1306_SETCOMPINS** 0xDA
- #define **SSD1306_SETVCOMDETECT** 0xDB
- #define **SSD1306_SETDISPLAYCLOCKDIV** 0xD5
- #define **SSD1306_SETPRECHARGE** 0xD9
- #define **SSD1306_SETMULTIPLEX** 0xA8
- #define **SSD1306_SETLOWCOLUMN** 0x00
- #define **SSD1306_SETHIGHCOLUMN** 0x10
- #define **SSD1306_SETSTARTLINE** 0x40
- #define **SSD1306_MEMORYMODE** 0x20
- #define **SSD1306_COLUMNADDR** 0x21
- #define **SSD1306_PAGEADDR** 0x22
- #define **SSD1306_COMSCANINC** 0xC0
- #define **SSD1306_COMSCANDEC** 0xC8
- #define **SSD1306_SEGREMAP** 0xA0
- #define **SSD1306_CHARGEPUMP** 0x8D
- #define **SSD1306_EXTERNALVCC** 0x1
- #define **SSD1306_SWITCHCAPVCC** 0x2
- #define **SSD1306_ACTIVATE_SCROLL** 0x2F
- #define **SSD1306_DEACTIVATE_SCROLL** 0x2E
- #define **SSD1306_SET_VERTICAL_SCROLL_AREA** 0xA3

- #define **SSD1306_RIGHT_HORIZONTAL_SCROLL** 0x26
- #define **SSD1306_LEFT_HORIZONTAL_SCROLL** 0x27
- #define **SSD1306_VERTICAL_AND_RIGHT_HORIZONTAL_SCROLL** 0x29
- #define **SSD1306_VERTICAL_AND_LEFT_HORIZONTAL_SCROLL** 0x2A
- #define **DATA_MODE** 0x40
- #define **ROWS** 64
- #define **COLUMNS** 128
- #define **PAGES** (ROWS / 8)
- #define **MAX_PAGE** (PAGES - 1)
- #define **MAX_ROW** (ROWS - 1)
- #define **MAX_COL** (COLUMNS - 1)
- #define **CHARS** (COLUMNS / FONT8x8_WIDTH)
- #define **COMMAND_MODE** 0x80
- #define **SET_COLUMN_ADDRESS** 0x21
- #define **SET_PAGE_ADDRESS** 0x22
- #define **FRAME_BUF_OFFSET** 13
- #define **FONT_8x8** 0
- #define **FONT_16x12_0** 1
- #define **FONT_16x12_1** 2
- #define **TIMEOUT_INIT** 400

Functions

- void **SSD1306_InitScreen** (SPI_HandleTypeDef ∗)
- void **SSD1306_writeString** (uint8_t col, const char ∗text)
- void **SSD1306_writeInt** (uint8_t col, int32_t num)
- HAL_StatusTypeDef **SSD1306_WriteRow** (uint8_t page)
- void **SSD1306_ClearBuf** ()
- HAL_StatusTypeDef **SSD1306_IsReady** ()
- void **SSD1306_setFont** (uint8_t font)
- void **SSD1306_setInvert** (uint8_t invert)
- void **SSD1306_clearDisplay** ()
- void **SSD1306_ResetI2C** ()
- HAL_StatusTypeDef **SSD1306_spiWrite** (uint8_t ∗buf, uint16_t num_bytes)
- HAL_StatusTypeDef **SSD1306_spiWriteDMA** (uint8_t ∗buf, uint16_t num_bytes)
- HAL_StatusTypeDef **SSD1306_writeFrameBufRow** (uint8_t page)
- void **SSD1306_writeCharToBuf** (uint8_t col, char chr)
- HAL_StatusTypeDef **SSD1306_command1** (uint8_t c)
- HAL_StatusTypeDef **SSD1306_sendCommand** (uint8_t command, uint8_t param1, uint8_t param2)
- HAL_StatusTypeDef **SSD1306_sendDataByte** (uint8_t data)
- HAL_StatusTypeDef **SSD1306_sendData** (uint8_t ∗data, uint16_t len)
- HAL_StatusTypeDef **SSD1306_setPageAddress** (uint8_t start, uint8_t end)
- HAL_StatusTypeDef **SSD1306_setColumnAddress** (uint8_t start, uint8_t end)
- void **SSD1306_SetupFrameBuf** ()

Variables

- SPI_HandleTypeDef ∗ **ssd1306_spi**

7.15.1 Detailed Description

760 Pizza Press SSD1306 display driver

Author

Austin Brown

Date

2022-08-05

Copyright

7.16 SSD1306.h

Go to the documentation of this file.

```
00001
00011 #ifndef SSD1306_SIMPLE_H
00012 #define SSD1306_SIMPLE_H
00013
00014 #include "font_8x8.h"
00015 #include "font_16x12.h"
00016 #include <stdio.h>
00017 #include <string.h>
00018 #include "main.h"
00019
00020
00021 #define SSD1306_128_64
00022 //   #define SSD1306_128_32
00023 //   #define SSD1306_96_16
00024 /*=========================================================================*/
00025
00026 #define SSD1306_SLAVE_ADDR 0b01111000
00027
00028 #if defined SSD1306_128_64 && defined SSD1306_128_32
00029   #error "Only one SSD1306 display can be specified at once in SSD1306.h"
00030 #endif
00031 #if !defined SSD1306_128_64 && !defined SSD1306_128_32 && !defined SSD1306_96_16
00032   #error "At least one SSD1306 display must be specified in SSD1306.h"
00033 #endif
00034
00035 #if defined SSD1306_128_64
00036   #define SSD1306_LCDWIDTH                  128
00037   #define SSD1306_LCDHEIGHT                 64
00038 #endif
00039 #if defined SSD1306_128_32
00040   #define SSD1306_LCDWIDTH                  128
00041   #define SSD1306_LCDHEIGHT                 32
00042 #endif
00043 #if defined SSD1306_96_16
00044   #define SSD1306_LCDWIDTH                  96
00045   #define SSD1306_LCDHEIGHT                 16
00046 #endif
00047
00048 #define SSD1306_SETCONTRAST 0x81
00049 #define SSD1306_DISPLAYALLON_RESUME 0xA4
00050 #define SSD1306_DISPLAYALLON 0xA5
00051 #define SSD1306_NORMALDISPLAY 0xA6
00052 #define SSD1306_INVERTDISPLAY 0xA7
00053 #define SSD1306_DISPLAYOFF 0xAE
00054 #define SSD1306_DISPLAYON 0xAF
00055
00056 #define SSD1306_SETDISPLAYOFFSET 0xD3
00057 #define SSD1306_SETCOMPINS 0xDA
00058
00059 #define SSD1306_SETVCOMDETECT 0xDB
00060
00061 #define SSD1306_SETDISPLAYCLOCKDIV 0xD5
```

```
00062 #define SSD1306_SETPRECHARGE 0xD9
00063
00064 #define SSD1306_SETMULTIPLEX 0xA8
00065
00066 #define SSD1306_SETLOWCOLUMN 0x00
00067 #define SSD1306_SETHIGHCOLUMN 0x10
00068
00069 #define SSD1306_SETSTARTLINE 0x40
00070
00071 #define SSD1306_MEMORYMODE 0x20
00072 #define SSD1306_COLUMNADDR 0x21
00073 #define SSD1306_PAGEADDR   0x22
00074
00075 #define SSD1306_COMSCANINC 0xC0
00076 #define SSD1306_COMSCANDEC 0xC8
00077
00078 #define SSD1306_SEGREMAP 0xA0
00079
00080 #define SSD1306_CHARGEPUMP 0x8D
00081
00082 #define SSD1306_EXTERNALVCC 0x1
00083 #define SSD1306_SWITCHCAPVCC 0x2
00084
00085 // Scrolling #defines
00086 #define SSD1306_ACTIVATE_SCROLL 0x2F
00087 #define SSD1306_DEACTIVATE_SCROLL 0x2E
00088 #define SSD1306_SET_VERTICAL_SCROLL_AREA 0xA3
00089 #define SSD1306_RIGHT_HORIZONTAL_SCROLL 0x26
00090 #define SSD1306_LEFT_HORIZONTAL_SCROLL 0x27
00091 #define SSD1306_VERTICAL_AND_RIGHT_HORIZONTAL_SCROLL 0x29
00092 #define SSD1306_VERTICAL_AND_LEFT_HORIZONTAL_SCROLL 0x2A
00093
00094
00095 #define DATA_MODE                  0x40
00096
00097
00098 // Display dimensions
00099 #define ROWS                       64
00100 #define COLUMNS                    128
00101 #define PAGES                      (ROWS / 8)
00102 #define MAX_PAGE                   (PAGES - 1)
00103 #define MAX_ROW                    (ROWS - 1)
00104 #define MAX_COL                    (COLUMNS - 1)
00105
00106 // Character dimensions 8x8 font
00107 #define CHARS                      (COLUMNS / FONT8x8_WIDTH)  // =
00108
00109 // Command and Datamode
00110 #define COMMAND_MODE               0x80 // continuation bit is set! Just means we could stream
      cotinous coomands
00111                                        // as is done in in triple comand mode
00112
00113
00114 #define SET_COLUMN_ADDRESS         0x21 // takes two bytes, start address and end address of display
      data RAM
00115 #define SET_PAGE_ADDRESS           0x22 // takes two bytes, start address and end address of display
      data RAM
00116
00117
00118 #define FRAME_BUF_OFFSET 13
00119
00120 #define FONT_8x8 0
00121 #define FONT_16x12_0 1
00122 #define FONT_16x12_1 2
00123
00124 #define TIMEOUT_INIT 400
00125
00126
00127 void SSD1306_InitScreen(SPI_HandleTypeDef*) ;
00128 void SSD1306_writeString(uint8_t col, const char* text) ;
00129 void SSD1306_writeInt( uint8_t col, int32_t num);
00130 HAL_StatusTypeDef SSD1306_WriteRow( uint8_t page );
00131 void SSD1306_ClearBuf();
00132 HAL_StatusTypeDef SSD1306_IsReady();
00133 void SSD1306_setFont( uint8_t font );
00134 void SSD1306_setInvert( uint8_t invert );
00135
00136 // internal functions
00137 void SSD1306_clearDisplay();
00138 void SSD1306_ResetI2C();
00139
00140 HAL_StatusTypeDef SSD1306_spiWrite( uint8_t *buf, uint16_t num_bytes);
00141 HAL_StatusTypeDef SSD1306_spiWriteDMA( uint8_t *buf, uint16_t num_bytes);
00142
00143
00144 //HAL_StatusTypeDef SSD1306_i2cWrite( uint8_t *buf, uint16_t num_bytes);
00145 //HAL_StatusTypeDef SSD1306_DMAi2cWrite( uint8_t *buf, uint16_t num_bytes);
```

```
00146
00147 HAL_StatusTypeDef SSD1306_writeFrameBufRow( uint8_t page );
00148 void SSD1306_writeCharToBuf( uint8_t col, char chr );
00149
00150 HAL_StatusTypeDef SSD1306_command1(uint8_t c);
00151 HAL_StatusTypeDef SSD1306_sendCommand(uint8_t command, uint8_t param1, uint8_t param2);
00152 HAL_StatusTypeDef SSD1306_sendDataByte(uint8_t data);
00153 HAL_StatusTypeDef SSD1306_sendData(uint8_t* data, uint16_t len);
00154
00155
00156 HAL_StatusTypeDef SSD1306_setPageAddress(uint8_t start, uint8_t end);
00157 HAL_StatusTypeDef SSD1306_setColumnAddress(uint8_t start, uint8_t end);
00158 // uint32_t SSD1306_writeChar(char chr);
00159 void SSD1306_SetupFrameBuf();
00160
00161
00162 //extern I2C_HandleTypeDef *ssd1306_i2c;
00163 extern SPI_HandleTypeDef *ssd1306_spi;
00164 //extern uint8_t framebuf[COLUMNS+FRAME_BUF_OFFSET];   // add control commands. Only a single row is
      buffered at a time.
00165 //extern int timeout_cnt;
00166 //extern uint8_t _char_width;
00167 //extern uint8_t _font;
00168
00169
00170 //char *convert(unsigned int num, int base);
00171
00172
00173
00174 #endif
```

7.17 Core/Inc/stm32f0xx_hal_conf.h File Reference

HAL configuration file.

```
#include "stm32f0xx_hal_rcc.h"
#include "stm32f0xx_hal_gpio.h"
#include "stm32f0xx_hal_exti.h"
#include "stm32f0xx_hal_dma.h"
#include "stm32f0xx_hal_cortex.h"
#include "stm32f0xx_hal_adc.h"
#include "stm32f0xx_hal_flash.h"
#include "stm32f0xx_hal_i2c.h"
#include "stm32f0xx_hal_iwdg.h"
#include "stm32f0xx_hal_pwr.h"
#include "stm32f0xx_hal_rtc.h"
#include "stm32f0xx_hal_spi.h"
#include "stm32f0xx_hal_tim.h"
#include "stm32f0xx_hal_uart.h"
```

Macros

- #define **HAL_MODULE_ENABLED**

  This is the list of modules to be used in the HAL driver.
- #define **HAL_ADC_MODULE_ENABLED**
- #define **HAL_IWDG_MODULE_ENABLED**
- #define **HAL_RTC_MODULE_ENABLED**
- #define **HAL_SPI_MODULE_ENABLED**
- #define **HAL_TIM_MODULE_ENABLED**
- #define **HAL_UART_MODULE_ENABLED**
- #define **HAL_CORTEX_MODULE_ENABLED**
- #define **HAL_DMA_MODULE_ENABLED**

- #define **HAL_FLASH_MODULE_ENABLED**
- #define **HAL_GPIO_MODULE_ENABLED**
- #define **HAL_EXTI_MODULE_ENABLED**
- #define **HAL_PWR_MODULE_ENABLED**
- #define **HAL_RCC_MODULE_ENABLED**
- #define **HAL_I2C_MODULE_ENABLED**
- #define HSE_VALUE ((uint32_t)8000000)

  Adjust the value of External High Speed oscillator (HSE) used in your application. This value is used by the RCC HAL module to compute the system frequency (when HSE is used as system clock source, directly or through the PLL).
- #define HSE_STARTUP_TIMEOUT ((uint32_t)100)

  In the following line adjust the External High Speed oscillator (HSE) Startup Timeout value.
- #define HSI_VALUE ((uint32_t)8000000)

  Internal High Speed oscillator (HSI) value. This value is used by the RCC HAL module to compute the system frequency (when HSI is used as system clock source, directly or through the PLL).
- #define HSI_STARTUP_TIMEOUT ((uint32_t)5000)

  In the following line adjust the Internal High Speed oscillator (HSI) Startup Timeout value.
- #define HSI14_VALUE ((uint32_t)14000000)

  Internal High Speed oscillator for ADC (HSI14) value.
- #define HSI48_VALUE ((uint32_t)48000000)

  Internal High Speed oscillator for USB (HSI48) value.
- #define **LSI_VALUE** ((uint32_t)40000)

  Internal Low Speed oscillator (LSI) value.
- #define LSE_VALUE ((uint32_t)32768)

  External Low Speed oscillator (LSI) value.
- #define LSE_STARTUP_TIMEOUT ((uint32_t)5000)

  Time out for LSE start up value in ms.
- #define VDD_VALUE ((uint32_t)3300)

  This is the HAL system configuration section.
- #define TICK_INT_PRIORITY ((uint32_t)3)
- #define **USE_RTOS** 0
- #define **PREFETCH_ENABLE** 1
- #define **INSTRUCTION_CACHE_ENABLE** 0
- #define **DATA_CACHE_ENABLE** 0
- #define **USE_SPI_CRC** 0U
- #define **USE_HAL_ADC_REGISTER_CALLBACKS** 0U /∗ ADC register callback disabled ∗/
- #define **USE_HAL_CAN_REGISTER_CALLBACKS** 0U /∗ CAN register callback disabled ∗/
- #define **USE_HAL_COMP_REGISTER_CALLBACKS** 0U /∗ COMP register callback disabled ∗/
- #define **USE_HAL_CEC_REGISTER_CALLBACKS** 0U /∗ CEC register callback disabled ∗/
- #define **USE_HAL_DAC_REGISTER_CALLBACKS** 0U /∗ DAC register callback disabled ∗/
- #define **USE_HAL_I2C_REGISTER_CALLBACKS** 0U /∗ I2C register callback disabled ∗/
- #define **USE_HAL_SMBUS_REGISTER_CALLBACKS** 0U /∗ SMBUS register callback disabled ∗/
- #define **USE_HAL_UART_REGISTER_CALLBACKS** 0U /∗ UART register callback disabled ∗/
- #define **USE_HAL_USART_REGISTER_CALLBACKS** 0U /∗ USART register callback disabled ∗/
- #define **USE_HAL_IRDA_REGISTER_CALLBACKS** 0U /∗ IRDA register callback disabled ∗/
- #define **USE_HAL_SMARTCARD_REGISTER_CALLBACKS** 0U /∗ SMARTCARD register callback disabled ∗/
- #define **USE_HAL_WWDG_REGISTER_CALLBACKS** 0U /∗ WWDG register callback disabled ∗/
- #define **USE_HAL_RTC_REGISTER_CALLBACKS** 0U /∗ RTC register callback disabled ∗/
- #define **USE_HAL_SPI_REGISTER_CALLBACKS** 0U /∗ SPI register callback disabled ∗/
- #define **USE_HAL_I2S_REGISTER_CALLBACKS** 0U /∗ I2S register callback disabled ∗/
- #define **USE_HAL_TIM_REGISTER_CALLBACKS** 0U /∗ TIM register callback disabled ∗/
- #define **USE_HAL_TSC_REGISTER_CALLBACKS** 0U /∗ TSC register callback disabled ∗/
- #define **USE_HAL_PCD_REGISTER_CALLBACKS** 0U /∗ PCD register callback disabled ∗/
- #define assert_param(expr) ((void)0U)

  Uncomment the line below to expanse the "assert_param" macro in the HAL drivers code.

7.17.1 Detailed Description

HAL configuration file.

Attention

7.17.2 Macro Definition Documentation

7.17.2.1 assert_param

```
#define assert_param(
             expr ) ((void)0U)
```

Uncomment the line below to expanse the "assert_param" macro in the HAL drivers code.

Include module's header file

7.17.2.2 HSE_STARTUP_TIMEOUT

```
#define HSE_STARTUP_TIMEOUT ((uint32_t)100)
```

In the following line adjust the External High Speed oscillator (HSE) Startup Timeout value.

Time out for HSE start up, in ms

7.17.2.3 HSE_VALUE

```
#define HSE_VALUE ((uint32_t)8000000)
```

Adjust the value of External High Speed oscillator (HSE) used in your application. This value is used by the RCC HAL module to compute the system frequency (when HSE is used as system clock source, directly or through the PLL).

Value of the External oscillator in Hz

7.17.2.4 HSI14_VALUE

```
#define HSI14_VALUE ((uint32_t)14000000)
```

Internal High Speed oscillator for ADC (HSI14) value.

Value of the Internal High Speed oscillator for ADC in Hz. The real value may vary depending on the variations in voltage and temperature.

7.17.2.5 HSI48_VALUE

```
#define HSI48_VALUE ((uint32_t)48000000)
```

Internal High Speed oscillator for USB (HSI48) value.

Value of the Internal High Speed oscillator for USB in Hz. The real value may vary depending on the variations in voltage and temperature.

7.17.2.6 HSI_STARTUP_TIMEOUT

```
#define HSI_STARTUP_TIMEOUT ((uint32_t)5000)
```

In the following line adjust the Internal High Speed oscillator (HSI) Startup Timeout value.

Time out for HSI start up

7.17.2.7 HSI_VALUE

```
#define HSI_VALUE ((uint32_t)8000000)
```

Internal High Speed oscillator (HSI) value. This value is used by the RCC HAL module to compute the system frequency (when HSI is used as system clock source, directly or through the PLL).

Value of the Internal oscillator in Hz

7.17.2.8 LSE_STARTUP_TIMEOUT

```
#define LSE_STARTUP_TIMEOUT ((uint32_t)5000)
```

Time out for LSE start up value in ms.

Time out for LSE start up, in ms

7.17.2.9 LSE_VALUE

```
#define LSE_VALUE ((uint32_t)32768)
```

External Low Speed oscillator (LSI) value.

$<$ Value of the Internal Low Speed oscillator in Hz The real value may vary depending on the variations in voltage and temperature.
Value of the External Low Speed oscillator in Hz

7.17.2.10 TICK_INT_PRIORITY

```
#define TICK_INT_PRIORITY ((uint32_t)3)
```

tick interrupt priority (lowest by default)

7.17.2.11 VDD_VALUE

```
#define VDD_VALUE ((uint32_t)3300)
```

This is the HAL system configuration section.

Value of VDD in mv

7.18 stm32f0xx_hal_conf.h

Go to the documentation of this file.

```
00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00020 /* Define to prevent recursive inclusion -------------------------------------*/
00021 #ifndef __STM32F0xx_HAL_CONF_H
00022 #define __STM32F0xx_HAL_CONF_H
00023
00024 #ifdef __cplusplus
00025  extern "C" {
00026 #endif
00027
00028 /* Exported types ------------------------------------------------------------*/
00029 /* Exported constants --------------------------------------------------------*/
00030
00031 /* ########################## Module Selection ############################## */
00035 #define HAL_MODULE_ENABLED
00036   #define HAL_ADC_MODULE_ENABLED
00037 /*#define HAL_CRYP_MODULE_ENABLED   */
00038 /*#define HAL_CAN_MODULE_ENABLED   */
00039 /*#define HAL_CEC_MODULE_ENABLED   */
00040 /*#define HAL_COMP_MODULE_ENABLED   */
00041 /*#define HAL_CRC_MODULE_ENABLED   */
00042 /*#define HAL_CRYP_MODULE_ENABLED   */
00043 /*#define HAL_TSC_MODULE_ENABLED   */
00044 /*#define HAL_DAC_MODULE_ENABLED   */
00045 /*#define HAL_I2S_MODULE_ENABLED   */
00046 #define HAL_IWDG_MODULE_ENABLED
00047 /*#define HAL_LCD_MODULE_ENABLED   */
00048 /*#define HAL_LPTIM_MODULE_ENABLED   */
00049 /*#define HAL_RNG_MODULE_ENABLED   */
00050 #define HAL_RTC_MODULE_ENABLED
00051 #define HAL_SPI_MODULE_ENABLED
00052 #define HAL_TIM_MODULE_ENABLED
00053 #define HAL_UART_MODULE_ENABLED
00054 /*#define HAL_USART_MODULE_ENABLED   */
00055 /*#define HAL_IRDA_MODULE_ENABLED   */
00056 /*#define HAL_SMARTCARD_MODULE_ENABLED   */
00057 /*#define HAL_SMBUS_MODULE_ENABLED   */
00058 /*#define HAL_WWDG_MODULE_ENABLED   */
00059 /*#define HAL_PCD_MODULE_ENABLED   */
00060 #define HAL_CORTEX_MODULE_ENABLED
00061 #define HAL_DMA_MODULE_ENABLED
00062 #define HAL_FLASH_MODULE_ENABLED
00063 #define HAL_GPIO_MODULE_ENABLED
00064 #define HAL_EXTI_MODULE_ENABLED
00065 #define HAL_PWR_MODULE_ENABLED
00066 #define HAL_RCC_MODULE_ENABLED
00067 #define HAL_I2C_MODULE_ENABLED
00068
00069 /* ########################## HSE/HSI Values adaptation ##################### */
00075 #if !defined  (HSE_VALUE)
00076   #define HSE_VALUE    ((uint32_t)8000000)
00077 #endif /* HSE_VALUE */
00078
00083 #if !defined  (HSE_STARTUP_TIMEOUT)
00084   #define HSE_STARTUP_TIMEOUT    ((uint32_t)100)
00085 #endif /* HSE_STARTUP_TIMEOUT */
00086
00092 #if !defined  (HSI_VALUE)
00093   #define HSI_VALUE    ((uint32_t)8000000)
00094 #endif /* HSI_VALUE */
00095
00100 #if !defined  (HSI_STARTUP_TIMEOUT)
00101  #define HSI_STARTUP_TIMEOUT   ((uint32_t)5000)
00102 #endif /* HSI_STARTUP_TIMEOUT */
00103
```

```
00107 #if !defined  (HSI14_VALUE)
00108 #define HSI14_VALUE ((uint32_t)14000000)
00111 #endif /* HSI14_VALUE */
00112
00116 #if !defined  (HSI48_VALUE)
00117   #define HSI48_VALUE ((uint32_t)48000000)
00120 #endif /* HSI48_VALUE */
00121
00125 #if !defined  (LSI_VALUE)
00126   #define LSI_VALUE  ((uint32_t)40000)
00127 #endif /* LSI_VALUE */
00133 #if !defined  (LSE_VALUE)
00134   #define LSE_VALUE  ((uint32_t)32768)
00135 #endif /* LSE_VALUE */
00136
00140 #if !defined  (LSE_STARTUP_TIMEOUT)
00141   #define LSE_STARTUP_TIMEOUT    ((uint32_t)5000)
00142 #endif /* LSE_STARTUP_TIMEOUT */
00143
00144 /* Tip: To avoid modifying this file each time you need to use different HSE,
00145    ===  you can define the HSE value in your toolchain compiler preprocessor. */
00146
00147 /* ########################### System Configuration ######################### */
00151 #define  VDD_VALUE                    ((uint32_t)3300)
00152 #define  TICK_INT_PRIORITY            ((uint32_t)3)
00153                                                                              /*  Warning: Must be set
      to higher priority for HAL_Delay()  */
00154                                                                              /*  and HAL_GetTick()
      usage under interrupt context          */
00155 #define  USE_RTOS                     0
00156 #define  PREFETCH_ENABLE              1
00157 #define  INSTRUCTION_CACHE_ENABLE     0
00158 #define  DATA_CACHE_ENABLE            0
00159 #define  USE_SPI_CRC                     0U
00160
00161 #define  USE_HAL_ADC_REGISTER_CALLBACKS         0U /* ADC register callback disabled       */
00162 #define  USE_HAL_CAN_REGISTER_CALLBACKS         0U /* CAN register callback disabled       */
00163 #define  USE_HAL_COMP_REGISTER_CALLBACKS        0U /* COMP register callback disabled      */
00164 #define  USE_HAL_CEC_REGISTER_CALLBACKS         0U /* CEC register callback disabled       */
00165 #define  USE_HAL_DAC_REGISTER_CALLBACKS         0U /* DAC register callback disabled       */
00166 #define  USE_HAL_I2C_REGISTER_CALLBACKS         0U /* I2C register callback disabled       */
00167 #define  USE_HAL_SMBUS_REGISTER_CALLBACKS       0U /* SMBUS register callback disabled     */
00168 #define  USE_HAL_UART_REGISTER_CALLBACKS        0U /* UART register callback disabled      */
00169 #define  USE_HAL_USART_REGISTER_CALLBACKS       0U /* USART register callback disabled     */
00170 #define  USE_HAL_IRDA_REGISTER_CALLBACKS        0U /* IRDA register callback disabled      */
00171 #define  USE_HAL_SMARTCARD_REGISTER_CALLBACKS   0U /* SMARTCARD register callback disabled */
00172 #define  USE_HAL_WWDG_REGISTER_CALLBACKS        0U /* WWDG register callback disabled      */
00173 #define  USE_HAL_RTC_REGISTER_CALLBACKS         0U /* RTC register callback disabled       */
00174 #define  USE_HAL_SPI_REGISTER_CALLBACKS         0U /* SPI register callback disabled       */
00175 #define  USE_HAL_I2S_REGISTER_CALLBACKS         0U /* I2S register callback disabled       */
00176 #define  USE_HAL_TIM_REGISTER_CALLBACKS         0U /* TIM register callback disabled       */
00177 #define  USE_HAL_TSC_REGISTER_CALLBACKS         0U /* TSC register callback disabled       */
00178 #define  USE_HAL_PCD_REGISTER_CALLBACKS         0U /* PCD register callback disabled       */
00179
00180 /* ########################### Assert Selection ############################## */
00185 /* #define USE_FULL_ASSERT   1U */
00186
00187 /* Includes ------------------------------------------------------------------*/
00192 #ifdef HAL_RCC_MODULE_ENABLED
00193  #include "stm32f0xx_hal_rcc.h"
00194 #endif /* HAL_RCC_MODULE_ENABLED */
00195
00196 #ifdef HAL_GPIO_MODULE_ENABLED
00197  #include "stm32f0xx_hal_gpio.h"
00198 #endif /* HAL_GPIO_MODULE_ENABLED */
00199
00200 #ifdef HAL_EXTI_MODULE_ENABLED
00201   #include "stm32f0xx_hal_exti.h"
00202 #endif /* HAL_EXTI_MODULE_ENABLED */
00203
00204 #ifdef HAL_DMA_MODULE_ENABLED
00205   #include "stm32f0xx_hal_dma.h"
00206 #endif /* HAL_DMA_MODULE_ENABLED */
00207
00208 #ifdef HAL_CORTEX_MODULE_ENABLED
00209  #include "stm32f0xx_hal_cortex.h"
00210 #endif /* HAL_CORTEX_MODULE_ENABLED */
00211
00212 #ifdef HAL_ADC_MODULE_ENABLED
00213  #include "stm32f0xx_hal_adc.h"
00214 #endif /* HAL_ADC_MODULE_ENABLED */
00215
00216 #ifdef HAL_CAN_MODULE_ENABLED
00217  #include "stm32f0xx_hal_can.h"
00218 #endif /* HAL_CAN_MODULE_ENABLED */
00219
00220 #ifdef HAL_CEC_MODULE_ENABLED
```

```
00221  #include "stm32f0xx_hal_cec.h"
00222 #endif /* HAL_CEC_MODULE_ENABLED */
00223
00224 #ifdef HAL_COMP_MODULE_ENABLED
00225  #include "stm32f0xx_hal_comp.h"
00226 #endif /* HAL_COMP_MODULE_ENABLED */
00227
00228 #ifdef HAL_CRC_MODULE_ENABLED
00229  #include "stm32f0xx_hal_crc.h"
00230 #endif /* HAL_CRC_MODULE_ENABLED */
00231
00232 #ifdef HAL_DAC_MODULE_ENABLED
00233  #include "stm32f0xx_hal_dac.h"
00234 #endif /* HAL_DAC_MODULE_ENABLED */
00235
00236 #ifdef HAL_FLASH_MODULE_ENABLED
00237  #include "stm32f0xx_hal_flash.h"
00238 #endif /* HAL_FLASH_MODULE_ENABLED */
00239
00240 #ifdef HAL_I2C_MODULE_ENABLED
00241  #include "stm32f0xx_hal_i2c.h"
00242 #endif /* HAL_I2C_MODULE_ENABLED */
00243
00244 #ifdef HAL_I2S_MODULE_ENABLED
00245  #include "stm32f0xx_hal_i2s.h"
00246 #endif /* HAL_I2S_MODULE_ENABLED */
00247
00248 #ifdef HAL_IRDA_MODULE_ENABLED
00249  #include "stm32f0xx_hal_irda.h"
00250 #endif /* HAL_IRDA_MODULE_ENABLED */
00251
00252 #ifdef HAL_IWDG_MODULE_ENABLED
00253  #include "stm32f0xx_hal_iwdg.h"
00254 #endif /* HAL_IWDG_MODULE_ENABLED */
00255
00256 #ifdef HAL_PCD_MODULE_ENABLED
00257  #include "stm32f0xx_hal_pcd.h"
00258 #endif /* HAL_PCD_MODULE_ENABLED */
00259
00260 #ifdef HAL_PWR_MODULE_ENABLED
00261  #include "stm32f0xx_hal_pwr.h"
00262 #endif /* HAL_PWR_MODULE_ENABLED */
00263
00264 #ifdef HAL_RTC_MODULE_ENABLED
00265  #include "stm32f0xx_hal_rtc.h"
00266 #endif /* HAL_RTC_MODULE_ENABLED */
00267
00268 #ifdef HAL_SMARTCARD_MODULE_ENABLED
00269  #include "stm32f0xx_hal_smartcard.h"
00270 #endif /* HAL_SMARTCARD_MODULE_ENABLED */
00271
00272 #ifdef HAL_SMBUS_MODULE_ENABLED
00273  #include "stm32f0xx_hal_smbus.h"
00274 #endif /* HAL_SMBUS_MODULE_ENABLED */
00275
00276 #ifdef HAL_SPI_MODULE_ENABLED
00277  #include "stm32f0xx_hal_spi.h"
00278 #endif /* HAL_SPI_MODULE_ENABLED */
00279
00280 #ifdef HAL_TIM_MODULE_ENABLED
00281  #include "stm32f0xx_hal_tim.h"
00282 #endif /* HAL_TIM_MODULE_ENABLED */
00283
00284 #ifdef HAL_TSC_MODULE_ENABLED
00285  #include "stm32f0xx_hal_tsc.h"
00286 #endif /* HAL_TSC_MODULE_ENABLED */
00287
00288 #ifdef HAL_UART_MODULE_ENABLED
00289  #include "stm32f0xx_hal_uart.h"
00290 #endif /* HAL_UART_MODULE_ENABLED */
00291
00292 #ifdef HAL_USART_MODULE_ENABLED
00293  #include "stm32f0xx_hal_usart.h"
00294 #endif /* HAL_USART_MODULE_ENABLED */
00295
00296 #ifdef HAL_WWDG_MODULE_ENABLED
00297  #include "stm32f0xx_hal_wwdg.h"
00298 #endif /* HAL_WWDG_MODULE_ENABLED */
00299
00300 /* Exported macro ------------------------------------------------------------*/
00301 #ifdef  USE_FULL_ASSERT
00310   #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t *)__FILE__, __LINE__))
00311 /* Exported functions ------------------------------------------------------- */
00312   void assert_failed(uint8_t* file, uint32_t line);
00313 #else
00314   #define assert_param(expr) ((void)0U)
00315 #endif /* USE_FULL_ASSERT */
```

```
00316
00317 #ifdef __cplusplus
00318 }
00319 #endif
00320
00321 #endif /* __STM32F0xx_HAL_CONF_H */
00322
```

7.19 Core/Inc/stm32f0xx_it.h File Reference

This file contains the headers of the interrupt handlers.

Functions

- void **NMI_Handler** (void)

  This function handles Non maskable interrupt.
- void **HardFault_Handler** (void)

  This function handles Hard fault interrupt.
- void **SVC_Handler** (void)

  This function handles System service call via SWI instruction.
- void **PendSV_Handler** (void)

  This function handles Pendable request for system service.
- void **SysTick_Handler** (void)

  This function handles System tick timer.
- void **DMA1_Channel1_IRQHandler** (void)

  This function handles DMA1 channel 1 interrupt.
- void **DMA1_Channel4_5_IRQHandler** (void)

  This function handles DMA1 channel 4 and 5 interrupts.
- void **ADC1_COMP_IRQHandler** (void)

  This function handles ADC and COMP interrupts (COMP interrupts through EXTI lines 21 and 22).
- void **TIM1_BRK_UP_TRG_COM_IRQHandler** (void)

  This function handles TIM1 break, update, trigger and commutation interrupts.
- void **SPI1_IRQHandler** (void)

  This function handles SPI1 global interrupt.
- void **SPI2_IRQHandler** (void)

  This function handles SPI2 global interrupt.
- void **USART2_IRQHandler** (void)

  This function handles USART2 global interrupt.

7.19.1 Detailed Description

This file contains the headers of the interrupt handlers.

Attention

7.20 stm32f0xx_it.h

Go to the documentation of this file.

```
00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00020 /* Define to prevent recursive inclusion -------------------------------------*/
00021 #ifndef __STM32F0xx_IT_H
00022 #define __STM32F0xx_IT_H
00023
00024 #ifdef __cplusplus
00025  extern "C" {
00026 #endif
00027
00028 /* Private includes ----------------------------------------------------------*/
00029 /* USER CODE BEGIN Includes */
00030
00031 /* USER CODE END Includes */
00032
00033 /* Exported types ------------------------------------------------------------*/
00034 /* USER CODE BEGIN ET */
00035
00036 /* USER CODE END ET */
00037
00038 /* Exported constants --------------------------------------------------------*/
00039 /* USER CODE BEGIN EC */
00040
00041 /* USER CODE END EC */
00042
00043 /* Exported macro ------------------------------------------------------------*/
00044 /* USER CODE BEGIN EM */
00045
00046 /* USER CODE END EM */
00047
00048 /* Exported functions prototypes ---------------------------------------------*/
00049 void NMI_Handler(void);
00050 void HardFault_Handler(void);
00051 void SVC_Handler(void);
00052 void PendSV_Handler(void);
00053 void SysTick_Handler(void);
00054 void DMA1_Channel1_IRQHandler(void);
00055 void DMA1_Channel4_5_IRQHandler(void);
00056 void ADC1_COMP_IRQHandler(void);
00057 void TIM1_BRK_UP_TRG_COM_IRQHandler(void);
00058 void SPI1_IRQHandler(void);
00059 void SPI2_IRQHandler(void);
00060 void USART2_IRQHandler(void);
00061 /* USER CODE BEGIN EFP */
00062
00063 /* USER CODE END EFP */
00064
00065 #ifdef __cplusplus
00066 }
00067 #endif
00068
00069 #endif /* __STM32F0xx_IT_H */
```

7.21 Core/Inc/structs.h File Reference

760 Pizza Press structures

```
#include "main.h"
```

Classes

- struct Button

  Debounced button state.
- struct PressSetpoint

  Mechanical press setpoint data.
- struct PressState

Mechanical press state data.

- struct ThermalSetpoint

  Thermal press setpoint data.
- struct ThermalState

  Thermal press state data.
- union Config
- struct Press

  Container for all press mechanical and thermal setpoint/state.
- struct MotorPI

  PI controller parameters and state.
- struct __MenuItem

  Menu item structure–contains all data needed to display a menu item.

Typedefs

- typedef struct __MenuItem **MenuItem**

  Menu item structure–contains all data needed to display a menu item.

Enumerations

- enum PressMode {
  PRESS_READY , **PRESS_ERROR** , **PRESS_DOWN** , **PRESS_DWELL** ,
  PRESS_UP , **PRESS_DONE** , **PRESS_JOG** }

  State machine states for pressing dough.
- enum PressCycleMode { **PRESS_FASTDROP** , **PRESS_PERIOD1** , **PRESS_TAPS** , **PRESS_PERIOD2** }

  State machine states for tapping the dough.
- enum MenuType {
  MENU , **MENU_NUM** , **MENU_FLAG** , **MENU_TEMP** ,
  MENU_RESET , **MENU_TEMP_UNITS** , **MENU_STATUS** , **MENU_JOG** ,
  MENU_DEBUG , **MENU_RESET_COUNT** , **MENU_CYCLE** , **MENU_OTHER** }

  Enum of menu types.

7.21.1 Detailed Description

760 Pizza Press structures

Author

Aaron Yeiser

Date

2022-08-10

Copyright

7.22 structs.h

Go to the documentation of this file.

```
00001
00011 #ifndef STRUCTS_H_
00012 #define STRUCTS_H_
00013
00014 #include "main.h"
00015
00017 typedef enum {
00018     PRESS_READY,
00019     PRESS_ERROR,
00020     PRESS_DOWN,
00021     PRESS_DWELL,
00022     PRESS_UP,
00023     PRESS_DONE,
00024     PRESS_JOG
00025 } PressMode;
00026
00028 typedef enum {
00029     PRESS_FASTDROP,
00030     PRESS_PERIOD1,
00031     PRESS_TAPS,
00032     PRESS_PERIOD2
00033 } PressCycleMode;
00034
00036 typedef enum {
00037     MENU,
00038     MENU_NUM,
00039     MENU_FLAG,
00040     MENU_TEMP,
00041     MENU_RESET,
00042     MENU_TEMP_UNITS,
00043     MENU_STATUS,
00044     MENU_JOG,
00045     MENU_DEBUG,
00046     MENU_RESET_COUNT,
00047     MENU_CYCLE,
00048     MENU_OTHER,
00049 } MenuType;
00050
00052 typedef struct {
00053     int ctr;
00054     int repeat_ctr;
00055     bool state;
00056     bool rising_edge_flag;
00057     bool falling_edge_flag;
00058 } Button;
00059
00061 typedef struct {
00062     int16_t burps;
00063     int16_t press_ticks1;
00064     int16_t press_ticks2;
00065     bool auto_mode;
00066     bool enable;
00067 } PressSetpoint;
00068
00070 typedef struct {
00071     PressMode mode;
00072     PressCycleMode cycle;
00073     bool overload_flag;
00074     int16_t burp_ctr;
00075     int16_t ticks_until_next;
00076     float motor_setpoint;
00077     float motor_slew_limited_setpoint;
00078     float current_limit;
00079     uint32_t error_code;
00080 } PressState;
00081
00083 typedef struct {
00084     float top_temp;
00085     float bottom_temp;
00086     bool enable;
00087 } ThermalSetpoint;
00088
00090 typedef struct {
00091     union {
00092         float temp_buf[4];
00093         struct {
00094             float top1, bottom1, top2, bottom2;
00095         };
00096     };
00097     float top_temp;
00098     float bottom_temp;
00099     float top_threshold;
```

```
00100     float bottom_threshold;
00101     bool top_ready;
00102     bool bottom_ready;
00103     uint16_t bad_read_countdown[4];
00104     uint8_t error;
00105     uint32_t error_code;
00106     bool top_ssr_on;
00107     bool bottom_ssr_on;
00108 } ThermalState;
00109
00112 typedef union {
00113     uint32_t regs[5];
00114     struct {
00115         uint16_t flags; //reg0
00116         int16_t top_temp;
00117         int16_t bottom_temp; //reg1
00118         int16_t press_time1;
00119         int16_t press_time2; //reg2
00120         int16_t burps;
00121         uint32_t ctr; //reg3
00122     };
00123 } Config;
00124
00126 typedef struct {
00127     PressSetpoint press_setpoint;
00128     PressState press_state;
00129     ThermalSetpoint thermal_setpoint;
00130     ThermalState thermal_state;
00131     Config config;
00132 } Press;
00133
00135 typedef struct {
00136     float KP;
00137     float TI;
00138     float accum;
00139     float max_accum;
00140 } MotorPI;
00141
00143 typedef struct __MenuItem{
00147     MenuType type;
00148
00149     uint16_t length;
00150     int16_t upper;
00151
00152     uint16_t index;
00153     int16_t lower;
00154
00155     int16_t step;
00156     int16_t flag;
00157
00158
00159     int16_t value;
00160     int16_t* target;
00161
00162     const char* name;
00163     const char* titlename;
00164
00165     struct __MenuItem* parent;
00166
00167     struct __MenuItem* items[16];
00168
00169     HAL_StatusTypeDef (*display)(struct __MenuItem*);
00170 } MenuItem;
00171
00172 #endif
```

7.23 Core/Src/config.c File Reference

760 Pizza Press configuration constants, backup, and restore

```
#include "config.h"
```

Functions

- void **backup_settings** (Config *config)

Write config settings to RTC backup registers.

- void restore_settings (Config *config)
- void reset_defaults (Config *config)
- void **config_to_setpoints** (Press *press)

  Process config flags and set press temperature setpoints.

Variables

- Press **press**

7.23.1 Detailed Description

760 Pizza Press configuration constants, backup, and restore

Author

Aaron Yeiser

Date

2022-08-10

Copyright

7.23.2 Function Documentation

7.23.2.1 reset_defaults()

```
void reset_defaults (
            Config * config )
```

Reset configuration to default. restore_settings() must be called to write to RTC registers

7.23.2.2 restore_settings()

```
void restore_settings (
            Config * config )
```

Read settings from RTC backup registers If RTC settings are invalid, we restore to default

7.24 Core/Src/control.c File Reference

760 Pizza Press mechanical and thermo controls

```
#include "control.h"
```

Functions

- void **HAL_ADC_ConvCpltCallback** (ADC_HandleTypeDef ∗hadc)

  Signal the ADC is ready when new data has occurred.
- void **HAL_TIM_PeriodElapsedCallback** (TIM_HandleTypeDef ∗htim)

  This gets called every millisecond.
- uint32_t check_interlocks (Press ∗press)

  Check press safety interlocks.
- void motor_state_machine (TIM_HandleTypeDef ∗htim, Press ∗press)

  Press mechanical state machine.
- float get_shunt_current (ADC_HandleTypeDef ∗hadc)

  Read the ADC current in Amps.
- float motor_pi_update (MotorPI ∗state, float err)

  Run PI controller for motor control.
- void motor_pwm_update (TIM_HandleTypeDef ∗htim, Press ∗press, float current)

  Set motor duty cycle with slew rate limit and current limit.
- HAL_StatusTypeDef read_thermocouples (SPI_HandleTypeDef ∗hspi, Press ∗press)

  Read SPI thermocouple readers.
- void thermal_control_loop (SPI_HandleTypeDef ∗hspi, Press ∗press)

  run thermal bang-bang control loop
- int getTopTempDisplay (Press ∗press)

  Top temperature rounded to int and in the correct units.
- int getBottomTempDisplay (Press ∗press)

  Bottom temperature rounded to int and in the correct units.

Variables

- uint32_t **ticks** = 0
- uint16_t **adc_output**
- float **shunt_current**
- float **max_current**
- float **shunt_current_sqr_filt** = 0.0f
- uint32_t **press_count** = 0
- bool **cycle_mode** = 0
- bool **cycle_state** = 0
- uint32_t **cycle_ticks** = 5000
- uint8_t **active_thermocouple** = 0
- uint8_t **thermo_buf** [4]
- volatile bool **adc_ready**
- volatile uint16_t **buzzer_ctr** = 0
- volatile uint16_t **white_led_ctr** = 0
- volatile uint16_t **blue_led_ctr** = 0

7.24.1 Detailed Description

760 Pizza Press mechanical and thermo controls

Author

Aaron Yeiser

Date

2022-08-05

Copyright

7.24.2 Function Documentation

7.24.2.1 check_interlocks()

```
uint32_t check_interlocks (
            Press * press )
```

Check press safety interlocks.

Parameters

press	Press state

ERR_INTERLOCK flag is set if the tray is open (interlock = 1) and the press is not homed to the top of travel ERR↩ _INTERLOCK flag is set if the tray is open and the press state is not READY or DONE ERR_BAD_SWITCH flag is set if the top and bottom switches are both tripped

Returns

error state flags

7.24.2.2 get_shunt_current()

```
float get_shunt_current (
            ADC_HandleTypeDef * hadc )
```

Read the ADC current in Amps.

Parameters

hadc	the ADC handle to read

Returns

float the current

7.24.2.3 getBottomTempDisplay()

```
int getBottomTempDisplay (
            Press * press )
```

Bottom temperature rounded to int and in the correct units.

Parameters

press	Press state

Returns

int Temperature to display

7.24.2.4 getTopTempDisplay()

```
int getTopTempDisplay (
            Press * press )
```

Top temperature rounded to int and in the correct units.

Parameters

press	Press state

Returns

int Temperature to display

7.24.2.5 motor_pi_update()

```
float motor_pi_update (
            MotorPI * state,
            float err )
```

Run PI controller for motor control.

Parameters

state	PI controller state
err	setpoint - measured error

Returns

controller effort

Note

this is not currently used

7.24.2.6 motor_pwm_update()

```
void motor_pwm_update (
            TIM_HandleTypeDef * htim,
            Press * press,
            float current )
```

Set motor duty cycle with slew rate limit and current limit.

Parameters

htim	PWM timer handle
press	Press state
current	Measured press current

7.24.2.7 motor_state_machine()

```
void motor_state_machine (
            TIM_HandleTypeDef * htim,
            Press * press )
```

Press mechanical state machine.

Parameters

htim	PWM timer for motor control
press	Press state

Note

press->press_state.ticks_until_next is used to execute an action after some time delay

PRESS_READY: The press is sitting at top stroke. It will start descending when buttons are pressed All relevant state variables are reset

PRESS_ERROR: Something bad happened. Slowly jog the press up to top of stroke at low current PRESS_DONE state is entered once the press is homed

PRESS_DOWN: Move the press down If cycle mode is PRESS_FASTDROP (exiting READY mode) move down quickly for PRESS_TIME_FASTDROP ticks Once fastdrop is done we limit the drop speed to avoid crashing the press

In manual mode we continue pressing until the bottom limit switch is tripped or the buttons released In auto mode we continue pressing until a timeout is tripped or bottom limit switch is tripped

PRESS_DWELL: Press is all the way down Handle dough tapping in auto mode (count number of taps remaining) Start upward motion if buttons released (manual mode) or tapping timeout (auto mode)

PRESS_UP: Move the press up (slow if tapping dough, fast otherwise) Move the press down if in manual mode or buttons pressed Move the press down if tapping dough and tap counter is nonzero

PRESS_DONE: Immediately goes to PRESS_READY after buttons released This prevents the press from immediately cycling in manual mode

PRESS_JOG: Jog mode, used to move the platen up and down with menu buttons

7.24.2.8 read_thermocouples()

```
HAL_StatusTypeDef read_thermocouples (
            SPI_HandleTypeDef * hspi,
            Press * press )
```

Read SPI thermocouple readers.

Parameters

hspi	SPI handle
press	Press state

Returns

HAL_StatusTypeDef SPI read status

Note

we cycle through thermocouples, only one of four TCs is read per tick

7.24.2.9 thermal_control_loop()

```
void thermal_control_loop (
            SPI_HandleTypeDef * hspi,
            Press * press )
```

run thermal bang-bang control loop

Parameters

hspi	SPI handle for reading thermocouples
press	Press state

7.25 Core/Src/debounce.c File Reference

760 Pizza Press debounced buttons

```
#include "debounce.h"
```

Functions

- bool debounce (Button *button, bool state)

  Debounce a button. This function must be called frequently (at least 1kHz)
- void debounce_menu_buttons (void)
- void **debounce_activate_buttons** (void)

  Debounce the activate buttons (on press sides)
- void **debounce_interlock** (void)

  Debounce the press interlock.

Variables

- Button **menu_up_button**
- Button **menu_down_button**
- Button **menu_enter_button**
- Button **activate_left_button**
- Button **activate_right_button**
- Button **press_top_limit**
- Button **press_bottom_limit**
- Button **tray_interlock**

7.25.1 Detailed Description

760 Pizza Press debounced buttons

Author

Aaron Yeiser

Date

2022-08-05

Copyright

7.25.2 Function Documentation

7.25.2.1 debounce()

```
bool debounce (
            Button * button,
            bool state )
```

Debounce a button. This function must be called frequently (at least 1kHz)

If the button is held down this will also generate a rising edge flag every REPEAT_INTERVAL after a delay of REPEAT_TIME

Parameters

button	The button debounced state
state	Measured state of the physical button

Returns

bool Debounced button state

7.25.2.2 debounce_menu_buttons()

```
void debounce_menu_buttons (
            void  )
```

Debounce all of the menu up/down/enter buttons Note that menu buttons are active low but the debounced states are active high

7.26 Core/Src/main.c File Reference

: Main program body

```
#include "main.h"
#include "control.h"
#include "menu.h"
#include "structs.h"
```

Macros

- #define **DEBUG**
- #define **TX_BUF_LEN** 256

Functions

- void SystemClock_Config (void)

  System Clock Configuration.
- int main (void)

  The application entry point.
- void Error_Handler (void)

  This function is executed in case of error occurrence.

Variables

- ADC_HandleTypeDef **hadc**
- DMA_HandleTypeDef **hdma_adc**
- IWDG_HandleTypeDef **hiwdg**
- RTC_HandleTypeDef **hrtc**
- SPI_HandleTypeDef **hspi1**
- SPI_HandleTypeDef **hspi2**
- DMA_HandleTypeDef **hdma_spi2_tx**
- TIM_HandleTypeDef **htim1**
- TIM_HandleTypeDef **htim2**
- UART_HandleTypeDef **huart2**
- char **uart_tx_buf** [TX_BUF_LEN]
- uint32_t **menu_last_used** =0
- bool **menu_awake** = 0
- uint32_t **last_button_press** =0
- float **shunt_current**
- float **max_current**
- uint16_t **adc_output**

7.26.1 Detailed Description

: Main program body

Attention

7.26.2 Function Documentation

7.26.2.1 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

Return values

None	

7.26.2.2 main()

```
int main (
             void  )
```

The application entry point.

Return values

int	

7.26.2.3 SystemClock_Config()

```
void SystemClock_Config (
             void  )
```

System Clock Configuration.

Return values

None	

Initializes the RCC Oscillators according to the specified parameters in the RCC_OscInitTypeDef structure.

Initializes the CPU, AHB and APB buses clocks

7.27 Core/Src/menu.c File Reference

760 Pizza Press screen menus

```
#include "menu.h"
#include "control.h"
```

Functions

- void **init_menus** (void)

  Initialize menu linkage.
- void **link_menus** (MenuItem *parent, MenuItem *child)

  operation to link parent and child menu items
- MenuItem * menu_up (MenuItem *item)

  Called when menu up button is pressed.
- MenuItem * menu_down (MenuItem *item)

  Called when menu down button is pressed.
- MenuItem * menu_enter (MenuItem *item)

  Called when menu enter button is pressed.
- HAL_StatusTypeDef **menu_return_home** (void)
- HAL_StatusTypeDef **thermocouple_readout** (void)
- HAL_StatusTypeDef **debug_display** (MenuItem *item)
- HAL_StatusTypeDef **lifetime_display** (MenuItem *item)
- HAL_StatusTypeDef **status_display** (MenuItem *item)
- HAL_StatusTypeDef **generic_display** (MenuItem *item)
- HAL_StatusTypeDef **temperature_display** (MenuItem *item)
- HAL_StatusTypeDef **press_time_display** (MenuItem *item)
- HAL_StatusTypeDef **manual_mode_display** (MenuItem *item)
- HAL_StatusTypeDef **reset_display** (MenuItem *item)
- HAL_StatusTypeDef **units_display** (MenuItem *item)
- HAL_StatusTypeDef **jog_display** (MenuItem *item)
- HAL_StatusTypeDef **write_row_innerfunc** (void)
- HAL_StatusTypeDef write_row (uint8_t rownum)

  write RAM text buffer row to SSD1306
- void set_row (const char *str, uint8_t rownum, uint8_t font)

  set RAM text buffer for screen

Variables

- MenuItem status_menu
- MenuItem main_menu
- MenuItem temperature_menu
- MenuItem pressmode_menu
- MenuItem options_menu
- MenuItem top_temp_menu
- MenuItem bottom_temp_menu
- MenuItem press_reset_count
- MenuItem mode_menu
- MenuItem press_time1_menu
- MenuItem press_time2_menu
- MenuItem burps_menu
- MenuItem eco_mode_menu
- MenuItem buzzer_menu
- MenuItem service_menu
- MenuItem reset_menu
- MenuItem units_menu
- MenuItem jog_menu
- MenuItem lifetime_menu
- MenuItem debug_menu
- MenuItem cycle_menu
- MenuItem * **current_menu** = &status_menu
- uint8_t **display_row** = 0
- char **screen_buf** [256]
- uint8_t **screen_fonts** [8]
- uint8_t **invert_row** [8]
- uint32_t **press_count**
- bool **cycle_mode**
- float **shunt_current**
- bool **cycle_state**
- uint8_t **eco_flash_ticks** = 0
- bool **busy_flag** = false

7.27.1 Detailed Description

760 Pizza Press screen menus

Author

Aaron Yeiser

Date

2022-08-05

Copyright

7.27.2 Function Documentation

7.27.2.1 menu_down()

```
MenuItem * menu_down (
            MenuItem * item )
```

Called when menu down button is pressed.

Parameters

the	current menu item

Returns

the next menu item to go to

7.27.2.2 menu_enter()

```
MenuItem * menu_enter (
            MenuItem * item )
```

Called when menu enter button is pressed.

Parameters

the	current menu item

Returns

the next menu item to go to

7.27.2.3 menu_up()

```
MenuItem * menu_up (
            MenuItem * item )
```

Called when menu up button is pressed.

Parameters

the	current menu item

Returns

the next menu item to go to

7.27.2.4 set_row()

```
void set_row (
            const char * str,
            uint8_t rownum,
            uint8_t font )
```

set RAM text buffer for screen

Parameters

str	Text to write to the row
rownum	Row (0-7) to write to
font	0 is small, 1 is big

7.27.2.5 write_row()

```
HAL_StatusTypeDef write_row (
            uint8_t rownum )
```

write RAM text buffer row to SSD1306

Parameters

rownum	the row to write to (0-7)

Returns

HAL_StatusTypeDef the status of writing to the row

Note

this is a blocking function

7.27.3 Variable Documentation

7.27.3.1 bottom_temp_menu

```
MenuItem bottom_temp_menu
```

Initial value:

```
= {
        .type=MENU_TEMP,
        .name="Bottom Temp",
        .titlename="BottomTemp",
        .lower=TEMP_LOWER_LIM_F,
        .upper=TEMP_UPPER_LIM_F,
        .step=1,
        .display=&temperature_display,
        .target=&(press.config.bottom_temp)
}
```

7.27.3.2 burps_menu

```
MenuItem burps_menu
```

Initial value:

```
= {
        .type=MENU_NUM,
        .name="Taps",
        .titlename=NULL,
        .lower=BURPS_LOWER_LIM,
        .upper=BURPS_UPPER_LIM,
        .step=1,
        .display=&generic_display,
        .target=&(press.config.burps)
}
```

7.27.3.3 buzzer_menu

```
MenuItem buzzer_menu
```

Initial value:

```
= {
        .type=MENU_FLAG,
        .name="Buzzer",
        .titlename=NULL,
        .display=&generic_display,
        .flag=CONFIG_BUZZER_FLAG,
        .target=(int16_t*) &(press.config.flags)
}
```

7.27.3.4 cycle_menu

```
MenuItem cycle_menu
```

Initial value:

```
= {
        .type = MENU_CYCLE,
        .name = "Cycle",
        .titlename=NULL,
        .display = &reset_display
}
```

7.27.3.5 debug_menu

```
MenuItem debug_menu
```

Initial value:

```
= {
        .type = MENU_DEBUG,
        .name = "Diagnostics",
        .titlename=NULL,
        .display = &debug_display
}
```

7.27.3.6 eco_mode_menu

```
MenuItem eco_mode_menu
```

Initial value:

```
= {
        .type=MENU_FLAG,
        .name="Eco Mode",
        .titlename=NULL,
        .display=&generic_display,
        .flag=CONFIG_ECO_FLAG,
        .target=(int16_t*) &(press.config.flags)
}
```

7.27.3.7 jog_menu

```
MenuItem jog_menu
```

Initial value:

```
= {
        .type = MENU_JOG,
        .name = "Jog",
        .titlename=NULL,
        .display = &jog_display
}
```

7.27.3.8 lifetime_menu

```
MenuItem lifetime_menu
```

Initial value:

```
= {
        .type = MENU_DEBUG,
        .name = "Lifetime Cycles",
        .titlename="Lifetime",
        .display = &lifetime_display
}
```

7.27.3.9 main_menu

```
MenuItem main_menu
```

Initial value:

```
= {
        .type=MENU,
        .name="Main Menu",
        .titlename=NULL,
        .display=&generic_display
}
```

7.27.3.10 mode_menu

```
MenuItem mode_menu
```

Initial value:

```
= {
        .type=MENU_FLAG,
        .name="Auto/Manual Mode",
        .titlename="Mode Set",
        .display=&manual_mode_display,
        .flag=CONFIG_MODE_FLAG,
        .target=(int16_t*) &(press.config.flags)
}
```

7.27.3.11 options_menu

```
MenuItem options_menu
```

Initial value:

```
= {
        .type=MENU,
        .name="Options",
        .titlename=NULL,
        .display=&generic_display
}
```

7.27.3.12 press_reset_count

```
MenuItem press_reset_count
```

Initial value:

```
= {
        .type=MENU_RESET_COUNT,
        .name="Reset Count",
        .titlename="ResetCount",
        .display=&reset_display,
        .value = 0
}
```

7.27.3.13 press_time1_menu

```
MenuItem press_time1_menu
```

Initial value:

```
= {
        .type=MENU_NUM,
        .name="1st Press Time",
        .titlename="1st Press",
        .lower=PRESS_TIME_LOWER_LIM,
        .upper=PRESS_TIME_UPPER_LIM,
        .step=500,
        .display=&press_time_display,
        .target=&(press.config.press_time1)
}
```

7.27.3.14 press_time2_menu

```
MenuItem press_time2_menu
```

Initial value:

```
= {
        .type=MENU_NUM,
        .name="2nd Press Time",
        .titlename="2nd Press",
        .lower=0,
        .upper=PRESS_TIME_UPPER_LIM,
        .step=500,
        .display=&press_time_display,
        .target=&(press.config.press_time2)
}
```

7.27.3.15 pressmode_menu

```
MenuItem pressmode_menu
```

Initial value:

```
= {
        .type=MENU,
        .name="Press Mode",
        .titlename=NULL,
        .display=&generic_display
}
```

7.27.3.16 reset_menu

```
MenuItem reset_menu
```

Initial value:

```
= {
        .type=MENU_RESET,
        .name="Reset All?",
        .titlename=NULL,
        .display=&reset_display,
        .value = 0
}
```

7.27.3.17 service_menu

```
MenuItem service_menu
```

Initial value:

```
= {
        .type=MENU,
        .name="Service",
        .titlename=NULL,
        .display=&generic_display
}
```

7.27.3.18 status_menu

```
MenuItem status_menu
```

Initial value:

```
= {
        .type=MENU_STATUS,
        .parent=NULL,
        .display=&status_display
}
```

7.27.3.19 temperature_menu

```
MenuItem temperature_menu
```

Initial value:

```
= {
        .type=MENU,
        .name="Platen Temp Adj",
        .titlename="PlatenTemp",
        .display=&generic_display
}
```

7.27.3.20 top_temp_menu

```
MenuItem top_temp_menu
```

Initial value:

```
= {
        .type=MENU_TEMP,
        .name="Top Temp",
        .titlename="TopTemp",
        .lower=TEMP_LOWER_LIM_F,
        .upper=TEMP_UPPER_LIM_F,
        .step=1,
        .display=&temperature_display,
        .target=&(press.config.top_temp)
}
```

7.27.3.21 units_menu

```
MenuItem units_menu
```

Initial value:

```
= {
        .type = MENU_TEMP_UNITS,
        .name = "Temp Units",
        .titlename=NULL,
        .display=&units_display,
        .target=(int16_t*) &(press.config.flags),
        .flag = CONFIG_UNITS_FLAG
}
```

7.28 Core/Src/SSD1306.c File Reference

760 Pizza Press SSD1306 display driver

```
#include <SSD1306.h>
```

Functions

- void **SSD1306_InitScreen** (SPI_HandleTypeDef *hspi)
- HAL_StatusTypeDef **SSD1306_spiWriteDMA** (uint8_t *buf, uint16_t num_bytes)
- HAL_StatusTypeDef **SSD1306_spiWrite** (uint8_t *buf, uint16_t num_bytes)
- HAL_StatusTypeDef **SSD1306_command1** (uint8_t c)
- HAL_StatusTypeDef **SSD1306_sendCommand** (uint8_t command, uint8_t param1, uint8_t param2)
- HAL_StatusTypeDef **SSD1306_sendDataByte** (uint8_t data)
- HAL_StatusTypeDef **SSD1306_sendData** (uint8_t *data, uint16_t len)
- HAL_StatusTypeDef **SSD1306_setPageAddress** (uint8_t start, uint8_t end)
- HAL_StatusTypeDef **SSD1306_setColumnAddress** (uint8_t start, uint8_t end)
- void **SSD1306_writeString** (uint8_t col, const char *text)
- void **SSD1306_writeInt** (uint8_t col, int32_t num)
- void **SSD1306_setFont** (uint8_t font)
- void **SSD1306_setInvert** (uint8_t invert)
- void **SSD1306_writeCharToBuf** (uint8_t col, char chr)
- HAL_StatusTypeDef **SSD1306_writeFrameBufRow** (uint8_t page)
- HAL_StatusTypeDef **SSD1306_WriteRow** (uint8_t page)
- void **SSD1306_ClearBuf** ()
- void **SSD1306_clearDisplay** ()
- void **SSD1306_SetupFrameBuf** ()

Variables

- uint8_t **framebuf** [COLUMNS+FRAME_BUF_OFFSET]
- int **timeout_cnt**
- uint8_t **_char_width**
- uint8_t **_font**
- uint8_t **_invert** = 0
- SPI_HandleTypeDef * **ssd1306_spi**

7.28.1 Detailed Description

760 Pizza Press SSD1306 display driver

Author

Austin Brown

Date

2022-08-05

Copyright

7.29 Core/Src/stm32f0xx_hal_msp.c File Reference

This file provides code for the MSP Initialization and de-Initialization codes.

```
#include "main.h"
```

Functions

- void HAL_TIM_MspPostInit (TIM_HandleTypeDef ∗htim)
- void HAL_MspInit (void)
- void HAL_ADC_MspInit (ADC_HandleTypeDef ∗hadc)

  ADC MSP Initialization This function configures the hardware resources used in this example.
- void HAL_ADC_MspDeInit (ADC_HandleTypeDef ∗hadc)

  ADC MSP De-Initialization This function freeze the hardware resources used in this example.
- void HAL_RTC_MspInit (RTC_HandleTypeDef ∗hrtc)

  RTC MSP Initialization This function configures the hardware resources used in this example.
- void HAL_RTC_MspDeInit (RTC_HandleTypeDef ∗hrtc)

  RTC MSP De-Initialization This function freeze the hardware resources used in this example.
- void HAL_SPI_MspInit (SPI_HandleTypeDef ∗hspi)

  SPI MSP Initialization This function configures the hardware resources used in this example.
- void HAL_SPI_MspDeInit (SPI_HandleTypeDef ∗hspi)

  SPI MSP De-Initialization This function freeze the hardware resources used in this example.
- void HAL_TIM_PWM_MspInit (TIM_HandleTypeDef ∗htim_pwm)

  TIM_PWM MSP Initialization This function configures the hardware resources used in this example.
- void HAL_TIM_PWM_MspDeInit (TIM_HandleTypeDef ∗htim_pwm)

  TIM_PWM MSP De-Initialization This function freeze the hardware resources used in this example.
- void HAL_UART_MspInit (UART_HandleTypeDef ∗huart)

  UART MSP Initialization This function configures the hardware resources used in this example.
- void HAL_UART_MspDeInit (UART_HandleTypeDef ∗huart)

  UART MSP De-Initialization This function freeze the hardware resources used in this example.

Variables

- DMA_HandleTypeDef **hdma_adc**
- DMA_HandleTypeDef **hdma_spi2_tx**

7.29.1 Detailed Description

This file provides code for the MSP Initialization and de-Initialization codes.

Attention

7.29.2 Function Documentation

7.29.2.1 HAL_ADC_MspDeInit()

```
void HAL_ADC_MspDeInit (
            ADC_HandleTypeDef * hadc )
```

ADC MSP De-Initialization This function freeze the hardware resources used in this example.

Parameters

hadc	ADC handle pointer

Return values

None	

ADC GPIO Configuration PA0 ------> ADC_IN0 PA1 ------> ADC_IN1

7.29.2.2 HAL_ADC_MspInit()

```
void HAL_ADC_MspInit (
             ADC_HandleTypeDef * hadc )
```

ADC MSP Initialization This function configures the hardware resources used in this example.

Parameters

hadc	ADC handle pointer

Return values

None	

ADC GPIO Configuration PA0 ------> ADC_IN0 PA1 ------> ADC_IN1

7.29.2.3 HAL_MspInit()

```
void HAL_MspInit (
             void  )
```

Initializes the Global MSP.

7.29.2.4 HAL_RTC_MspDeInit()

```
void HAL_RTC_MspDeInit (
             RTC_HandleTypeDef * hrtc )
```

RTC MSP De-Initialization This function freeze the hardware resources used in this example.

Parameters

hrtc	RTC handle pointer

Return values

None	

7.29.2.5 HAL_RTC_MspInit()

```
void HAL_RTC_MspInit (
            RTC_HandleTypeDef * hrtc )
```

RTC MSP Initialization This function configures the hardware resources used in this example.

Parameters

hrtc	RTC handle pointer

Return values

None	

7.29.2.6 HAL_SPI_MspDeInit()

```
void HAL_SPI_MspDeInit (
            SPI_HandleTypeDef * hspi )
```

SPI MSP De-Initialization This function freeze the hardware resources used in this example.

Parameters

hspi	SPI handle pointer

Return values

None	

SPI1 GPIO Configuration PB3 ------> SPI1_SCK PB4 ------> SPI1_MISO PB5 ------> SPI1_MOSI

SPI2 GPIO Configuration PB13 ------> SPI2_SCK PB15 ------> SPI2_MOSI

7.29.2.7 HAL_SPI_MspInit()

```
void HAL_SPI_MspInit (
            SPI_HandleTypeDef * hspi )
```

SPI MSP Initialization This function configures the hardware resources used in this example.

Parameters

hspi	SPI handle pointer

Return values

None	

SPI1 GPIO Configuration PB3 ------> SPI1_SCK PB4 ------> SPI1_MISO PB5 ------> SPI1_MOSI

SPI2 GPIO Configuration PB13 ------> SPI2_SCK PB15 ------> SPI2_MOSI

7.29.2.8 HAL_TIM_MspPostInit()

```
void HAL_TIM_MspPostInit (
            TIM_HandleTypeDef * htim )
```

TIM1 GPIO Configuration PA7 ------> TIM1_CH1N PB0 ------> TIM1_CH2N PA8 ------> TIM1_CH1 PA9 ------> TIM1_CH2

TIM2 GPIO Configuration PB11 ------> TIM2_CH4

7.29.2.9 HAL_TIM_PWM_MspDeInit()

```
void HAL_TIM_PWM_MspDeInit (
            TIM_HandleTypeDef * htim_pwm )
```

TIM_PWM MSP De-Initialization This function freeze the hardware resources used in this example.

Parameters

htim_pwm	TIM_PWM handle pointer

Return values

None	

7.29.2.10 HAL_TIM_PWM_MspInit()

```
void HAL_TIM_PWM_MspInit (
            TIM_HandleTypeDef * htim_pwm )
```

TIM_PWM MSP Initialization This function configures the hardware resources used in this example.

Parameters

htim_pwm	TIM_PWM handle pointer

Return values

None	

7.29.2.11 HAL_UART_MspDeInit()

```
void HAL_UART_MspDeInit (
             UART_HandleTypeDef * huart )
```

UART MSP De-Initialization This function freeze the hardware resources used in this example.

Parameters

huart	UART handle pointer

Return values

None	

USART2 GPIO Configuration PA2 ------> USART2_TX PA3 ------> USART2_RX

7.29.2.12 HAL_UART_MspInit()

```
void HAL_UART_MspInit (
             UART_HandleTypeDef * huart )
```

UART MSP Initialization This function configures the hardware resources used in this example.

Parameters

huart	UART handle pointer

Return values

None	

USART2 GPIO Configuration PA2 ------> USART2_TX PA3 ------> USART2_RX

7.30 Core/Src/stm32f0xx_it.c File Reference

Interrupt Service Routines.

```
#include "main.h"
#include "stm32f0xx_it.h"
```

Functions

- void **NMI_Handler** (void)

  This function handles Non maskable interrupt.
- void **HardFault_Handler** (void)

  This function handles Hard fault interrupt.
- void **SVC_Handler** (void)

  This function handles System service call via SWI instruction.
- void **PendSV_Handler** (void)

  This function handles Pendable request for system service.
- void **SysTick_Handler** (void)

  This function handles System tick timer.
- void **DMA1_Channel1_IRQHandler** (void)

  This function handles DMA1 channel 1 interrupt.
- void **DMA1_Channel4_5_IRQHandler** (void)

  This function handles DMA1 channel 4 and 5 interrupts.
- void **ADC1_COMP_IRQHandler** (void)

  This function handles ADC and COMP interrupts (COMP interrupts through EXTI lines 21 and 22).
- void **TIM1_BRK_UP_TRG_COM_IRQHandler** (void)

  This function handles TIM1 break, update, trigger and commutation interrupts.
- void **SPI1_IRQHandler** (void)

  This function handles SPI1 global interrupt.
- void **SPI2_IRQHandler** (void)

  This function handles SPI2 global interrupt.
- void **USART2_IRQHandler** (void)

  This function handles USART2 global interrupt.

Variables

- DMA_HandleTypeDef **hdma_adc**
- ADC_HandleTypeDef **hadc**
- DMA_HandleTypeDef **hdma_spi2_tx**
- SPI_HandleTypeDef **hspi1**
- SPI_HandleTypeDef **hspi2**
- TIM_HandleTypeDef **htim1**
- UART_HandleTypeDef **huart2**

7.30.1 Detailed Description

Interrupt Service Routines.

Attention

7.31 Core/Src/syscalls.c File Reference

STM32CubeIDE Minimal System calls file.

```
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>
```

Functions

- int **__io_putchar** (int ch) __attribute__((weak))
- int **__io_getchar** (void)
- void **initialise_monitor_handles** ()
- int **_getpid** (void)
- int **_kill** (int pid, int sig)
- void **_exit** (int status)
- **__attribute__** ((weak))
- int **_close** (int file)
- int **_fstat** (int file, struct stat ∗st)
- int **_isatty** (int file)
- int **_lseek** (int file, int ptr, int dir)
- int **_open** (char ∗path, int flags,...)
- int **_wait** (int ∗status)
- int **_unlink** (char ∗name)
- int **_times** (struct tms ∗buf)
- int **_stat** (char ∗file, struct stat ∗st)
- int **_link** (char ∗old, char ∗new)
- int **_fork** (void)
- int **_execve** (char ∗name, char ∗∗argv, char ∗∗env)

Variables

- char ∗∗ **environ** = __env

7.31.1 Detailed Description

STM32CubeIDE Minimal System calls file.

Author

Auto-generated by STM32CubeIDE

```
For more information about which c-functions
need which of these lowlevel functions
please consult the Newlib libc-manual
```

Attention

7.32 Core/Src/sysmem.c File Reference

STM32CubeIDE System Memory calls file.

```
#include <errno.h>
#include <stdint.h>
```

Functions

- void * _sbrk (ptrdiff_t incr)

  _sbrk() allocates memory to the newlib heap and is used by malloc and others from the C library

7.32.1 Detailed Description

STM32CubeIDE System Memory calls file.

Author

Generated by STM32CubeIDE

```
For more information about which C functions
need which of these lowlevel functions
please consult the newlib libc manual
```

Attention

7.32.2 Function Documentation

7.32.2.1 _sbrk()

```
void * _sbrk (
            ptrdiff_t incr )
```

_sbrk() allocates memory to the newlib heap and is used by malloc and others from the C library

```
* ##########################################################################
* #  .data  #  .bss  #       newlib heap       #          MSP stack          #
* #         #        #                         # Reserved by _Min_Stack_Size #
* ##########################################################################
* ^-- RAM start      ^-- _end                             _estack, RAM end --^
*
```

This implementation starts allocating at the '_end' linker symbol The '_Min_Stack_Size' linker symbol reserves a memory for the MSP stack The implementation considers '_estack' linker symbol to be RAM end NOTE: If the MSP stack, at any point during execution, grows larger than the reserved size, please increase the '_Min_Stack_Size'.

Parameters

incr	Memory size

Returns

Pointer to allocated memory

7.33 Core/Src/system_stm32f0xx.c File Reference

CMSIS Cortex-M0 Device Peripheral Access Layer System Source File.

```
#include "stm32f0xx.h"
```

Macros

- #define HSE_VALUE ((uint32_t)8000000)
- #define HSI_VALUE ((uint32_t)8000000)
- #define HSI48_VALUE ((uint32_t)48000000)

Functions

- void SystemInit (void)

  Setup the microcontroller system.
- void SystemCoreClockUpdate (void)

  Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

Variables

- uint32_t **SystemCoreClock** = 8000000
- const uint8_t **AHBPrescTable** [16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
- const uint8_t **APBPrescTable** [8] = {0, 0, 0, 0, 1, 2, 3, 4}

7.33.1 Detailed Description

CMSIS Cortex-M0 Device Peripheral Access Layer System Source File.

Author

MCD Application Team

1. This file provides two functions and one global variable to be called from user application:
   - SystemInit(): This function is called at startup just after reset and before branch to main program. This call is made inside the "startup_stm32f0xx.s" file.
   - SystemCoreClock variable: Contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.
   - SystemCoreClockUpdate(): Updates the variable SystemCoreClock and must be called whenever the core clock is changed during program execution.

Attention

Index