

Low Voltage Variable Frequency Drive

2.1

Generado por Doxygen 1.15.0

1 Jerarquía de directorios	1
1.1 Directorios	1
2 Índice de estructuras de datos	3
2.1 Estructuras de datos	3
3 Índice de archivos	5
3.1 Lista de archivos	5
4 Documentación de directorios	7
4.1 Referencia del directorio main/adc	7
4.2 Referencia del directorio main/display	7
4.3 Referencia del directorio main/io_control	8
4.4 Referencia del directorio main	8
4.5 Referencia del directorio main/nvs	8
4.6 Referencia del directorio main/rtc	9
4.7 Referencia del directorio main/system	9
4.8 Referencia del directorio main/wifi	9
5 Documentación de estructuras de datos	11
5.1 Referencia de la estructura bitmap_t	11
5.1.1 Descripción detallada	11
5.1.2 Documentación de campos	11
5.1.2.1 bitmap	11
5.1.2.2 h	12
5.1.2.3 w	12
5.1.2.4 x	12
5.1.2.5 y	12
5.2 Referencia de la estructura frequency_settings_SH1106_t	12
5.2.1 Descripción detallada	13
5.2.2 Documentación de campos	13
5.2.2.1 edit	13
5.2.2.2 edit_flag	13
5.2.2.3 edit_variable	13
5.2.2.4 frequency_settings	13
5.2.2.5 multiplier	13
5.3 Referencia de la estructura frequency_settings_t	13
5.3.1 Descripción detallada	14
5.3.2 Documentación de campos	14
5.3.2.1 acceleration	14
5.3.2.2 desacceleration	14
5.3.2.3 freq_regime	14
5.3.2.4 input_variable	14
5.4 Referencia de la unión MCP23017_DEFVAL_t	14

5.4.1 Descripción detallada	15
5.4.2 Documentación de campos	15
5.4.2.1 all	15
5.4.2.2 [struct]	15
5.4.2.3 DEF0	15
5.4.2.4 DEF1	15
5.4.2.5 DEF2	16
5.4.2.6 DEF3	16
5.4.2.7 DEF4	16
5.4.2.8 DEF5	16
5.4.2.9 DEF6	16
5.4.2.10 DEF7	16
5.5 Referencia de la unión MCP23017_GPINTEN_t	17
5.5.1 Descripción detallada	17
5.5.2 Documentación de campos	17
5.5.2.1 all	17
5.5.2.2 [struct]	17
5.5.2.3 GPINT0	17
5.5.2.4 GPINT1	18
5.5.2.5 GPINT2	18
5.5.2.6 GPINT3	18
5.5.2.7 GPINT4	18
5.5.2.8 GPINT5	18
5.5.2.9 GPINT6	18
5.5.2.10 GPINT7	19
5.6 Referencia de la unión MCP23017_GPIO_t	19
5.6.1 Descripción detallada	19
5.6.2 Documentación de campos	19
5.6.2.1 all	19
5.6.2.2 [struct]	19
5.6.2.3 GP0	20
5.6.2.4 GP1	20
5.6.2.5 GP2	20
5.6.2.6 GP3	20
5.6.2.7 GP4	20
5.6.2.8 GP5	20
5.6.2.9 GP6	21
5.6.2.10 GP7	21
5.7 Referencia de la unión MCP23017_GPPU_t	21
5.7.1 Descripción detallada	21
5.7.2 Documentación de campos	21
5.7.2.1 all	21

5.7.2.2 [struct]	22
5.7.2.3 PU0	22
5.7.2.4 PU1	22
5.7.2.5 PU2	22
5.7.2.6 PU3	22
5.7.2.7 PU4	22
5.7.2.8 PU5	23
5.7.2.9 PU6	23
5.7.2.10 PU7	23
5.8 Referencia de la unión MCP23017_INTCAP_t	23
5.8.1 Descripción detallada	24
5.8.2 Documentación de campos	24
5.8.2.1 all	24
5.8.2.2 [struct]	24
5.8.2.3 ICP0	24
5.8.2.4 ICP1	24
5.8.2.5 ICP2	24
5.8.2.6 ICP3	24
5.8.2.7 ICP4	25
5.8.2.8 ICP5	25
5.8.2.9 ICP6	25
5.8.2.10 ICP7	25
5.9 Referencia de la unión MCP23017_INTCON_t	25
5.9.1 Descripción detallada	26
5.9.2 Documentación de campos	26
5.9.2.1 all	26
5.9.2.2 [struct]	26
5.9.2.3 IOC0	26
5.9.2.4 IOC1	26
5.9.2.5 IOC2	26
5.9.2.6 IOC3	26
5.9.2.7 IOC4	27
5.9.2.8 IOC5	27
5.9.2.9 IOC6	27
5.9.2.10 IOC7	27
5.10 Referencia de la unión MCP23017_INTF_t	27
5.10.1 Descripción detallada	28
5.10.2 Documentación de campos	28
5.10.2.1 all	28
5.10.2.2 [struct]	28
5.10.2.3 INT0	28
5.10.2.4 INT1	28

5.10.2.5 INT2	28
5.10.2.6 INT3	28
5.10.2.7 INT4	29
5.10.2.8 INT5	29
5.10.2.9 INT6	29
5.10.2.10 INT7	29
5.11 Referencia de la unión MCP23017_IOCON_t	29
5.11.1 Descripción detallada	30
5.11.2 Documentación de campos	30
5.11.2.1 all	30
5.11.2.2 BANK	30
5.11.2.3 [struct]	30
5.11.2.4 DISSLW	30
5.11.2.5 INTPOL	30
5.11.2.6 MIRROR	30
5.11.2.7 ODR	31
5.11.2.8 reserved	31
5.11.2.9 reserved_2	31
5.11.2.10 SEQOP	31
5.12 Referencia de la unión MCP23017_IODIR_t	31
5.12.1 Descripción detallada	32
5.12.2 Documentación de campos	32
5.12.2.1 all	32
5.12.2.2 [struct]	32
5.12.2.3 IO0	32
5.12.2.4 IO1	32
5.12.2.5 IO2	32
5.12.2.6 IO3	32
5.12.2.7 IO4	33
5.12.2.8 IO5	33
5.12.2.9 IO6	33
5.12.2.10 IO7	33
5.13 Referencia de la unión MCP23017_IPOL_t	33
5.13.1 Descripción detallada	34
5.13.2 Documentación de campos	34
5.13.2.1 all	34
5.13.2.2 [struct]	34
5.13.2.3 IP0	34
5.13.2.4 IP1	34
5.13.2.5 IP2	34
5.13.2.6 IP3	34
5.13.2.7 IP4	35

5.13.2.8 IP5	35
5.13.2.9 IP6	35
5.13.2.10 IP7	35
5.14 Referencia de la unión MCP23017_OLAT_t	35
5.14.1 Descripción detallada	36
5.14.2 Documentación de campos	36
5.14.2.1 all	36
5.14.2.2 [struct]	36
5.14.2.3 OL0	36
5.14.2.4 OL1	36
5.14.2.5 OL2	36
5.14.2.6 OL3	36
5.14.2.7 OL4	37
5.14.2.8 OL5	37
5.14.2.9 OL6	37
5.14.2.10 OL7	37
5.15 Referencia de la estructura rtc_alarms_t	37
5.15.1 Descripción detallada	37
5.15.2 Documentación de campos	38
5.15.2.1 start_timer	38
5.15.2.2 stop_timer	38
5.16 Referencia de la estructura seecurity_settings_SH1106_t	38
5.16.1 Descripción detallada	38
5.16.2 Documentación de campos	38
5.16.2.1 edit	38
5.16.2.2 edit_flag	39
5.16.2.3 edit_variable	39
5.16.2.4 multiplier	39
5.16.2.5 seecurity_settings	39
5.17 Referencia de la estructura seecurity_settings_t	39
5.17.1 Descripción detallada	39
5.17.2 Documentación de campos	40
5.17.2.1 ibus_max	40
5.17.2.2 vbus_min	40
5.18 Referencia de la estructura sh1106_t	40
5.18.1 Descripción detallada	40
5.18.2 Documentación de campos	40
5.18.2.1 buffer	40
5.18.2.2 height	40
5.18.2.3 rotation	41
5.18.2.4 width	41
5.19 Referencia de la estructura spi_cmd_item_t	41

5.19.1 Descripción detallada	41
5.19.2 Documentación de campos	41
5.19.2.1 getValue	41
5.19.2.2 request	42
5.19.2.3 setValue	42
5.20 Referencia de la estructura system_status_t	42
5.20.1 Descripción detallada	42
5.20.2 Documentación de campos	42
5.20.2.1 acceleration	42
5.20.2.2 desacceleration	43
5.20.2.3 emergency_signals	43
5.20.2.4 frequency	43
5.20.2.5 frequency_destiny	43
5.20.2.6 ibus_max	43
5.20.2.7 inputs_status	43
5.20.2.8 status	43
5.20.2.9 vbus_min	44
5.21 Referencia de la estructura time_settings_SH1106_t	44
5.21.1 Descripción detallada	44
5.21.2 Documentación de campos	44
5.21.2.1 edit	44
5.21.2.2 edit_flag	44
5.21.2.3 edit_variable	44
5.21.2.4 multiplier	45
5.21.2.5 time_settings	45
5.22 Referencia de la estructura time_settings_t	45
5.22.1 Descripción detallada	45
5.22.2 Documentación de campos	45
5.22.2.1 time_start	45
5.22.2.2 time_stop	45
5.22.2.3 time_system	45
6 Documentación de archivos	47
6.1 Referencia del archivo main/adc/adc.c	47
6.1.1 Descripción detallada	48
6.1.2 Documentación de «define»	48
6.1.2.1 ADC_ATTEN_USED	48
6.1.2.2 ADC_CH_GPIO34	48
6.1.2.3 ADC_CH_GPIO35	49
6.1.2.4 ADC_CH_GPIO36	49
6.1.2.5 ADC_CH_GPIO39	49
6.1.2.6 ADC_PERIOD_MS	49

6.1.2.7 ADC_UNIT_USED	49
6.1.3 Documentación de funciones	50
6.1.3.1 adc_cali_try_init()	50
6.1.3.2 adc_init()	51
6.1.3.3 adc_task()	51
6.1.3.4 readADC()	52
6.1.4 Documentación de variables	53
6.1.4.1 bus_meas_evt_queue	53
6.1.4.2 calibration_3V3_source	53
6.1.4.3 calibration_3V3_source_ok	53
6.1.4.4 calibration_5V_source	53
6.1.4.5 calibration_5V_source_ok	53
6.1.4.6 calibration_bus_voltage	53
6.1.4.7 calibration_bus_voltage_ok	54
6.1.4.8 calibration_current	54
6.1.4.9 calibration_current_ok	54
6.1.4.10 s_adc	54
6.1.4.11 TAG	54
6.2 adc.c	54
6.3 Referencia del archivo main/adc/adc.h	57
6.3.1 Descripción detallada	57
6.3.2 Documentación de funciones	58
6.3.2.1 adc_init()	58
6.3.2.2 adc_task()	58
6.3.2.3 readADC()	58
6.4 adc.h	59
6.5 Referencia del archivo main/display/display.c	59
6.5.1 Descripción detallada	61
6.5.2 Documentación de «define»	61
6.5.2.1 EDIT_ACCELERATION_INDX	61
6.5.2.2 EDIT_DESACCELERATION_INDX	62
6.5.2.3 EDIT_FREQUENCY_INDX	62
6.5.2.4 EDIT_HFIN_LINE_INDX	62
6.5.2.5 EDIT_HINI_LINE_INDX	62
6.5.2.6 EDIT_IBUS_LINE_INDX	62
6.5.2.7 EDIT_INPUT_VARIATION_INDX	62
6.5.2.8 EDIT_TIME_LINE_INDX	62
6.5.2.9 EDIT_VBUS_LINE_INDX	62
6.5.2.10 FREC_LINE_INDX	63
6.5.2.11 FREQUENCY CUADRATIC VARIABLE	63
6.5.2.12 FREQUENCY LINEAR VARIABLE	63
6.5.2.13 HFIN_LINE_INDX	63

6.5.2.14 HINI_LINE_INDX	63
6.5.2.15 I2C_DISPLAY_MASTER_NUM	63
6.5.2.16 I2C_MASTER_FREQ_HZ	63
6.5.2.17 I2C_MASTER_RX_BUF_DISABLE	63
6.5.2.18 I2C_MASTER_SCL_IO	64
6.5.2.19 I2C_MASTER_SDA_IO	64
6.5.2.20 I2C_MASTER_TX_BUF_DISABLE	64
6.5.2.21 IBUS_LINE_INDX	64
6.5.2.22 VBUS_LINE_INDX	64
6.5.3 Documentación de enumeraciones	64
6.5.3.1 screen_selected_e	64
6.5.4 Documentación de funciones	65
6.5.4.1 DisplayEventPost()	65
6.5.4.2 get_system_acceleration()	65
6.5.4.3 get_system_desacceleration()	65
6.5.4.4 get_system_frequency()	65
6.5.4.5 i2c_master_init()	66
6.5.4.6 sh1106_clear_buffer()	66
6.5.4.7 sh1106_frequency_edit_variables()	66
6.5.4.8 sh1106_init()	66
6.5.4.9 sh1106_main_screen()	67
6.5.4.10 sh1106_print_emergency()	67
6.5.4.11 sh1106_print_frame()	67
6.5.4.12 sh1106_refresh()	67
6.5.4.13 sh1106_security_edit_variables()	67
6.5.4.14 sh1106_select_edit_variables()	68
6.5.4.15 sh1106_splash_screen()	68
6.5.4.16 sh1106_time_edit_variables()	68
6.5.4.17 system_variables_save()	69
6.5.4.18 task_display()	69
6.5.5 Documentación de variables	70
6.5.5.1 blink	70
6.5.5.2 button_evt_queue	70
6.5.5.3 init_seq	70
6.5.5.4 oled	70
6.5.5.5 screen_displayed	70
6.5.5.6 system_frequency_settings	71
6.5.5.7 system_seccurity_settings	71
6.5.5.8 system_time_settings	71
6.5.5.9 TAG	71
6.6 display.c	71
6.7 Referencia del archivo main/display/display.h	81

6.7.1 Descripción detallada	82
6.7.2 Documentación de funciones	82
6.7.2.1 DisplayEventPost()	82
6.7.2.2 get_system_acceleration()	83
6.7.2.3 get_system_desacceleration()	83
6.7.2.4 get_system_frequency()	83
6.7.2.5 sh1106_init()	83
6.7.2.6 system_variables_save()	84
6.7.2.7 task_display()	84
6.8 display.h	85
6.9 Referencia del archivo main/display/sh1106_graphics.c	85
6.9.1 Descripción detallada	86
6.9.2 Documentación de «typedef»	87
6.9.2.1 bitmap_t	87
6.9.3 Documentación de funciones	87
6.9.3.1 sh1106_draw_arrow()	87
6.9.3.2 sh1106_draw_bitmap()	87
6.9.3.3 sh1106_draw_fail()	87
6.9.3.4 sh1106_draw_line()	88
6.9.3.5 sh1106_draw_text()	88
6.9.3.6 sh1106_draw_utn_logo()	88
6.9.4 Documentación de variables	89
6.9.4.1 arrow_bmp	89
6.9.4.2 fail_bmp	89
6.9.4.3 font5x7_close_excl	89
6.9.4.4 font5x7_close_quest	89
6.9.4.5 font5x7_comma	89
6.9.4.6 font5x7_dot	89
6.9.4.7 font5x7_double_dot	90
6.9.4.8 font5x7_minus	90
6.9.4.9 font5x7_rowmajor	90
6.9.4.10 font5x7_rowminor	91
6.9.4.11 font5x7_rownumber	91
6.9.4.12 font5x7_space	92
6.9.4.13 font8x14_close_excl	92
6.9.4.14 font8x14_close_quest	92
6.9.4.15 font8x14_comma	92
6.9.4.16 font8x14_dot	92
6.9.4.17 font8x14_double_dot	93
6.9.4.18 font8x14_minus	93
6.9.4.19 font8x14_rowmajor	93
6.9.4.20 font8x14_rowminor	94

6.9.4.21 font8x14_rownumber	94
6.9.4.22 font8x14_space	95
6.9.4.23 line_bmp	95
6.9.4.24 logo16_utn_bmp	95
6.9.4.25 slash	95
6.10 sh1106_graphics.c	96
6.11 Referencia del archivo main/display/sh1106_graphics.h	101
6.11.1 Descripción detallada	102
6.11.2 Documentación de «typedef»	102
6.11.2.1 sh1106_t	102
6.11.3 Documentación de funciones	102
6.11.3.1 sh1106_draw_arrow()	102
6.11.3.2 sh1106_draw_fail()	103
6.11.3.3 sh1106_draw_line()	103
6.11.3.4 sh1106_draw_text()	103
6.11.3.5 sh1106_draw_utn_logo()	104
6.12 sh1106_graphics.h	104
6.13 Referencia del archivo main/display/sh1106_i2c.c	104
6.13.1 Descripción detallada	105
6.13.2 Documentación de «define»	105
6.13.2.1 CMD_CONTROL	105
6.13.2.2 DATA_CONTROL	105
6.13.2.3 I2C_DISPLAY	106
6.13.2.4 SH1106_ADDR	106
6.13.3 Documentación de funciones	106
6.13.3.1 sh1106_write()	106
6.14 sh1106_i2c.c	106
6.15 Referencia del archivo main/display/sh1106_i2c.h	107
6.15.1 Descripción detallada	107
6.15.2 Documentación de enumeraciones	107
6.15.2.1 sh1106_comm_type_t	107
6.15.3 Documentación de funciones	108
6.15.3.1 sh1106_write()	108
6.16 sh1106_i2c.h	108
6.17 Referencia del archivo main/io_control/io_control.c	109
6.17.1 Descripción detallada	111
6.17.2 Documentación de «define»	111
6.17.2.1 BUZZER_PIN	111
6.17.2.2 DUTY_MAX_PCT	111
6.17.2.3 DUTY_MIN_PCT	111
6.17.2.4 FREQ_SEL_1_PIN	112
6.17.2.5 FREQ_SEL_2_PIN	112

6.17.2.6 FREQ_SEL_3_PIN	112
6.17.2.7 FREQ_SEL_MASK	112
6.17.2.8 INT_A_PIN	112
6.17.2.9 INT_B_PIN	112
6.17.2.10 MATRIX_SW1	113
6.17.2.11 MATRIX_SW2	113
6.17.2.12 MATRIX_SW3	113
6.17.2.13 MATRIX_SW4	113
6.17.2.14 MATRIX_SW5	113
6.17.2.15 MATRIX_SW6	113
6.17.2.16 MATRIX_SW7	114
6.17.2.17 MATRIX_SW8	114
6.17.2.18 PWM_CHANNEL	114
6.17.2.19 PWM_FREQ_HZ	114
6.17.2.20 PWM_GPIO	114
6.17.2.21 PWM_MODE	114
6.17.2.22 PWM_RES	115
6.17.2.23 PWM_STEP_MS	115
6.17.2.24 PWM_TIMER	115
6.17.2.25 RELAY_PIN	115
6.17.2.26 START_BUTTON_PIN	115
6.17.2.27 STOP_BUTTON_PIN	115
6.17.2.28 STOP_PIN	116
6.17.2.29 STOP_SW_MASK	116
6.17.2.30 SWITCH_BUTTON_BACK	116
6.17.2.31 SWITCH_BUTTON_DOWN	116
6.17.2.32 SWITCH_BUTTON_LEFT	116
6.17.2.33 SWITCH_BUTTON_MENU	116
6.17.2.34 SWITCH_BUTTON_OK	117
6.17.2.35 SWITCH_BUTTON_RIGHT	117
6.17.2.36 SWITCH_BUTTON_SAVE	117
6.17.2.37 SWITCH_BUTTON_UP	117
6.17.2.38 TERMO_SW_MASK	117
6.17.2.39 TERMO_SW_PIN	117
6.17.3 Documentación de funciones	117
6.17.3.1 BuzzerEventPost()	117
6.17.3.2 freq_0_10V_output()	118
6.17.3.3 gpio_init_interrupts()	118
6.17.3.4 GPIO_interrupt_attendance_task()	119
6.17.3.5 gpio_isr_handler()	119
6.17.3.6 MCP23017_buzzer_control()	119
6.17.3.7 MCP23017_keyboard_control()	119

6.17.3.8 MCP23017_relay_control()	119
6.17.3.9 RelayEventPost()	119
6.17.3.10 set_freq_output()	120
6.17.4 Documentación de variables	120
6.17.4.1 buzzer_evt_queue	120
6.17.4.2 GPIO_evt_queue	120
6.17.4.3 mcp_porta	121
6.17.4.4 mcp_portb	121
6.17.4.5 relay_evt_queue	121
6.17.4.6 TAG	121
6.18 io_control.c	121
6.19 Referencia del archivo main/io_control/io_control.h	128
6.19.1 Descripción detallada	129
6.19.2 Documentación de funciones	129
6.19.2.1 BuzzerEventPost()	129
6.19.2.2 GPIO_interrupt_attendance_task()	129
6.19.2.3 RelayEventPost()	130
6.19.2.4 set_freq_output()	131
6.20 io_control.h	131
6.21 Referencia del archivo main/io_control/MCP23017.c	132
6.21.1 Descripción detallada	133
6.21.2 Documentación de «define»	133
6.21.2.1 I2C_IO_MASTER_NUM	133
6.21.2.2 I2C_MASTER_FREQ_HZ	133
6.21.2.3 I2C_MASTER_RX_BUF_DISABLE	134
6.21.2.4 I2C_MASTER_SCL_IO	134
6.21.2.5 I2C_MASTER_SDA_IO	134
6.21.2.6 I2C_MASTER_TIMEOUT_MS	134
6.21.2.7 I2C_MASTER_TX_BUF_DISABLE	134
6.21.3 Documentación de funciones	134
6.21.3.1 i2c_is_hardware_free()	134
6.21.3.2 i2c_master_init()	134
6.21.3.3 i2c_release_hardware()	135
6.21.3.4 MCP23017_INIT()	135
6.21.3.5 mcp_get_on_interrupt_input()	135
6.21.3.6 mcp_interrupt_flag()	136
6.21.3.7 mcp_read_port()	136
6.21.3.8 mcp_register_read()	137
6.21.3.9 mcp_register_write()	137
6.21.3.10 mcp_write_output_pin()	138
6.21.3.11 mcp_write_output_port()	138
6.21.4 Documentación de variables	139

6.21.4.1 TAG	139
6.21.4.2 xI2C_Mutex	139
6.22 MCP23017.c	139
6.23 Referencia del archivo main/io_control/MCP23017.h	143
6.23.1 Descripción detallada	144
6.23.2 Documentación de funciones	144
6.23.2.1 MCP23017_INIT()	144
6.23.2.2 mcp_get_on_interrupt_input()	145
6.23.2.3 mcp_interrupt_flag()	146
6.23.2.4 mcp_read_port()	146
6.23.2.5 mcp_write_output_pin()	147
6.23.2.6 mcp_write_output_port()	147
6.24 MCP23017.h	148
6.25 Referencia del archivo main/io_control/mcp23017_defs.h	148
6.25.1 Documentación de «define»	150
6.25.1.1 __MCP23017_ADDRESS__	150
6.25.1.2 __MCP23017_BANK_SEQUENTIAL__	150
6.25.1.3 __MCP23017_BANK_SPLIT__	150
6.25.1.4 __MCP23017_DISSLLW_DISABLED__	151
6.25.1.5 __MCP23017_DISSLLW_ENABLED__	151
6.25.1.6 __MCP23017_FUNC_MODE__	151
6.25.1.7 __MCP23017_GPINTEN_DISABLE__	151
6.25.1.8 __MCP23017_GPINTEN_ENABLE__	151
6.25.1.9 __MCP23017_GPPU_PULL_UP_DISABLE__	151
6.25.1.10 __MCP23017_GPPU_PULL_UP_ENABLE__	152
6.25.1.11 __MCP23017_INTCON_CHANGE__	152
6.25.1.12 __MCP23017_INTCON_DEFVAL__	152
6.25.1.13 __MCP23017_INTERRUPT_DISABLE__	152
6.25.1.14 __MCP23017_INTERRUPT_ENABLE__	152
6.25.1.15 __MCP23017_INTPOL_POLARITY_HIGH__	152
6.25.1.16 __MCP23017_INTPOL_POLARITY_LOW__	153
6.25.1.17 __MCP23017_IODIR_INPUT__	153
6.25.1.18 __MCP23017_IODIR_OUTPUT__	153
6.25.1.19 __MCP23017_MIRROR_CONNECTED__	153
6.25.1.20 __MCP23017_MIRROR_UNCONNECTED__	153
6.25.1.21 __MCP23017_ODR_ACTIVE_DRIVER_OUTPUT__	153
6.25.1.22 __MCP23017_ODR_OPEN_DRAIN_OUTPUT__	154
6.25.1.23 __MCP23017_OPPOSITE_LOGIC__	154
6.25.1.24 __MCP23017_POSITIVE_LOGIC__	154
6.25.1.25 __MCP23017_READ__	154
6.25.1.26 __MCP23017_READ_OPCODE__	154
6.25.1.27 __MCP23017_SEQOP_DISABLED__	154

6.25.1.28 __MCP23017_SEQOP_ENABLED_	155
6.25.1.29 __MCP23017_WRITE_	155
6.25.1.30 __MCP23017_WRITE_OPCODE_	155
6.25.1.31 MCP23017_DEFVALA_REGISTER	155
6.25.1.32 MCP23017_DEFVALB_REGISTER	155
6.25.1.33 MCP23017_GPINTENA_REGISTER	155
6.25.1.34 MCP23017_GPINTENB_REGISTER	156
6.25.1.35 MCP23017_GPIOA_REGISTER	156
6.25.1.36 MCP23017_GPIOB_REGISTER	156
6.25.1.37 MCP23017_GPPUA_REGISTER	156
6.25.1.38 MCP23017_GPPUB_REGISTER	156
6.25.1.39 MCP23017_INTCAPA_REGISTER	156
6.25.1.40 MCP23017_INTCAPB_REGISTER	157
6.25.1.41 MCP23017_INTCONA_REGISTER	157
6.25.1.42 MCP23017_INTCONB_REGISTER	157
6.25.1.43 MCP23017_INTFA_REGISTER	157
6.25.1.44 MCP23017_INTFB_REGISTER	157
6.25.1.45 MCP23017_IOCON_REGISTER	157
6.25.1.46 MCP23017_IODIRA_REGISTER	158
6.25.1.47 MCP23017_IODIRB_REGISTER	158
6.25.1.48 MCP23017_IPOLA_REGISTER	158
6.25.1.49 MCP23017_IPOLB_REGISTER	158
6.25.1.50 MCP23017_OLATA_REGISTER	158
6.25.1.51 MCP23017_OLATB_REGISTER	158
6.25.2 Documentación de enumeraciones	158
6.25.2.1 mcp_port_e	158
6.26 mcp23017_defs.h	159
6.27 Referencia del archivo main/LVFV_system.h	162
6.27.1 Descripción detallada	163
6.27.2 Documentación de «define»	164
6.27.2.1 LINE_INCREMENT	164
6.27.2.2 SH1106_SIZE_1	164
6.27.2.3 SH1106_SIZE_2	164
6.27.2.4 VARIABLE_EIGHT	164
6.27.2.5 VARIABLE_FIFTH	164
6.27.2.6 VARIABLE_FIRST	164
6.27.2.7 VARIABLE_FOURTH	164
6.27.2.8 VARIABLE_SECOND	165
6.27.2.9 VARIABLE_SEVENTH	165
6.27.2.10 VARIABLE_SIXTH	165
6.27.2.11 VARIABLE_THIRD	165
6.27.3 Documentación de «typedef»	165

6.27.3.1 frequency_settings_SH1106_t	165
6.27.3.2 frequency_settings_t	165
6.27.3.3 security_settings_SH1106_t	165
6.27.3.4 security_settings_t	165
6.27.3.5 sh1106_variable_lines_e	166
6.27.3.6 system_status_e	166
6.27.3.7 system_status_t	166
6.27.3.8 time_settings_SH1106_t	166
6.27.3.9 time_settings_t	166
6.27.4 Documentación de enumeraciones	166
6.27.4.1 sh1106_variable_lines_e	166
6.27.4.2 system_status_e	166
6.27.4.3 systemSignal_e	167
6.28 LVFV_system.h	168
6.29 Referencia del archivo main/main.c	170
6.29.1 Descripción detallada	170
6.29.2 Documentación de funciones	171
6.29.2.1 app_main()	171
6.29.3 Documentación de variables	171
6.29.3.1 TAG	171
6.30 main.c	171
6.31 Referencia del archivo main/nvs/nvs.c	172
6.31.1 Descripción detallada	173
6.31.2 Documentación de «define»	173
6.31.2.1 load_acceleration	173
6.31.2.2 load_desacceleration	174
6.31.2.3 load_frequency	174
6.31.2.4 load_hour_fin	174
6.31.2.5 load_hour_ini	174
6.31.2.6 load_ibus_max	174
6.31.2.7 load_input_variable	175
6.31.2.8 load_min_fin	175
6.31.2.9 load_min_ini	175
6.31.2.10 load_vbus_min	175
6.31.2.11 save_acceleration	175
6.31.2.12 save_desacceleration	176
6.31.2.13 save_frequency	176
6.31.2.14 save_hour_fin	176
6.31.2.15 save_hour_ini	176
6.31.2.16 save_ibus_max	176
6.31.2.17 save_input_variable	177
6.31.2.18 save_min_fin	177

6.31.2.19 save_min_ini	177
6.31.2.20 save_vbus_min	177
6.31.3 Documentación de funciones	177
6.31.3.1 load_16()	177
6.31.3.2 load_variables()	178
6.31.3.3 nvs_init_once()	179
6.31.3.4 save_16()	179
6.31.3.5 save_variables()	180
6.31.4 Documentación de variables	181
6.31.4.1 TAG	181
6.32 nvs.c	181
6.33 Referencia del archivo main/nvs/nvs.h	185
6.33.1 Descripción detallada	185
6.33.2 Documentación de funciones	186
6.33.2.1 load_variables()	186
6.33.2.2 nvs_init_once()	186
6.33.2.3 save_variables()	187
6.34 nvs.h	188
6.35 Referencia del archivo main/rtc/rtc.c	188
6.35.1 Descripción detallada	189
6.35.2 Documentación de funciones	189
6.35.2.1 alarm_start_cb()	189
6.35.2.2 alarm_stop_cb()	189
6.35.2.3 getTime()	190
6.35.2.4 initRTC()	190
6.35.2.5 program_alarm()	190
6.35.2.6 rtc_schedule_alarms()	191
6.35.2.7 setTime()	191
6.35.3 Documentación de variables	192
6.35.3.1 alarm_settings	192
6.35.3.2 rtc_alarms	192
6.35.3.3 TAG	192
6.35.3.4 time_start	192
6.35.3.5 time_stop	192
6.36 rtc.c	193
6.37 Referencia del archivo main/rtc/rtc.h	195
6.37.1 Descripción detallada	195
6.37.2 Documentación de funciones	196
6.37.2.1 getTime()	196
6.37.2.2 initRTC()	196
6.37.2.3 rtc_schedule_alarms()	196
6.37.2.4 setTime()	197

6.38 rtc.h	197
6.39 Referencia del archivo main/system/sysAdmin.c	198
6.39.1 Descripción detallada	199
6.39.2 Documentación de funciones	199
6.39.2.1 accelerating()	199
6.39.2.2 change_frequency()	199
6.39.2.3 desaccelerating()	200
6.39.2.4 engine_emergency_stop()	200
6.39.2.5 engine_emergency_stop_release()	200
6.39.2.6 engine_start()	201
6.39.2.7 engine_stop()	201
6.39.2.8 get_status()	201
6.39.2.9 set_frequency_table()	202
6.39.2.10 set_system_settings()	202
6.39.2.11 update_meas()	203
6.39.3 Documentación de variables	204
6.39.3.1 accelerating_handle	204
6.39.3.2 desaccelerating_handle	204
6.39.3.3 frequency_table	204
6.39.3.4 system_seccurity_settings	204
6.39.3.5 system_status	205
6.39.3.6 TAG	205
6.40 sysAdmin.c	205
6.41 Referencia del archivo main/system/sysAdmin.h	208
6.41.1 Descripción detallada	208
6.41.2 Documentación de funciones	209
6.41.2.1 change_frequency()	209
6.41.2.2 engine_emergency_stop()	209
6.41.2.3 engine_emergency_stop_release()	209
6.41.2.4 engine_start()	210
6.41.2.5 engine_stop()	210
6.41.2.6 get_status()	210
6.41.2.7 set_frequency_table()	211
6.41.2.8 set_system_settings()	211
6.41.2.9 update_meas()	212
6.42 sysAdmin.h	213
6.43 Referencia del archivo main/system/sysControl.c	213
6.43.1 Descripción detallada	215
6.43.2 Documentación de «define»	215
6.43.2.1 PIN_NUM_CLK	215
6.43.2.2 PIN_NUM_CS	215
6.43.2.3 PIN_NUM_MISO	215

6.43.2.4 PIN_NUM_MOSI	215
6.43.2.5 SPI_CLOCK_HZ	216
6.43.2.6 SPI_HOST_USED	216
6.43.2.7 SPI_QUEUE_TX_DEPTH	216
6.43.3 Documentación de funciones	216
6.43.3.1 SPI_communication()	216
6.43.3.2 SPI_Init()	216
6.43.3.3 SPI_SendRequest()	217
6.43.3.4 SystemEventPost()	217
6.43.4 Documentación de variables	218
6.43.4.1 spi_handle	218
6.43.4.2 system_event_queue	218
6.43.4.3 TAG	218
6.44 sysControl.c	218
6.45 Referencia del archivo main/system/sysControl.h	222
6.45.1 Descripción detallada	223
6.45.2 Documentación de enumeraciones	223
6.45.2.1 SPI_Request	223
6.45.2.2 SPI_Response	224
6.45.3 Documentación de funciones	224
6.45.3.1 SPI_communication()	224
6.45.3.2 SPI_Init()	226
6.45.3.3 SystemEventPost()	226
6.46 sysControl.h	227
6.47 Referencia del archivo main/wifi/form.c	227
6.47.1 Documentación de variables	228
6.47.1.1 HTML_FORM	228
6.48 form.c	228
6.49 Referencia del archivo main/wifi/form.h	230
6.49.1 Documentación de variables	230
6.49.1.1 HTML_FORM	230
6.50 form.h	230
6.51 Referencia del archivo main/wifi/wifi.c	230
6.51.1 Documentación de funciones	231
6.51.1.1 get_param_value()	231
6.51.1.2 hex2int()	232
6.51.1.3 root_get_handler()	232
6.51.1.4 save_post_handler()	232
6.51.1.5 start_webserver()	232
6.51.1.6 url_decode()	232
6.51.1.7 wifi_init_softap()	233
6.51.2 Documentación de variables	233

6.51.2.1 TAG	233
6.52 wifi.c	233
6.53 Referencia del archivo main/wifi/wifi.h	236
6.53.1 Documentación de funciones	237
6.53.1.1 start_webserver()	237
6.53.1.2 wifi_init_softap()	237
6.54 wifi.h	237
Índice alfabético	239

Capítulo 1

Jerarquía de directorios

1.1. Directories

adc	7
adc.c	47
adc.h	57
display	7
display.c	59
display.h	81
sh1106_graphics.c	85
sh1106_graphics.h	101
sh1106_i2c.c	104
sh1106_i2c.h	107
io_control	8
io_control.c	109
io_control.h	128
MCP23017.c	132
MCP23017.h	143
mcp23017_defs.h	148
main	8
adc	7
adc.c	47
adc.h	57
display	7
display.c	59
display.h	81
sh1106_graphics.c	85
sh1106_graphics.h	101
sh1106_i2c.c	104
sh1106_i2c.h	107
io_control	8
io_control.c	109
io_control.h	128
MCP23017.c	132
MCP23017.h	143
mcp23017_defs.h	148
nvs	8
nvs.c	172
nvs.h	185
rtc	9

rtc.c	188
rtc.h	195
system	9
sysAdmin.c	198
sysAdmin.h	208
sysControl.c	213
sysControl.h	222
wifi	9
form.c	227
form.h	230
wifi.c	230
wifi.h	236
LVFV_system.h	162
main.c	170
nvs	8
nvs.c	172
nvs.h	185
rtc	9
rtc.c	188
rtc.h	195
system	9
sysAdmin.c	198
sysAdmin.h	208
sysControl.c	213
sysControl.h	222
wifi	9
form.c	227
form.h	230
wifi.c	230
wifi.h	236

Capítulo 2

Índice de estructuras de datos

2.1. Estructuras de datos

Lista de estructuras con breves descripciones:

bitmap_t	Estructura que representa un carácter cualquiera. Donde se le debe inicializar un puntero a alguno de los caracteres, el alto, ancho, y su posición en x e y	11
frequency_settings_SH1106_t		12
frequency_settings_t		13
MCP23017_DEFVAL_t		14
MCP23017_GPINTEN_t		17
MCP23017_GPIO_t		19
MCP23017_GPPU_t		21
MCP23017_INTCAP_t		23
MCP23017_INTCON_t		25
MCP23017_INTF_t		27
MCP23017_IOCON_t		29
MCP23017_IODIR_t		31
MCP23017_IPOL_t		33
MCP23017_OLAT_t		35
rtc_alarms_t		37
seccurity_settings_SH1106_t		38
seccurity_settings_t		39
sh1106_t		40
spi_cmd_item_t	Estructura de datos que contiene un comando a enviar por SPI, un argumento que se desea enviar al STM32 y un dato que puede recibirse como respuesta en consecuencia	41
system_status_t	Estructura con las variables necesarias para establecer las condiciones de trabajo del motor .	42
time_settings_SH1106_t		44
time_settings_t		45

Capítulo 3

Índice de archivos

3.1. Lista de archivos

Lista de todos los archivos con breves descripciones:

main/ LVFV_system.h	Declaraciones de estructuras, variables, definiciones y estados generales del sistema utilizados en varios de los archivos	162
main/ main.c	Inicialización de tareas y periféricos que no tienen una tarea específica. El resto de los periféricos se inicializan en las tareas que son utilizados	170
main/adc/ adc.c	Funcion de inicialización y tarea lectura del ADC	47
main/adc/ adc.h	Declaración de funcion de inicialización y tarea lectura del ADC	57
main/display/ display.c	Funciones de consulta de los seteos hechos por el usuario, tarea que controla el display y posteо de eventos para controlar el display	59
main/display/ display.h	Declaración de funciones que de consulta de los seteos hechos por el usuario, tarea que controla el display y posteо de eventos para controlar el display	81
main/display/sh1106_graphics/ c	Funciones que permiten operar sobre el display	85
main/display/sh1106_graphics/ h	Declaración de funciones que permiten operar sobre el display	101
main/display/sh1106_i2c/ c	Funciones de hardware para el puerto I2C	104
main/display/sh1106_i2c/ h	Declaración de funciones de hardware para el puerto I2C	107
main/io_control/ io_control.c	Funciones de control del relay, buzzer y tarea del uso del expansor de entradas y salidas y entradas y salidas aisladas directas al ESP32	109
main/io_control/ io_control.h	Declaración de funciones de control del relay, buzzer y tarea del uso del expansor de entradas y salidas y entradas y salidas aisladas directas al ESP32	128
main/io_control/ MCP23017.c	Declaración de funciones de inicialización, lectura y escritura a través del puerto I2C el dispositivo MCP23017	132
main/io_control/ MCP23017.h	Declaración de funciones de inicialización, lectura y escritura a través del puerto I2C el dispositivo MCP23017	143

main/io_control/ mcp23017_defs.h	148
main/nvs/ nvs.c	
Funciones de lectura y escritura de variables no volátiles. Carga en sistema además las variables de seguridad al momento de ser guardadas en la memoria	172
main/nvs/ nvs.h	
Declaración de funciones de lectura y escritura de memoria no volatil. Útiles para los reset y no tener que reconfigurar siempre las variables básicas del sistema	185
main/rtc/ rtc.c	
Funciones de lectura y escritura del RTC y alarmas	188
main/rtc/ rtc.h	
Header con prototipos de funciones básicas para el lectura y escritura del RTC y alarmas	195
main/system/ sysAdmin.c	
Archivo de funciones que permiten controlar las variables fundamentales del sistema que controla el motor, solo para poder ser representadas en el display. Las variables reales de sistema se encuentran el el STM32	198
main/system/ sysAdmin.h	
Header con las funciones de funcionamiento del sistema	208
main/system/ sysControl.c	
Archivo de funciones que controlan el sistema físico a través del puerto SPI contra el STM32. En función de las respuestas del STM32, es que se controlan las variables internas de sistema del ESP32	213
main/system/ sysControl.h	
Declraraciones de funciones que controlan el sistema del STM32. Además se encontrará los comandos y respuestas que admite el STM32	222
main/wifi/ form.c	227
main/wifi/ form.h	230
main/wifi/ wifi.c	230
main/wifi/ wifi.h	236

Capítulo 4

Documentación de directorios

4.1. Referencia del directorio main/adc

Archivos

- archivo [adc.c](#)
Función de inicialización y tarea lectura del ADC.
- archivo [adc.h](#)
Declaración de función de inicialización y tarea lectura del ADC.

4.2. Referencia del directorio main/display

Archivos

- archivo [display.c](#)
Funciones de consulta de los seteos hechos por el usuario, tarea que controla el display y posteo de eventos para controlar el display.
- archivo [display.h](#)
Declaración de funciones que de consulta de los seteos hechos por el usuario, tarea que controla el display y posteo de eventos para controlar el display.
- archivo [sh1106_graphics.c](#)
Funciones que permiten operar sobre el display.
- archivo [sh1106_graphics.h](#)
Declaración de funciones que permiten operar sobre el display.
- archivo [sh1106_i2c.c](#)
Funciones de hardware para el puerto I2C.
- archivo [sh1106_i2c.h](#)
Declaración de funciones de hardware para el puerto I2C.

4.3. Referencia del directorio main/io_control

Archivos

- archivo [io_control.c](#)
Funciones de control del relay, buzzer y tarea del uso del expansor de entradas y salidas y entradas y salidas aisladas directas al ESP32.
- archivo [io_control.h](#)
Declaración de funciones de control del relay, buzzer y tarea del uso del expansor de entradas y salidas y entradas y salidas aisladas directas al ESP32.
- archivo [MCP23017.c](#)
Declaración de funciones de inicialización, lectura y escritura a través del puerto I2C el dispositivo MCP23017.
- archivo [MCP23017.h](#)
Declaración de funciones de inicialización, lectura y escritura a través del puerto I2C el dispositivo MCP23017.
- archivo [mcp23017_defs.h](#)

4.4. Referencia del directorio main

Directorio

- directorio [adc](#)
- directorio [display](#)
- directorio [io_control](#)
- directorio [nvs](#)
- directorio [rtc](#)
- directorio [system](#)
- directorio [wifi](#)

Archivos

- archivo [LVFV_system.h](#)
Declaraciones de estructuras, variables, definiciones y estados generales del sistema utilizados en varios de los archivos.
- archivo [main.c](#)
Inicialización de tareas y periféricos que no tienen una tarea específica. El resto de los periféricos se inicializan en las tareas que son utilizados.

4.5. Referencia del directorio main/nvs

Archivos

- archivo [nvs.c](#)
Funciones de lectura y escritura de variables no volátiles. Carga en sistema además las variables de seguridad al momento de ser guardadas en la memoria.
- archivo [nvs.h](#)
Declaración de funciones de lectura y escritura de memoria no volátil. Útiles para los reset y no tener que reconfigurar siempre las variables básicas del sistema.

4.6. Referencia del directorio main/rtc

Archivos

- archivo [rtc.c](#)

Funciones de lectura y escritura del RTC y alarmas.

- archivo [rtc.h](#)

Header con prototipos de funciones básicas para el lectura y escritura del RTC y alarmas.

4.7. Referencia del directorio main/system

Archivos

- archivo [sysAdmin.c](#)

Archivo de funciones que permiten controlar las variables fundamentales del sistema que controla el motor, solo para poder ser representadas en el display. Las variables reales de sistema se encuentran en el STM32.

- archivo [sysAdmin.h](#)

Header con las funciones de funcionamiento del sistema.

- archivo [sysControl.c](#)

Archivo de funciones que controlan el sistema físico a través del puerto SPI contra el STM32. En función de las respuestas del STM32, es que se controlan las variables internas de sistema del ESP32.

- archivo [sysControl.h](#)

Declaraciones de funciones que controlan el sistema del STM32. Además se encontrarán los comandos y respuestas que admite el STM32.

4.8. Referencia del directorio main/wifi

Archivos

- archivo [form.c](#)

- archivo [form.h](#)

- archivo [wifi.c](#)

- archivo [wifi.h](#)

Capítulo 5

Documentación de estructuras de datos

5.1. Referencia de la estructura `bitmap_t`

Estructura que representa un carácter cualquiera. Donde de le debe inicializar un puntero a alguno de los caracteres, el alto, ancho, y su posición en x e y.

Campos de datos

- `const uint8_t * bitmap`
- `uint8_t w`
- `uint8_t h`
- `int x`
- `int y`

5.1.1. Descripción detallada

Estructura que representa un carácter cualquiera. Donde de le debe inicializar un puntero a alguno de los caracteres, el alto, ancho, y su posición en x e y.

Definición en la línea 176 del archivo [sh1106_graphics.c](#).

5.1.2. Documentación de campos

5.1.2.1. `bitmap`

```
const uint8_t* bitmap
```

Puntero al carácter

Definición en la línea 177 del archivo [sh1106_graphics.c](#).

5.1.2.2. h

```
uint8_t h
```

Alto del carácter

Definición en la línea 179 del archivo [sh1106_graphics.c](#).

5.1.2.3. w

```
uint8_t w
```

Ancho del carácter

Definición en la línea 178 del archivo [sh1106_graphics.c](#).

5.1.2.4. x

```
int x
```

Posición en X del carácter

Definición en la línea 180 del archivo [sh1106_graphics.c](#).

5.1.2.5. y

```
int y
```

Posición en Y del carácter

Definición en la línea 181 del archivo [sh1106_graphics.c](#).

La documentación de esta estructura está generada del siguiente archivo:

- [main/display/sh1106_graphics.c](#)

5.2. Referencia de la estructura frequency_settings_SH1106_t

```
#include <LVFV_system.h>
```

Campos de datos

- [frequency_settings_t frequency_settings](#)
- [sh1106_variable_lines_e * edit](#)
- [uint8_t edit_variable](#)
- [uint8_t * edit_flag](#)
- [uint8_t * multiplier](#)

5.2.1. Descripción detallada

Definición en la línea 127 del archivo [LVFV_system.h](#).

5.2.2. Documentación de campos

5.2.2.1. edit

```
sh1106_variable_lines_e* edit
```

Definición en la línea 129 del archivo [LVFV_system.h](#).

5.2.2.2. edit_flag

```
uint8_t* edit_flag
```

Definición en la línea 131 del archivo [LVFV_system.h](#).

5.2.2.3. edit_variable

```
uint8_t edit_variable
```

Definición en la línea 130 del archivo [LVFV_system.h](#).

5.2.2.4. frequency_settings

```
frequency_settings_t frequency_settings
```

Definición en la línea 128 del archivo [LVFV_system.h](#).

5.2.2.5. multiplier

```
uint8_t* multiplier
```

Definición en la línea 132 del archivo [LVFV_system.h](#).

La documentación de esta estructura está generada del siguiente archivo:

- main/[LVFV_system.h](#)

5.3. Referencia de la estructura frequency_settings_t

```
#include <LVFV_system.h>
```

Campos de datos

- `uint16_t freq_regime`
- `uint16_t acceleration`
- `uint16_t desacceleration`
- `uint16_t input_variable`

5.3.1. Descripción detallada

Definición en la línea 86 del archivo [LVFV_system.h](#).

5.3.2. Documentación de campos

5.3.2.1. acceleration

`uint16_t acceleration`

Definición en la línea 88 del archivo [LVFV_system.h](#).

5.3.2.2. desacceleration

`uint16_t desacceleration`

Definición en la línea 89 del archivo [LVFV_system.h](#).

5.3.2.3. freq_regime

`uint16_t freq_regime`

Definición en la línea 87 del archivo [LVFV_system.h](#).

5.3.2.4. input_variable

`uint16_t input_variable`

Definición en la línea 90 del archivo [LVFV_system.h](#).

La documentación de esta estructura está generada del siguiente archivo:

- main/[LVFV_system.h](#)

5.4. Referencia de la unión MCP23017_DEFVAL_t

```
#include <mcp23017_defs.h>
```

Campos de datos

```
■ uint8_t all
■ struct {
    uint8_t DEF0: 1
    uint8_t DEF1: 1
    uint8_t DEF2: 1
    uint8_t DEF3: 1
    uint8_t DEF4: 1
    uint8_t DEF5: 1
    uint8_t DEF6: 1
    uint8_t DEF7: 1
} bits
```

5.4.1. Descripción detallada

Definición en la línea 237 del archivo [mcp23017_defs.h](#).

5.4.2. Documentación de campos

5.4.2.1. all

```
uint8_t all
```

Unificación de la variable

Definición en la línea 238 del archivo [mcp23017_defs.h](#).

5.4.2.2. [struct]

```
struct { ... } bits
```

5.4.2.3. DEF0

```
uint8_t DEF0
```

Bit 0 de DEFVAL

Definición en la línea 240 del archivo [mcp23017_defs.h](#).

5.4.2.4. DEF1

```
uint8_t DEF1
```

Bit 1 de DEFVAL

Definición en la línea 241 del archivo [mcp23017_defs.h](#).

5.4.2.5. DEF2

uint8_t DEF2

Bit 2 de DEFVAL

Definición en la línea 242 del archivo [mcp23017_defs.h](#).

5.4.2.6. DEF3

uint8_t DEF3

Bit 3 de DEFVAL

Definición en la línea 243 del archivo [mcp23017_defs.h](#).

5.4.2.7. DEF4

uint8_t DEF4

Bit 4 de DEFVAL

Definición en la línea 244 del archivo [mcp23017_defs.h](#).

5.4.2.8. DEF5

uint8_t DEF5

Bit 5 de DEFVAL

Definición en la línea 245 del archivo [mcp23017_defs.h](#).

5.4.2.9. DEF6

uint8_t DEF6

Bit 6 de DEFVAL

Definición en la línea 246 del archivo [mcp23017_defs.h](#).

5.4.2.10. DEF7

uint8_t DEF7

Bit 7 de DEFVAL

Definición en la línea 247 del archivo [mcp23017_defs.h](#).

La documentación de esta unión está generada del siguiente archivo:

- [main/io_control/mcp23017_defs.h](#)

5.5. Referencia de la unión MCP23017_GPINTEN_t

```
#include <mcp23017_defs.h>
```

Campos de datos

- uint8_t all
- struct {
 - uint8_t GPINT0: 1
 - uint8_t GPINT1: 1
 - uint8_t GPINT2: 1
 - uint8_t GPINT3: 1
 - uint8_t GPINT4: 1
 - uint8_t GPINT5: 1
 - uint8_t GPINT6: 1
 - uint8_t GPINT7: 1}
- bits

5.5.1. Descripción detallada

Definición en la línea 153 del archivo [mcp23017_defs.h](#).

5.5.2. Documentación de campos

5.5.2.1. all

uint8_t all

Unificación de la variable

Definición en la línea 154 del archivo [mcp23017_defs.h](#).

5.5.2.2. [struct]

struct { ... } bits

5.5.2.3. GPINT0

uint8_t GPINT0

Bit 0 de GPINTEN

Definición en la línea 156 del archivo [mcp23017_defs.h](#).

5.5.2.4. GPINT1

```
uint8_t GPINT1
```

Bit 1 de GPINTEN

Definición en la línea 157 del archivo [mcp23017_defs.h](#).

5.5.2.5. GPINT2

```
uint8_t GPINT2
```

Bit 2 de GPINTEN

Definición en la línea 158 del archivo [mcp23017_defs.h](#).

5.5.2.6. GPINT3

```
uint8_t GPINT3
```

Bit 3 de GPINTEN

Definición en la línea 159 del archivo [mcp23017_defs.h](#).

5.5.2.7. GPINT4

```
uint8_t GPINT4
```

Bit 4 de GPINTEN

Definición en la línea 160 del archivo [mcp23017_defs.h](#).

5.5.2.8. GPINT5

```
uint8_t GPINT5
```

Bit 5 de GPINTEN

Definición en la línea 161 del archivo [mcp23017_defs.h](#).

5.5.2.9. GPINT6

```
uint8_t GPINT6
```

Bit 6 de GPINTEN

Definición en la línea 162 del archivo [mcp23017_defs.h](#).

5.5.2.10. GPINT7

```
uint8_t GPINT7
```

Bit 7 de GPINTEN

Definición en la línea 163 del archivo [mcp23017_defs.h](#).

La documentación de esta unión está generada del siguiente archivo:

- [main/io_control/mcp23017_defs.h](#)

5.6. Referencia de la unión MCP23017_GPIO_t

```
#include <mcp23017_defs.h>
```

Campos de datos

- `uint8_t all`
- `struct {`
 - `uint8_t GP0: 1`
 - `uint8_t GP1: 1`
 - `uint8_t GP2: 1`
 - `uint8_t GP3: 1`
 - `uint8_t GP4: 1`
 - `uint8_t GP5: 1`
 - `uint8_t GP6: 1`
 - `uint8_t GP7: 1``}` `bits`

5.6.1. Descripción detallada

Definición en la línea 195 del archivo [mcp23017_defs.h](#).

5.6.2. Documentación de campos

5.6.2.1. all

```
uint8_t all
```

Unificación de la variable

Definición en la línea 196 del archivo [mcp23017_defs.h](#).

5.6.2.2. [struct]

```
struct { ... } bits
```

5.6.2.3. GP0

```
uint8_t GP0
```

Bit 0 de GPIO

Definición en la línea 198 del archivo [mcp23017_defs.h](#).

5.6.2.4. GP1

```
uint8_t GP1
```

Bit 1 de GPIO

Definición en la línea 199 del archivo [mcp23017_defs.h](#).

5.6.2.5. GP2

```
uint8_t GP2
```

Bit 2 de GPIO

Definición en la línea 200 del archivo [mcp23017_defs.h](#).

5.6.2.6. GP3

```
uint8_t GP3
```

Bit 3 de GPIO

Definición en la línea 201 del archivo [mcp23017_defs.h](#).

5.6.2.7. GP4

```
uint8_t GP4
```

Bit 4 de GPIO

Definición en la línea 202 del archivo [mcp23017_defs.h](#).

5.6.2.8. GP5

```
uint8_t GP5
```

Bit 5 de GPIO

Definición en la línea 203 del archivo [mcp23017_defs.h](#).

5.6.2.9. GP6

uint8_t GP6

Bit 6 de GPIO

Definición en la línea 204 del archivo [mcp23017_defs.h](#).

5.6.2.10. GP7

uint8_t GP7

Bit 7 de GPIO

Definición en la línea 205 del archivo [mcp23017_defs.h](#).

La documentación de esta unión está generada del siguiente archivo:

- [main/io_control/mcp23017_defs.h](#)

5.7. Referencia de la unión MCP23017_GPPU_t

```
#include <mcp23017_defs.h>
```

Campos de datos

- uint8_t all
- struct {
 - uint8_t PU0: 1
 - uint8_t PU1: 1
 - uint8_t PU2: 1
 - uint8_t PU3: 1
 - uint8_t PU4: 1
 - uint8_t PU5: 1
 - uint8_t PU6: 1
 - uint8_t PU7: 1}
- bits

5.7.1. Descripción detallada

Definición en la línea 174 del archivo [mcp23017_defs.h](#).

5.7.2. Documentación de campos

5.7.2.1. all

uint8_t all

Unificación de la variable

Definición en la línea 175 del archivo [mcp23017_defs.h](#).

5.7.2.2. [struct]

```
struct { ... } bits
```

5.7.2.3. PU0

uint8_t PU0

Bit 0 de GPPU

Definición en la línea 177 del archivo [mcp23017_defs.h](#).

5.7.2.4. PU1

```
uint8_t PU1
```

Bit 1 de GPPU

Definición en la línea 178 del archivo [mcp23017_defs.h](#).

5.7.2.5. PU2

```
uint8_t PU2
```

Bit 2 de GPPU

Definición en la línea 179 del archivo [mcp23017_defs.h](#).

5.7.2.6. PU3

```
uint8_t PU3
```

Bit 3 de GPPU

Definición en la línea 180 del archivo [mcp23017_defs.h](#).

5.7.2.7. PU4

```
uint8_t PU4
```

Bit 4 de GPPU

Definición en la línea 181 del archivo [mcp23017_defs.h](#).

5.7.2.8. PU5

```
uint8_t PU5
```

Bit 5 de GPPU

Definición en la línea 182 del archivo [mcp23017_defs.h](#).

5.7.2.9. PU6

```
uint8_t PU6
```

Bit 6 de GPPU

Definición en la línea 183 del archivo [mcp23017_defs.h](#).

5.7.2.10. PU7

```
uint8_t PU7
```

Bit 7 de GPPU

Definición en la línea 184 del archivo [mcp23017_defs.h](#).

La documentación de esta unión está generada del siguiente archivo:

- [main/io_control/mcp23017_defs.h](#)

5.8. Referencia de la unión MCP23017_INTCAP_t

```
#include <mcp23017_defs.h>
```

Campos de datos

- `uint8_t all`
- `struct {`
 - `uint8_t ICP0: 1`
 - `uint8_t ICP1: 1`
 - `uint8_t ICP2: 1`
 - `uint8_t ICP3: 1`
 - `uint8_t ICP4: 1`
 - `uint8_t ICP5: 1`
 - `uint8_t ICP6: 1`
 - `uint8_t ICP7: 1``}` `bits`

5.8.1. Descripción detallada

Definición en la línea [321](#) del archivo [mcp23017_defs.h](#).

5.8.2. Documentación de campos

5.8.2.1. all

```
uint8_t all
```

Unificación de la variable

Definición en la línea [322](#) del archivo [mcp23017_defs.h](#).

5.8.2.2. [struct]

```
struct { ... } bits
```

5.8.2.3. ICP0

```
uint8_t ICP0
```

Bit 0 de INTCAP

Definición en la línea [324](#) del archivo [mcp23017_defs.h](#).

5.8.2.4. ICP1

```
uint8_t ICP1
```

Bit 1 de INTCAP

Definición en la línea [325](#) del archivo [mcp23017_defs.h](#).

5.8.2.5. ICP2

```
uint8_t ICP2
```

Bit 2 de INTCAP

Definición en la línea [326](#) del archivo [mcp23017_defs.h](#).

5.8.2.6. ICP3

```
uint8_t ICP3
```

Bit 3 de INTCAP

Definición en la línea [327](#) del archivo [mcp23017_defs.h](#).

5.8.2.7. ICP4

```
uint8_t ICP4
```

Bit 4 de INTCAP

Definición en la línea 328 del archivo [mcp23017_defs.h](#).

5.8.2.8. ICP5

```
uint8_t ICP5
```

Bit 5 de INTCAP

Definición en la línea 329 del archivo [mcp23017_defs.h](#).

5.8.2.9. ICP6

```
uint8_t ICP6
```

Bit 6 de INTCAP

Definición en la línea 330 del archivo [mcp23017_defs.h](#).

5.8.2.10. ICP7

```
uint8_t ICP7
```

Bit 7 de INTCAP

Definición en la línea 331 del archivo [mcp23017_defs.h](#).

La documentación de esta unión está generada del siguiente archivo:

- [main/io_control/mcp23017_defs.h](#)

5.9. Referencia de la unión MCP23017_INTCON_t

```
#include <mcp23017_defs.h>
```

Campos de datos

- `uint8_t all`
- `struct {`
 - `uint8_t IOC0: 1`
 - `uint8_t IOC1: 1`
 - `uint8_t IOC2: 1`
 - `uint8_t IOC3: 1`
 - `uint8_t IOC4: 1`
 - `uint8_t IOC5: 1`
 - `uint8_t IOC6: 1`
 - `uint8_t IOC7: 1``}` **bits**

5.9.1. Descripción detallada

Definición en la línea [258](#) del archivo [mcp23017_defs.h](#).

5.9.2. Documentación de campos

5.9.2.1. all

```
uint8_t all
```

Unificación de la variable

Definición en la línea [259](#) del archivo [mcp23017_defs.h](#).

5.9.2.2. [struct]

```
struct { ... } bits
```

5.9.2.3. IOC0

```
uint8_t IOC0
```

Bit 0 de INTCON

Definición en la línea [261](#) del archivo [mcp23017_defs.h](#).

5.9.2.4. IOC1

```
uint8_t IOC1
```

Bit 1 de INTCON

Definición en la línea [262](#) del archivo [mcp23017_defs.h](#).

5.9.2.5. IOC2

```
uint8_t IOC2
```

Bit 2 de INTCON

Definición en la línea [263](#) del archivo [mcp23017_defs.h](#).

5.9.2.6. IOC3

```
uint8_t IOC3
```

Bit 3 de INTCON

Definición en la línea [264](#) del archivo [mcp23017_defs.h](#).

5.9.2.7. IOC4

```
uint8_t IOC4
```

Bit 4 de INTCON

Definición en la línea 265 del archivo [mcp23017_defs.h](#).

5.9.2.8. IOC5

```
uint8_t IOC5
```

Bit 5 de INTCON

Definición en la línea 266 del archivo [mcp23017_defs.h](#).

5.9.2.9. IOC6

```
uint8_t IOC6
```

Bit 6 de INTCON

Definición en la línea 267 del archivo [mcp23017_defs.h](#).

5.9.2.10. IOC7

```
uint8_t IOC7
```

Bit 7 de INTCON

Definición en la línea 268 del archivo [mcp23017_defs.h](#).

La documentación de esta unión está generada del siguiente archivo:

- [main/io_control/mcp23017_defs.h](#)

5.10. Referencia de la unión MCP23017_INTF_t

```
#include <mcp23017_defs.h>
```

Campos de datos

- `uint8_t all`
- `struct {`
 - `uint8_t INT0: 1`
 - `uint8_t INT1: 1`
 - `uint8_t INT2: 1`
 - `uint8_t INT3: 1`
 - `uint8_t INT4: 1`
 - `uint8_t INT5: 1`
 - `uint8_t INT6: 1`
 - `uint8_t INT7: 1``}` **bits**

5.10.1. Descripción detallada

Definición en la línea 300 del archivo [mcp23017_defs.h](#).

5.10.2. Documentación de campos

5.10.2.1. all

```
uint8_t all
```

Unificación de la variable

Definición en la línea 301 del archivo [mcp23017_defs.h](#).

5.10.2.2. [struct]

```
struct { ... } bits
```

5.10.2.3. INT0

```
uint8_t INT0
```

Bit 0 de INTF

Definición en la línea 303 del archivo [mcp23017_defs.h](#).

5.10.2.4. INT1

```
uint8_t INT1
```

Bit 1 de INTF

Definición en la línea 304 del archivo [mcp23017_defs.h](#).

5.10.2.5. INT2

```
uint8_t INT2
```

Bit 2 de INTF

Definición en la línea 305 del archivo [mcp23017_defs.h](#).

5.10.2.6. INT3

```
uint8_t INT3
```

Bit 3 de INTF

Definición en la línea 306 del archivo [mcp23017_defs.h](#).

5.10.2.7. INT4

```
uint8_t INT4
```

Bit 4 de INTF

Definición en la línea 307 del archivo [mcp23017_defs.h](#).

5.10.2.8. INT5

```
uint8_t INT5
```

Bit 5 de INTF

Definición en la línea 308 del archivo [mcp23017_defs.h](#).

5.10.2.9. INT6

```
uint8_t INT6
```

Bit 6 de INTF

Definición en la línea 309 del archivo [mcp23017_defs.h](#).

5.10.2.10. INT7

```
uint8_t INT7
```

Bit 7 de INTF

Definición en la línea 310 del archivo [mcp23017_defs.h](#).

La documentación de esta unión está generada del siguiente archivo:

- [main/io_control/mcp23017_defs.h](#)

5.11. Referencia de la unión MCP23017_IOCON_t

```
#include <mcp23017_defs.h>
```

Campos de datos

- `uint8_t all`
- `struct {`
 - `uint8_t reserved: 1`
 - `uint8_t INTPOL: 1`
 - `uint8_t ODR: 1`
 - `uint8_t reserved_2: 1`
 - `uint8_t DISSLW: 1`
 - `uint8_t SEQOP: 1`
 - `uint8_t MIRROR: 1`
 - `uint8_t BANK: 1``}` **bits**

5.11.1. Descripción detallada

Definición en la línea [279](#) del archivo [mcp23017_defs.h](#).

5.11.2. Documentación de campos

5.11.2.1. all

```
uint8_t all
```

Unificación de la variable

Definición en la línea [280](#) del archivo [mcp23017_defs.h](#).

5.11.2.2. BANK

```
uint8_t BANK
```

Bit 7 de IOCON - Configuración del banco de memoria. Separación o no de Puertos A y B

Definición en la línea [289](#) del archivo [mcp23017_defs.h](#).

5.11.2.3. [struct]

```
struct { ... } bits
```

5.11.2.4. DISSLW

```
uint8_t DISSLW
```

Bit 4 de IOCON - Configuración de velocidad del MCP23017

Definición en la línea [286](#) del archivo [mcp23017_defs.h](#).

5.11.2.5. INTPOL

```
uint8_t INTPOL
```

Bit 1 de IOCON - Configuración de polaridad de las entradas

Definición en la línea [283](#) del archivo [mcp23017_defs.h](#).

5.11.2.6. MIRROR

```
uint8_t MIRROR
```

Bit 6 de IOCON - Configuración de espejado de pines de interrupción

Definición en la línea [288](#) del archivo [mcp23017_defs.h](#).

5.11.2.7. ODR

```
uint8_t ODR
```

Bit 2 de IOCON - Configuración de formato de salida de pines INT

Definición en la línea 284 del archivo [mcp23017_defs.h](#).

5.11.2.8. reserved

```
uint8_t reserved
```

Bit 0 de IOCON - Sin uso

Definición en la línea 282 del archivo [mcp23017_defs.h](#).

5.11.2.9. reserved_2

```
uint8_t reserved_2
```

Bit 3 de IOCON - Sin uso

Definición en la línea 285 del archivo [mcp23017_defs.h](#).

5.11.2.10. SEQOP

```
uint8_t SEQOP
```

Bit 5 de IOCON - Configuración de autoincremento de banco de memoria luego de un acceso

Definición en la línea 287 del archivo [mcp23017_defs.h](#).

La documentación de esta unión está generada del siguiente archivo:

- [main/io_control/mcp23017_defs.h](#)

5.12. Referencia de la unión MCP23017_IODIR_t

```
#include <mcp23017_defs.h>
```

Campos de datos

- `uint8_t all`
- `struct {`
 - `uint8_t IO0: 1`
 - `uint8_t IO1: 1`
 - `uint8_t IO2: 1`
 - `uint8_t IO3: 1`
 - `uint8_t IO4: 1`
 - `uint8_t IO5: 1`
 - `uint8_t IO6: 1`
 - `uint8_t IO7: 1``}` **bits**

5.12.1. Descripción detallada

Definición en la línea 111 del archivo [mcp23017_defs.h](#).

5.12.2. Documentación de campos

5.12.2.1. all

```
uint8_t all
```

Unificación de la variable

Definición en la línea 112 del archivo [mcp23017_defs.h](#).

5.12.2.2. [struct]

```
struct { ... } bits
```

5.12.2.3. IO0

```
uint8_t IO0
```

Bit 0 de IODIR

Definición en la línea 114 del archivo [mcp23017_defs.h](#).

5.12.2.4. IO1

```
uint8_t IO1
```

Bit 1 de IODIR

Definición en la línea 115 del archivo [mcp23017_defs.h](#).

5.12.2.5. IO2

```
uint8_t IO2
```

Bit 2 de IODIR

Definición en la línea 116 del archivo [mcp23017_defs.h](#).

5.12.2.6. IO3

```
uint8_t IO3
```

Bit 3 de IODIR

Definición en la línea 117 del archivo [mcp23017_defs.h](#).

5.12.2.7. IO4

```
uint8_t IO4
```

Bit 4 de IODIR

Definición en la línea 118 del archivo [mcp23017_defs.h](#).

5.12.2.8. IO5

```
uint8_t IO5
```

Bit 5 de IODIR

Definición en la línea 119 del archivo [mcp23017_defs.h](#).

5.12.2.9. IO6

```
uint8_t IO6
```

Bit 6 de IODIR

Definición en la línea 120 del archivo [mcp23017_defs.h](#).

5.12.2.10. IO7

```
uint8_t IO7
```

Bit 7 de IODIR

Definición en la línea 121 del archivo [mcp23017_defs.h](#).

La documentación de esta unión está generada del siguiente archivo:

- [main/io_control/mcp23017_defs.h](#)

5.13. Referencia de la unión MCP23017_IPOL_t

```
#include <mcp23017_defs.h>
```

Campos de datos

- `uint8_t all`
- `struct {`
 - `uint8_t IP0: 1`
 - `uint8_t IP1: 1`
 - `uint8_t IP2: 1`
 - `uint8_t IP3: 1`
 - `uint8_t IP4: 1`
 - `uint8_t IP5: 1`
 - `uint8_t IP6: 1`
 - `uint8_t IP7: 1``}` **bits**

5.13.1. Descripción detallada

Definición en la línea 132 del archivo [mcp23017_defs.h](#).

5.13.2. Documentación de campos

5.13.2.1. all

```
uint8_t all
```

Unificación de la variable

Definición en la línea 133 del archivo [mcp23017_defs.h](#).

5.13.2.2. [struct]

```
struct { ... } bits
```

5.13.2.3. IPO

```
uint8_t IPO
```

Bit 0 de IPOL

Definición en la línea 135 del archivo [mcp23017_defs.h](#).

5.13.2.4. IP1

```
uint8_t IP1
```

Bit 1 de IPOL

Definición en la línea 136 del archivo [mcp23017_defs.h](#).

5.13.2.5. IP2

```
uint8_t IP2
```

Bit 2 de IPOL

Definición en la línea 137 del archivo [mcp23017_defs.h](#).

5.13.2.6. IP3

```
uint8_t IP3
```

Bit 3 de IPOL

Definición en la línea 138 del archivo [mcp23017_defs.h](#).

5.13.2.7. IP4

```
uint8_t IP4
```

Bit 4 de IPOL

Definición en la línea 139 del archivo [mcp23017_defs.h](#).

5.13.2.8. IP5

```
uint8_t IP5
```

Bit 5 de IPOL

Definición en la línea 140 del archivo [mcp23017_defs.h](#).

5.13.2.9. IP6

```
uint8_t IP6
```

Bit 6 de IPOL

Definición en la línea 141 del archivo [mcp23017_defs.h](#).

5.13.2.10. IP7

```
uint8_t IP7
```

Bit 7 de IPOL

Definición en la línea 142 del archivo [mcp23017_defs.h](#).

La documentación de esta unión está generada del siguiente archivo:

- [main/io_control/mcp23017_defs.h](#)

5.14. Referencia de la unión MCP23017_OLAT_t

```
#include <mcp23017_defs.h>
```

Campos de datos

- `uint8_t all`
- `struct {`
 - `uint8_t OL0: 1`
 - `uint8_t OL1: 1`
 - `uint8_t OL2: 1`
 - `uint8_t OL3: 1`
 - `uint8_t OL4: 1`
 - `uint8_t OL5: 1`
 - `uint8_t OL6: 1`
 - `uint8_t OL7: 1``}` **bits**

5.14.1. Descripción detallada

Definición en la línea [216](#) del archivo [mcp23017_defs.h](#).

5.14.2. Documentación de campos

5.14.2.1. all

```
uint8_t all
```

Unificación de la variable

Definición en la línea [217](#) del archivo [mcp23017_defs.h](#).

5.14.2.2. [struct]

```
struct { ... } bits
```

5.14.2.3. OL0

```
uint8_t OL0
```

Bit 0 de OLAT

Definición en la línea [219](#) del archivo [mcp23017_defs.h](#).

5.14.2.4. OL1

```
uint8_t OL1
```

Bit 1 de OLAT

Definición en la línea [220](#) del archivo [mcp23017_defs.h](#).

5.14.2.5. OL2

```
uint8_t OL2
```

Bit 2 de OLAT

Definición en la línea [221](#) del archivo [mcp23017_defs.h](#).

5.14.2.6. OL3

```
uint8_t OL3
```

Bit 3 de OLAT

Definición en la línea [222](#) del archivo [mcp23017_defs.h](#).

5.14.2.7. OL4

`uint8_t OL4`

Bit 4 de OLAT

Definición en la línea 223 del archivo [mcp23017_defs.h](#).

5.14.2.8. OL5

`uint8_t OL5`

Bit 5 de OLAT

Definición en la línea 224 del archivo [mcp23017_defs.h](#).

5.14.2.9. OL6

`uint8_t OL6`

Bit 6 de OLAT

Definición en la línea 225 del archivo [mcp23017_defs.h](#).

5.14.2.10. OL7

`uint8_t OL7`

Bit 7 de OLAT

Definición en la línea 226 del archivo [mcp23017_defs.h](#).

La documentación de esta unión está generada del siguiente archivo:

- [main/io_control/mcp23017_defs.h](#)

5.15. Referencia de la estructura `rtc_alarms_t`

Campos de datos

- `esp_timer_handle_t start_timer`
- `esp_timer_handle_t stop_timer`

5.15.1. Descripción detallada

Definición en la línea 24 del archivo [rtc.c](#).

5.15.2. Documentación de campos

5.15.2.1. start_timer

```
esp_timer_handle_t start_timer
```

Definición en la línea 25 del archivo [rtc.c](#).

5.15.2.2. stop_timer

```
esp_timer_handle_t stop_timer
```

Definición en la línea 26 del archivo [rtc.c](#).

La documentación de esta estructura está generada del siguiente archivo:

- main/rtc/[rtc.c](#)

5.16. Referencia de la estructura security_settings_SH1106_t

```
#include <LVFV_system.h>
```

Campos de datos

- `security_settings_t` `security_settings`
- `sh1106_variable_lines_e` * `edit`
- `uint8_t` `edit_variable`
- `uint8_t` * `edit_flag`
- `uint8_t` * `multiplier`

5.16.1. Descripción detallada

Definición en la línea 143 del archivo [LVFV_system.h](#).

5.16.2. Documentación de campos

5.16.2.1. edit

```
sh1106_variable_lines_e* edit
```

Definición en la línea 145 del archivo [LVFV_system.h](#).

5.16.2.2. `edit_flag`

```
uint8_t* edit_flag
```

Definición en la línea 147 del archivo [LVFV_system.h](#).

5.16.2.3. `edit_variable`

```
uint8_t edit_variable
```

Definición en la línea 146 del archivo [LVFV_system.h](#).

5.16.2.4. `multiplier`

```
uint8_t* multiplier
```

Definición en la línea 148 del archivo [LVFV_system.h](#).

5.16.2.5. `seccurity_settings`

```
seccurity_settings_t seccurity_settings
```

Definición en la línea 144 del archivo [LVFV_system.h](#).

La documentación de esta estructura está generada del siguiente archivo:

- main/[LVFV_system.h](#)

5.17. Referencia de la estructura `seccurity_settings_t`

```
#include <LVFV_system.h>
```

Campos de datos

- `uint16_t vbus_min`
- `uint16_t ibus_max`

5.17.1. Descripción detallada

Definición en la línea 99 del archivo [LVFV_system.h](#).

5.17.2. Documentación de campos

5.17.2.1. ibus_max

```
uint16_t ibus_max
```

Definición en la línea 101 del archivo [LVFV_system.h](#).

5.17.2.2. vbus_min

```
uint16_t vbus_min
```

Definición en la línea 100 del archivo [LVFV_system.h](#).

La documentación de esta estructura está generada del siguiente archivo:

- main/[LVFV_system.h](#)

5.18. Referencia de la estructura sh1106_t

```
#include <sh1106_graphics.h>
```

Campos de datos

- uint8_t width
- uint8_t height
- uint8_t rotation
- uint8_t buffer [128 *8]

5.18.1. Descripción detallada

Definición en la línea 13 del archivo [sh1106_graphics.h](#).

5.18.2. Documentación de campos

5.18.2.1. buffer

```
uint8_t buffer[128 *8]
```

Definición en la línea 17 del archivo [sh1106_graphics.h](#).

5.18.2.2. height

```
uint8_t height
```

Definición en la línea 15 del archivo [sh1106_graphics.h](#).

5.18.2.3. rotation

```
uint8_t rotation
```

Definición en la línea 16 del archivo [sh1106_graphics.h](#).

5.18.2.4. width

```
uint8_t width
```

Definición en la línea 14 del archivo [sh1106_graphics.h](#).

La documentación de esta estructura está generada del siguiente archivo:

- [main/display/sh1106_graphics.h](#)

5.19. Referencia de la estructura spi_cmd_item_t

Estructura de datos que contiene un comando a enviar por SPI, un argumento que se desea enviar al STM32 y un dato que que pueda recibirse como respuesta en consecuencia.

```
#include <sysControl.h>
```

Campos de datos

- [SPI_Request request](#)
- int [getValue](#)
- int [setValue](#)

5.19.1. Descripción detallada

Estructura de datos que contiene un comando a enviar por SPI, un argumento que se desea enviar al STM32 y un dato que que pueda recibirse como respuesta en consecuencia.

Nota

`getValue` no siempre recibe un dato como respuesta, solo en los comandos "GET"

Definición en la línea 59 del archivo [sysControl.h](#).

5.19.2. Documentación de campos

5.19.2.1. getValue

```
int getValue
```

Definición en la línea 61 del archivo [sysControl.h](#).

5.19.2.2. request

```
SPI_Request request
```

Definición en la línea 60 del archivo [sysControl.h](#).

5.19.2.3. setValue

```
int setValue
```

Definición en la línea 62 del archivo [sysControl.h](#).

La documentación de esta estructura está generada del siguiente archivo:

- main/system/[sysControl.h](#)

5.20. Referencia de la estructura system_status_t

Estructura con las variables necesarias para establecer las condiciones de trabajo del motor.

```
#include <LVFV_system.h>
```

Campos de datos

- uint16_t [frequency](#)
- uint16_t [frequency_destiny](#)
- uint16_t [vbus_min](#)
- uint16_t [ibus_max](#)
- uint16_t [acceleration](#)
- uint16_t [desacceleration](#)
- uint8_t [inputs_status](#)
- [system_status_e](#) [status](#)
- uint8_t [emergency_signals](#)

5.20.1. Descripción detallada

Estructura con las variables necesarias para establecer las condiciones de trabajo del motor.

Definición en la línea 109 del archivo [LVFV_system.h](#).

5.20.2. Documentación de campos

5.20.2.1. acceleration

```
uint16_t acceleration
```

Definición en la línea 114 del archivo [LVFV_system.h](#).

5.20.2.2. desacceleration

```
uint16_t desacceleration
```

Definición en la línea 115 del archivo [LVFV_system.h](#).

5.20.2.3. emergency_signals

```
uint8_t emergency_signals
```

Definición en la línea 118 del archivo [LVFV_system.h](#).

5.20.2.4. frequency

```
uint16_t frequency
```

Definición en la línea 110 del archivo [LVFV_system.h](#).

5.20.2.5. frequency_destiny

```
uint16_t frequency_destiny
```

Definición en la línea 111 del archivo [LVFV_system.h](#).

5.20.2.6. ibus_max

```
uint16_t ibus_max
```

Definición en la línea 113 del archivo [LVFV_system.h](#).

5.20.2.7. inputs_status

```
uint8_t inputs_status
```

Definición en la línea 116 del archivo [LVFV_system.h](#).

5.20.2.8. status

```
system_status_e status
```

Definición en la línea 117 del archivo [LVFV_system.h](#).

5.20.2.9. vbus_min

```
uint16_t vbus_min
```

Definición en la línea 112 del archivo [LVFV_system.h](#).

La documentación de esta estructura está generada del siguiente archivo:

- main/[LVFV_system.h](#)

5.21. Referencia de la estructura time_settings_SH1106_t

```
#include <LVFV_system.h>
```

Campos de datos

- `time_settings_t` `time_settings`
- `sh1106_variable_lines_e` * `edit`
- `uint8_t` `edit_variable`
- `uint8_t` * `edit_flag`
- `uint8_t` * `multiplier`

5.21.1. Descripción detallada

Definición en la línea 135 del archivo [LVFV_system.h](#).

5.21.2. Documentación de campos

5.21.2.1. edit

```
sh1106_variable_lines_e* edit
```

Definición en la línea 137 del archivo [LVFV_system.h](#).

5.21.2.2. edit_flag

```
uint8_t* edit_flag
```

Definición en la línea 139 del archivo [LVFV_system.h](#).

5.21.2.3. edit_variable

```
uint8_t edit_variable
```

Definición en la línea 138 del archivo [LVFV_system.h](#).

5.21.2.4. multiplier

```
uint8_t* multiplier
```

Definición en la línea 140 del archivo [LVFV_system.h](#).

5.21.2.5. time_settings

```
time_settings_t time_settings
```

Definición en la línea 136 del archivo [LVFV_system.h](#).

La documentación de esta estructura está generada del siguiente archivo:

- main/[LVFV_system.h](#)

5.22. Referencia de la estructura time_settings_t

```
#include <LVFV_system.h>
```

Campos de datos

- struct tm * [time_system](#)
- struct tm * [time_start](#)
- struct tm * [time_stop](#)

5.22.1. Descripción detallada

Definición en la línea 93 del archivo [LVFV_system.h](#).

5.22.2. Documentación de campos

5.22.2.1. time_start

```
struct tm* time_start
```

Definición en la línea 95 del archivo [LVFV_system.h](#).

5.22.2.2. time_stop

```
struct tm* time_stop
```

Definición en la línea 96 del archivo [LVFV_system.h](#).

5.22.2.3. time_system

```
struct tm* time_system
```

Definición en la línea 94 del archivo [LVFV_system.h](#).

La documentación de esta estructura está generada del siguiente archivo:

- main/[LVFV_system.h](#)

Capítulo 6

Documentación de archivos

6.1. Referencia del archivo main/adc/adc.c

Funcion de inicialización y tarea lectura del ADC.

```
#include <stdio.h>
#include <math.h>
#include "freertos/FreRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "esp_log.h"
#include "esp_err.h"
#include "esp_adc/adc_oneshot.h"
#include "esp_adc/adc_cali.h"
#include "esp_adc/adc_cali_scheme.h"
#include "esp_adc/adc_continuous.h"
#include "../LVFV_system.h"
#include "../adc/adc.h"
#include "../System/SysAdmin.h"
```

defines

- #define ADC_PERIOD_MS 100
- #define ADC_UNIT_USED ADC_UNIT_1
- #define ADC_CH_GPIO34 ADC_CHANNEL_6
- #define ADC_CH_GPIO35 ADC_CHANNEL_7
- #define ADC_CH_GPIO36 ADC_CHANNEL_0
- #define ADC_CH_GPIO39 ADC_CHANNEL_3
- #define ADC_ATTEN_USED ((adc_atten_t) 3)

Funciones

- static esp_err_t **adc_cali_try_init** (adc_unit_t unit, adc_channel_t ch, adc_atten_t atten, adc_cali_handle_t *out)
Estandariza las mediciones del ADC en función de la tensión que representa el pin físico en milis voltios.
- esp_err_t **adc_init** (void)
Configura los 4 ADC necesarios para el sistema y los calibra para obtener las mediciones en desde 0 a 3125mV.
- void **adc_task** (void *arg)
Función programada como alarma para finalizar el funcionamiento del motor.
- bool **readADC** (void)
Desencola mediciones obtenidas desde el ADC para tensión y corriente del bus de continua, compara con los parámetros seteados y dispara el estado de emergencia si es necesario.

Variables

- static const char * **TAG** = "ADC"
- static adc_oneshot_unit_handle_t **s_adc** = NULL
- static adc_cali_handle_t **calibration_bus_voltage** = NULL
- static adc_cali_handle_t **calibration_current** = NULL
- static adc_cali_handle_t **calibration_5V_source** = NULL
- static adc_cali_handle_t **calibration_3V3_source** = NULL
- static bool **calibration_bus_voltage_ok** = false
- static bool **calibration_current_ok** = false
- static bool **calibration_5V_source_ok** = false
- static bool **calibration_3V3_source_ok** = false
- QueueHandle_t **bus_meas_evt_queue** = NULL

6.1.1. Descripción detallada

Función de inicialización y tarea lectura del ADC.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo [adc.c](#).

6.1.2. Documentación de «define»

6.1.2.1. ADC_ATTEN_USED

```
#define ADC_ATTEN_USED ( (adc_atten_t) 3 )
```

Definición en la línea [36](#) del archivo [adc.c](#).

6.1.2.2. ADC_CH_GPIO34

```
#define ADC_CH_GPIO34 ADC_CHANNEL_6
```

Definición en la línea [31](#) del archivo [adc.c](#).

6.1.2.3. ADC_CH_GPIO35

```
#define ADC_CH_GPIO35 ADC_CHANNEL_7
```

Definición en la línea 32 del archivo [adc.c](#).

6.1.2.4. ADC_CH_GPIO36

```
#define ADC_CH_GPIO36 ADC_CHANNEL_0
```

Definición en la línea 33 del archivo [adc.c](#).

6.1.2.5. ADC_CH_GPIO39

```
#define ADC_CH_GPIO39 ADC_CHANNEL_3
```

Definición en la línea 34 del archivo [adc.c](#).

6.1.2.6. ADC_PERIOD_MS

```
#define ADC_PERIOD_MS 100
```

Parámetros

in	arg	Sin uso.
----	-----	----------

Valores devueltos

- | | |
|---|--|
| - | ESP_OK: Si inicialización y calibración de todos los ADC fue satisfactoria <ul style="list-style-type: none">ESP_ERR_INVALID_ARG: Si la inicialización de algún ADC fallóESP_ERR_INVALID_STATE: Si las calibraciones de algún ADC falló |
|---|--|

Definición en la línea 28 del archivo [adc.c](#).

6.1.2.7. ADC_UNIT_USED

```
#define ADC_UNIT_USED ADC_UNIT_1
```

Definición en la línea 30 del archivo [adc.c](#).

6.1.3. Documentación de funciones

6.1.3.1. adc_cali_try_init()

```
esp_err_t adc_cali_try_init (
    adc_unit_t unit,
    adc_channel_t ch,
    adc_atten_t atten,
    adc_cali_handle_t * out) [static]
```

Estandariza las mediciones del ADC en función de la tensión que representa el pin físico en mili volts.

Denera la configuración necesaria para que las mediciones pasen de cuentas a mili volts desde 0 a 3125 mili volts

Parámetros

in	<i>uint</i>	Unidad del ADC utilizada. Puede tomar valores ADC_UNIT_1 o ADC_UNIT_2
in	<i>ch</i>	Canal del ADC utilizado. Puede tomar valores ADC_CHANNEL_0, ADC_CHANNEL_1, ADC_CHANNEL_2, ADC_CHANNEL_3, ADC_CHANNEL_4, ADC_CHANNEL_5, ADC_CHANNEL_6, ADC_CHANNEL_7, ADC_CHANNEL_8 o ADC_CHANNEL_9.
in	<i>atten</i>	Atenuación configurada en el ADC. Puede tomar valores ADC_ATTEN_DB_0, ADC_ATTEN_DB_2_5, ADC_ATTEN_DB_6 o ADC_ATTEN_DB_12
in	<i>out</i>	Puntero al handler del ADC utilizado.

Valores devueltos

- ESP_OK: Si la calibración del ADC fue satisfactoria
 - ESP_ERR_INVALID_ARG: Si los argumentos de calibración fueron incorrectos
 - ESP_ERR_NO_MEM: No hay suficiente memoria
 - ESP_ERR_NOT_SUPPORTED: Los eFose no están correctamente inicializados
 - ESP_ERR_INVALID_ARG: Si out es un puntero nulo

Definición en la línea 80 del archivo [adc.c](#).

6.1.3.2. adc_init()

```
esp_err_t adc_init (
    void )
```

Configura los 4 ADC necesarios para el sistema y los calibra para obtener las mediciones en desde 0 a 3125mV.

Configura por las entradas el canal 0 (GPIO36) el nivel de tensión de fuente de 5V, 3 (GPIO39) el nivel de tensión de fuente de 3.3V, 6 (GPIO34) tensión de bus de contínua y 7 (GPIO35) corriente de bus de contínua cada

Definición en la línea 97 del archivo [adc.c](#).

6.1.3.3. adc_task()

```
void adc_task (
    void * arg)
```

Función programada como alarma para finalizar el funcionamiento del motor.

Lee por las entradas el canal 0 (GPIO36) el nivel de tensión de fuente de 5V, 3 (GPIO39) el nivel de tensión de fuente de 3.3V, 6 (GPIO34) tensión de bus de contínua y 7 (GPIO35) corriente de bus de contínua cada

Definición en la línea 146 del archivo [adc.c](#).

6.1.3.4. `readADC()`

```
bool readADC (
    void )
```

Desencola mediciones obtenidas desde el ADC para tensión y corriente del bus de continua, compara con los parámetros seteados y dispara el estado de emergencia si es necesario.

Consulta si existe alguna medición encolada en `bus_meas_evt_queue` sin ser bloqueante y envía esas mediciones a analizar por el sistema para evaluar si corresponde o no un estado de emergencia.

Valores devueltos

- | | |
|---|--|
| - | true: Si el sistema está inicializado y las mediciones pueden ser obtenidas <ul style="list-style-type: none">• false: Si el sistema todavía no inició y las mediciones que pudo haber obtenido fueron inválidas |
|---|--|

Definición en la línea 239 del archivo [adc.c](#).

6.1.4. Documentación de variables

6.1.4.1. bus_meas_evt_queue

```
QueueHandle_t bus_meas_evt_queue = NULL
```

Definición en la línea 52 del archivo [adc.c](#).

6.1.4.2. calibration_3V3_source

```
adc_cali_handle_t calibration_3V3_source = NULL [static]
```

Definición en la línea 45 del archivo [adc.c](#).

6.1.4.3. calibration_3V3_source_ok

```
bool calibration_3V3_source_ok = false [static]
```

Definición en la línea 50 del archivo [adc.c](#).

6.1.4.4. calibration_5V_source

```
adc_cali_handle_t calibration_5V_source = NULL [static]
```

Definición en la línea 44 del archivo [adc.c](#).

6.1.4.5. calibration_5V_source_ok

```
bool calibration_5V_source_ok = false [static]
```

Definición en la línea 49 del archivo [adc.c](#).

6.1.4.6. calibration_bus_voltage

```
adc_cali_handle_t calibration_bus_voltage = NULL [static]
```

Definición en la línea 42 del archivo [adc.c](#).

6.1.4.7. `calibration_bus_voltage_ok`

```
bool calibration_bus_voltage_ok = false [static]
```

Definición en la línea 47 del archivo [adc.c](#).

6.1.4.8. `calibration_current`

```
adc_cali_handle_t calibration_current = NULL [static]
```

Definición en la línea 43 del archivo [adc.c](#).

6.1.4.9. `calibration_current_ok`

```
bool calibration_current_ok = false [static]
```

Definición en la línea 48 del archivo [adc.c](#).

6.1.4.10. `s_adc`

```
adc_oneshot_unit_handle_t s_adc = NULL [static]
```

Definición en la línea 40 del archivo [adc.c](#).

6.1.4.11. `TAG`

```
const char* TAG = "ADC" [static]
```

Definición en la línea 38 del archivo [adc.c](#).

6.2. `adc.c`

[Ir a la documentación de este archivo.](#)

```
00001
00009 #include <stdio.h>
00010 #include <math.h>
00011
00012 #include "freertos/FreeRTOS.h"
00013 #include "freertos/task.h"
00014 #include "freertos/queue.h"
00015
00016 #include "esp_log.h"
00017 #include "esp_err.h"
00018
00019 #include "esp_adcadc_oneshot.h"
00020 #include "esp_adcadc.h"
00021 #include "esp_adcadc_cali_scheme.h"
00022 #include "esp_adcadc_continuous.h"
00023
00024 #include "../LVFV_system.h"
00025 #include "../adc/adc.h"
00026 #include "../System/SysAdmin.h"
00027
00028 #define ADC_PERIOD_MS          100
00029
00030 #define ADC_UNIT_USED           ADC_UNIT_1
```

```

00031 #define ADC_CH_GPIO34          ADC_CHANNEL_6    // GPIO34
00032 #define ADC_CH_GPIO35          ADC_CHANNEL_7    // GPIO35
00033 #define ADC_CH_GPIO36          ADC_CHANNEL_0    // GPIO36
00034 #define ADC_CH_GPIO39          ADC_CHANNEL_3    // GPIO39
00035
00036 #define ADC_ATTEN_USED        ( (adc_atten_t) 3 ) // ~3.3-3.6 V FS aprox.
00037
00038 static const char *TAG = "ADC";
00039
00040 static adc_oneshot_unit_handle_t s_adc = NULL;
00041
00042 static adc_cali_handle_t calibration_bus_voltage = NULL;
00043 static adc_cali_handle_t calibration_current = NULL;
00044 static adc_cali_handle_t calibration_5V_source = NULL;
00045 static adc_cali_handle_t calibration_3V3_source = NULL;
00046
00047 static bool calibration_bus_voltage_ok = false;
00048 static bool calibration_current_ok = false;
00049 static bool calibration_5V_source_ok = false;
00050 static bool calibration_3V3_source_ok = false;
00051
00052 QueueHandle_t bus_meas_evt_queue = NULL;
00053
00080 static esp_err_t adc_cali_try_init(adc_unit_t unit, adc_channel_t ch, adc_atten_t atten,
00081     adc_cali_handle_t *out) {
00082     esp_err_t err;
00083
00084     if (out == NULL) {
00085         return ESP_ERR_INVALID_ARG;
00086     }
00087
00088     adc_cali_line_fitting_config_t cfg2 = {
00089         .unit_id = unit,
00090         .atten = atten,
00091         .bitwidth = ADC_BITWIDTH_DEFAULT,
00092     };
00093     err = adc_cali_create_scheme_line_fitting(&cfg2, out);
00094
00095     return err;
00096 }
00097 esp_err_t adc_init(void) {
00098
00099     esp_err_t retval;
00100
00101     adc_oneshot_unit_init_cfg_t unit_cfg = {
00102         .unit_id = ADC_UNIT_USED,
00103     };
00104
00105     adc_oneshot_new_unit(&unit_cfg, &s_adc);
00106
00107     adc_oneshot_chan_cfg_t ch_cfg = {
00108         .bitwidth = ADC_BITWIDTH_DEFAULT,
00109         .atten = ADC_ATTEN_USED,
00110     };
00111
00112     retval = adc_oneshot_config_channel(s_adc, ADC_CH_GPIO34, &ch_cfg);
00113     if (retval != ESP_OK) {
00114         return retval;
00115     }
00116
00117     retval = adc_oneshot_config_channel(s_adc, ADC_CH_GPIO35, &ch_cfg);
00118     if (retval != ESP_OK) {
00119         return retval;
00120     }
00121
00122     retval = adc_oneshot_config_channel(s_adc, ADC_CH_GPIO36, &ch_cfg);
00123     if (retval != ESP_OK) {
00124         return retval;
00125     }
00126
00127     retval = adc_oneshot_config_channel(s_adc, ADC_CH_GPIO39, &ch_cfg);
00128     if (retval != ESP_OK) {
00129         return retval;
00130     }
00131
00132     /* Calibración (si el chip lo soporta) */
00133     calibration_5V_source_ok = adc_cali_try_init(ADC_UNIT_USED, ADC_CH_GPIO36, ADC_ATTEN_USED,
00134         &calibration_5V_source);
00135     calibration_3V3_source_ok = adc_cali_try_init(ADC_UNIT_USED, ADC_CH_GPIO39, ADC_ATTEN_USED,
00136         &calibration_3V3_source);
00137     calibration_bus_voltage_ok = adc_cali_try_init(ADC_UNIT_USED, ADC_CH_GPIO34, ADC_ATTEN_USED,
00138         &calibration_bus_voltage);
00139     calibration_current_ok = adc_cali_try_init(ADC_UNIT_USED, ADC_CH_GPIO35, ADC_ATTEN_USED,
00140         &calibration_current);
00141
00142     ESP_LOGI(TAG, "Calibración: 5v Source =%s", calibration_5V_source_ok == ESP_OK ? "ON" : "OFF");

```

```

00139     ESP_LOGI(TAG, "Calibración: 3.3v Source =%s", calibration_3V3_source_ok == ESP_OK ? "ON" : "OFF");
00140     ESP_LOGI(TAG, "Calibración: DC bus voltage =%s", calibration_bus_voltage_ok == ESP_OK ? "ON" :
00141         "OFF");
00142     ESP_LOGI(TAG, "Calibración: DC bus current =%s", calibration_current_ok == ESP_OK ? "ON" : "OFF");
00143     return ESP_OK;
00144 }
00145
00146 void adc_task(void *arg) {
00147
00148     uint16_t vbus_vector[20] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}, ibus_vector[20] =
00149         {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
00150     uint16_t vector_index = 0;
00151     int raw_meas_bus_voltage = 0, raw_meas_current = 0, raw_meas_5V_source = 0, raw_meas_3V3_source =
0;
00152     int meas_bus_voltage = 0, meas_current = 0, meas_5V_source = 0, meas_3V3_source = 0;
00153
00154     uint32_t vbus_sum = 0, ibus_sum = 0;
00155
00156     bool vector_filled = false;
00157
00158     security_settings_t bus_meas;
00159
00160     if (adc_init() != ESP_OK) {
00161         ESP_LOGE(TAG, "adc_init falló; no se inicializaron correctamente los ADC");
00162         ESP_ERROR_CHECK(ESP_FAIL);
00163     }
00164
00165     if (bus_meas_evt_queue == NULL) {
00166         bus_meas_evt_queue = xQueueCreate(1, sizeof(security_settings_t));
00167         if (bus_meas_evt_queue == NULL) {
00168             ESP_LOGE(TAG, "No se pudo crear la cola de mediciones de bus");
00169             return;
00170         }
00171     }
00172
00173     vTaskDelay(pdMS_TO_TICKS(2000));
00174
00175     while (1) {
00176         if (adc_oneshot_read(s_adc, ADC_CH_GPIO34, &raw_meas_bus_voltage) == ESP_OK) {
00177             if (calibration_bus_voltage_ok == ESP_OK) {
00178                 adc_cali_raw_to_voltage(calibration_bus_voltage, raw_meas_bus_voltage,
00179                     &meas_bus_voltage);
00180
00181                 vbus_sum -= vbus_vector[vector_index];
00182                 vbus_vector[vector_index] = (int) truncf(meas_bus_voltage / 9.014);
00183                 vbus_sum += vbus_vector[vector_index];
00184
00185                 if (vector_filled) {
00186                     bus_meas.vbus_min = (uint16_t) (vbus_sum / 20);
00187                 }
00188             } else {
00189                 ESP_LOGE(TAG, "Error de lectura tensión");
00190             }
00191
00192             if (adc_oneshot_read(s_adc, ADC_CH_GPIO35, &raw_meas_current) == ESP_OK) {
00193
00194                 if (calibration_current_ok == ESP_OK) {
00195                     adc_cali_raw_to_voltage(calibration_current, raw_meas_current, &meas_current);
00196
00197                     ibus_sum -= ibus_vector[vector_index];
00198                     ibus_vector[vector_index] = abs(meas_current - 2500);
00199                     ibus_sum += ibus_vector[vector_index];
00200
00201                     if (vector_filled) {
00202                         bus_meas.ibus_max = (uint16_t) (ibus_sum / 20) * 5;
00203                     }
00204                 } else {
00205                     ESP_LOGE(TAG, "Error de lectura corriente");
00206                 }
00207
00208             if (adc_oneshot_read(s_adc, ADC_CH_GPIO39, &raw_meas_3V3_source) == ESP_OK) {
00209                 if (calibration_3V3_source_ok == ESP_OK) {
00210                     adc_cali_raw_to_voltage(calibration_3V3_source, raw_meas_3V3_source,
00211                         &meas_3V3_source);
00212                 }
00213             } else {
00214                 ESP_LOGE(TAG, "Error de lectura corriente");
00215             }
00216
00217             if (adc_oneshot_read(s_adc, ADC_CH_GPIO36, &raw_meas_5V_source) == ESP_OK) {
00218                 if (calibration_5V_source_ok == ESP_OK) {
00219                     adc_cali_raw_to_voltage(calibration_5V_source, raw_meas_5V_source, &meas_5V_source);
00220                 }
00221
00222         }
00223     }
00224 }
```

```

00221         } else {
00222             ESP_LOGE(TAG, "Error de lectura corriente");
00223         }
00224
00225         vector_index++;
00226         if (vector_index >= 20) {
00227             vector_index = 0;
00228             vector_filled = true;
00229         }
00230
00231         if (vector_filled && bus_meas.ibus_max != 0 && bus_meas.vbus_min != 0 ) {
00232             xQueueSend(bus_meas_evt_queue, &bus_meas, 0);
00233         }
00234
00235         vTaskDelay(pdMS_TO_TICKS(ADC_PERIOD_MS));
00236     }
00237 }
00238
00239 bool readADC(void) {
00240     security_settings_t bus_meas;
00241     bool meas_updated = false;
00242     if (xQueueReceive( bus_meas_evt_queue, &bus_meas, pdMS_TO_TICKS(0) ) ) {
00243         update_meas(bus_meas.vbus_min, bus_meas.ibus_max);
00244         meas_updated = true;
00245     }
00246     return meas_updated;
00247 }
```

6.3. Referencia del archivo main/adc/adc.h

Declaración de función de inicialización y tarea lectura del ADC.

```
#include "../LVFV_system.h"
```

Funciones

- void **adc_task** (void *arg)
Función programada como alarma para finalizar el funcionamiento del motor.
- esp_err_t **adc_init** (void)
Configura los 4 ADC necesarios para el sistema y los calibra para obtener las mediciones en desde 0 a 3125mV.
- bool **readADC** (void)
Desencola mediciones obtenidas desde el ADC para tensión y corriente del bus de continua, compara con los parámetros seteados y dispara el estado de emergencia si es necesario.

6.3.1. Descripción detallada

Declaración de función de inicialización y tarea lectura del ADC.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo [adc.h](#).

6.3.2. Documentación de funciones

6.3.2.1. adc_init()

```
esp_err_t adc_init (
    void )
```

Configura los 4 ADC necesarios para el sistema y los calibra para obtener las mediciones en desde 0 a 3125mV.

Configura por las entradas el canal 0 (GPIO36) el nivel de tensión de fuente de 5V, 3 (GPIO39) el nivel de tensión de fuente de 3.3V, 6 (GPIO34) tensión de bus de contínua y 7 (GPIO35) corriente de bus de contínua cada

Definición en la línea [97](#) del archivo [adc.c](#).

6.3.2.2. adc_task()

```
void adc_task (
    void * arg)
```

Función programada como alarma para finalizar el funcionamiento del motor.

Lee por las entradas el canal 0 (GPIO36) el nivel de tensión de fuente de 5V, 3 (GPIO39) el nivel de tensión de fuente de 3.3V, 6 (GPIO34) tensión de bus de contínua y 7 (GPIO35) corriente de bus de contínua cada

Definición en la línea [146](#) del archivo [adc.c](#).

6.3.2.3. readADC()

```
bool readADC (
    void )
```

Desencola mediciones obtenidas desde el ADC para tensión y corriente del bus de contínua, compara con los parámetros seteados y dispara el estado de emergencia si es necesario.

Consulta si existe alguna medición encolada en `bus_meas_evt_queue` sin ser bloqueante y envía esas mediciones a analizar por el sistema para evaluar si corresponde o no un estado de emergencia.

Valores devueltos

- | |
|---|
| <ul style="list-style-type: none"> - true: Si el sistema está inicializado y las mediciones pueden ser obtenidas <ul style="list-style-type: none"> • false: Si el sistema todavía no inició y las mediciones que pudo haber obtenido fueron inválidas |
|---|

Definición en la línea [239](#) del archivo [adc.c](#).

6.4. adc.h

[Ir a la documentación de este archivo.](#)

```
00001
00009 #ifndef __ADC_H__
00010
00011 #define __ADC_H__
00012
00013 #include "../LVFV_system.h"
00014
00025 void adc_task(void *arg);
00026
00039 esp_err_t adc_init(void);
00040
00052 bool readADC(void);
00053
00054 #endif
```

6.5. Referencia del archivo main/display/display.c

Funciones de consulta de los seteos hechos por el usuario, tarea que controla el display y posteo de eventos para controlar el display.

```
#include "esp_log.h"
#include "esp_err.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include <string.h>
#include <math.h>
#include "esp_system.h"
#include "LVFV_system.h"
#include "driver/i2c.h"
#include "./display/display.h"
#include "./display/sh1106_i2c.h"
#include "./display/sh1106_graphics.h"
#include "./io_control/io_control.h"
#include "../system/SysControl.h"
#include "../system/SysAdmin.h"
#include "../rtc/rtc.h"
#include "../nvs/nvs.h"
```

defines

- #define I2C_DISPLAY_MASTER_NUM I2C_NUM_0
- #define I2C_MASTER_SDA_IO 18
- #define I2C_MASTER_SCL_IO 5
- #define I2C_MASTER_FREQ_HZ 400000
- #define I2C_MASTER_TX_BUF_DISABLE 0
- #define I2C_MASTER_RX_BUF_DISABLE 0
- #define FREQUENCY_LINEAR_VARIABLE 0
- #define FREQUENCY_CUADRATIC_VARIABLE 1
- #define FREC_LINE_INDX VARIABLE_FIRST
- #define VBUS_LINE_INDX VARIABLE_SECOND
- #define IBUS_LINE_INDX VARIABLE_THIRD
- #define HINI_LINE_INDX VARIABLE_FOURTH

- #define HFIN_LINE_INDX VARIABLE_FIFTH
- #define EDIT_FREQUENCY_INDX VARIABLE_FIRST
- #define EDIT_ACCELERATION_INDX VARIABLE_SECOND
- #define EDIT_DESACCELERATION_INDX VARIABLE_THIRD
- #define EDIT_INPUT_VARIATION_INDX VARIABLE_FOURTH
- #define EDIT_TIME_LINE_INDX VARIABLE_SECOND
- #define EDIT_HINI_LINE_INDX VARIABLE_THIRD
- #define EDIT_HFIN_LINE_INDX VARIABLE_FOURTH
- #define EDIT_VBUS_LINE_INDX VARIABLE_SECOND
- #define EDIT_IBUS_LINE_INDX VARIABLE_THIRD

Enumeraciones

- enum screen_selected_e {
 SCREEN_MAIN , SCREEN_SELECT_VARIABLE , SCREEN_TIME_EDIT , SCREEN_SECURITY_EDIT ,
 SCREEN_FREQUENCY_EDIT }

Listado de posibles pantallas que puede ver el usuario.

Funciones

- static void i2c_master_init (void)

Inicializa el bus I2C para comunicarse con el display SH1106.
- static void sh1106_clear_buffer ()

Vacia el buffer de la pantalla para iniciar un nuevo dibujo.
- static esp_err_t sh1106_refresh ()

Función que imprime la pantalla con los elementos agregados por el usuario.
- static void sh1106_print_frame ()

Función que agrega al buffer de la pantalla un marco fijo que incluye la hora, la línea horizontal divisoria y el logo de la UTN.
- static void sh1106_time_edit_variables (time_settings_SH1106_t *variables)

Imprime en pantalla las variables de tiempo a editar: Hora del sistema, hora de inicio y hora de fin.
- static void sh1106_security_edit_variables (security_settings_SH1106_t *variables)

Imprime en pantalla las variables de seguridad a editar: Tensión mínima en la bus de contínua y corriente máxima en el bus de continua.
- static void sh1106_frequency_edit_variables (frequency_settings_SH1106_t *variables)

Imprime en pantalla las variables de frecuencia a editar: Frecuencia de régimen, aceleración, desaceleración y variación de las entradas aisladas para seleccionar la velocidad de régimen.
- static void sh1106_select_edit_variables (sh1106_variable_lines_e variable)

Permite al usuario seleccionar entre las tres pantallas de edición de variables: Frecuencia, tiempo y seguridad.
- static void sh1106_splash_screen ()

Pantalla de splash que se imprime al iniciar el sistema.
- static void sh1106_main_screen ()

Pantalla principal del sistema. Allí se muestra la hora del sistema, la frecuencia a la que se encuentra el variador, la hora de inicio y fin de movimiento del motor.
- static void sh1106_print_emergency ()

Entrega el valor de frecuencia de régimen seteado por el usuario.
- uint16_t get_system_frequency ()

Entrega el valor de aceleración seteado por el usuario.
- uint16_t get_system_desacceleration ()

Entrega el valor de desaceleración seteado por el usuario.

- `esp_err_t sh1106_init ()`
Función que inicializa el display para comenzar a imprimir.
- `esp_err_t system_variables_save (frequency_settings_SH1106_t *frequency_settings, time_settings_SH1106_t *time_settings, seccurity_settings_SH1106_t *seccurity_settings)`
Copia parámetros recibidos desde la UI al estado del sistema, los aplica en tiempo de ejecución y los persiste en NVS.
- `void task_display (void *pvParameters)`
Tarea de display que controla las diferentes pantallas de muestra, selección y edición de las variables de sistema.
- `esp_err_t DisplayEventPost (systemSignal_e event)`
Función que permite encolar acciones para que ejecute la tarea de display.

Variables

- `QueueHandle_t button_evt_queue`
- `frequency_settings_t system_frequency_settings`
- `time_settings_t system_time_settings`
- `seccurity_settings_t system_seccurity_settings`
- `static screen_selected_e screen_displayed = SCREEN_MAIN`
- `static const uint8_t init_seq []`
- `sh1106_t oled`
- `static const char * TAG = "DISPLAY"`
- `static uint8_t blink`

6.5.1. Descripción detallada

Funciones de consulta de los seteos hechos por el usuario, tarea que controla el display y posteo de eventos para controlar el display.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo [display.c](#).

6.5.2. Documentación de «define»

6.5.2.1. EDIT_ACCELERATION_INDX

```
#define EDIT_ACCELERATION_INDX VARIABLE_SECOND
```

Definición en la línea [49](#) del archivo [display.c](#).

6.5.2.2. EDIT_DESACCELERATION_INDX

```
#define EDIT_DESACCELERATION_INDX VARIABLE_THIRD
```

Definición en la línea 50 del archivo [display.c](#).

6.5.2.3. EDIT_FREQUENCY_INDX

```
#define EDIT_FREQUENCY_INDX VARIABLE_FIRST
```

Definición en la línea 48 del archivo [display.c](#).

6.5.2.4. EDIT_HFIN_LINE_INDX

```
#define EDIT_HFIN_LINE_INDX VARIABLE_FOURTH
```

Definición en la línea 55 del archivo [display.c](#).

6.5.2.5. EDIT_HINI_LINE_INDX

```
#define EDIT_HINI_LINE_INDX VARIABLE_THIRD
```

Definición en la línea 54 del archivo [display.c](#).

6.5.2.6. EDIT_IBUS_LINE_INDX

```
#define EDIT_IBUS_LINE_INDX VARIABLE_THIRD
```

Definición en la línea 58 del archivo [display.c](#).

6.5.2.7. EDIT_INPUT_VARIATION_INDX

```
#define EDIT_INPUT_VARIATION_INDX VARIABLE_FOURTH
```

Definición en la línea 51 del archivo [display.c](#).

6.5.2.8. EDIT_TIME_LINE_INDX

```
#define EDIT_TIME_LINE_INDX VARIABLE_SECOND
```

Definición en la línea 53 del archivo [display.c](#).

6.5.2.9. EDIT_VBUS_LINE_INDX

```
#define EDIT_VBUS_LINE_INDX VARIABLE_SECOND
```

Definición en la línea 57 del archivo [display.c](#).

6.5.2.10. FREC_LINE_INDX

```
#define FREC_LINE_INDX VARIABLE_FIRST
```

Definición en la línea [42](#) del archivo [display.c](#).

6.5.2.11. FREQUENCY CUADRATIC VARIABLE

```
#define FREQUENCY CUADRATIC VARIABLE 1
```

Definición en la línea [40](#) del archivo [display.c](#).

6.5.2.12. FREQUENCY_LINEAR_VARIABLE

```
#define FREQUENCY_LINEAR_VARIABLE 0
```

Definición en la línea [39](#) del archivo [display.c](#).

6.5.2.13. HFIN_LINE_INDX

```
#define HFIN_LINE_INDX VARIABLE_FIFTH
```

Definición en la línea [46](#) del archivo [display.c](#).

6.5.2.14. HINI_LINE_INDX

```
#define HINI_LINE_INDX VARIABLE_FOURTH
```

Definición en la línea [45](#) del archivo [display.c](#).

6.5.2.15. I2C_DISPLAY_MASTER_NUM

```
#define I2C_DISPLAY_MASTER_NUM I2C_NUM_0
```

Definición en la línea [32](#) del archivo [display.c](#).

6.5.2.16. I2C_MASTER_FREQ_HZ

```
#define I2C_MASTER_FREQ_HZ 400000
```

Definición en la línea [35](#) del archivo [display.c](#).

6.5.2.17. I2C_MASTER_RX_BUF_DISABLE

```
#define I2C_MASTER_RX_BUF_DISABLE 0
```

Definición en la línea [37](#) del archivo [display.c](#).

6.5.2.18. I2C_MASTER_SCL_IO

```
#define I2C_MASTER_SCL_IO 5
```

Definición en la línea 34 del archivo [display.c](#).

6.5.2.19. I2C_MASTER_SDA_IO

```
#define I2C_MASTER_SDA_IO 18
```

Definición en la línea 33 del archivo [display.c](#).

6.5.2.20. I2C_MASTER_TX_BUF_DISABLE

```
#define I2C_MASTER_TX_BUF_DISABLE 0
```

Definición en la línea 36 del archivo [display.c](#).

6.5.2.21. IBUS_LINE_INDX

```
#define IBUS_LINE_INDX VARIABLE_THIRD
```

Definición en la línea 44 del archivo [display.c](#).

6.5.2.22. VBUS_LINE_INDX

```
#define VBUS_LINE_INDX VARIABLE_SECOND
```

Definición en la línea 43 del archivo [display.c](#).

6.5.3. Documentación de enumeraciones

6.5.3.1. screen_selected_e

```
enum screen_selected_e
```

Listado de posibles pantallas que puede ver el usuario.

Valores de enumeraciones

SCREEN_MAIN	
SCREEN_SELECT_VARIABLE	
SCREEN_TIME_EDIT	
SCREEN_SECURITY_EDIT	

SCREEN_FREQUENCY_EDIT

Definición en la línea 65 del archivo [display.c](#).

6.5.4. Documentación de funciones

6.5.4.1. DisplayEventPost()

```
esp_err_t DisplayEventPost (
    systemSignal_e event)
```

Función que permite encolar acciones para que ejecute la tarea de display.

Valores devueltos

<i>pdTRUE</i>	Si se encoló exitosamente errQUEUE_FULL: Cualquier tipo de falla
---------------	--

Definición en la línea 900 del archivo [display.c](#).

6.5.4.2. get_system_acceleration()

```
uint16_t get_system_acceleration ()
```

Entrega el valor de aceleración seteado por el usuario.

Valores devueltos

<i>Valor</i>	de aceleración configurado por el usuario
--------------	---

Definición en la línea 458 del archivo [display.c](#).

6.5.4.3. get_system_desacceleration()

```
uint16_t get_system_desacceleration ()
```

Entrega el valor de desaceleración seteado por el usuario.

Valores devueltos

<i>Valor</i>	de desaceleración configurado por el usuario
--------------	--

Definición en la línea 462 del archivo [display.c](#).

6.5.4.4. get_system_frequency()

```
uint16_t get_system_frequency ()
```

Entrega el valor de frecuencia de regimen seteado por el usuario.

Valores devueltos

<i>Valor</i>	de frecuencia de regimen configurado por el usuario
--------------	---

Definición en la línea 454 del archivo [display.c](#).

6.5.4.5. `i2c_master_init()`

```
void i2c_master_init (
    void ) [static]
```

Inicializa el bus I2C para comunicarse con el display SH1106.

Definición en la línea 187 del archivo [display.c](#).

6.5.4.6. `sh1106_clear_buffer()`

```
void sh1106_clear_buffer () [static]
```

Vacía el buffer de la pantalla para iniciar un nuevo dibujo.

Definición en la línea 200 del archivo [display.c](#).

6.5.4.7. `sh1106_frequency_edit_variables()`

```
void sh1106_frequency_edit_variables (
    frequency_settings_SH1106_t * variables) [static]
```

Imprime en pantalla las variables de frecuencia a editar: Frecuencia de régimen, aceleración, desaceleración y variación de las entradas aisladas para seleccionar la velocidad de régimen.

Parámetros

<i>in, out</i>	<i>variables</i>	Puntero a la estructura de veriables a modificar por el usuario
----------------	------------------	---

Definición en la línea 334 del archivo [display.c](#).

6.5.4.8. `sh1106_init()`

```
esp_err_t sh1106_init ()
```

Función que inicializa el display para comenzar a imprimir.

Valores devueltos

<i>ESP_OK</i>	Si inicializa correctamente ESP_FAIL Si alguno de los comandos de inicialización falla
---------------	--

Definición en la línea 466 del archivo [display.c](#).

6.5.4.9. sh1106_main_screen()

```
void sh1106_main_screen () [static]
```

Pantalla principal del sistema. Allí se muestra la hora del sistema, la frecuencia a la que se encuentra el variador, la hora de inicio y fin de movimiento del motor.

Definición en la línea 425 del archivo [display.c](#).

6.5.4.10. sh1106_print_emergency()

```
void sh1106_print_emergency () [static]
```

Definición en la línea 235 del archivo [display.c](#).

6.5.4.11. sh1106_print_frame()

```
void sh1106_print_frame () [static]
```

Función que agrega al buffer de la pantalla un marco fijo que incluye la hora, la línea horizontal divisoria y el logo de la UTN.

Definición en la línea 227 del archivo [display.c](#).

6.5.4.12. sh1106_refresh()

```
esp_err_t sh1106_refresh () [static]
```

Función que imprime la pantalla con los elementos agregados por el usuario.

Valores devueltos

<i>ESP_OK</i>	si todo funciona bien
---------------	-----------------------

Definición en la línea 204 del archivo [display.c](#).

6.5.4.13. sh1106_security_edit_variables()

```
void sh1106_security_edit_variables (
    security_settings_SH1106_t * variables) [static]
```

Imprime en pantalla las variables de seguridad a editar: Tensión mínima en la bus de continúa y corriente máxima en el bus de continua.

Parámetros

in, out	variables	Puntero a la estructura de veriables a modificar por el usuario
---------	-----------	---

Definición en la línea 298 del archivo [display.c](#).

6.5.4.14. sh1106_select_edit_variables()

```
void sh1106_select_edit_variables (
    sh1106_variable_lines_e variable) [static]
```

Permite al usuario seleccionar entre las tres pantallas de edición de variables: Frecuencia, tiempo y seguridad.

Parámetros

in, out	variables	Selector de menúes
---------	-----------	--------------------

Definición en la línea 402 del archivo [display.c](#).

6.5.4.15. sh1106_splash_screen()

```
void sh1106_splash_screen () [static]
```

Pantalla de splash que se imprime al iniciar el sistema.

Definición en la línea 414 del archivo [display.c](#).

6.5.4.16. sh1106_time_edit_variables()

```
void sh1106_time_edit_variables (
    time_settings_SH1106_t * variables) [static]
```

Imprime en pantalla las variables de tiempo a editar: Hora del sistema, hora de inicio y hora de fin.

Parámetros

in, out	variables	Puntero a la estructura de veriables a modificar por el usuario
---------	-----------	---

Definición en la línea 244 del archivo [display.c](#).

6.5.4.17. system_variables_save()

```
esp_err_t system_variables_save (
    frequency_settings_SH1106_t * frequency_settings,
    time_settings_SH1106_t * time_settings,
    security_settings_SH1106_t * security_settings)
```

Copia parámetros recibidos desde la UI al estado del sistema, los aplica en tiempo de ejecución y los persiste en NVS.

Flujo: 1) Valida punteros de entrada. 2) Copia por valor los campos de frecuencia y seguridad a los structs globales `system_frequency_settings` y `system_security_settings`. 3) Copia por valor solo las *campos* de hora (tm_hour/min/sec) hacia las estructuras de tiempo globales apuntadas por `system_time_settings.time_*`. (No guarda punteros entrantes: solo lee sus valores). 4) Aplica cambios en runtime:

- `setTime` (`system_time_settings.time_system`) para RTC.
- `set_frequency_table(...)` para tabla de modulación según entrada/fo.
- `rtc_schedule_alarms(&system_time_settings)` programa alarmas start/stop. 5) Persiste todo en NVS con `save_variables(...)`. 6) Notifica/actualiza al resto del sistema con `set_system_settings(...)`.

Parámetros

<code>frequency_settings</code>	Parámetros de frecuencia y rampas (Hz, Hz/s).
<code>time_settings</code>	Tiempos de sistema/start/stop (struct tm vía punteros).
<code>security_settings</code>	Límites de seguridad (Vbus min, Ibus máx).

Devuelve

`ESP_OK` si el flujo se ejecutó; `ESP_ERR_INVALID_ARG` si algún puntero es `NULL`.

Definición en la línea 485 del archivo [display.c](#).

6.5.4.18. task_display()

```
void task_display (
    void * pvParameters)
```

Tarea de display que controla las diferentes pantallas de muestra, selección y edición de las variables de sistema.

Parámetros

in	<code>pvParameters</code>	Variable sin uso
----	---------------------------	------------------

Definición en la línea 523 del archivo [display.c](#).

6.5.5. Documentación de variables

6.5.5.1. blink

```
uint8_t blink [static]
```

Definición en la línea 110 del archivo [display.c](#).

6.5.5.2. button_evt_queue

```
QueueHandle_t button_evt_queue
```

Definición en la línea 73 del archivo [display.c](#).

6.5.5.3. init_seq

```
const uint8_t init_seq[] [static]
```

Valor inicial:

```
= {  
    0xAE,  
    0xD5,  
    0x80,  
    0xA8,  
    0x3F,  
    0xD3,  
    0x00,  
    0x00,  
    0x40,  
    0xAD,  
    0x8B,  
    0xA1,  
    0xC8,  
    0xDA,  
    0x12,  
    0x81,  
    0xCF,  
    0xD9,  
    0xF1,  
    0xDB,  
    0x40,  
    0xA4,  
    0xA6,  
    0xAF  
}
```

Definición en la línea 81 del archivo [display.c](#).

6.5.5.4. oled

```
sh1106_t oled
```

Definición en la línea 107 del archivo [display.c](#).

6.5.5.5. screen_displayed

```
screen_selected_e screen_displayed = SCREEN_MAIN [static]
```

Definición en la línea 79 del archivo [display.c](#).

6.5.5.6. system_frequency_settings

```
frequency_settings_t system_frequency_settings
```

Definición en la línea 75 del archivo [display.c](#).

6.5.5.7. system_seccurity_settings

```
seccurity_settings_t system_seccurity_settings
```

Definición en la línea 77 del archivo [display.c](#).

6.5.5.8. system_time_settings

```
time_settings_t system_time_settings
```

Definición en la línea 76 del archivo [display.c](#).

6.5.5.9. TAG

```
const char* TAG = "DISPLAY" [static]
```

Definición en la línea 109 del archivo [display.c](#).

6.6. display.c

[Ir a la documentación de este archivo.](#)

```

00001
00009 #include "esp_log.h"
00010 #include "esp_err.h"
00011
00012 #include "freertos/FreeRTOS.h"
00013 #include "freertos/task.h"
00014 #include "freertos/queue.h"
00015
00016 #include <string.h>
00017 #include <math.h>
00018 #include "esp_system.h"
00019 #include "LVFV_system.h"
00020
00021 #include "driver/i2c.h"
00022 #include "./display/display.h"
00023 #include "./display/sh1106_i2c.h"
00024 #include "./display/sh1106_graphics.h"
00025
00026 #include "./io_control/io_control.h"
00027 #include "../system/SysControl.h"
00028 #include "../system/SysAdmin.h"
00029 #include "../rtc/rtc.h"
00030 #include "../nvs/nvs.h"
00031
00032 #define I2C_DISPLAY_MASTER_NUM           I2C_NUM_0 // Número de puerto I2C usado
00033 #define I2C_MASTER_SDA_IO             18 // Pin SDA del puerto I2C ->
00034 #define I2C_MASTER_SCL_IO             GPIO21 // Pin SCL del puerto I2C ->
00035 #define I2C_MASTER_FREQ_HZ            500000 // Frecuencia de clock del puerto
00036 #define I2C_MASTER_TX_BUF_DISABLE      0 // Largo del buffer de transmisión
00037                                     del puerto I2C - Como se usa en modo master, es una variable ignorada

```

```

00037 #define I2C_MASTER_RX_BUF_DISABLE 0 // Largo del buffer de recepción
      del puerto I2C - Como se usa en modo master, es una variable ignorada
00038
00039 #define FREQUENCY_LINEAR_VARIABLE 0
00040 #define FREQUENCY_CUADRATIC_VARIABLE 1
00041
00042 #define FREC_LINE_INDX VARIABLE_FIRST
00043 #define VBUS_LINE_INDX VARIABLE_SECOND
00044 #define IBUS_LINE_INDX VARIABLE_THIRD
00045 #define HINI_LINE_INDX VARIABLE_FOURTH
00046 #define HFIN_LINE_INDX VARIABLE_FIFTH
00047
00048 #define EDIT_FREQUENCY_INDX VARIABLE_FIRST
00049 #define EDIT_ACCELERATION_INDX VARIABLE_SECOND
00050 #define EDIT_DESACCELERATION_INDX VARIABLE_THIRD
00051 #define EDIT_INPUT_VARIATION_INDX VARIABLE_FOURTH
00052
00053 #define EDIT_TIME_LINE_INDX VARIABLE_SECOND
00054 #define EDIT_HINI_LINE_INDX VARIABLE_THIRD
00055 #define EDIT_HFIN_LINE_INDX VARIABLE_FOURTH
00056
00057 #define EDIT_VBUS_LINE_INDX VARIABLE_SECOND
00058 #define EDIT_IBUS_LINE_INDX VARIABLE_THIRD
00059
00060 typedef enum {
00061     SCREEN_MAIN, // 0 - Pantalla principal
00062     SCREEN_SELECT_VARIABLE, // 1 - Pantalla de selección de
00063     pantallas de edición
00064     SCREEN_TIME_EDIT, // 2 - Pantalla de edición de
00065     variables de tiempo
00066     SCREEN_SECURITY_EDIT, // 3 - Pantalla de edición de
00067     variables de seguridad
00068     SCREEN_FREQUENCY_EDIT, // 4 - Pantalla de edición de
00069     variables de frecuencia, aceleración y desaceleración
00070 } screen_selected_e;
00071
00072
00073 QueueHandle_t button_evt_queue; // Cola de comandos de las
      entradas digitales. Tiene capacidad para un solo mensaje del tamaño
      systemSignal_e
00074
00075 frequency_settings_t system_frequency_settings; // Estructura con las variables de
      frecuencia del sistema
00076 time_settings_t system_time_settings; // Estructura con las variables de
      tiempo del sistema
00077 seccurity_settings_t system_seccurity_settings; // Estructura con las variables de
      seguridad del sistema
00078
00079 static screen_selected_e screen_displayed = SCREEN_MAIN; // Pantalla actualmente mostrada
00080
00081 static const uint8_t init_seq[] = { // Secuencia de inicialización del
      display
00082     0xAE, // DISPLAYOFF
00083     0xD5, // SETDISPLAYCLOCKDIV
00084     0x80,
00085     0xA8, // SETMULTIPLEX
00086     0x3F,
00087     0xD3, // SETDISPLAYOFFSET
00088     0x00,
00089     0x40, // SETSTARTLINE
00090     0xAD, // SETCHARGEPUOMP
00091     0x8B,
00092     0xA1, // SEGREMAP
00093     0xC8, // COMSCANDEC
00094     0xDA, // SETCOMPINS
00095     0x12,
00096     0x81, // SETCONTRAST
00097     0xCF,
00098     0xD9, // SETPRECHARGE
00099     0xF1,
00100    0xDB, // SETVCOMDETECT
00101    0x40,
00102    0xA4, // DISPLAYALLON_RESUME
00103    0xA6, // NORMALDISPLAY
00104    0xAF // DISPLAYON
00105 };
00106
00107 sh1106_t oled; // Estructura con la configuración
      del display
00108
00109 static const char *TAG = "DISPLAY"; // Tag de ESP_LOG
00110 static uint8_t blink; // Variable de parpadeo para
      edición de variables
00111
00112 static void i2c_master_init(void);
00113 static void sh1106_clear_buffer();
00114 static esp_err_t sh1106_refresh();
00115 static void sh1106_print_frame();
00116 static void sh1106_time_edit_variables(time_settings_SH1106_t *variables);

```

```

00155 static void sh1106_security_edit_variables(security_settings_SH1106_t *variables);
00164 static void sh1106_frequency_edit_variables(frequency_settings_SH1106_t *variables);
00173 static void sh1106_select_edit_variables(sh1106_variable_lines_e variable);
00179 static void sh1106_splash_screen();
00185 static void sh1106_main_screen( );
00186
00187 static void i2c_master_init(void) {
00188     i2c_config_t conf = {
00189         .mode = I2C_MODE_MASTER,
00190         .sda_io_num = I2C_MASTER_SDA_IO,
00191         .sda_pullup_en = GPIO_PULLUP_ENABLE,
00192         .scl_io_num = I2C_MASTER_SCL_IO,
00193         .scl_pullup_en = GPIO_PULLUP_ENABLE,
00194         .master.clk_speed = I2C_MASTER_FREQ_HZ,
00195     };
00196     ESP_ERROR_CHECK(i2c_param_config(I2C_DISPLAY_MASTER_NUM, &conf));
00197     ESP_ERROR_CHECK(i2c_driver_install(I2C_DISPLAY_MASTER_NUM, conf.mode, I2C_MASTER_RX_BUF_DISABLE,
00198                                     I2C_MASTER_TX_BUF_DISABLE, 0));
00199
00200 static void sh1106_clear_buffer() {
00201     memset(oled.buffer, 0, sizeof(oled.buffer));
00202 }
00203
00204 static esp_err_t sh1106_refresh() {
00205     uint8_t page;
00206     // El SH1106 espera que se setee la dirección de la columna luego se manden datos página por
00207     // página
00208     for (page = 0; page < 8; page++) {
00209         // Dirección de página
00210         uint8_t command = 0xB0 + page;
00211         ESP_ERROR_CHECK(sh1106_write(&command, 1, SH1106_COMM_CMD));
00212         // Dirección de columna baja y alta (offset 2 que ajusta SH1106)
00213         command = 0x02;
00214         ESP_ERROR_CHECK(sh1106_write(&command, 1, SH1106_COMM_CMD)); // col lo (2)
00215         command = 0x10;
00216         ESP_ERROR_CHECK(sh1106_write(&command, 1, SH1106_COMM_CMD)); // col hi
00217         // Enviar 128 bytes de datos para esta página
00218         uint8_t *page_data = &oled.buffer[page * 128];
00219         ESP_ERROR_CHECK(sh1106_write(page_data, 128, SH1106_COMM_DATA));
00220     }
00221     if (page == 8) {
00222         return ESP_OK;
00223     }
00224     return ESP_FAIL;
00225 }
00226
00227 static void sh1106_print_frame() {
00228     char str[13];
00229     sh1106_draw_utn_logo(&oled, 1, 1);
00230     sh1106_draw_line(&oled, 0, 17);
00231     strftime(str, sizeof(str), "%H:%M:%S", system_time_settings.time_system);
00232     sh1106_draw_text(&oled, str, 34, 0, SH1106_SIZE_2);
00233 }
00234
00235 static void sh1106_print_emergency() {
00236     sh1106_clear_buffer();
00237     sh1106_print_frame();
00238     sh1106_draw_text(&oled, " Parada", 25, 25, SH1106_SIZE_2);
00239     sh1106_draw_text(&oled, "EMERGENCIA", 25, VARIABLE_FOURTH, SH1106_SIZE_2);
00240
00241     ESP_ERROR_CHECK(sh1106_refresh());
00242 }
00243
00244 static void sh1106_time_edit_variables(time_settings_SH1106_t *variables) {
00245     char hora_str[12];
00246     uint8_t blink_compare = 4;
00247     if (*variables->edit_flag) {
00248         blink_compare = 12;
00249     }
00250
00251     sh1106_clear_buffer();
00252     sh1106_print_frame();
00253
00254     sh1106_draw_text(&oled, "Time:", 15, EDIT_TIME_LINE_IDX, SH1106_SIZE_1);
00255     sh1106_draw_text(&oled, "Ini:", 15, EDIT_HINI_LINE_IDX, SH1106_SIZE_1);
00256     sh1106_draw_text(&oled, "Fin:", 15, EDIT_HFIN_LINE_IDX, SH1106_SIZE_1);
00257
00258     if (*variables->edit) >= VARIABLE_SECOND && *variables->edit <= VARIABLE_FOURTH ) {
00259         sh1106_draw_arrow(&oled, 1, (int) VARIABLE_SECOND);
00260     } else if ( *variables->edit) >= VARIABLE_FIFTH && *variables->edit <= VARIABLE_SIXTH ) {
00261         sh1106_draw_arrow(&oled, 1, (int) VARIABLE_THIRD);
00262     } else if ( *variables->edit) >= VARIABLE_SEVENTH && *variables->edit <= VARIABLE_EIGHTH ) {
00263         sh1106_draw_arrow(&oled, 1, (int) VARIABLE_FOURTH);
00264     }
00265 }
```

```

00266     if ( *(variables->edit) == VARIABLE_SECOND && blink < blink_compare ) {
00267         sprintf(hora_str, " :%02d:%02d", variables->time_settings.time_system->tm_min,
00268             variables->time_settings.time_system->tm_sec);
00269     } else if( *(variables->edit) == VARIABLE_THIRD && blink < blink_compare ) {
00270         sprintf(hora_str, "%02d: :%02d", variables->time_settings.time_system->tm_hour,
00271             variables->time_settings.time_system->tm_sec);
00272     } else if( *(variables->edit) == VARIABLE_FOURTH && blink < blink_compare ) {
00273         sprintf(hora_str, "%02d:%02d: ", variables->time_settings.time_system->tm_hour,
00274             variables->time_settings.time_system->tm_min);
00275     } else {
00276         strftime(hora_str, sizeof(hora_str), "%H:%M:%S", variables->time_settings.time_system);
00277     }
00278     sh1106_draw_text( &oled, hora_str, 64, EDIT_TIME_LINE_INDX, SH1106_SIZE_1);
00279
00280     if ( *(variables->edit) == VARIABLE_FIFTH && blink < blink_compare ) {
00281         sprintf(hora_str, " :%02d", variables->time_settings.time_start->tm_min);
00282     } else if( *(variables->edit) == VARIABLE_SIXTH && blink < blink_compare ) {
00283         sprintf(hora_str, "%02d: ", variables->time_settings.time_start->tm_hour);
00284     } else {
00285         sprintf(hora_str, "%02d:%02d", variables->time_settings.time_start->tm_hour,
00286             variables->time_settings.time_start->tm_min);
00287     }
00288     sh1106_draw_text( &oled, hora_str, 64, EDIT_HINI_LINE_INDX, SH1106_SIZE_1);
00289
00290     if ( *(variables->edit) == VARIABLE_SEVENTH && blink < blink_compare ) {
00291         sprintf(hora_str, " :%02d", variables->time_settings.time_stop->tm_min);
00292     } else if ( *(variables->edit) == VARIABLE_EIGTH && blink < blink_compare ) {
00293         sprintf(hora_str, "%02d: ", variables->time_settings.time_stop->tm_hour);
00294     } else {
00295         sprintf(hora_str, "%02d:%02d", variables->time_settings.time_stop->tm_hour,
00296             variables->time_settings.time_stop->tm_min);
00297     }
00298     sh1106_draw_text( &oled, hora_str, 64, EDIT_HFIN_LINE_INDX, SH1106_SIZE_1);
00299
00300     ESP_ERROR_CHECK(sh1106_refresh());
00301 }
00302
00303 static void sh1106_security_edit_variables(seccurity_settings_SH1106_t *variables) {
00304     char num_str[12];
00305
00306     sh1106_clear_buffer( &oled );
00307     sh1106_print_frame();
00308
00309     sh1106_draw_text( &oled, "V. min:", 15, EDIT_VBUS_LINE_INDX, SH1106_SIZE_1);
00310     sh1106_draw_text( &oled, "V", 117, EDIT_VBUS_LINE_INDX, SH1106_SIZE_1);
00311     sh1106_draw_text( &oled, "I. max:", 15, EDIT_IBUS_LINE_INDX, SH1106_SIZE_1);
00312     sh1106_draw_text( &oled, "A", 117, EDIT_IBUS_LINE_INDX, SH1106_SIZE_1);
00313     sh1106_draw_arrow( &oled, 1, (int) *(variables->edit));
00314
00315     if ( *(variables->edit) == EDIT_VBUS_LINE_INDX && *(variables->edit_flag) ) {
00316         if ( blink >= 12 ) {
00317             sprintf(num_str, "%03d", variables->seccurity_settings.vbus_min);
00318             sh1106_draw_text( &oled, num_str, 85, EDIT_VBUS_LINE_INDX, SH1106_SIZE_1);
00319         }
00320     } else {
00321         sprintf(num_str, "%03d", variables->seccurity_settings.vbus_min);
00322         sh1106_draw_text( &oled, num_str, 85, EDIT_VBUS_LINE_INDX, SH1106_SIZE_1);
00323     }
00324
00325     if ( *(variables->edit) == EDIT_IBUS_LINE_INDX && *(variables->edit_flag) ) {
00326         if ( blink >= 12 ) {
00327             sprintf(num_str, "%03d", variables->seccurity_settings.ibus_max);
00328             sh1106_draw_text( &oled, num_str, 85, EDIT_IBUS_LINE_INDX, SH1106_SIZE_1);
00329         }
00330     } else {
00331         sprintf(num_str, "%03d", variables->seccurity_settings.ibus_max);
00332         sh1106_draw_text( &oled, num_str, 85, EDIT_IBUS_LINE_INDX, SH1106_SIZE_1);
00333     }
00334     sh1106_draw_arrow( &oled, 1, (int) variables->edit);
00335
00336     ESP_ERROR_CHECK(sh1106_refresh());
00337 }
00338
00339 static void sh1106_frequency_edit_variables(frequency_settings_SH1106_t *variables) {
00340     char num_str[12];
00341
00342     sh1106_clear_buffer();
00343     sh1106_print_frame();
00344
00345     sh1106_draw_text( &oled, "Frec:", 15, VARIABLE_FIRST, SH1106_SIZE_1);
00346     sh1106_draw_text( &oled, "Hz", 95, VARIABLE_FIRST, SH1106_SIZE_1);
00347     sh1106_draw_text( &oled, "Acel:", 15, VARIABLE_SECOND, SH1106_SIZE_1);
00348     sh1106_draw_text( &oled, "Hz/s", 95, VARIABLE_SECOND, SH1106_SIZE_1);
00349     sh1106_draw_text( &oled, "Desa:", 15, VARIABLE_THIRD, SH1106_SIZE_1);
00350     sh1106_draw_text( &oled, "Hz/s", 95, VARIABLE_THIRD, SH1106_SIZE_1);
00351     sh1106_draw_text( &oled, "Var entradas", 18, VARIABLE_FOURTH, SH1106_SIZE_1);
00352 }
```

```

00348     if ( *(variables->edit) != VARIABLE_FOURTH ) {
00349         sh1106_draw_arrow( &oled, 1, (int) *(variables->edit));
00350     } else {
00351         sh1106_draw_arrow( &oled, 1, (int) VARIABLE_FIFTH);
00352     }
00353
00354     if ( *(variables->edit) == EDIT_FREQUENCY_INDX && *(variables->edit_flag) ) {
00355         if ( blink >= 12 ) {
00356             sprintf(num_str, "%03d", variables->frequency_settings.freq_regime);
00357             sh1106_draw_text( &oled, num_str, 70, VARIABLE_FIRST, SH1106_SIZE_1);
00358         }
00359     } else {
00360         sprintf(num_str, "%03d", variables->frequency_settings.freq_regime);
00361         sh1106_draw_text( &oled, num_str, 70, VARIABLE_FIRST, SH1106_SIZE_1);
00362     }
00363
00364     if ( *(variables->edit) == EDIT_ACCELERATION_INDX && *(variables->edit_flag) ) {
00365         if ( blink >= 12 ) {
00366             sprintf(num_str, "%02d", variables->frequency_settings.acceleration);
00367             sh1106_draw_text( &oled, num_str, 70, VARIABLE_SECOND, SH1106_SIZE_1);
00368         }
00369     } else {
00370         sprintf(num_str, "%02d", variables->frequency_settings.acceleration);
00371         sh1106_draw_text( &oled, num_str, 70, VARIABLE_SECOND, SH1106_SIZE_1);
00372     }
00373
00374     if ( *(variables->edit) == EDIT_DESACCELERATION_INDX && *(variables->edit_flag) ) {
00375         if ( blink >= 12 ) {
00376             sprintf(num_str, "%02d", variables->frequency_settings.desacceleration);
00377             sh1106_draw_text( &oled, num_str, 70, VARIABLE_THIRD, SH1106_SIZE_1);
00378         }
00379     } else {
00380         sprintf(num_str, "%02d", variables->frequency_settings.desacceleration);
00381         sh1106_draw_text( &oled, num_str, 70, VARIABLE_THIRD, SH1106_SIZE_1);
00382     }
00383     if ( *(variables->edit) == EDIT_INPUT_VARIATION_INDX && *(variables->edit_flag) ) {
00384         if ( blink >= 12 ) {
00385             if ( variables->frequency_settings.input_variable == 1 ) {
00386                 sh1106_draw_text( &oled, " Lineal", 18, VARIABLE_FIFTH, SH1106_SIZE_1);
00387             } else if ( variables->frequency_settings.input_variable == 2 ) {
00388                 sh1106_draw_text( &oled, " Cuadratica", 18, VARIABLE_FIFTH, SH1106_SIZE_1);
00389             }
00390         }
00391     } else {
00392         if ( variables->frequency_settings.input_variable == 1 ) {
00393             sh1106_draw_text( &oled, " Lineal", 18, VARIABLE_FIFTH, SH1106_SIZE_1);
00394         } else if ( variables->frequency_settings.input_variable == 2 ) {
00395             sh1106_draw_text( &oled, " Cuadratica", 18, VARIABLE_FIFTH, SH1106_SIZE_1);
00396         }
00397     }
00398
00399     ESP_ERROR_CHECK(sh1106_refresh());
00400 }
00401
00402 static void sh1106_select_edit_variables(sh1106_variable_lines_e variable) {
00403
00404     sh1106_clear_buffer();
00405     sh1106_print_frame();
00406
00407     sh1106_draw_arrow( &oled, 1, (int) variable);
00408     sh1106_draw_text( &oled, "Frecuencias", 15, VARIABLE_SECOND, SH1106_SIZE_1);
00409     sh1106_draw_text( &oled, "Seguridad", 15, VARIABLE_THIRD, SH1106_SIZE_1);
00410     sh1106_draw_text( &oled, "Horarios", 15, VARIABLE_FOURTH, SH1106_SIZE_1);
00411     ESP_ERROR_CHECK(sh1106_refresh());
00412 }
00413
00414 static void sh1106_splash_screen() {
00415
00416     sh1106_clear_buffer();
00417     sh1106_draw_utn_logo( &oled, 56, 10);
00418
00419     sh1106_draw_text( &oled, "      UTN FRBA", 0, 30, SH1106_SIZE_1);
00420     sh1106_draw_text( &oled, "Proy. Final 2025", 0, 40, SH1106_SIZE_1);
00421     sh1106_draw_text( &oled, "Andrenacci-Carra", 0, 50, SH1106_SIZE_1);
00422     ESP_ERROR_CHECK(sh1106_refresh());
00423 }
00424
00425 static void sh1106_main_screen( ) {
00426
00427     char hora_str[13];
00428     system_status_t s_e;
00429     get_status(&s_e);
00430
00431     sh1106_clear_buffer();
00432     sh1106_print_frame();
00433
00434     sh1106_draw_text( &oled, "Frecuen.:", 5, VARIABLE_FIRST, SH1106_SIZE_1);

```

```

00435     sh1106_draw_text( &oled, "V. Bus DC:", 5, VARIABLE_SECOND, SH1106_SIZE_1);
00436     sh1106_draw_text( &oled, "I. Out DC:", 5, VARIABLE_THIRD, SH1106_SIZE_1);
00437     sh1106_draw_text( &oled, "Arranque:", 5, VARIABLE_FOURTH, SH1106_SIZE_1);
00438     sh1106_draw_text( &oled, "Parada:", 5, VARIABLE_FIFTH, SH1106_SIZE_1);
00439
00440     sprintf(hora_str, "%03d", s_e.frequency);
00441     sh1106_draw_text( &oled, hora_str, 86, VARIABLE_FIRST, SH1106_SIZE_1);
00442     sprintf(hora_str, "%03d", s_e.vbus_min);
00443     sh1106_draw_text( &oled, hora_str, 86, VARIABLE_SECOND, SH1106_SIZE_1);
00444     sprintf(hora_str, "%04d", s_e.ibus_max);
00445     sh1106_draw_text( &oled, hora_str, 86, VARIABLE_THIRD, SH1106_SIZE_1);
00446     sprintf(hora_str, "%02d: %02d", system_time_settings.time_start->tm_hour,
00447             system_time_settings.time_start->tm_min );
00448     sh1106_draw_text( &oled, hora_str, 86, VARIABLE_FOURTH, SH1106_SIZE_1);
00449     sprintf(hora_str, "%02d: %02d", system_time_settings.time_stop->tm_hour,
00450             system_time_settings.time_stop->tm_min );
00451     sh1106_draw_text( &oled, hora_str, 86, VARIABLE_FIFTH, SH1106_SIZE_1);
00452 }
00453
00454 uint16_t get_system_frequency() {
00455     return system_frequency_settings.freq_regime;
00456 }
00457
00458 uint16_t get_system_acceleration() {
00459     return system_frequency_settings.acceleration;
00460 }
00461
00462 uint16_t get_system_desacceleration() {
00463     return system_frequency_settings.desacceleration;
00464 }
00465
00466 esp_err_t sh1106_init() {
00467
00468     oled.width = 128;
00469     oled.height = 64;
00470     oled.rotation = 0;
00471
00472     i2c_master_init();
00473
00474     for (size_t i = 0; i < sizeof(init_seq); i++) {
00475         if ( sh1106_write(&init_seq[i], 1, SH1106_COMM_CMD) != ESP_OK ) {
00476             ESP_LOGE(TAG, "Error enviando comando de inicialización %zu: 0x%02X", i, init_seq[i]);
00477             return ESP_FAIL;
00478         }
00479     }
00480     sh1106_clear_buffer();
00481     sh1106_splash_screen();
00482     return sh1106_refresh();
00483 }
00484
00485 esp_err_t system_variables_save(frequency_settings_SH1106_t *frequency_settings,
00486     time_settings_SH1106_t *time_settings, seccurity_settings_SH1106_t *seccurity_settings) {
00487
00488     ESP_LOGI(TAG, "Guardando variables del sistema desde el display");
00489     if ( frequency_settings == NULL || time_settings == NULL || seccurity_settings == NULL ) {
00490         return ESP_ERR_INVALID_ARG;
00491     }
00492
00493     if ( time_settings->time_settings.time_system == NULL || time_settings->time_settings.time_start
00494         == NULL || time_settings->time_settings.time_stop == NULL ) {
00495         return ESP_ERR_INVALID_ARG;
00496     }
00497
00498     system_frequency_settings.freq_regime = frequency_settings->frequency_settings.freq_regime;
00499     system_frequency_settings.acceleration = frequency_settings->frequency_settings.acceleration;
00500     system_frequency_settings.desacceleration =
00501         frequency_settings->frequency_settings.desacceleration;
00502     system_frequency_settings.input_variable = frequency_settings->frequency_settings.input_variable;
00503     system_seccurity_settings.vbus_min = seccurity_settings->seccurity_settings.vbus_min;
00504     system_seccurity_settings.ibus_max = seccurity_settings->seccurity_settings.ibus_max;
00505     system_time_settings.time_system->tm_hour = time_settings->time_settings.time_system->tm_hour;
00506     system_time_settings.time_system->tm_min = time_settings->time_settings.time_system->tm_min;
00507     system_time_settings.time_system->tm_sec = time_settings->time_settings.time_system->tm_sec;
00508     system_time_settings.time_start->tm_hour = time_settings->time_settings.time_start->tm_hour;
00509     system_time_settings.time_start->tm_min = time_settings->time_settings.time_start->tm_min;
00510     system_time_settings.time_start->tm_sec = 0;
00511     system_time_settings.time_stop->tm_hour = time_settings->time_settings.time_stop->tm_hour;
00512     system_time_settings.time_stop->tm_min = time_settings->time_settings.time_stop->tm_min;
00513     system_time_settings.time_stop->tm_sec = 0;
00514
00515     setTime( system_time_settings.time_system );
00516     set_frequency_table(system_frequency_settings.input_variable,
00517         system_frequency_settings.freq_regime);
00518     rtc_schedule_alarms(&system_time_settings);
00519
00520     if ( save_variables( &system_frequency_settings, &system_time_settings,

```

```
00516     &system_seccurity_settings) != ESP_OK ) {
00517         ESP_LOGE(TAG, "Algo falló guardando los valores en NVS");
00518     set_system_settings( &system_frequency_settings, &system_seccurity_settings);
00519     ESP_LOGI(TAG, "Configuración guardada correctamente");
00520     return ESP_OK;
00521 }
00522
00523 void task_display(void *pvParameters) {
00524
00525     uint8_t new_button; // Variable Queue
00526
00527     sh1106_variable_lines_e top_variable = first, bottom_variable = fifth; // Identificador de la
00528     variable seleccionada
00529     sh1106_variable_lines_e variable_lines = VARIABLE_FIRST;
00530
00531     uint8_t top_multiplier = 100, bottom_multiplier = 1; // Incremental de la
00532     variable seleccionada
00533     uint8_t multiplier = 1;
00534
00535     uint32_t edit_variable_min = 0, edit_variable_max = 0; // Variable
00536     seleccionada
00537     uint16_t *edit_variable = NULL;
00538
00539     uint8_t edit = 0; // Edición o selección
00540
00541     struct tm timeinfo = {
00542         .tm_hour = 0,
00543         .tm_min = 0,
00544         .tm_sec = 0,
00545         .tm_mday = 0,
00546         .tm_mon = 0,
00547         .tm_year = 0
00548     };
00549     struct tm time_start = {
00550         .tm_hour = 0,
00551         .tm_min = 0,
00552         .tm_sec = 0,
00553         .tm_mday = 0,
00554         .tm_mon = 0,
00555         .tm_year = 0
00556     };
00557     struct tm time_stop = {
00558         .tm_hour = 0,
00559         .tm_min = 0,
00560         .tm_sec = 0,
00561         .tm_mday = 0,
00562         .tm_mon = 0,
00563         .tm_year = 0
00564     };
00565     struct tm timestamp_edit = {
00566         .tm_hour = 0,
00567         .tm_min = 0,
00568         .tm_sec = 0,
00569         .tm_mday = 0,
00570         .tm_mon = 0,
00571         .tm_year = 0
00572     };
00573     struct tm time_start_edit = {
00574         .tm_hour = 0,
00575         .tm_min = 0,
00576         .tm_sec = 0,
00577         .tm_mday = 0,
00578         .tm_mon = 0,
00579         .tm_year = 0
00580     };
00581     struct tm time_stop_edit = {
00582         .tm_hour = 0,
00583         .tm_min = 0,
00584         .tm_sec = 0,
00585         .tm_mday = 0,
00586         .tm_mon = 0,
00587         .tm_year = 0
00588     };
00589     if ( sh1106_init() != ESP_OK ) {
00590         ESP_LOGE(TAG, "Error al inicializar el display SH1106");
00591         ESP_ERROR_CHECK(ESP_FAIL);
00592     }
00593     ESP_LOGI(TAG, "Inicialización del display SH1106 exitosa");
00594
00595     time_settings_SH1106_t time_edit;
00596     security_settings_SH1106_t seccurity_edit;
00597     frequency_settings_SH1106_t frequency_edit;
00598 }
```

```
00599     frequency_edit.multiplier = &multiplier;
00600     frequency_edit.edit = &variable_lines;
00601     frequency_edit.edit_flag = &edit;
00602
00603     security_edit.multiplier = &multiplier;
00604     security_edit.edit = &variable_lines;
00605     security_edit.edit_flag = &edit;
00606
00607     time_edit.multiplier = &multiplier;
00608     time_edit.edit_flag = &edit;
00609     time_edit.time_settings.time_system = &timestring_edit;
00610     time_edit.time_settings.time_start = &time_start_edit;
00611     time_edit.time_settings.time_stop = &time_stop_edit;
00612     time_edit.edit = &variable_lines;
00613     time_edit.time_settings.time_start = &time_start_edit;
00614     time_edit.time_settings.time_stop = &time_stop_edit;
00615
00616     system_time_settings.time_start = &time_start;
00617     system_time_settings.time_system = &timeinfo;
00618     system_time_settings.time_stop = &time_stop;
00619     ESP_LOGI(TAG, "Variables de edición inicializadas");
00620
00621     setTime( &timeinfo );
00622     rtc_schedule_alarms(&system_time_settings);
00623     ESP_LOGI(TAG, "Variables temporales inicializadas");
00624     set_frequency_table(system_frequency_settings.input_variable,
00625     system_frequency_settings.freq_regime);
00626     ESP_LOGI(TAG, "Tablas de frecuencia inicializadas");
00627
00628     if (button_evt_queue == NULL) {
00629         button_evt_queue = xQueueCreate(1, sizeof(uint32_t));
00630         if (button_evt_queue == NULL) {
00631             ESP_LOGE(TAG, "No se pudo crear la cola de interrupciones");
00632             ESP_ERROR_CHECK(ESP_FAIL);
00633         }
00634     }
00635     ESP_LOGI(TAG, "Queue de eventos inicializadas");
00636
00637     load_variables( &system_frequency_settings, &system_time_settings, &system_seccurity_settings);
00638     set_system_settings( &system_frequency_settings, &system_seccurity_settings);
00639
00640     vTaskDelay(pdMS_TO_TICKS(2000));
00641
00642     while (1) {
00643         getTime(&timeinfo);
00644         blink++;
00645         if ( blink >= 24 ) {
00646             blink = 0;
00647         }
00648         if ( xQueueReceive( button_evt_queue, &new_button, pdMS_TO_TICKS(5) ) ) {
00649             switch (new_button) {
00650                 case BUTTON_MENU:
00651                     if ( screen_displayed == SCREEN_MAIN ) {
00652                         screen_displayed = SCREEN_SELECT_VARIABLE;
00653                         variable_lines = VARIABLE_SECOND;
00654                         top_variable = VARIABLE_SECOND;
00655                         bottom_variable = VARIABLE_FOURTH;
00656
00657                         frequency_edit.frequency_settings.freq_regime =
00658                         system_frequency_settings.freq_regime;
00659                         frequency_edit.frequency_settings.acceleration =
00660                         system_frequency_settings.acceleration;
00661                         frequency_edit.frequency_settings.desacceleration =
00662                         system_frequency_settings.desacceleration;
00663                         frequency_edit.frequency_settings.input_variable =
00664                         system_frequency_settings.input_variable;
00665
00666                         security_edit.security_settings.vbus_min =
00667                         system_seccurity_settings.vbus_min;
00668                         security_edit.security_settings.ibus_max =
00669                         system_seccurity_settings.ibus_max;
00670
00671                         time_edit.time_settings.time_system->tm_hour =
00672                         system_time_settings.time_system->tm_hour;
00673                         time_edit.time_settings.time_system->tm_min =
00674                         system_time_settings.time_system->tm_min;
00675                         time_edit.time_settings.time_system->tm_sec =
00676                         system_time_settings.time_system->tm_sec;
00677
00678                         time_edit.time_settings.time_start->tm_hour =
00679                         system_time_settings.time_start->tm_hour;
00680                         time_edit.time_settings.time_start->tm_min =
00681                         system_time_settings.time_start->tm_min;
00682                         time_edit.time_settings.time_start->tm_sec = 0;
00683
00684                         time_edit.time_settings.time_stop->tm_hour =
00685                         system_time_settings.time_stop->tm_hour;
```

```
00673             time_edit.time_settings.time_stop->tm_min =
00674     system_time_settings.time_stop->tm_min;
00675
00676         }
00677         break;
00678     case BUTTON_OK:
00679         if ( screen_displayed == SCREEN_SELECT_VARIABLE ) {
00680             if ( variable_lines == VARIABLE_SECOND ) {
00681                 screen_displayed = SCREEN_FREQUENCY_EDIT;
00682                 variable_lines = VARIABLE_FIRST;
00683                 top_variable = VARIABLE_FIRST;
00684                 bottom_variable = VARIABLE_FOURTH;
00685
00686                 variable_lines = VARIABLE_FIRST;
00687             } else if ( variable_lines == VARIABLE_THIRD ) {
00688                 top_variable = VARIABLE_SECOND;
00689                 bottom_variable = VARIABLE_THIRD;
00690                 screen_displayed = SCREEN_SECURITY_EDIT;
00691                 variable_lines = VARIABLE_SECOND;
00692
00693                 variable_lines = VARIABLE_SECOND;
00694             } else if ( variable_lines == VARIABLE_FOURTH ) {
00695                 top_variable = VARIABLE_SECOND;
00696                 bottom_variable = VARIABLE_EIGHTH;
00697                 variable_lines = VARIABLE_SECOND;
00698                 screen_displayed = SCREEN_TIME_EDIT;
00699
00700                 variable_lines = VARIABLE_SECOND;
00701             }
00702             multiplier = 1;
00703         } else if ( screen_displayed == SCREEN_FREQUENCY_EDIT ) {
00704             if ( edit_variable == NULL ) {
00705                 if ( variable_lines == EDIT_FREQUENCY_INDX ) {
00706                     edit_variable = (uint16_t*)
00707                         &(frequency_edit.frequency_settings.freq_regime);
00708                     edit_variable_min = 5;
00709                     edit_variable_max = 150;
00710                 } else if ( variable_lines == EDIT_ACCELERATION_INDX ) {
00711                     edit_variable = (uint16_t*)
00712                         &(frequency_edit.frequency_settings.acceleration);
00713                     edit_variable_min = 1;
00714                     edit_variable_max = 150;
00715                 } else if ( variable_lines == EDIT_DESACCELERATION_INDX ) {
00716                     edit_variable = (uint16_t*)
00717                         &(frequency_edit.frequency_settings.desacceleration);
00718                     edit_variable_min = 1;
00719                     edit_variable_max = 150;
00720                 } else if ( variable_lines == EDIT_INPUT_VARIATION_INDX ) {
00721                     edit_variable = (uint16_t*)
00722                         &(frequency_edit.frequency_settings.input_variable);
00723                     edit_variable_min = 1;
00724                     edit_variable_max = 2;
00725                     top_multiplier = 1;
00726                     bottom_multiplier = 1;
00727                 }
00728             } else {
00729                 edit = 1;
00730                 multiplier = 1;
00731                 top_multiplier = 100;
00732                 bottom_multiplier = 1;
00733             }
00734         } else if ( screen_displayed == SCREEN_SECURITY_EDIT ) {
00735             if ( edit_variable == NULL ) {
00736                 if ( variable_lines == EDIT_VBUS_LINE_INDX ) {
00737                     edit_variable = (uint16_t*)
00738                         &(security_edit.security_settings.vbus_min);
00739                     edit_variable_min = 250;
00740                     edit_variable_max = 360;
00741                 } else if ( variable_lines == EDIT_IBUS_LINE_INDX ) {
00742                     edit_variable = (uint16_t*)
00743                         &(security_edit.security_settings.ibus_max);
00744                     edit_variable_min = 500;
00745                     edit_variable_max = 2000;
00746                 }
00747                 edit = 1;
00748                 multiplier = 1;
00749                 top_multiplier = 100;
00750                 bottom_multiplier = 1;
00751             } else {
00752                 edit = 0;
00753                 edit_variable = NULL;
00754                 edit_variable_min = 0;
```

```

00753                     edit_variable_max = 0;
00754                 }
00755             } else if ( screen_displayed == SCREEN_TIME_EDIT ) {
00756                 if ( edit_variable == NULL ) {
00757                     if ( variable_lines == VARIABLE_SECOND ) {
00758                         edit_variable = (uint16_t*)
00759                         &(time_edit.time_settings.time_system->tm_hour);
00760                         edit_variable_min = 0;
00761                         edit_variable_max = 23;
00762                     } else if ( variable_lines == VARIABLE_THIRD ) {
00763                         edit_variable = (uint16_t*)
00764                         &(time_edit.time_settings.time_system->tm_min);
00765                         edit_variable_min = 0;
00766                         edit_variable_max = 59;
00767                     } else if ( variable_lines == VARIABLE_FOURTH ) {
00768                         edit_variable = (uint16_t*)
00769                         &(time_edit.time_settings.time_system->tm_sec);
00770                         edit_variable_min = 0;
00771                         edit_variable_max = 59;
00772                     } else if ( variable_lines == VARIABLE_FIFTH ) {
00773                         edit_variable = (uint16_t*)
00774                         &(time_edit.time_settings.time_start->tm_hour);
00775                         edit_variable_min = 0;
00776                         edit_variable_max = 23;
00777                     } else if ( variable_lines == VARIABLE_SIXTH ) {
00778                         edit_variable = (uint16_t*)
00779                         &(time_edit.time_settings.time_start->tm_min);
00780                         edit_variable_min = 0;
00781                         edit_variable_max = 59;
00782                     } else if ( variable_lines == VARIABLE_SEVENTH ) {
00783                         edit_variable = (uint16_t*)
00784                         &(time_edit.time_settings.time_stop->tm_hour);
00785                         edit_variable_min = 0;
00786                         edit_variable_max = 23;
00787                     } else if ( variable_lines == VARIABLE_EIGHTH ) {
00788                         edit_variable = (uint16_t*)
00789                         &(time_edit.time_settings.time_stop->tm_min);
00790                         edit_variable_min = 0;
00791                         edit_variable_max = 59;
00792                     }
00793                     edit = 1;
00794                     multiplier = 1;
00795                     top_multiplier = 10;
00796                     bottom_multiplier = 1;
00797                 } else {
00798                     edit = 0;
00799                     edit_variable = NULL;
00800                     edit_variable_min = 0;
00801                     edit_variable_max = 0;
00802                 }
00803             } else if ( screen_displayed == SCREEN_MAIN ) {
00804                 SystemEventPost(START_PRESSED);
00805             }
00806             break;
00807         case BUTTON_BACK:
00808             if ( screen_displayed == SCREEN_SELECT_VARIABLE ) {
00809                 screen_displayed = SCREEN_MAIN;
00810             } else if ( screen_displayed == SCREEN_SECURITY_EDIT || screen_displayed ==
00811             SCREEN_FREQUENCY_EDIT || screen_displayed == SCREEN_TIME_EDIT ) {
00812                 screen_displayed = SCREEN_SELECT_VARIABLE;
00813                 variable_lines = VARIABLE_SECOND;
00814                 top_variable = VARIABLE_SECOND;
00815                 bottom_variable = VARIABLE_FOURTH;
00816                 edit_variable = NULL;
00817                 edit_variable_min = 0;
00818                 edit_variable_max = 0;
00819             } else if ( screen_displayed == SCREEN_MAIN ) {
00820                 SystemEventPost(STOP_PRESSED);
00821             }
00822             break;
00823         case BUTTON_UP:
00824             if ( edit_variable != NULL ) {
00825                 (*edit_variable) += multiplier;
00826                 if ( *edit_variable >= edit_variable_max ) {
00827                     *edit_variable = edit_variable_max;
00828                 }
00829             } else {
00830                 if ( variable_lines == top_variable ) {
00831                     variable_lines = bottom_variable;
00832                 } else {
00833                     variable_lines -= LINE_INCREMENT;
00834                 }
00835             }
00836             break;
00837         case BUTTON_DOWN:
00838             if ( edit_variable != NULL ) {
00839                 (*edit_variable) -= multiplier;
00840             }
00841         }

```

```

00832             if ( *edit_variable < edit_variable_min || *edit_variable > edit_variable_max
00833         ) {
00834             *edit_variable = edit_variable_min;
00835         }
00836     } else {
00837         if ( variable_lines == bottom_variable ) {
00838             variable_lines = top_variable;
00839         } else {
00840             variable_lines += LINE_INCREMENT;
00841         }
00842     }
00843     break;
00844 case BUTTON_LEFT:
00845     if ( multiplier < top_multiplier ) {
00846         multiplier *= 10;
00847     }
00848     if ( multiplier > top_multiplier ) {
00849         multiplier = top_multiplier;
00850     }
00851     break;
00852 case BUTTON_RIGHT:
00853     if ( multiplier > bottom_multiplier ) {
00854         multiplier /= 10;
00855     }
00856     if (multiplier < bottom_multiplier ) {
00857         multiplier = bottom_multiplier;
00858     }
00859     break;
00860 case BUTTON_SAVE:
00861     ESP_LOGI(TAG, "Guardando valores...");
00862
00863     system_variables_save(&frequency_edit, &time_edit, &seccurity_edit);
00864
00865     screen_displayed = SCREEN_MAIN;
00866     edit = 0;
00867     edit_variable = NULL;
00868     edit_variable_min = 0;
00869     edit_variable_max = 0;
00870     break;
00871 }
00872 }
00873
00874 switch (screen_displayed) {
00875     case SCREEN_MAIN:
00876         system_status_t s_e;
00877         get_status(&s_e);
00878         if ( ( s_e.status == SYSTEM_EMERGENCY || s_e.status == SYSTEM_EMERGENCY_OK ) && blink
< 12 ) {
00879             sh1106_print_emergency();
00880         } else {
00881             sh1106_main_screen();
00882         }
00883         break;
00884     case SCREEN_SELECT_VARIABLE:
00885         sh1106_select_edit_variables(variable_lines);
00886         break;
00887     case SCREEN_TIME_EDIT:
00888         sh1106_time_edit_variables(&time_edit);
00889         break;
00890     case SCREEN_SECURITY_EDIT:
00891         sh1106_security_edit_variables(&seccurity_edit);
00892         break;
00893     case SCREEN_FREQUENCY_EDIT:
00894         sh1106_frequency_edit_variables(&frequency_edit);
00895         break;
00896     }
00897 }
00898 }
00899
00900 esp_err_t DisplayEventPost(systemSignal_e event) {
00901     if (button_evt_queue == NULL) {
00902         return ESP_FAIL;
00903     }
00904     return xQueueSend(button_evt_queue, &event, 0);
00905 }
```

6.7. Referencia del archivo main/display/display.h

Declaración de funciones que de consulta de los seteos hechos por el usuario, tarea que controla el display y posteo de eventos para controlar el display.

```
#include <stdint.h>
#include <stdbool.h>
#include "../LVFV_system.h"
```

Funciones

- `uint16_t get_system_frequency ()`
Entrega el valor de frecuencia de regimen seteado por el usuario.
- `uint16_t get_system_acceleration ()`
Entrega el valor de aceleración seteado por el usuario.
- `uint16_t get_system_desacceleration ()`
Entrega el valor de desaceleración seteado por el usuario.
- `esp_err_t sh1106_init ()`
Función que inicializa el display para comenzar a imprimir.
- `void task_display (void *pvParameters)`
Tarea de display que controla las diferentes pantallas de muestra, selección y edición de las variables de sistema.
- `esp_err_t DisplayEventPost (systemSignal_e event)`
Función que permite encolar acciones para que ejecute la tarea de display.
- `esp_err_t system_variables_save (frequency_settings_SH1106_t *frequency_settings, time_settings_SH1106_t *time_settings, seccurity_settings_SH1106_t *seccurity_settings)`
Copia parámetros recibidos desde la UI al estado del sistema, los aplica en tiempo de ejecución y los persiste en NVS.

6.7.1. Descripción detallada

Declaración de funciones que de consulta de los seteos hechos por el usuario, tarea que controla el display y posteo de eventos para controlar el display.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo [display.h](#).

6.7.2. Documentación de funciones

6.7.2.1. DisplayEventPost()

```
esp_err_t DisplayEventPost (
    systemSignal_e event)
```

Función que permite encolar acciones para que ejecute la tarea de display.

Valores devueltos

<i>pdTRUE</i>	Si se encoló exitosamente errQUEUE_FULL: Cualquier tipo de falla
---------------	--

Definición en la línea 900 del archivo [display.c](#).

6.7.2.2. `get_system_acceleration()`

```
uint16_t get_system_acceleration ()
```

Entrega el valor de aceleración seteado por el usuario.

Valores devueltos

<i>Valor</i>	de aceleración configurado por el usuario
--------------	---

Definición en la línea 458 del archivo [display.c](#).

6.7.2.3. `get_system_desacceleration()`

```
uint16_t get_system_desacceleration ()
```

Entrega el valor de desaceleración seteado por el usuario.

Valores devueltos

<i>Valor</i>	de desaceleración configurado por el usuario
--------------	--

Definición en la línea 462 del archivo [display.c](#).

6.7.2.4. `get_system_frequency()`

```
uint16_t get_system_frequency ()
```

Entrega el valor de frecuencia de regimen seteado por el usuario.

Valores devueltos

<i>Valor</i>	de frecuencia de regimen configurado por el usuario
--------------	---

Definición en la línea 454 del archivo [display.c](#).

6.7.2.5. `sh1106_init()`

```
esp_err_t sh1106_init ()
```

Función que inicializa el display para comenzar a imprimir.

Valores devueltos

<code>ESP_OK</code>	Si inicializa correctamente ESP_FAIL Si alguno de los comandos de inicialización falla
---------------------	--

Definición en la línea 466 del archivo [display.c](#).

6.7.2.6. `system_variables_save()`

```
esp_err_t system_variables_save (
    frequency_settings_SH1106_t * frequency_settings,
    time_settings_SH1106_t * time_settings,
    security_settings_SH1106_t * security_settings)
```

Copia parámetros recibidos desde la UI al estado del sistema, los aplica en tiempo de ejecución y los persiste en NVS.

Flujo: 1) Valida punteros de entrada. 2) Copia por valor los campos de frecuencia y seguridad a los structs globales `system_frequency_settings` y `system_security_settings`. 3) Copia por valor solo las *campos* de hora (tm_hour/min/sec) hacia las estructuras de tiempo globales apuntadas por `system_time_settings.time_*`. (No guarda punteros entrantes: solo lee sus valores). 4) Aplica cambios en runtime:

- `setTime` (`system_time_settings.time_system`) para RTC.
- `set_frequency_table` (...) para tabla de modulación según entrada/fo.
- `rtc_schedule_alarms` (&`system_time_settings`) programa alarmas start/stop. 5) Persiste todo en NVS con `save_variables` (...).
- 6) Notifica/actualiza al resto del sistema con `set_system_settings` (...).

Parámetros

<code>frequency_settings</code>	Parámetros de frecuencia y rampas (Hz, Hz/s).
<code>time_settings</code>	Tiempos de sistema/start/stop (struct tm vía punteros).
<code>security_settings</code>	Límites de seguridad (Vbus min, Ibus máx).

Devuelve

`ESP_OK` si el flujo se ejecutó; `ESP_ERR_INVALID_ARG` si algún puntero es NULL.

Definición en la línea 485 del archivo [display.c](#).

6.7.2.7. `task_display()`

```
void task_display (
    void * pvParameters)
```

Tarea de display que controla las diferentes pantallas de muestra, selección y edición de las variables de sistema.

Parámetros

in	<i>pvParameters</i>	Variable sin uso
----	---------------------	------------------

Definición en la línea 523 del archivo [display.c](#).

6.8. display.h

[Ir a la documentación de este archivo.](#)

```

00001
00009 #ifndef SH1106_H
00010 #define SH1106_H
00011
00012 #include <stdint.h>
00013 #include <stdbool.h>
00014 #include "../LVFV_system.h"
00015
00024 uint16_t get_system_frequency();
00033 uint16_t get_system_acceleration();
00042 uint16_t get_system_desacceleration();
00052 esp_err_t sh1106_init();
00061 void task_display(void *pvParameters);
00071 esp_err_t DisplayEventPost(systemSignal_e event);
00072
00096 esp_err_t system_variables_save(frequency_settings_SH1106_t *frequency_settings,
                                     time_settings_SH1106_t *time_settings, security_settings_SH1106_t *security_settings);
00097
00098 #endif

```

6.9. Referencia del archivo main/display/sh1106_graphics.c

Funciones que permiten operar sobre el display.

```

#include <string.h>
#include "esp_err.h"
#include "./sh1106_graphics.h"
#include "../LVFV_system.h"

```

Estructuras de datos

- struct [bitmap_t](#)

Estructura que representa un carácter cualquiera. Donde debe inicializar un puntero a alguno de los caracteres, el alto, ancho, y su posición en x e y.

typedefs

- [typedef struct bitmap_t bitmap_t](#)

Funciones

- static void [sh1106_draw_bitmap \(sh1106_t *oled, bitmap_t *bitmap\)](#)

La función escribe en el bitmap de la pantalla los caracteres que el usuario desea imprimir.

- void [sh1106_draw_text \(sh1106_t *oled, const char *text, int x, int y, uint8_t size\)](#)
- void [sh1106_draw_utn_logo \(sh1106_t *oled, int x, int y\)](#)
- void [sh1106_draw_arrow \(sh1106_t *oled, int x, int y\)](#)
- void [sh1106_draw_fail \(sh1106_t *oled\)](#)

Función que dibuja un signo de exclamación en la esquina superior derecha de la pantalla para expresar una falla en el sistema.

- void [sh1106_draw_line \(sh1106_t *oled, int x, int y\)](#)

Variables

- static const unsigned char `logo16_utn.bmp` [] = {0x71,0x8E,0x71,0x8E,0x79,0x9E,0x3D,0xBC,0x1F,0xF8,0x07,0xE0,0x07,0xE0,0x1F,0xF8,0x3D,0xBC,0x79,0x9E,0x71,0x8E,0x71,0x8E}
- static const unsigned char `fail.bmp` [] = {0x03,0xC0,0x0C,0x30,0x33,0xD8,0x6E,0x6C,0xDC,0x06,0xDE,0x03,0xCF,0xE3,0xC7,0xF3,0xC0,0x7B,0x60,0x76,0x36,0x6C,0x1B,0xD8,0x0C,0x30,0x03,0xC0}
- static const unsigned char `slash` [] = {0x06,0x0E,0x1C,0x38,0x70,0xE0,0xC0}
- static const unsigned char `line.bmp` [] = {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF}
- static const unsigned char `arrow.bmp` [] = {0x00,0x20,0x00,0x30,0x00,0x38,0x1F,0xFC,0x00,0x38,0x00,0x30,0x00,0x20}
- static const uint8_t `font5x7_space` [7] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00}
- static const uint8_t `font5x7_double_dot` [7] = {0x18,0x18,0x00,0x00,0x18,0x18,0x00}
- static const uint8_t `font8x14_double_dot` [14] = {0x00,0x38,0x38,0x38,0x00,0x00,0x00,0x00,0x38,0x38,0x38,0x00,0x00}
- static const uint8_t `font5x7_dot` [7] = {0x00,0x00,0x00,0x00,0x00,0x18,0x18}
- static const uint8_t `font8x14_dot` [14] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x38}
- static const uint8_t `font5x7_comma` [7] = {0x00,0x00,0x00,0x00,0x18,0x18,0x30}
- static const uint8_t `font8x14_comma` [14] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x70}
- static const uint8_t `font5x7_close_quest` [7] = {0x3C,0x66,0x06,0x0C,0x18,0x00,0x18}
- static const uint8_t `font8x14_close_quest` [14] = {0x3C,0x3C,0x66,0x66,0x06,0x06,0x0C,0x0C,0x18,0x18,0x00,0x00,0x18,0x18}
- static const uint8_t `font5x7_close_excl` [7] = {0x18,0x18,0x18,0x18,0x18,0x18,0x18}
- static const uint8_t `font8x14_close_excl` [14] = {0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18}
- static const uint8_t `font5x7_minus` [7] = {0x00,0x00,0x00,0x7E,0x00,0x00,0x00}
- static const uint8_t `font8x14_minus` [14] = {0x00,0x00,0x00,0x00,0x00,0x00,0x7E,0x7E,0x00,0x00,0x00,0x00,0x00}
- static const uint8_t `font5x7_rowmajor` [][7]
- static const uint8_t `font5x7_rowminor` [][7]
- static const uint8_t `font5x7_rownumber` [][7]
- static const uint8_t `font8x14_rowmajor` [][14]
- static const uint8_t `font8x14_rowminor` [][14]
- static const uint8_t `font8x14_rownumber` [][14]

6.9.1. Descripción detallada

Funciones que permiten operar sobre el display.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo [sh1106_graphics.c](#).

6.9.2. Documentación de «typedef»

6.9.2.1. bitmap_t

```
typedef struct bitmap_t bitmap_t
```

6.9.3. Documentación de funciones

6.9.3.1. sh1106_draw_arrow()

```
void sh1106_draw_arrow (
    sh1106_t * oled,
    int x,
    int y)
```

Parámetros

out	<i>oled</i>	Puntero a la estructura que con la información que se enviará al display
in	<i>x</i>	Coordenada en x donde se desea diujar el logo de la UTN
in	<i>y</i>	Coordenada en y donde se desea diujar el logo de la UTN

Definición en la línea 324 del archivo [sh1106_graphics.c](#).

6.9.3.2. sh1106_draw_bitmap()

```
void sh1106_draw_bitmap (
    sh1106_t * oled,
    bitmap_t * bitmap) [static]
```

La función escribe en el bitmap de la pantalla los caracteres que el usuario desea imprimir.

Copiando bit a bit el buffer en las posiciones x e y indicadas

Parámetros

out	<i>oled</i>	Puntero a la estructura que representa todo lo que se enviará al display
in	<i>bitmap</i>	Puntero a la estructura del carácter que se desea escribir en el display

Definición en la línea 199 del archivo [sh1106_graphics.c](#).

6.9.3.3. sh1106_draw_fail()

```
void sh1106_draw_fail (
    sh1106_t * oled)
```

Función que dibuja un signo de exclamación en la esquina superior derecha de la pantalla para expresar una falla en el sistema.

Parámetros

<code>out</code>	<code>oled</code>	Puntero a la estructura que con la información que se enviará al display
------------------	-------------------	--

Definición en la línea 335 del archivo [sh1106_graphics.c](#).

6.9.3.4. sh1106_draw_line()

```
void sh1106_draw_line (
    sh1106_t * oled,
    int x,
    int y)
```

Parámetros

<code>out</code>	<code>oled</code>	Puntero a la estructura que con la información que se enviará al display
<code>in</code>	<code>x</code>	Coordenada en x donde se desea dibujar una línea del ancho del display
<code>in</code>	<code>y</code>	Coordenada en y donde se desea dibujar una línea del ancho del display

Definición en la línea 346 del archivo [sh1106_graphics.c](#).

6.9.3.5. sh1106_draw_text()

```
void sh1106_draw_text (
    sh1106_t * oled,
    const char * text,
    int x,
    int y,
    uint8_t size)
```

Parámetros

<code>out</code>	<code>oled</code>	Puntero a la estructura que con la información que se enviará al display
<code>out</code>	<code>oled</code>	Puntero a la estructura que con la información que se enviará al display
<code>in</code>	<code>x</code>	Coordenada en x donde se desea dibujar el texto de caracteres alfanuméricos
<code>in</code>	<code>y</code>	Coordenada en y donde se desea dibujar el texto de caracteres alfanuméricos
<code>in</code>	<code>size</code>	Largo del texto que se desea imprimir expresado en caracteres

Definición en la línea 219 del archivo [sh1106_graphics.c](#).

6.9.3.6. sh1106_draw_utn_logo()

```
void sh1106_draw_utn_logo (
    sh1106_t * oled,
    int x,
    int y)
```

Parámetros

out	<i>oled</i>	Puntero a la estructura que con la información que se enviará al display
in	<i>x</i>	Coordenada en x donde se desea diujar el texto de caracteres alfanuméricos
in	<i>y</i>	Coordenada en y donde se desea diujar el texto de caracteres alfanuméricos

Definición en la línea 313 del archivo [sh1106_graphics.c](#).

6.9.4. Documentación de variables

6.9.4.1. arrow_bmp

```
const unsigned char arrow_bmp[ ] = {0x00,0x20,0x00,0x30,0x00,0x38,0x1F,0xFC,0x00,0x38,0x00,0x30,0x00,0x20}
[static]
```

Definición en la línea 18 del archivo [sh1106_graphics.c](#).

6.9.4.2. fail_bmp

```
const unsigned char fail_bmp[ ] = {0x03,0xC0,0x0C,0x30,0x33,0xD8,0x6E,0x6C,0xDC,0x06,0xDE,0x03,0x←
CF,0xE3,0xC7,0xF3,0xC0,0x7B,0x60,0x76,0x36,0x6C,0x1B,0xD8,0x0C,0x30,0x03,0xC0 } [static]
```

Definición en la línea 15 del archivo [sh1106_graphics.c](#).

6.9.4.3. font5x7_close_excl

```
const uint8_t font5x7_close_excl[7] = {0x18,0x18,0x18,0x18,0x18,0x00,0x18} [static]
```

Definición en la línea 29 del archivo [sh1106_graphics.c](#).

6.9.4.4. font5x7_close_quest

```
const uint8_t font5x7_close_quest[7] = {0x3C,0x66,0x06,0x0C,0x18,0x00,0x18} [static]
```

Definición en la línea 27 del archivo [sh1106_graphics.c](#).

6.9.4.5. font5x7_comma

```
const uint8_t font5x7_comma[7] = {0x00,0x00,0x00,0x00,0x18,0x18,0x30} [static]
```

Definición en la línea 25 del archivo [sh1106_graphics.c](#).

6.9.4.6. font5x7_dot

```
const uint8_t font5x7_dot[7] = {0x00,0x00,0x00,0x00,0x00,0x18,0x18} [static]
```

Definición en la línea 23 del archivo [sh1106_graphics.c](#).

6.9.4.7. `font5x7_double_dot`

```
const uint8_t font5x7_double_dot[7] = {0x18,0x18,0x00,0x00,0x18,0x18,0x00} [static]
```

Definición en la línea 21 del archivo [sh1106_graphics.c](#).

6.9.4.8. `font5x7_minus`

```
const uint8_t font5x7_minus[7] = {0x00,0x00,0x00,0x7E,0x00,0x00,0x00} [static]
```

Definición en la línea 31 del archivo [sh1106_graphics.c](#).

6.9.4.9. `font5x7_rowmajor`

```
const uint8_t font5x7_rowmajor[][7] [static]
```

Valor inicial:

```
= {

{0x1E,0x33,0x33,0x3F,0x33,0x33,0x33},
{0x3E,0x33,0x33,0x3E,0x33,0x33,0x3E},
{0x1E,0x33,0x30,0x30,0x30,0x33,0x1E},
{0x3C,0x36,0x33,0x33,0x36,0x3C},
{0x3F,0x30,0x30,0x3E,0x30,0x30,0x3F},
{0x3F,0x30,0x30,0x3E,0x30,0x30,0x30},
{0x1E,0x33,0x30,0x37,0x33,0x33,0x1F},
{0x33,0x33,0x33,0x3F,0x33,0x33,0x33},
{0x1E,0x0C,0x0C,0x0C,0x0C,0x0C,0x1E},
{0x0F,0x06,0x06,0x06,0x06,0x36,0x1C},
{0x33,0x36,0x3C,0x38,0x3C,0x36,0x33},
{0x30,0x30,0x30,0x30,0x30,0x30,0x3F},
{0x33,0x3F,0x3F,0x33,0x33,0x33,0x33},
{0x33,0x3B,0x3F,0x37,0x33,0x33,0x33},
{0x1E,0x33,0x33,0x33,0x33,0x33,0x1E},
{0x3E,0x33,0x33,0x3E,0x30,0x30,0x30},
{0x1E,0x33,0x33,0x33,0x37,0x36,0x1D},
{0x3E,0x33,0x33,0x3E,0x3C,0x36,0x33},
{0x1E,0x33,0x30,0x1E,0x03,0x33,0x1E},
{0x3F,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C},
{0x33,0x33,0x33,0x33,0x33,0x33,0x1E},
{0x33,0x33,0x33,0x33,0x33,0x1E,0x0C},
{0x33,0x33,0x33,0x33,0x3F,0x3F,0x33},
{0x33,0x33,0x1E,0x0C,0x1E,0x33,0x33},
{0x33,0x33,0x33,0x1E,0x0C,0x0C,0x0C},
{0x3F,0x03,0x06,0x0C,0x18,0x30,0x3F}
```

Definición en la línea 33 del archivo [sh1106_graphics.c](#).

6.9.4.10. font5x7_rowminor

```
const uint8_t font5x7_rowminor[ ][7] [static]
```

Valor inicial:

```
= {  
    {0x00,0x00,0x3C,0x06,0x3E,0x66,0x3E},  
    {0x60,0x60,0x7C,0x66,0x66,0x66,0x7C},  
    {0x00,0x00,0x3C,0x66,0x60,0x66,0x3C},  
    {0x06,0x06,0x3E,0x66,0x66,0x66,0x3E},  
    {0x00,0x00,0x3C,0x66,0x7E,0x60,0x3C},  
    {0x1C,0x30,0x30,0x7C,0x30,0x30,0x30},  
    {0x00,0x3E,0x66,0x66,0x3E,0x06,0x7C},  
    {0x60,0x60,0x7C,0x66,0x66,0x66,0x66},  
    {0x18,0x00,0x38,0x18,0x18,0x18,0x3C},  
    {0x06,0x00,0x06,0x06,0x66,0x66,0x3C},  
    {0x60,0x60,0x66,0x6C,0x78,0x6C,0x66},  
    {0x38,0x18,0x18,0x18,0x18,0x18,0x3C},  
    {0x00,0x00,0x6C,0x7E,0x7E,0x6C,0x6C},  
    {0x00,0x00,0x7C,0x66,0x66,0x66,0x66},  
    {0x00,0x00,0x3C,0x66,0x66,0x66,0x3C},  
    {0x00,0x7C,0x66,0x66,0x7C,0x60,0x60},  
    {0x00,0x3E,0x66,0x66,0x3E,0x06,0x06},  
    {0x00,0x00,0x6C,0x76,0x60,0x60,0x60},  
    {0x00,0x00,0x3E,0x60,0x3C,0x06,0x7C},  
    {0x30,0x30,0x7C,0x30,0x30,0x30,0x1C},  
    {0x00,0x00,0x66,0x66,0x66,0x66,0x3E},  
    {0x00,0x00,0x66,0x66,0x3C,0x18},  
    {0x00,0x00,0x66,0x66,0x7E,0x7E,0x6C},  
    {0x00,0x00,0x66,0x3C,0x18,0x3C,0x66},  
    {0x00,0x66,0x66,0x66,0x3E,0x06,0x7C},  
    {0x00,0x7E,0x0C,0x18,0x30,0x7E,0x00}  
}
```

Definición en la línea [62](#) del archivo [sh1106_graphics.c](#).

6.9.4.11. font5x7_rownumber

```
const uint8_t font5x7_rownumber[ ][7] [static]
```

Valor inicial:

```
= {  
    {0x1E,0x33,0x37,0x3B,0x33,0x33,0x1E},  
    {0x0C,0x1C,0x0C,0x0C,0x0C,0x0C,0x1E},  
    {0x1E,0x33,0x03,0x0E,0x18,0x30,0x3F},
```

```

{0x1E,0x33,0x03,0x0E,0x03,0x33,0x1E},
{0x06,0x0E,0x1E,0x36,0x3F,0x06,0x06},
{0x3F,0x30,0x3E,0x03,0x03,0x33,0x1E},
{0x1E,0x30,0x3E,0x33,0x33,0x33,0x1E},
{0x3F,0x03,0x06,0x0C,0x18,0x18,0x18},
{0x1E,0x33,0x33,0x1E,0x33,0x33,0x1E},
{0x1E,0x33,0x33,0x1F,0x03,0x03,0x1E}

}

```

Definición en la línea 90 del archivo [sh1106_graphics.c](#).

6.9.4.12. font5x7_space

```
const uint8_t font5x7_space[7] = {0x00,0x00,0x00,0x00,0x00,0x00} [static]
```

Definición en la línea 19 del archivo [sh1106_graphics.c](#).

6.9.4.13. font8x14_close_excl

```
const uint8_t font8x14_close_excl[14] = {0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x00,0x00,0x18,0x18}
[static]
```

Definición en la línea 30 del archivo [sh1106_graphics.c](#).

6.9.4.14. font8x14_close_quest

```
const uint8_t font8x14_close_quest[14] = {0x3C,0x3C,0x66,0x66,0x06,0x06,0x0C,0x0C,0x18,0x18,0x00,0x00,0x18,0x18}
[static]
```

Definición en la línea 28 del archivo [sh1106_graphics.c](#).

6.9.4.15. font8x14_comma

```
const uint8_t font8x14_comma[14] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x38,0x70}
[static]
```

Definición en la línea 26 del archivo [sh1106_graphics.c](#).

6.9.4.16. font8x14_dot

```
const uint8_t font8x14_dot[14] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x38}
[static]
```

Definición en la línea 24 del archivo [sh1106_graphics.c](#).

6.9.4.17. font8x14_double_dot

```
const uint8_t font8x14_double_dot[14] = {0x00,0x38,0x38,0x38,0x00,0x00,0x00,0x00,0x00,0x38,0x38,0x38,0x00,0x00,0x00}
[static]
```

Definición en la línea 22 del archivo [sh1106_graphics.c](#).

6.9.4.18. font8x14_minus

```
const uint8_t font8x14_minus[14] = {0x00,0x00,0x00,0x00,0x00,0x7E,0x7E,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}
[static]
```

Definición en la línea 32 del archivo [sh1106_graphics.c](#).

6.9.4.19. font8x14_rowmajor

```
const uint8_t font8x14_rowmajor[ ][14] [static]
```

Valor inicial:

```
= {
    {0x1E,0x1E,0x33,0x33,0x33,0x33,0x3F,0x3F,0x33,0x33,0x33,0x33,0x33,0x33,0x33},
    {0x3E,0x3E,0x33,0x33,0x33,0x33,0x3E,0x3E,0x33,0x33,0x33,0x3E,0x3E},
    {0x1E,0x1E,0x33,0x33,0x30,0x30,0x30,0x30,0x30,0x33,0x33,0x1E,0x1E},
    {0x3C,0x3C,0x36,0x36,0x33,0x33,0x33,0x33,0x33,0x36,0x36,0x3C,0x3C},
    {0x3F,0x3F,0x30,0x30,0x30,0x30,0x3E,0x30,0x30,0x30,0x30,0x3F,0x3F},
    {0x3F,0x3F,0x30,0x30,0x30,0x30,0x3E,0x3E,0x30,0x30,0x30,0x30,0x30},
    {0x1E,0x1E,0x33,0x30,0x30,0x37,0x37,0x33,0x33,0x33,0x1F,0x1F},
    {0x33,0x33,0x33,0x33,0x33,0x3F,0x3F,0x33,0x33,0x33,0x33,0x33},
    {0x1E,0x1E,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x1E,0x1E},
    {0x0F,0x0F,0x06,0x06,0x06,0x06,0x06,0x06,0x06,0x36,0x36,0x1C,0x1C},
    {0x33,0x33,0x36,0x36,0x3C,0x3C,0x38,0x38,0x3C,0x3C,0x36,0x36,0x33,0x33},
    {0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x3F,0x3F},
    {0x33,0x33,0x3F,0x3F,0x3F,0x3F,0x33,0x33,0x33,0x33,0x33,0x33,0x33},
    {0x33,0x33,0x3B,0x3B,0x3F,0x3F,0x37,0x33,0x33,0x33,0x33,0x33,0x33},
    {0x1E,0x1E,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x1E,0x1E},
    {0x3E,0x3E,0x33,0x33,0x33,0x33,0x3E,0x3E,0x30,0x30,0x30,0x30,0x30},
    {0x1E,0x1E,0x33,0x33,0x33,0x33,0x33,0x37,0x37,0x36,0x36,0x1D,0x1D},
    {0x3E,0x3E,0x33,0x33,0x33,0x33,0x3E,0x3E,0x3C,0x3C,0x36,0x36,0x33,0x33},
    {0x1E,0x1E,0x33,0x30,0x30,0x1E,0x1E,0x03,0x03,0x33,0x33,0x1E,0x1E},
    {0x3F,0x3F,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C},
    {0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x1E,0x1E},
    {0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x1E,0x0C,0x0C},
    {0x33,0x33,0x33,0x33,0x33,0x33,0x3F,0x3F,0x3F,0x3F,0x33,0x33,0x33},
    {0x33,0x33,0x33,0x33,0x1E,0x1E,0x0C,0x0C,0x1E,0x1E,0x33,0x33,0x33},
    {0x33,0x33,0x33,0x33,0x1E,0x1E,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C},
    {0x3F,0x3F,0x03,0x03,0x06,0x06,0x0C,0x0C,0x18,0x18,0x30,0x30,0x3F,0x3F},
}
```

Definición en la línea 102 del archivo [sh1106_graphics.c](#).

6.9.4.20. font8x14_rowminor

```
const uint8_t font8x14_rowminor[ ][14] [static]
```

Valor inicial:

```
= {
    {0x00, 0x00, 0x00, 0x00, 0x3C, 0x3C, 0x06, 0x06, 0x3E, 0x3E, 0x66, 0x66, 0x3E, 0x3E},
    {0x60, 0x60, 0x60, 0x60, 0x7C, 0x7C, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x7C, 0x7C},
    {0x00, 0x00, 0x00, 0x00, 0x3C, 0x3C, 0x66, 0x66, 0x60, 0x60, 0x66, 0x66, 0x3C, 0x3C},
    {0x06, 0x06, 0x06, 0x06, 0x3E, 0x3E, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x3E, 0x3E},
    {0x00, 0x00, 0x00, 0x00, 0x3C, 0x3C, 0x66, 0x66, 0x7E, 0x7E, 0x60, 0x60, 0x3C, 0x3C},
    {0x1C, 0x1C, 0x30, 0x30, 0x30, 0x30, 0x7C, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30},
    {0x00, 0x00, 0x3E, 0x3E, 0x66, 0x66, 0x66, 0x66, 0x3E, 0x3E, 0x06, 0x06, 0x7C, 0x7C},
    {0x60, 0x60, 0x60, 0x60, 0x7C, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66},
    {0x18, 0x18, 0x00, 0x00, 0x38, 0x38, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x3C, 0x3C},
    {0x06, 0x06, 0x00, 0x00, 0x06, 0x06, 0x06, 0x06, 0x06, 0x66, 0x66, 0x3C, 0x3C},
    {0x60, 0x60, 0x60, 0x60, 0x66, 0x66, 0x6C, 0x6C, 0x78, 0x78, 0x6C, 0x6C, 0x66, 0x66},
    {0x38, 0x38, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x3C, 0x3C},
    {0x00, 0x00, 0x00, 0x00, 0x6C, 0x6C, 0x7E, 0x7E, 0x7E, 0x6C, 0x6C, 0x6C, 0x6C},
    {0x00, 0x00, 0x00, 0x00, 0x7C, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66},
    {0x00, 0x00, 0x00, 0x3C, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x3C, 0x3C},
    {0x00, 0x00, 0x7C, 0x66, 0x66, 0x66, 0x66, 0x7C, 0x60, 0x60, 0x60, 0x60},
    {0x00, 0x00, 0x3E, 0x3E, 0x66, 0x66, 0x66, 0x3E, 0x06, 0x06, 0x06, 0x06},
    {0x00, 0x00, 0x00, 0x00, 0x6C, 0x6C, 0x76, 0x76, 0x60, 0x60, 0x60, 0x60, 0x60},
    {0x00, 0x00, 0x00, 0x00, 0x3E, 0x60, 0x60, 0x3C, 0x06, 0x06, 0x7C, 0x7C},
    {0x30, 0x30, 0x30, 0x30, 0x7C, 0x30, 0x30, 0x30, 0x30, 0x30, 0x1C, 0x1C},
    {0x00, 0x00, 0x00, 0x00, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x3E, 0x3E},
    {0x00, 0x00, 0x00, 0x00, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x3C, 0x18},
    {0x00, 0x00, 0x00, 0x00, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66},
    {0x00, 0x00, 0x00, 0x00, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x7E, 0x7E, 0x6C, 0x6C},
    {0x00, 0x00, 0x00, 0x00, 0x66, 0x66, 0x3C, 0x18, 0x18, 0x3C, 0x3C, 0x66, 0x66},
    {0x00, 0x00, 0x66, 0x66, 0x66, 0x66, 0x66, 0x3E, 0x06, 0x06, 0x7C, 0x7C},
    {0x00, 0x00, 0x00, 0x00, 0x7E, 0x7E, 0x0C, 0x0C, 0x18, 0x18, 0x30, 0x30, 0x7E, 0x7E},
}
```

}

Definición en la línea 130 del archivo [sh1106_graphics.c](#).

6.9.4.21. font8x14_rownumber

```
const uint8_t font8x14_rownumber[ ][14] [static]
```

Valor inicial:

```
= {
    {0x3C, 0x66, 0xC3, 0xC3, 0xC3, 0xC3, 0xC3, 0xC3, 0xC3, 0x66, 0x3C, 0x00},
    {0x18, 0x38, 0x78, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x7E, 0x00},
    {0x3C, 0x66, 0xC3, 0x03, 0x06, 0x0C, 0x18, 0x30, 0x60, 0xC0, 0xC3, 0xFF, 0xFE, 0x00},
```

```
{0x7E,0xC3,0x03,0x03,0x06,0x3C,0x06,0x03,0x03,0x03,0xC3,0x66,0x3C,0x00},  
{0x06,0x0E,0x1E,0x36,0x66,0xC6,0x86,0xFF,0xFF,0x06,0x06,0x06,0x06,0x00},  
{0xFE,0xC0,0xC0,0xC0,0xFC,0xC6,0x03,0x03,0x03,0xC3,0x66,0x3C,0x00},  
{0x3C,0x66,0xC3,0xC0,0xC0,0xFC,0xC6,0xC3,0xC3,0xC3,0x66,0x3C,0x00},  
{0xFF,0xC3,0x03,0x06,0x06,0x0C,0x18,0x18,0x30,0x30,0x60,0x60,0x60,0x00},  
{0x3C,0x66,0xC3,0xC3,0x66,0x3C,0x66,0xC3,0xC3,0x66,0x3C,0x00},  
{0x3C,0x66,0xC3,0xC3,0x67,0x3F,0x03,0x03,0xC3,0x66,0x3C,0x00}  
}
```

Definición en la línea 158 del archivo [sh1106_graphics.c](#).

6.9.4.22. font8x14_space

```
const uint8_t font8x14_space[14] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};  
[static]
```

Definición en la línea 20 del archivo [sh1106_graphics.c](#).

6.9.4.23. line_bmp

```
const unsigned char line_bmp[] = {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF};  
[static]
```

Definición en la línea 17 del archivo [sh1106_graphics.c](#).

6.9.4.24. logo16_utn_bmp

```
const unsigned char logo16_utn_bmp[] = {0x71,0x8E,0x71,0x8E,0x79,0x9E,0x3D,0xBC,0x1F,0x8E,0x71,0xE0,0x07,0xE0,0x07,0x1F,0xF8,0x3D,0xBC,0x79,0x9E,0x71,0x8E,0x71,0x8E};  
[static]
```

Definición en la línea 14 del archivo [sh1106_graphics.c](#).

6.9.4.25. slash

```
const unsigned char slash[] = {0x06,0x0E,0x1C,0x38,0x70,0xE0,0xC0};  
[static]
```

Definición en la línea 16 del archivo [sh1106_graphics.c](#).

6.10. sh1106_graphics.c

[Ir a la documentación de este archivo.](#)

```

00001
00009 #include <string.h>
00010 #include "esp_err.h"
00011 #include "./sh1106_graphics.h"
00012 #include "../LVFV_system.h"
00013
00014 static const unsigned char logo16_utn_bmp[] =
{0x71,0x8E,0x71,0x8E,0x79,0x9E,0x3D,0xBC,0x1F,0xF8,0x07,0xE0,0x07,0xE0,0x1F,0xF8,0x3D,0xBC,0x79,0x9E,0x71,0x8E,0x71,0x81
}; // Logo de la UTN 16 x 12
00015 static const unsigned char fail_bmp[] =
{0x03,0xC0,0x0C,0x30,0x33,0xD8,0x6E,0x6C,0xDC,0x06,0xDE,0x03,0xCF,0xE3,0xC7,0xF3,0xC0,0x7B,0x60,0x76,0x36,0x6C,0x1B,0xD0
}; // Icono de fail 24 x 26
00016 static const unsigned char slash[] = {0x06,0x0E,0x1C,0x38,0x70,0xE0,0xC0}; // Slash para separar miles 7 x 7
00017 static const unsigned char line_bmp[] =
{0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF}; // Linea horizontal 16 x 8
00018 static const unsigned char arrow_bmp[] =
{0x00,0x20,0x00,0x30,0x00,0x38,0x1F,0xFC,0x00,0x38,0x00,0x30,0x00,0x20}; // Flecha hacia la derecha 15 x 7
00019 static const uint8_t font5x7_space[7] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00}; // Espacio 5 x 7
00020 static const uint8_t font8x14_space[14] =
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}; // Espacio 8 x 14
00021 static const uint8_t font5x7_double_dot[7] = {0x18,0x18,0x00,0x00,0x18,0x18,0x00}; // Dos puntos 5 x 7
00022 static const uint8_t font8x14_double_dot[14] =
{0x00,0x38,0x38,0x00,0x00,0x00,0x00,0x00,0x38,0x38,0x00,0x00}; // Dos puntos 8 x 14
00023 static const uint8_t font5x7_dot[7] = {0x00,0x00,0x00,0x00,0x00,0x18,0x18}; // Punto 5 x 7
00024 static const uint8_t font8x14_dot[14] =
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x38}; // Punto 8 x 14
00025 static const uint8_t font5x7_comma[7] = {0x00,0x00,0x00,0x00,0x18,0x18,0x30}; // Coma 5 x 7
00026 static const uint8_t font8x14_comma[14] =
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x70}; // Coma 8 x 14
00027 static const uint8_t font5x7_close_quest[7] = {0x3C,0x66,0x06,0x0C,0x18,0x00,0x18}; // Signo de interrogación cerrado 5 x 7
00028 static const uint8_t font8x14_close_quest[14] =
{0x3C,0x3C,0x66,0x66,0x06,0x06,0x0C,0x18,0x18,0x00,0x00,0x18,0x18}; // Signo de interrogación cerrado 8 x 14
00029 static const uint8_t font5x7_close_excl[7] = {0x18,0x18,0x18,0x18,0x18,0x00,0x18}; // Signo de exclamación cerrado 5 x 7
00030 static const uint8_t font8x14_close_excl[14] =
{0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x00,0x00,0x18,0x18}; // Signo de exclamación cerrado 8 x 14
00031 static const uint8_t font5x7_minus[7] = {0x00,0x00,0x00,0x7E,0x00,0x00,0x00}; // Menos 5 x 7
00032 static const uint8_t font8x14_minus[14] =
{0x00,0x00,0x00,0x00,0x00,0x7E,0x7E,0x00,0x00,0x00,0x00,0x00,0x00}; // Menos 8 x 14
00033 static const uint8_t font5x7_rowmajor[] [7] = {
// Alfabeto mayúsculas 5 x 7
00034 // A-Z
00035 {0x1E,0x33,0x33,0x3F,0x33,0x33,0x33},
// A
00036 {0x3E,0x33,0x33,0x3E,0x33,0x33,0x3E},
// B
00037 {0x1E,0x33,0x30,0x30,0x30,0x33,0x1E},
// C
00038 {0x3C,0x36,0x33,0x33,0x36,0x3C},
// D
00039 {0x3F,0x30,0x30,0x3E,0x30,0x30,0x3F},
// E
00040 {0x3F,0x30,0x30,0x3E,0x30,0x30,0x30},
// F
00041 {0x1E,0x33,0x30,0x37,0x33,0x33,0x1F},
// G
00042 {0x33,0x33,0x33,0x3F,0x33,0x33,0x33},
// H
00043 {0x1E,0x0C,0x0C,0x0C,0x0C,0x0C,0x1E},
// I
00044 {0x0F,0x06,0x06,0x06,0x06,0x36,0x1C},
// J
00045 {0x33,0x36,0x3C,0x38,0x3C,0x36,0x33},
// K
00046 {0x30,0x30,0x30,0x30,0x30,0x30,0x3F},
// L
}

```

```

00047     {0x33,0x3F,0x3F,0x33,0x33,0x33,0x33},
00048     // M
00049     {0x33,0x3B,0x3F,0x37,0x33,0x33,0x33},
00050     // N
00051     {0x1E,0x33,0x33,0x33,0x33,0x1E},
00052     // O
00053     {0x3E,0x33,0x33,0x3E,0x30,0x30,0x30},
00054     // P
00055     {0x1E,0x33,0x33,0x33,0x37,0x36,0x1D},
00056     // Q
00057     {0x3E,0x33,0x33,0x3E,0x3C,0x36,0x33},
00058     // R
00059     {0x1E,0x33,0x30,0x1E,0x03,0x33,0x1E},
00060     // S
00061     {0x3F,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C},
00062     // T
00063     {0x33,0x33,0x33,0x33,0x33,0x33,0x1E},
00064     // U
00065     {0x33,0x33,0x33,0x33,0x33,0x1E,0x0C},
00066     // V
00067     {0x33,0x33,0x33,0x33,0x3F,0x3F,0x33},
00068     // W
00069     {0x33,0x33,0x1E,0x0C,0x1E,0x33,0x33},
00070     // X
00071     {0x33,0x33,0x33,0x1E,0x0C,0x0C,0x0C},
00072     // Y
00073     {0x3F,0x03,0x06,0x0C,0x18,0x30,0x3F}
00074     // Z
00075     };
00076     static const uint8_t font5x7_rowminor[] [7] = {
00077     // Alfabeto minúsculas 5 x 7
00078     {0x00,0x00,0x3C,0x06,0x3E,0x66,0x3E},
00079     // a
00080     {0x60,0x60,0x7C,0x66,0x66,0x66,0x7C},
00081     // b
00082     {0x00,0x00,0x3C,0x66,0x60,0x66,0x3C},
00083     // c
00084     {0x06,0x06,0x3E,0x66,0x66,0x66,0x3E},
00085     // d
00086     {0x00,0x00,0x3C,0x66,0x7E,0x60,0x3C},
00087     // e
00088     {0x1C,0x30,0x30,0x7C,0x30,0x30,0x30},
00089     // f
00090     {0x00,0x3E,0x66,0x66,0x3E,0x06,0x7C},
00091     // g
00092     {0x60,0x60,0x7C,0x66,0x66,0x66,0x66},
00093     // h
00094     {0x18,0x00,0x38,0x18,0x18,0x18,0x3C},
00095     // i
00096     {0x06,0x00,0x06,0x06,0x66,0x66,0x3C},
00097     // j
00098     {0x60,0x60,0x66,0x6C,0x78,0x6C,0x66},
00099     // k
00100     {0x38,0x18,0x18,0x18,0x18,0x18,0x3C},
00101     // l
00102     {0x00,0x00,0x6C,0x7E,0x7E,0x6C,0x6C},
00103     // m
00104     {0x00,0x00,0x7C,0x66,0x66,0x66,0x66},
00105     // n
00106     {0x00,0x00,0x3C,0x66,0x66,0x66,0x3C},
00107     // o
00108     {0x00,0x7C,0x66,0x66,0x7C,0x60,0x60},
00109     // p
00110     {0x00,0x3E,0x66,0x66,0x3E,0x06,0x06},
00111     // q
00112     {0x00,0x00,0x6C,0x76,0x60,0x60,0x60},
00113     // r
00114     {0x00,0x00,0x3E,0x60,0x3C,0x06,0x7C},
00115     // s
00116     {0x30,0x30,0x7C,0x30,0x30,0x30,0x1C},
00117     // t
00118     {0x00,0x00,0x66,0x66,0x66,0x66,0x3E},
00119     // u
00120     {0x00,0x00,0x66,0x66,0x66,0x3C,0x18},
00121     // v
00122     {0x00,0x00,0x66,0x66,0x7E,0x7E,0x6C},
00123     // w
00124     {0x00,0x00,0x66,0x3C,0x18,0x3C,0x66},
00125     // x
00126     {0x00,0x66,0x66,0x66,0x3E,0x06,0x7C},
00127     // y
00128     {0x00,0x7E,0x0C,0x18,0x30,0x7E,0x00}
00129     // z
00130     };
00131     static const uint8_t font5x7_rownumber[] [7] = {
00132     // Números 5 x 7
00133     {0x1E,0x33,0x37,0x3B,0x33,0x33,0x1E},
00134     // 0
00135     {0x33,0x33,0x33,0x33,0x33,0x33,0x33},
00136     // 1
00137     {0x33,0x33,0x33,0x33,0x33,0x33,0x33},
00138     // 2
00139     {0x33,0x33,0x33,0x33,0x33,0x33,0x33},
00140     // 3
00141     {0x33,0x33,0x33,0x33,0x33,0x33,0x33},
00142     // 4
00143     {0x33,0x33,0x33,0x33,0x33,0x33,0x33},
00144     // 5
00145     {0x33,0x33,0x33,0x33,0x33,0x33,0x33},
00146     // 6
00147     {0x33,0x33,0x33,0x33,0x33,0x33,0x33},
00148     // 7
00149     {0x33,0x33,0x33,0x33,0x33,0x33,0x33},
00150     // 8
00151     {0x33,0x33,0x33,0x33,0x33,0x33,0x33},
00152     // 9
00153     {0x33,0x33,0x33,0x33,0x33,0x33,0x33}
00154     // .
00155     };

```

```

    // 0
00092 {0x0C,0x1C,0x0C,0x0C,0x0C,0x1E},
    // 1
00093 {0x1E,0x33,0x03,0x0E,0x18,0x30,0x3F},
    // 2
00094 {0x1E,0x33,0x03,0x0E,0x03,0x33,0x1E},
    // 3
00095 {0x06,0x0E,0x1E,0x36,0x3F,0x06,0x06},
    // 4
00096 {0x3F,0x30,0x3E,0x03,0x03,0x33,0x1E},
    // 5
00097 {0x1E,0x30,0x3E,0x33,0x33,0x33,0x1E},
    // 6
00098 {0x3F,0x03,0x06,0x0C,0x18,0x18,0x18},
    // 7
00099 {0x1E,0x33,0x33,0x1E,0x33,0x33,0x1E},
    // 8
00100 {0x1E,0x33,0x33,0x1F,0x03,0x03,0x1E}
    // 9
00101 };
00102 static const uint8_t font8x14_rowmajor[][14] = {
    // Alfabeto mayúsculas 8 x 14
00103 {0x1E,0x1E,0x33,0x33,0x33,0x3F,0x3F,0x33,0x33,0x33,0x33,0x33,0x33},
    // A
00104 {0x3E,0x3E,0x33,0x33,0x33,0x3E,0x3E,0x33,0x33,0x33,0x3E,0x3E},
    // B
00105 {0x1E,0x1E,0x33,0x33,0x30,0x30,0x30,0x30,0x30,0x33,0x33,0x1E,0x1E},
    // C
00106 {0x3C,0x3C,0x36,0x36,0x33,0x33,0x33,0x33,0x33,0x36,0x36,0x3C,0x3C},
    // D
00107 {0x3F,0x3F,0x30,0x30,0x30,0x3E,0x3E,0x30,0x30,0x30,0x30,0x3F,0x3F},
    // E
00108 {0x3F,0x3F,0x30,0x30,0x30,0x3E,0x3E,0x30,0x30,0x30,0x30,0x30,0x30},
    // F
00109 {0x1E,0x1E,0x33,0x33,0x30,0x30,0x37,0x37,0x33,0x33,0x33,0x1F,0x1F},
    // G
00110 {0x33,0x33,0x33,0x33,0x33,0x3F,0x3F,0x33,0x33,0x33,0x33,0x33,0x33},
    // H
00111 {0x1E,0x1E,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x1E,0x1E},
    // I
00112 {0x0F,0x0F,0x06,0x06,0x06,0x06,0x06,0x06,0x06,0x36,0x36,0x1C,0x1C},
    // J
00113 {0x33,0x33,0x36,0x36,0x3C,0x3C,0x38,0x38,0x3C,0x3C,0x36,0x36,0x33,0x33},
    // K
00114 {0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x3F,0x3F},
    // L
00115 {0x33,0x33,0x3F,0x3F,0x3F,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33},
    // M
00116 {0x33,0x33,0x3B,0x3B,0x3F,0x3F,0x37,0x37,0x33,0x33,0x33,0x33,0x33},
    // N
00117 {0x1E,0x1E,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x1E,0x1E},
    // O
00118 {0x3E,0x3E,0x33,0x33,0x33,0x3E,0x3E,0x30,0x30,0x30,0x30,0x30},
    // P
00119 {0x1E,0x1E,0x33,0x33,0x33,0x33,0x33,0x37,0x37,0x33,0x33,0x33,0x33},
    // Q
00120 {0x3E,0x3E,0x33,0x33,0x33,0x3E,0x3E,0x3C,0x3C,0x36,0x36,0x33,0x33},
    // R
00121 {0x1E,0x1E,0x33,0x33,0x30,0x1E,0x1E,0x03,0x03,0x33,0x33,0x1E,0x1E},
    // S
00122 {0x3F,0x3F,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C},
    // T
00123 {0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x1E,0x1E},
    // U
00124 {0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x1E,0x1E,0x0C,0x0C},
    // V
00125 {0x33,0x33,0x33,0x33,0x33,0x3F,0x3F,0x3F,0x3F,0x33,0x33,0x33,0x33},
    // W
00126 {0x33,0x33,0x33,0x33,0x1E,0x1E,0x0C,0x0C,0x1E,0x33,0x33,0x33,0x33},
    // X
00127 {0x33,0x33,0x33,0x33,0x1E,0x1E,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C},
    // Y
00128 {0x3F,0x3F,0x03,0x03,0x06,0x06,0x0C,0x0C,0x18,0x18,0x30,0x30,0x3F,0x3F},
    // Z
00129 };
00130 static const uint8_t font8x14_rowminor[][14] = {
    // Alfabeto minúsculas 8 x 14
00131 {0x00,0x00,0x00,0x00,0x3C,0x3C,0x06,0x06,0x3E,0x3E,0x66,0x66,0x3E,0x3E},
    // a
00132 {0x60,0x60,0x60,0x60,0x7C,0x7C,0x66,0x66,0x66,0x66,0x66,0x66,0x7C,0x7C},
    // b
00133 {0x00,0x00,0x00,0x00,0x3C,0x3C,0x66,0x66,0x60,0x60,0x66,0x66,0x3C,0x3C},
    // c
00134 {0x06,0x06,0x06,0x06,0x3E,0x3E,0x66,0x66,0x66,0x66,0x66,0x66,0x3E,0x3E},
    // d
00135 {0x00,0x00,0x00,0x00,0x3C,0x3C,0x66,0x66,0x7E,0x7E,0x60,0x60,0x3C,0x3C},
    // e

```

```

00136     {0x1C,0x1C,0x30,0x30,0x30,0x7C,0x7C,0x30,0x30,0x30,0x30,0x30,0x30},
00137     // f
00138     {0x00,0x00,0x3E,0x3E,0x66,0x66,0x66,0x3E,0x06,0x06,0x7C,0x7C},
00139     // g
00140     {0x60,0x60,0x60,0x60,0x7C,0x7C,0x66,0x66,0x66,0x66,0x66,0x66,0x66,0x66},
00141     // h
00142     {0x18,0x18,0x00,0x00,0x38,0x38,0x18,0x18,0x18,0x18,0x18,0x18,0x3C,0x3C},
00143     // i
00144     {0x06,0x06,0x00,0x00,0x06,0x06,0x06,0x06,0x06,0x66,0x66,0x3C,0x3C},
00145     // j
00146     {0x60,0x60,0x60,0x66,0x66,0x6C,0x78,0x78,0x6C,0x6C,0x66,0x66},
00147     // k
00148     {0x38,0x38,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x3C,0x3C},
00149     // l
00150     {0x00,0x00,0x00,0x00,0x6C,0x6C,0x7E,0x7E,0x7E,0x6C,0x6C,0x6C,0x6C},
00151     // m
00152     {0x00,0x00,0x00,0x00,0x7C,0x7C,0x66,0x66,0x66,0x66,0x66,0x66,0x66,0x66},
00153     // n
00154     {0x00,0x00,0x7C,0x7C,0x66,0x66,0x66,0x7C,0x7C,0x60,0x60,0x60,0x60},
00155     // o
00156     {0x00,0x00,0x3E,0x3E,0x60,0x60,0x3C,0x3C,0x06,0x06,0x7C,0x7C},
00157     // p
00158 static const uint8_t font8x14_rownumber[][14] = {
00159     // Números 8 x 14
00160     {0x3C,0x66,0xC3,0xC3,0xC3,0xC3,0xC3,0xC3,0xC3,0xC3,0x66,0x3C,0x00},
00161     // 0
00162     {0x18,0x38,0x78,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x7E,0x00},
00163     // 1
00164     {0x3C,0x66,0xC3,0x03,0x06,0x0C,0x18,0x30,0x60,0xC0,0xC3,0xFF,0xFE,0x00},
00165     // 2
00166     {0x7E,0xC3,0x03,0x06,0x3C,0x06,0x03,0x03,0xC3,0x66,0x3C,0x00},
00167     // 3
00168     {0x06,0x0E,0x1E,0x36,0x66,0xC6,0x86,0xFF,0xFF,0x06,0x06,0x06,0x00},
00169     // 4
00170     {0xFE,0xC0,0xC0,0xFC,0xC6,0x03,0x03,0x03,0xC3,0x66,0x3C,0x00},
00171     // 5
00172     {0x3C,0x66,0xC3,0xC0,0xC0,0xFC,0xC6,0xC3,0xC3,0xC3,0x66,0x3C,0x00},
00173     // 6
00174     {0xFF,0xC3,0x03,0x06,0x0C,0x18,0x30,0x60,0x60,0x60,0x00},
00175     // 7
00176     {0x3C,0x66,0xC3,0xC3,0x66,0x3C,0x66,0xC3,0xC3,0x66,0x3C,0x00},
00177     // 8
00178     {0x3C,0x66,0xC3,0xC3,0x67,0x3F,0x03,0x03,0xC3,0x66,0x3C,0x00}
00179     // 9
00180 } bitmap_t;
00181
00182 typedef struct bitmap_t {
00183     const uint8_t *bitmap;
00184     uint8_t w;
00185     uint8_t h;
00186     int x;
00187     int y;
00188 } bitmap_t;
00189
00190 static void sh1106_draw_bitmap( sh1106_t *oled, bitmap_t *bitmap);
00191
00192 static void sh1106_draw_bitmap( sh1106_t *oled, bitmap_t *bitmap) {
00193     // Dibujado sencillo: copia bit por bit en el buffer
00194     // (Implementar según formato del bitmap)
00195     for (uint8_t by = 0; by < bitmap->h; by++) {
00196         for (uint8_t bx = 0; bx < bitmap->w; bx++) {
00197             uint8_t byte = bitmap->bitmap[by * ((bitmap->w + 7) / 8) + (bx / 8)];
00198             uint8_t bit = (byte >> (7 - (bx % 8))) & 0x01;
00199             if (bit) {
00200                 int px = bitmap->x + bx;
00201                 int py = bitmap->y + by;
00202             }
00203         }
00204     }
00205 }
```

```

00209         if (px >= 0 && px < oled->width && py >= 0 && py < oled->height) {
00210             uint8_t page = py / 8;
00211             uint8_t bitpos = py % 8;
00212             oled->buffer[page * oled->width + px] |= (1 << bitpos);
00213         }
00214     }
00215 }
00216 }
00217 }
00218
00219 void sh1106_draw_text( sh1106_t *oled, const char *text, int x, int y, uint8_t size) {
00220     uint8_t x_diff = 0;
00221     bitmap_t bitmap;
00222     if (size == SH1106_SIZE_1) {
00223         bitmap.w = 8;
00224         bitmap.h = 7;
00225     } else if (size == SH1106_SIZE_2) {
00226         bitmap.w = 8;
00227         bitmap.h = 14;
00228     } else {
00229         return;
00230     }
00231     bitmap.x = x;
00232     bitmap.y = y;
00233     for(uint8_t x_diff = 0, letter = 0; letter < strlen(text); x_diff++, letter++) {
00234
00235         if (x + 8 * x_diff > 120) {
00236             x_diff = 0;
00237             y_diff += 9;
00238         }
00239         bitmap.x = x + 8 * x_diff;
00240         bitmap.y = y + y_diff;
00241
00242         if (text[letter] >= 'A' && text[letter] <= 'Z') {
00243             if (size == SH1106_SIZE_1) {
00244                 bitmap.bitmap = font5x7_rowmajor[text[letter] - 'A'];
00245             } else if (size == SH1106_SIZE_2) {
00246                 bitmap.bitmap = font8x14_rowmajor[text[letter] - 'A'];
00247             }
00248         } else if (text[letter] >= 'a' && text[letter] <= 'z') {
00249             if (size == SH1106_SIZE_1) {
00250                 bitmap.bitmap = font5x7_rowminor[text[letter] - 'a'];
00251             } else if (size == SH1106_SIZE_2) {
00252                 bitmap.bitmap = font8x14_rowminor[text[letter] - 'a'];
00253             }
00254         } else if (text[letter] >= '0' && text[letter] <= '9') {
00255             if (size == SH1106_SIZE_1) {
00256                 bitmap.bitmap = font5x7_rownumber[text[letter] - '0'];
00257             } else if (size == SH1106_SIZE_2) {
00258                 bitmap.bitmap = font8x14_rownumber[text[letter] - '0'];
00259             }
00260         } else if (text[letter] == ':') {
00261             if (size == SH1106_SIZE_1) {
00262                 bitmap.bitmap = font5x7_double_dot;
00263             } else if (size == SH1106_SIZE_2) {
00264                 bitmap.bitmap = font8x14_double_dot;
00265             }
00266         } else if (text[letter] == '.') {
00267             if (size == SH1106_SIZE_1) {
00268                 bitmap.bitmap = font5x7_dot;
00269             } else if (size == SH1106_SIZE_2) {
00270                 bitmap.bitmap = font8x14_dot;
00271             }
00272         } else if (text[letter] == ',') {
00273             if (size == SH1106_SIZE_1) {
00274                 bitmap.bitmap = font5x7_comma;
00275             } else if (size == SH1106_SIZE_2) {
00276                 bitmap.bitmap = font8x14_comma;
00277             }
00278         } else if (text[letter] == '?') {
00279             if (size == SH1106_SIZE_1) {
00280                 bitmap.bitmap = font5x7_close_quest;
00281             } else if (size == SH1106_SIZE_2) {
00282                 bitmap.bitmap = font8x14_close_quest;
00283             }
00284         } else if (text[letter] == '!') {
00285             if (size == SH1106_SIZE_1) {
00286                 bitmap.bitmap = font5x7_close_excl;
00287             } else if (size == SH1106_SIZE_2) {
00288                 bitmap.bitmap = font8x14_close_excl;
00289             }
00290         } else if (text[letter] == '-') {
00291             if (size == SH1106_SIZE_1) {
00292                 bitmap.bitmap = font5x7_minus;
00293             } else if (size == SH1106_SIZE_2) {
00294                 bitmap.bitmap = font8x14_minus;
00295             }
}

```

```

00296     } else if ( text[letter] == '/' ) {
00297         if ( size == SH1106_SIZE_1 ) {
00298             bitmap.bitmap = slash;
00299         } else if ( size == SH1106_SIZE_2 ) {
00300             bitmap.bitmap = font8x14_space;           // No existe caracter '/' en tamaño 2
00301         }
00302     } else {                                // Si es un espacio u otro carácter no soportado
00303         if ( size == SH1106_SIZE_1 ) {
00304             bitmap.bitmap = font5x7_space;
00305         } else if ( size == SH1106_SIZE_2 ) {
00306             bitmap.bitmap = font8x14_space;
00307         }
00308     }
00309     sh1106_draw_bitmap( oled, &bitmap);
00310 }
00311 }
00312
00313 void sh1106_draw_utn_logo( sh1106_t *oled, int x, int y) {
00314     bitmap_t bitmap = {
00315         .bitmap = logo16_utn_bmp,
00316         .w = 16,
00317         .h = 12,
00318         .x = x,
00319         .y = y
00320     };
00321     sh1106_draw_bitmap( oled, &bitmap);
00322 }
00323
00324 void sh1106_draw_arrow( sh1106_t *oled, int x, int y) {
00325     bitmap_t bitmap = {
00326         .bitmap = arrow_bmp,
00327         .w = 16,
00328         .h = 7,
00329         .x = x,
00330         .y = y
00331     };
00332     sh1106_draw_bitmap( oled, &bitmap);
00333 }
00334
00335 void sh1106_draw_fail( sh1106_t *oled ) {
00336     bitmap_t bitmap = {
00337         .bitmap = fail_bmp,
00338         .w = 16,
00339         .h = 14,
00340         .x = 109,
00341         .y = 1
00342     };
00343     sh1106_draw_bitmap( oled, &bitmap);
00344 }
00345
00346 void sh1106_draw_line( sh1106_t *oled, int x, int y) {
00347     bitmap_t bitmap = {
00348         .bitmap = line_bmp,
00349         .w = 128,
00350         .h = 1,
00351         .x = x,
00352         .y = y
00353     };
00354     sh1106_draw_bitmap( oled, &bitmap);
00355 }

```

6.11. Referencia del archivo main/display/sh1106_graphics.h

Declaración de funciones que permiten operar sobre el display.

Estructuras de datos

- struct `sh1106_t`

typedefs

- `typedef struct sh1106_t sh1106_t`

Funciones

- void [sh1106_draw_text](#) ([sh1106_t](#) *oled, const char *text, int x, int y, uint8_t size)
- void [sh1106_draw_utn_logo](#) ([sh1106_t](#) *oled, int x, int y)
- void [sh1106_draw_arrow](#) ([sh1106_t](#) *oled, int x, int y)
- void [sh1106_draw_fail](#) ([sh1106_t](#) *oled)
Función que dibuja un signo de exclamación en la esquina superior derecha de la pantalla para expresar una falla en el sistema.
- void [sh1106_draw_line](#) ([sh1106_t](#) *oled, int x, int y)

6.11.1. Descripción detallada

Declaración de funciones que permiten operar sobre el display.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo [sh1106_graphics.h](#).

6.11.2. Documentación de «typedef»

6.11.2.1. sh1106_t

```
typedef struct sh1106_t sh1106_t
```

6.11.3. Documentación de funciones

6.11.3.1. sh1106_draw_arrow()

```
void sh1106_draw_arrow (
    sh1106\_t * oled,
    int x,
    int y)
```

Parámetros

<code>out</code>	<code>oled</code>	Puntero a la estructura que con la información que se enviará al display
------------------	-------------------	--

in	x	Coordenada en x donde se desea dibujar el logo de la UTN	
in	y	Coordenada en y donde se desea dibujar el logo de la UTN	

Definición en la línea 324 del archivo [sh1106_graphics.c](#).

6.11.3.2. sh1106_draw_fail()

```
void sh1106_draw_fail (
    sh1106_t * oled)
```

Función que dibuja un signo de exclamación en la esquina superior derecha de la pantalla para expresar una falla en el sistema.

Parámetros

out	oled	Puntero a la estructura que con la información que se enviará al display
-----	------	--

Definición en la línea 335 del archivo [sh1106_graphics.c](#).

6.11.3.3. sh1106_draw_line()

```
void sh1106_draw_line (
    sh1106_t * oled,
    int x,
    int y)
```

Parámetros

out	oled	Puntero a la estructura que con la información que se enviará al display
in	x	Coordenada en x donde se desea dibujar una línea del ancho del display
in	y	Coordenada en y donde se desea dibujar una línea del ancho del display

Definición en la línea 346 del archivo [sh1106_graphics.c](#).

6.11.3.4. sh1106_draw_text()

```
void sh1106_draw_text (
    sh1106_t * oled,
    const char * text,
    int x,
    int y,
    uint8_t size)
```

Parámetros

out	<i>oled</i>	Puntero a la estructura que con la información que se enviará al display
out	<i>oled</i>	Puntero a la estructura que con la información que se enviará al display
in	<i>x</i>	Coordenada en x donde se desea diujar el texto de caracteres alfanuméricos
in	<i>y</i>	Coordenada en y donde se desea diujar el texto de caracteres alfanuméricos
in	<i>size</i>	Largo del texto que se desea imprimir expresado en caracteres

Definición en la línea 219 del archivo [sh1106_graphics.c](#).

6.11.3.5. sh1106_draw_utn_logo()

```
void sh1106_draw_utn_logo (
    sh1106_t * oled,
    int x,
    int y)
```

Parámetros

out	<i>oled</i>	Puntero a la estructura que con la información que se enviará al display
in	<i>x</i>	Coordenada en x donde se desea diujar el texto de caracteres alfanuméricos
in	<i>y</i>	Coordenada en y donde se desea diujar el texto de caracteres alfanuméricos

Definición en la línea 313 del archivo [sh1106_graphics.c](#).

6.12. sh1106_graphics.h

[Ir a la documentación de este archivo.](#)

```
00001
00009 #ifndef __SH1106_GRAPHICS_H__
00010
00011 #define __SH1106_GRAPHICS_H__
00012
00013 typedef struct sh1106_t{
00014     uint8_t width;
00015     uint8_t height;
00016     uint8_t rotation;
00017     uint8_t buffer[128 * 8]; // 128 x 64 píxeles, 8 páginas de 8 pixeles verticales
00018 } sh1106_t;
00019
00040 void sh1106_draw_text( sh1106_t *oled, const char *text, int x, int y, uint8_t size);
00056 void sh1106_draw_utn_logo( sh1106_t *oled, int x, int y);
00072 void sh1106_draw_arrow( sh1106_t *oled, int x, int y);
00082 void sh1106_draw_fail( sh1106_t *oled );
00098 void sh1106_draw_line( sh1106_t *oled, int x, int y);
00099
00100#endif
```

6.13. Referencia del archivo main/display/sh1106_i2c.c

Funciones de hardware para el puerto I2C.

```
#include "freertos/Freertos.h"
#include "esp_err.h"
#include "driver/i2c.h"
#include "./sh1106_i2c.h"
```

defines

- `#define I2C_DISPLAY I2C_NUM_0`
- `#define SH1106_ADDR 0x3C`
- `#define CMD_CONTROL 0x00`
- `#define DATA_CONTROL 0x40`

Funciones

- `esp_err_t sh1106_write (const uint8_t *data, size_t len, sh1106_comm_type_t comm_type)`

Función que envía datos o comandos al disposit SH1106 a través del puerto I2C.

6.13.1. Descripción detallada

Funciones de hardware para el puerto I2C.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo [sh1106_i2c.c](#).

6.13.2. Documentación de «define»

6.13.2.1. CMD_CONTROL

```
#define CMD_CONTROL 0x00
```

Definición en la línea 16 del archivo [sh1106_i2c.c](#).

6.13.2.2. DATA_CONTROL

```
#define DATA_CONTROL 0x40
```

Definición en la línea 17 del archivo [sh1106_i2c.c](#).

6.13.2.3. I2C_DISPLAY

```
#define I2C_DISPLAY I2C_NUM_0
```

Definición en la línea 14 del archivo [sh1106_i2c.c](#).

6.13.2.4. SH1106_ADDR

```
#define SH1106_ADDR 0x3C
```

Definición en la línea 15 del archivo [sh1106_i2c.c](#).

6.13.3. Documentación de funciones

6.13.3.1. sh1106_write()

```
esp_err_t sh1106_write (
    const uint8_t * data,
    size_t len,
    sh1106_comm_type_t comm_type)
```

Función que envía datos o comandos al display SH1106 a través del puerto I2C.

Parámetros

in	<i>data</i>	Dato que se desea escribir en el display
in	<i>len</i>	Largo del dato que se quiere escribir en el display
in	<i>comm_type</i>	Tipo de comunicación que se desea: Dato o Comando

Valores devueltos



Definición en la línea 19 del archivo [sh1106_i2c.c](#).

6.14. sh1106_i2c.c

[Ir a la documentación de este archivo.](#)

```
00001
00009 #include "freertos/FreeRTOS.h"
00010 #include "esp_err.h"
00011 #include "driver/i2c.h"
00012 #include "./sh1106_i2c.h"
00013
00014 #define I2C_DISPLAY      I2C_NUM_0          // Número de puerto I2C utilizado para el display
00015 #define SH1106_ADDR      0x3C              // Address del display. Puede variar: 0x3C o 0x3D según como se
                                                // configure el hardware
00016 #define CMD_CONTROL     0x00              // Control byte para comandos
00017 #define DATA_CONTROL    0x40              // Control byte para datos
00018
```

```

00019 esp_err_t sh1106_write(const uint8_t *data, size_t len, sh1106_comm_type_t comm_type) {
00020     i2c_cmd_handle_t cmdh = i2c_cmd_link_create();
00021     i2c_master_start(cmdh);
00022     i2c_master_write_byte(cmdh, (SH1106_ADDR << 1) | I2C_MASTER_WRITE, true);
00023
00024     if (comm_type == SH1106_COMM_CMD) {
00025         i2c_master_write_byte(cmdh, CMD_CONTROL, true);
00026         i2c_master_write_byte(cmdh, *data, true);
00027     } else {
00028         i2c_master_write_byte(cmdh, DATA_CONTROL, true);
00029         i2c_master_write(cmdh, data, len, true);
00030     }
00031
00032     i2c_master_stop(cmdh);
00033     esp_err_t ret = i2c_master_cmd_begin(I2C_DISPLAY, cmdh, pdMS_TO_TICKS(1000));
00034     i2c_cmd_link_delete(cmdh);
00035
00036 }

```

6.15. Referencia del archivo main/display/sh1106_i2c.h

Declaración de funciones de hardware para el puerto I2C.

Enumeraciones

- enum [sh1106_comm_type_t](#) { [SH1106_COMM_CMD](#) = 0 , [SH1106_COMM_DATA](#) }
- Tipo de comunicación I2C con el display SH1106: Comando o Dato.*

Funciones

- [esp_err_t sh1106_write](#) (const uint8_t *data, size_t len, [sh1106_comm_type_t](#) comm_type)
- Función que envía datos o comandos al display SH1106 a través del puerto I2C.*

6.15.1. Descripción detallada

Declaración de funciones de hardware para el puerto I2C.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo [sh1106_i2c.h](#).

6.15.2. Documentación de enumeraciones

6.15.2.1. sh1106_comm_type_t

enum [sh1106_comm_type_t](#)

Tipo de comunicación I2C con el display SH1106: Comando o Dato.

Valores de enumeraciones

SH1106_COMM_CMD	
SH1106_COMM_DATA	

Definición en la línea 18 del archivo [sh1106_i2c.h](#).

6.15.3. Documentación de funciones

6.15.3.1. sh1106_write()

```
esp_err_t sh1106_write (
    const uint8_t * data,
    size_t len,
    sh1106_comm_type_t comm_type)
```

Función que envía datos o comandos al display SH1106 a través del puerto I2C.

Parámetros

in	<i>data</i>	Dato que se desea escribir en el display
in	<i>len</i>	Largo del dato que se quiere escribir en el display
in	<i>comm_type</i>	Tipo de comunicación que se desea: Dato o Comando

Valores devueltos

--	--

Definición en la línea 19 del archivo [sh1106_i2c.c](#).

6.16. sh1106_i2c.h

[Ir a la documentación de este archivo.](#)

```
00001
00009 #ifndef __SH1106_I2C_H__
00010
00011 #define __SH1106_I2C_H__
00012
00018 typedef enum {
00019     SH1106_COMM_CMD = 0,
00020     SH1106_COMM_DATA
00021 } sh1106_comm_type_t;
00022
00039 esp_err_t sh1106_write(const uint8_t *data, size_t len, sh1106_comm_type_t comm_type);
00040
00041 #endif
```

6.17. Referencia del archivo main/io_control/io_control.c

Funciones de control del relay, buzzer y tarea del uso del expansor de entradas y salidas y entradas y salidas aisladas directas al ESP32.

```
#include "sdkconfig.h"
#include "esp_err.h"
#include "esp_log.h"
#include "driver/gpio.h"
#include "driver/ledc.h"
#include "display/display.h"
#include "freertos/FreRTOS.h"
#include "freertos/semphr.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include <inttypes.h>
#include <stdio.h>
#include <stdlib.h>
#include "mcp23017_defs.h"
#include "./MCP23017.h"
#include "./io_control.h"
#include "../system/sysAdmin.h"
#include "../system/sysControl.h"
```

defines

- #define INT_A_PIN GPIO_NUM_22
- #define BUZZER_PIN 0
- #define RELAY_PIN 1
- #define FREQ_SEL_1_PIN 2
- #define FREQ_SEL_2_PIN 3
- #define FREQ_SEL_3_PIN 4
- #define STOP_PIN 5
- #define TERMO_SW_PIN 6
- #define FREQ_SEL_MASK ((1 << FREQ_SEL_3_PIN) | (1 << FREQ_SEL_2_PIN) | (1 << FREQ_SEL_1_PIN))
- #define TERMO_SW_MASK (1 << TERMO_SW_PIN)
- #define STOP_SW_MASK (1 << STOP_PIN)
- #define INT_B_PIN GPIO_NUM_23
- #define MATRIX_SW1 0x17
- #define MATRIX_SW2 0x27
- #define MATRIX_SW3 0x1B
- #define MATRIX_SW4 0x2B
- #define MATRIX_SW5 0x1D
- #define MATRIX_SW6 0x2D
- #define MATRIX_SW7 0x1E
- #define MATRIX_SW8 0x2E
- #define SWITCH_BUTTON_BACK MATRIX_SW1
- #define SWITCH_BUTTON_UP MATRIX_SW2
- #define SWITCH_BUTTON_LEFT MATRIX_SW3
- #define SWITCH_BUTTON_RIGHT MATRIX_SW4
- #define SWITCH_BUTTON_DOWN MATRIX_SW5
- #define SWITCH_BUTTON_SAVE MATRIX_SW6

- #define SWITCH_BUTTON_OK MATRIX_SW7
- #define SWITCH_BUTTON_MENU MATRIX_SW8
- #define STOP_BUTTON_PIN GPIO_NUM_16
- #define START_BUTTON_PIN GPIO_NUM_17
- #define PWM_GPIO 4
- #define PWM_FREQ_HZ 1000
- #define PWM_TIMER LEDC_TIMER_0
- #define PWM_MODE LEDC_LOW_SPEED_MODE
- #define PWM_CHANNEL LEDC_CHANNEL_0
- #define PWM_RES LEDC_TIMER_10_BIT
- #define PWM_STEP_MS 250
- #define DUTY_MIN_PCT 54
- #define DUTY_MAX_PCT 98

Funciones

- static esp_err_t freq_0_10V_output ()

Inicializa el timer para el correcto funcionamiento del PWM que permite obtener la salida de 0 a 10V analógicos.
- static esp_err_t gpio_init_interrupts (void)

Inicializa las interrupciones para INTA y INTB del MCP23017 y para los botones aislados stop y start.
- static void MCP23017_buzzer_control (void *arg)

Tarea que espera comandos a través de la cola buzzer_evt_queue para hacer sonar al buzzer.
- static void MCP23017_keyboard_control (void *arg)

Tarea que controla las salidas del MCP23017 para hacer funcionar al teclado.
- static void MCP23017_relay_control (void *arg)

Tarea que espera comandos a través de la cola relay_evt_queue para activar o desactivar el relay.
- void IRAM_ATTR gpio_isr_handler (void *arg)

Convierte la frecuencia en Hz que está girando el motor en el duty necesario para poder operar el PWM de la salida 0-10V.
- void GPIO_interrupt_attendance_task (void *arg)

Tarea que en función de las interrupciones recibidas permite obtener cual de los pulsadores del teclado, entradas aisladas, termo switch o entradas del MCP2307 fue presionado.
- esp_err_t RelayEventPost (uint8_t relay_event)

Crea un evento para que el relay se active o desactive de acuerdo a la variable relay_event.
- esp_err_t BuzzerEventPost (uint32_t buzzer_time_on)

Envía una señal para hacer sonar el buzzer durante buzzer_time_on milisegundos.

Variables

- static const char * TAG = "IO_CTRL"
- QueueHandle_t buzzer_evt_queue = NULL
- QueueHandle_t relay_evt_queue = NULL
- QueueHandle_t GPIO_evt_queue = NULL
- uint8_t mcp_porta
- uint8_t mcp_portb

6.17.1. Descripción detallada

Funciones de control del relay, buzzer y tarea del uso del expensor de entradas y salidas y entradas y salidas aisladas directas al ESP32.

Autor

Andrenacci - Carra

Las tareas de relay y buzzer son tareas aisladas del resto del sistema y son accedidas a través de las funciones RelayEventPost y BuzzerEventPost. Esto genera que no sea necesaria que la cola de comandos deba ser utilizada en diversos archivos, protege los valores que envía antes de generar el procesamiento del dato desde las respectivas tareas. Cuenta además con la selección, elección y control de entradas que le corresponde a cada tecla del teclado, cada entrada digital aislada. Trabaja con el antirrubote de las entradas, evitando que se filtren pulsos de ruido que hayan sido detectadas como cambios en las entradas o evitar que se generen más de un evento cuando en realidad era uno solo. También corre la tarea que controla la salida de 0 a 10 volts de continua, la señal que representa indirectamente la frecuencia de operación del motor.

Versión

2.0

Fecha

2025-11-06

Definición en el archivo [io_control.c](#).

6.17.2. Documentación de «define»

6.17.2.1. BUZZER_PIN

```
#define BUZZER_PIN 0
```

Pin del puerto A que controla el buzzer de la HMI

Definición en la línea [41](#) del archivo [io_control.c](#).

6.17.2.2. DUTY_MAX_PCT

```
#define DUTY_MAX_PCT 98
```

Máximo duty que puede alcanzar el PWM para lograr los 0V de salida

Definición en la línea [84](#) del archivo [io_control.c](#).

6.17.2.3. DUTY_MIN_PCT

```
#define DUTY_MIN_PCT 54
```

Mínimo duty que puede alcanzar el PWM para lograr los 10V de salida

Definición en la línea [83](#) del archivo [io_control.c](#).

6.17.2.4. FREQ_SEL_1_PIN

```
#define FREQ_SEL_1_PIN 2
```

Pin del puerto A que controla la selección de frecuencia 1 del variador de frecuencia

Definición en la línea [43](#) del archivo [io_control.c](#).

6.17.2.5. FREQ_SEL_2_PIN

```
#define FREQ_SEL_2_PIN 3
```

Pin del puerto A que controla la selección de frecuencia 2 del variador de frecuencia

Definición en la línea [44](#) del archivo [io_control.c](#).

6.17.2.6. FREQ_SEL_3_PIN

```
#define FREQ_SEL_3_PIN 4
```

Pin del puerto A que controla la selección de frecuencia 3 del variador de frecuencia

Definición en la línea [45](#) del archivo [io_control.c](#).

6.17.2.7. FREQ_SEL_MASK

```
#define FREQ_SEL_MASK ( (1 << FREQ_SEL_3_PIN) | (1 << FREQ_SEL_2_PIN) | (1 << FREQ_SEL_1_PIN) )
```

Máscara que, comparada con el registro de flags de interrupción, permite identificar cambios en las entradas del variador de frecuencia

Definición en la línea [49](#) del archivo [io_control.c](#).

6.17.2.8. INT_A_PIN

```
#define INT_A_PIN GPIO_NUM_22
```

Configuración de pin físico correspondiente al flag de interrupciones del puerto A del MCP23017

Definición en la línea [39](#) del archivo [io_control.c](#).

6.17.2.9. INT_B_PIN

```
#define INT_B_PIN GPIO_NUM_23
```

Configuración de pin físico correspondiente al flag de interrupciones del puerto B del MCP23017

Definición en la línea [53](#) del archivo [io_control.c](#).

6.17.2.10. MATRIX_SW1

```
#define MATRIX_SW1 0x17
```

Representación de switch impreso como SW1 en la serigrafía por 0x17

Definición en la línea 55 del archivo [io_control.c](#).

6.17.2.11. MATRIX_SW2

```
#define MATRIX_SW2 0x27
```

Representación de switch impreso como SW2 en la serigrafía por 0x27

Definición en la línea 56 del archivo [io_control.c](#).

6.17.2.12. MATRIX_SW3

```
#define MATRIX_SW3 0x1B
```

Representación de switch impreso como SW3 en la serigrafía por 0x1B

Definición en la línea 57 del archivo [io_control.c](#).

6.17.2.13. MATRIX_SW4

```
#define MATRIX_SW4 0x2B
```

Representación de switch impreso como SW4 en la serigrafía por 0x2B

Definición en la línea 58 del archivo [io_control.c](#).

6.17.2.14. MATRIX_SW5

```
#define MATRIX_SW5 0x1D
```

Representación de switch impreso como SW5 en la serigrafía por 0x1D

Definición en la línea 59 del archivo [io_control.c](#).

6.17.2.15. MATRIX_SW6

```
#define MATRIX_SW6 0x2D
```

Representación de switch impreso como SW6 en la serigrafía por 0x2D

Definición en la línea 60 del archivo [io_control.c](#).

6.17.2.16. MATRIX_SW7

```
#define MATRIX_SW7 0x1E
```

Representación de switch impreso como SW7 en la serigrafía por 0x1E

Definición en la línea [61](#) del archivo [io_control.c](#).

6.17.2.17. MATRIX_SW8

```
#define MATRIX_SW8 0x2E
```

Representación de switch impreso como SW8 en la serigrafía por 0x2E

Definición en la línea [62](#) del archivo [io_control.c](#).

6.17.2.18. PWM_CHANNEL

```
#define PWM_CHANNEL LEDC_CHANNEL_0
```

Canal del timer utilizado para el PWM que controla la salida 0-10V

Definición en la línea [80](#) del archivo [io_control.c](#).

6.17.2.19. PWM_FREQ_HZ

```
#define PWM_FREQ_HZ 1000
```

Frecuencia de operación del timer para la salida 0-10V en 1KHz

Definición en la línea [77](#) del archivo [io_control.c](#).

6.17.2.20. PWM_GPIO

```
#define PWM_GPIO 4
```

GPIO4 es la utilizada para la salida de 0-10V en el PCB

Definición en la línea [76](#) del archivo [io_control.c](#).

6.17.2.21. PWM_MODE

```
#define PWM_MODE LEDC_LOW_SPEED_MODE
```

El requerimiento del timer para el control de los 0-10V no amerita un timer de alta velocidad

Definición en la línea [79](#) del archivo [io_control.c](#).

6.17.2.22. PWM_RES

```
#define PWM_RES LEDC_TIMER_10_BIT
```

Resolución de 10 bits (0–1023) para el ADC que

Definición en la línea 81 del archivo [io_control.c](#).

6.17.2.23. PWM_STEP_MS

```
#define PWM_STEP_MS 250
```

Cambio cada 500 ms

Definición en la línea 82 del archivo [io_control.c](#).

6.17.2.24. PWM_TIMER

```
#define PWM_TIMER LEDC_TIMER_0
```

Timer utilizado para el PWM que controla la salida 0-10V

Definición en la línea 78 del archivo [io_control.c](#).

6.17.2.25. RELAY_PIN

```
#define RELAY_PIN 1
```

Pin del puerto A que controla el relé de la HMI

Definición en la línea 42 del archivo [io_control.c](#).

6.17.2.26. START_BUTTON_PIN

```
#define START_BUTTON_PIN GPIO_NUM_17
```

Configuración de pin físico correspondiente al pulsado exterior de arranque del motor

Definición en la línea 74 del archivo [io_control.c](#).

6.17.2.27. STOP_BUTTON_PIN

```
#define STOP_BUTTON_PIN GPIO_NUM_16
```

Configuración de pin físico correspondiente al pulsado exterior de parada del motor

Definición en la línea 73 del archivo [io_control.c](#).

6.17.2.28. STOP_PIN

```
#define STOP_PIN 5
```

Pin del puerto A que controla el botón de parada delvariador de frecuencia

Definición en la línea 46 del archivo [io_control.c](#).

6.17.2.29. STOP_SW_MASK

```
#define STOP_SW_MASK (1 << STOP_PIN)
```

Máscara que, comparada con el registro de flags de interrupción, permite identificar cambios en el botón de parada del variador de frecuencia

Definición en la línea 51 del archivo [io_control.c](#).

6.17.2.30. SWITCH_BUTTON_BACK

```
#define SWITCH_BUTTON_BACK MATRIX_SW1
```

Asignación de botón al SW1 representado por 0x17

Definición en la línea 64 del archivo [io_control.c](#).

6.17.2.31. SWITCH_BUTTON_DOWN

```
#define SWITCH_BUTTON_DOWN MATRIX_SW5
```

Asignación de botón al SW5 representado por 0x1D

Definición en la línea 68 del archivo [io_control.c](#).

6.17.2.32. SWITCH_BUTTON_LEFT

```
#define SWITCH_BUTTON_LEFT MATRIX_SW3
```

Asignación de botón al SW3 representado por 0x1B

Definición en la línea 66 del archivo [io_control.c](#).

6.17.2.33. SWITCH_BUTTON_MENU

```
#define SWITCH_BUTTON_MENU MATRIX_SW8
```

Asignación de botón al SW8 representado por 0x2E

Definición en la línea 71 del archivo [io_control.c](#).

6.17.2.34. SWITCH_BUTTON_OK

```
#define SWITCH_BUTTON_OK MATRIX_SW7
```

Asignación de botón al SW7 representado por 0x1E

Definición en la línea 70 del archivo [io_control.c](#).

6.17.2.35. SWITCH_BUTTON_RIGHT

```
#define SWITCH_BUTTON_RIGHT MATRIX_SW4
```

Asignación de botón al SW4 representado por 0x2B

Definición en la línea 67 del archivo [io_control.c](#).

6.17.2.36. SWITCH_BUTTON_SAVE

```
#define SWITCH_BUTTON_SAVE MATRIX_SW6
```

Asignación de botón al SW6 representado por 0x2D

Definición en la línea 69 del archivo [io_control.c](#).

6.17.2.37. SWITCH_BUTTON_UP

```
#define SWITCH_BUTTON_UP MATRIX_SW2
```

Asignación de botón al SW2 representado por 0x27

Definición en la línea 65 del archivo [io_control.c](#).

6.17.2.38. TERMO_SW_MASK

```
#define TERMO_SW_MASK (1 << TERMO_SW_PIN)
```

Máscara que, comparada con el registro de flags de interrupción, permite identificar cambios en el interruptor térmico del variador de frecuencia

Definición en la línea 50 del archivo [io_control.c](#).

6.17.2.39. TERMO_SW_PIN

```
#define TERMO_SW_PIN 6
```

Pin del puerto A que controla el interruptor térmico del variador de frecuencia

Definición en la línea 47 del archivo [io_control.c](#).

6.17.3. Documentación de funciones

6.17.3.1. BuzzerEventPost()

```
esp_err_t BuzzerEventPost (
    uint32_t buzzer_time_on)
```

Envía una señal para hacer sonar el buzzer durante `buzzer_time_on` mili segundos.

Parámetros

in	buzzer_time_on	Tiempo en milisegundos durante el que el buzzer sonará.
----	----------------	---

Devuelve

- ESP_OK Evento creado exitosamente
- ESP_FAIL Error al crear el evento

Definición en la línea 601 del archivo [io_control.c](#).

6.17.3.2. freq_0_10V_output()

```
esp_err_t freq_0_10V_output () [static]
```

Inicializa el timer para el correcto funcionamiento del PWM que permite obtener la salida de 0 a 10V analógicos.

Devuelve

- ESP_OK: En caso de éxito.
- ESP_ERR_INVALID_ARG: Error al pasar parámetros de configuración de timer o canal de PWM.
- ESP_FAIL: No se pudo encontrar un pre divisor apropiado para la base de frecuencia y resolución de duty dado.
- ESP_ERR_INVALID_STATE: Timer no pudo ser desconfigurado porque el timer no se configuró previamente o está pausado.

Definición en la línea 174 del archivo [io_control.c](#).

6.17.3.3. gpio_init_interrupts()

```
esp_err_t gpio_init_interrupts (
    void ) [static]
```

Inicializa las interrupciones para INTA y INTB del MCP23017 y para los botones aislados stop y start.

Las interrupciones de INTA y INTB se dispararán cada vez que los pines pasen de estado alto a bajo, ninguno de ellos tendrá activo el pull-up o pull-down ya que los pines del MCP23017 trabajarán como pines activos. Las interrupciones de start y stop serán disparadas por flanco alto o bajo y tampoco tendrán activos ni sus pull-ups ni pull-downs

Devuelve

- ESP_OK: en caso de éxito
- ESP_FAIL: No se pudo inicializar alguna de las colas de eventos.
- ESP_ERR_INVALID_STATE: El handler de interrupciones no fue inicializado correctamente o se intentó inicializar la interrupción más de una vez
- ESP_ERR_INVALID_ARG: Error en la configuración de los parámetros o hay un error en los GPIO
- ESP_ERR_NO_MEM: No hay memoria suficiente para instalar las interrupciones
- ESP_ERR_NOT_FOUND: No hay interrupciones disponibles para instalar con la configuración solicitada

<

<

Definición en la línea 211 del archivo [io_control.c](#).

6.17.3.4. GPIO_interrupt_attendance_task()

```
void GPIO_interrupt_attendance_task (
    void * arg)
```

Tarea que en función de las interrupciones recibidas permite obtener cual de los pulsadores del teclado, entradas aisladas, termo switch o entradas del MCP2307 fue presionado.

Trabaja con un antirrebote de 200ms. Cuando una entrada cambia su estado, la tarea vuelve a leer las entradas luego de 200ms para asegurarse el valor leído y recién ahí envía la señal correspondiente

Definición en la línea 386 del archivo [io_control.c](#).

6.17.3.5. gpio_isr_handler()

```
void IRAM_ATTR gpio_isr_handler (
    void * arg)
```

Definición en la línea 164 del archivo [io_control.c](#).

6.17.3.6. MCP23017_buzzer_control()

```
void MCP23017_buzzer_control (
    void * arg) [static]
```

Tarea que espera comandos a través de la cola `buzzer_evt_queue` para hacer sonar al buzzer.

Lo que espera la tarea a través de la cola, es el tiempo en milisegundos que hará sonar el buzzer.

Definición en la línea 289 del archivo [io_control.c](#).

6.17.3.7. MCP23017_keyboard_control()

```
void MCP23017_keyboard_control (
    void * arg) [static]
```

Tarea que controla las salidas del MCP23017 para hacer funcionar al teclado.

Alternadamente activa y desactiva los pines 4 y 5 del puerto B siempre y cuando ninguna tecla esté siendo presionada.

Definición en la línea 313 del archivo [io_control.c](#).

6.17.3.8. MCP23017_relay_control()

```
void MCP23017_relay_control (
    void * arg) [static]
```

Tarea que espera comandos a través de la cola `relay_evt_queue` para activar o desactivar el relay.

Lo que espera la tarea a través de la cola, es un uno para activar el relay; cualquier otro número recibido desenergizará la bobina del relay.

Definición en la línea 334 del archivo [io_control.c](#).

6.17.3.9. RelayEventPost()

```
esp_err_t RelayEventPost (
    uint8_t relay_event)
```

Crea un evento para que el relay se active o desactive de acuerdo a la variable `relay_event`.

Con `relay_event` en 1 activa el relay, con 0 desactiva el relay. Cualquier otro valor retorna `ESP_FAIL`.

Parámetros

in	<i>relay_event</i>	Estado del relay
----	--------------------	------------------

Devuelve

- ESP_OK Evento creado exitosamente
- ESP_FAIL Error al crear el evento

Definición en la línea 593 del archivo [io_control.c](#).

6.17.3.10. `set_freq_output()`

```
esp_err_t set_freq_output (
    uint16_t freq)
```

Convierte la frecuencia en Hz que está girando el motor en el duty necesario para poder operar el PWM de la salida 0-10V.

Permite configurar el duty del PWM desde 0 a 450Hz siendo que, para 0Hz la salida será 0V y para 450HZ serán 10V.

Parámetros

in	<i>freq</i>	Es la frecuencia que está girando el motor.
----	-------------	---

Devuelve

- ESP_OK PWM configurado correctamente
- ESP_ERR_INVALID_ARG Error al configurar el duty del PWM

Definición en la línea 359 del archivo [io_control.c](#).

6.17.4. Documentación de variables

6.17.4.1. `buzzer_evt_queue`

```
QueueHandle_t buzzer_evt_queue = NULL
```

Cola de eventos para el control del buzzer. Espera tiempo de activación del buzzer, tiempo en el que se lo escuchará sonar

Definición en la línea 86 del archivo [io_control.c](#).

6.17.4.2. `GPIO_evt_queue`

```
QueueHandle_t GPIO_evt_queue = NULL
```

Cola de eventos para las entradas digitales aisladas y directas sobre el ESP32 y las que llegan al MCP23017. Solo puede encolar una acción desde dentro de este archivo

Definición en la línea 88 del archivo [io_control.c](#).

6.17.4.3. mcp_porta

```
uint8_t mcp_porta
```

Estado de entradas del puerto A del MCP23017

Definición en la línea 90 del archivo [io_control.c](#).

6.17.4.4. mcp_portb

```
uint8_t mcp_portb
```

Estado de entradas del puerto B del MCP23017

Definición en la línea 91 del archivo [io_control.c](#).

6.17.4.5. relay_evt_queue

```
QueueHandle_t relay_evt_queue = NULL
```

Cola de eventos para el control del relay. Admite solo 1 para energizarlo y 0 para desenergizarlo

Definición en la línea 87 del archivo [io_control.c](#).

6.17.4.6. TAG

```
const char* TAG = "IO_CTRL" [static]
```

Variable que se imprime en los ESP_LOG

Definición en la línea 37 del archivo [io_control.c](#).

6.18. io_control.c

[Ir a la documentación de este archivo.](#)

```
00001
00012 #include "sdkconfig.h"
00013
00014 #include "esp_err.h"
00015 #include "esp_log.h"
00016
00017 #include "driver/gpio.h"
00018 #include "driver/ledc.h"
00019 #include "display/display.h"
00020
00021 #include "freertos/FreeRTOS.h"
00022 #include "freertos/semphr.h"
00023 #include "freertos/task.h"
00024 #include "freertos/queue.h"
00025
00026 #include <inttypes.h>
00027 #include <stdio.h>
00028 #include <stdlib.h>
00029
00030 #include "mcp23017_defs.h"
00031
00032 #include "./MCP23017.h"
```

```
00033 #include "./io_control.h"
00034 #include "../system/sysAdmin.h"
00035 #include "../system/sysControl.h"
00036
00037 static const char *TAG = "IO_CTRL";
00038
00039 #define INT_A_PIN           GPIO_NUM_22
00040
00041 #define BUZZER_PIN          0
00042 #define RELAY_PIN            1
00043 #define FREQ_SEL_1_PIN        2
00044 #define FREQ_SEL_2_PIN        3
00045 #define FREQ_SEL_3_PIN        4
00046 #define STOP_PIN              5
00047 #define TERMO_SW_PIN         6
00048
00049 #define FREQ_SEL_MASK        ((1 << FREQ_SEL_3_PIN) | (1 << FREQ_SEL_2_PIN) | (1 <<
00050   FREQ_SEL_1_PIN))
00050 #define TERMO_SW_MASK        (1 << TERMO_SW_PIN)
00051 #define STOP_SW_MASK          (1 << STOP_PIN)
00052
00053 #define INT_B_PIN           GPIO_NUM_23
00054
00055 #define MATRIX_SW1            0x17
00056 #define MATRIX_SW2            0x27
00057 #define MATRIX_SW3            0x1B
00058 #define MATRIX_SW4            0x2B
00059 #define MATRIX_SW5            0x1D
00060 #define MATRIX_SW6            0x2D
00061 #define MATRIX_SW7            0x1E
00062 #define MATRIX_SW8            0x2E
00063
00064 #define SWITCH_BUTTON_BACK    MATRIX_SW1
00065 #define SWITCH_BUTTON_UP      MATRIX_SW2
00066 #define SWITCH_BUTTON_LEFT    MATRIX_SW3
00067 #define SWITCH_BUTTON_RIGHT   MATRIX_SW4
00068 #define SWITCH_BUTTON_DOWN    MATRIX_SW5
00069 #define SWITCH_BUTTON_SAVE    MATRIX_SW6
00070 #define SWITCH_BUTTON_OK      MATRIX_SW7
00071 #define SWITCH_BUTTON_MENU    MATRIX_SW8
00072
00073 #define STOP_BUTTON_PIN       GPIO_NUM_16
00074 #define START_BUTTON_PIN      GPIO_NUM_17
00075
00076 #define PWM_GPIO              4
00077 #define PWM_FREQ_HZ           1000
00078 #define PWM_TIMER             LEDC_TIMER_0
00079 #define PWM_MODE               LEDC_LOW_SPEED_MODE
00080 #define PWM_CHANNEL            LEDC_CHANNEL_0
00081 #define PWM_RES                LEDC_TIMER_10_BIT
00082 #define PWM_STEP_MS             250
```

```
00083 #define DUTY_MIN_PCT          54
00084 #define DUTY_MAX_PCT          98
00085
00086 QueueHandle_t buzzer_evt_queue = NULL;
00087 QueueHandle_t relay_evt_queue = NULL;
00088 QueueHandle_t GPIO_evt_queue = NULL;
00089
00090 uint8_t mcp_porta;
00091 uint8_t mcp_portb;
00092
00104 static esp_err_t freq_0_10V_output();
00105
00121 static esp_err_t gpio_init_interrupts(void);
00122
00131 static void MCP23017_buzzer_control(void* arg);
00132
00141 static void MCP23017_keyboard_control(void* arg);
00142
00151 static void MCP23017_relay_control(void* arg);
00152
00153
00164 void IRAM_ATTR gpio_isr_handler(void* arg) {
00165     uint32_t gpio_num = (uint32_t) arg;
00166     BaseType_t xHigherPriorityTaskWoken = pdFALSE;
00167
00168     xQueueSendFromISR(GPIO_evt_queue, &gpio_num, &xHigherPriorityTaskWoken);
00169     if (xHigherPriorityTaskWoken) {
00170         portYIELD_FROM_ISR();
00171     }
00172 }
00173
00174 static esp_err_t freq_0_10V_output() {
00175     esp_err_t retval;
00176     // Configurar el temporizador del PWM
00177     ledc_timer_config_t ledc_timer = {
00178         .speed_mode      = PWM_MODE,
00179         .timer_num       = PWM_TIMER,
00180         .duty_resolution = PWM_RES,
00181         .freq_hz         = PWM_FREQ_HZ,
00182         .clk_cfg         = LEDC_AUTO_CLK
00183     };
00184     retval = ledc_timer_config(&ledc_timer);
00185
00186     if (retval != ESP_OK) {
00187         ESP_LOGE(TAG, "Error al configurar el timer de la salida 0-10V");
00188         return retval;
00189     }
00190
00191     // Configurar el canal
00192     ledc_channel_config_t ledc_channel = {
00193         .gpio_num        = PWM_GPIO,
00194         .speed_mode      = PWM_MODE,
00195         .channel         = PWM_CHANNEL,
00196         .timer_sel       = PWM_TIMER,
00197         .duty            = 0,
00198         .hpoint          = 0
00199     };
00200
00201     retval = ledc_channel_config(&ledc_channel);
00202
00203     if (retval != ESP_OK) {
00204         ESP_LOGE(TAG, "Error al configurar el canal de la salida 0-10V");
00205         return retval;
00206     }
00207
00208     return ESP_OK;
00209 }
00210
00211 static esp_err_t gpio_init_interrupts(void) {
00212     esp_err_t retval;
00213     gpio_config_t io_conf_stop_start;
00214
00215     io_conf_stop_start.io_conf =
00216
00217     if (GPIO_evt_queue == NULL) {
00218         GPIO_evt_queue = xQueueCreate(1, sizeof(uint32_t));
00219         if (GPIO_evt_queue == NULL) {
```

```

00220         ESP_LOGE(TAG, "No se pudo crear la cola de interrupciones");
00221         return ESP_FAIL;
00222     }
00223 }
00224
00225     retval = gpio_install_isr_service(0);
00226     if (retval != ESP_OK) {
00227         ESP_LOGE(TAG, "Error al configurar la interrupción de GPIO para INTA y INTB del MCP23017");
00228         return retval;
00229     }
00230
00231 // Configuración de pines
00232 io_conf.intr_type = GPIO_INTR_NEGEDGE;
00233 io_conf.mode = GPIO_MODE_INPUT;
00234 io_conf.pin_bit_mask = (1ULL<<INT_A_PIN) | (1ULL<<INT_B_PIN);
00235 io_conf.pull_down_en = GPIO_PULLDOWN_DISABLE;
00236 io_conf.pull_up_en = GPIO_PULLUP_DISABLE;
00237
00238     retval = gpio_config(&io_conf);
00239     if (retval != ESP_OK) {
00240         ESP_LOGE(TAG, "Error al configurar los GPIO");
00241         return retval;
00242     }
00243
00244 // Asocio pines a la rutina ISR
00245     retval = gpio_isr_handler_add(INT_A_PIN, gpio_isr_handler, (void*) (uintptr_t) INT_A_PIN);
00246     if (retval != ESP_OK) {
00247         ESP_LOGE(TAG, "Error al configurar la interrupción de GPIO para pin INTA del MCP23017");
00248         return retval;
00249     }
00250
00251     retval = gpio_isr_handler_add(INT_B_PIN, gpio_isr_handler, (void*) (uintptr_t) INT_B_PIN);
00252     if (retval != ESP_OK) {
00253         ESP_LOGE(TAG, "Error al configurar la interrupción de GPIO para pin INTB del MCP23017");
00254         return retval;
00255     }
00256
00257     ESP_LOGI(TAG, "Interrupciones GPIO configuradas en %d (Int A MCP) y %d (Int B MCP)", INT_A_PIN,
00258             INT_B_PIN);
00259
00260 // Configuración de pines
00261 io_conf_stop_start.intr_type = GPIO_INTR_ANYEDGE;
00262 io_conf_stop_start.mode = GPIO_MODE_INPUT;
00263 io_conf_stop_start.pin_bit_mask = (1ULL<<STOP_BUTTON_PIN) | (1ULL<<START_BUTTON_PIN);
00264 io_conf_stop_start.pull_down_en = GPIO_PULLDOWN_DISABLE;
00265 io_conf_stop_start.pull_up_en = GPIO_PULLUP_DISABLE;
00266
00267     retval = gpio_config(&io_conf_stop_start);
00268     if (retval != ESP_OK) {
00269         ESP_LOGE(TAG, "Error al configurar los GPIO para botones aislados stop y start");
00270         return retval;
00271     }
00272
00273 // Asocio pines a la rutina ISR
00274     retval = gpio_isr_handler_add(STOP_BUTTON_PIN, gpio_isr_handler, (void*) (uintptr_t)
00275 STOP_BUTTON_PIN);
00276     if (retval != ESP_OK) {
00277         ESP_LOGE(TAG, "Error al configurar la interrupción de GPIO para botones aislados stop");
00278         return retval;
00279     }
00280
00281     retval = gpio_isr_handler_add(START_BUTTON_PIN, gpio_isr_handler, (void*) (uintptr_t)
00282 START_BUTTON_PIN);
00283     if (retval != ESP_OK) {
00284         ESP_LOGE(TAG, "Error al configurar la interrupción de GPIO para botones aislados start");
00285         return retval;
00286     }
00287
00288     ESP_LOGI(TAG, "Interrupciones GPIO configuradas en %d (Stop) y %d (Start)", STOP_BUTTON_PIN,
00289             START_BUTTON_PIN);
00290     return ESP_OK;
00291 }
00292
00293 static void MCP23017_buzzer_control(void* arg) {
00294     uint32_t delay_time;
00295     uint16_t buzzer_time;
00296
00297     if (buzzer_evt_queue == NULL) {
00298         buzzer_evt_queue = xQueueCreate(10, sizeof(uint32_t));
00299         if (buzzer_evt_queue == NULL) {
00300             ESP_LOGE(TAG, "No se pudo crear la cola de buzzer");
00301             return;
00302         }
00303     }
00304
00305     mcp_write_output_pin(PORTA, 0, 0);

```

```

00303     for(;;) {
00304         if(xQueueReceive(buzzer_evt_queue, &delay_time, portMAX_DELAY)) {
00305             buzzer_time = (uint16_t) delay_time;
00306             mcp_write_output_pin(PORTA, 0, 1);
00307             vTaskDelay(pdMS_TO_TICKS(buzzer_time));
00308             mcp_write_output_pin(PORTA, 0, 0);
00309         }
00310     }
00311 }
00312
00313 static void MCP23017_keyboard_control(void* arg) {
00314     ESP_LOGI(TAG, "Iniciando tarea de control de teclado");
00315
00316     for(;;) {
00317         vTaskDelay(pdMS_TO_TICKS(100));
00318         if ( mcp_portb & 0x10 ) {
00319             mcp_portb = mcp_portb & (~0x10);
00320             mcp_portb = mcp_portb | (0x20);
00321         } else {
00322             mcp_portb = mcp_portb & (~0x20);
00323             mcp_portb = mcp_portb | (0x10);
00324         }
00325         if ( ( mcp_portb & 0x0F ) != 0x0F ) {
00326             // ESP_LOGI(TAG, "Puerto B: %X. Botón pulsado...", mcp_portb);
00327             mcp_read_port(PORTB, &mcp_portb);
00328             continue;
00329         }
00330         mcp_write_output_port(PORTB, mcp_portb);
00331     }
00332 }
00333
00334 static void MCP23017_relay_control(void* arg) {
00335
00336     uint8_t io_num;
00337
00338     if (relay_evt_queue == NULL) {
00339         relay_evt_queue = xQueueCreate(1, sizeof(uint8_t));
00340         if (relay_evt_queue == NULL) {
00341             ESP_LOGE(TAG, "No se pudo crear la cola de relay");
00342             return;
00343         }
00344     }
00345
00346     mcp_write_output_pin(PORTA, 1, 0);
00347     for(;;) {
00348         if(xQueueReceive(relay_evt_queue, &io_num, portMAX_DELAY)) {
00349             ESP_LOGI(TAG, "New relay state: %d", io_num);
00350             if ( io_num == 1 ) {
00351                 mcp_write_output_pin(PORTA, 1, 1);
00352             } else {
00353                 mcp_write_output_pin(PORTA, 1, 0);
00354             }
00355         }
00356     }
00357 }
00358
00359 esp_err_t set_freq_output(uint16_t freq)
00360 {
00361     const uint16_t F_MAX      = 150;    // Hz
00362     const uint16_t DUTY_MAX   = 98;     // %
00363     const uint16_t DUTY_MIN   = 54;     // %
00364     const uint32_t PWM_MAX   = 1023;   // ticks (10 bits)
00365
00366     if (freq > F_MAX) freq = F_MAX;
00367
00368     // duty_pct = 98 - round( (44*freq)/150 )
00369     uint16_t drop = (uint16_t)((44u * freq + 75u) / 150u); // +75 para redondeo
00370     int duty_pct = (int)DUTY_MAX - (int)drop;
00371
00372     if (duty_pct < DUTY_MIN) duty_pct = DUTY_MIN;
00373     if (duty_pct > DUTY_MAX) duty_pct = DUTY_MAX;
00374
00375     // ticks = round( duty_pct/100 * 1023 )
00376     uint32_t ticks = (uint32_t)((duty_pct * PWM_MAX + 50u) / 100u);
00377
00378     if (ledc_get_duty(PWM_MODE, PWM_CHANNEL) != ticks) {
00379         ESP_ERROR_CHECK(ledc_set_duty(PWM_MODE, PWM_CHANNEL, ticks));
00380         ESP_ERROR_CHECK(ledc_update_duty(PWM_MODE, PWM_CHANNEL));
00381         ESP_LOGI("PWM", "Freq=%u Hz, Duty=%d % (ticks=%lu)", freq, duty_pct, (unsigned long)ticks);
00382     }
00383     return ESP_OK;
00384 }
00385
00386 void GPIO_interrupt_attendance_task(void* arg) {
00387     uint32_t io_num;
00388     uint32_t last_interrupt = 0;
00389     uint8_t scratch_data;

```

```

00390     uint8_t speed_selector;
00391
00392     if (MCP23017_INIT() == ESP_OK) {
00393         ESP_LOGI(TAG, "MCP23017 Tarea iniciada correctamente");
00394     } else {
00395         ESP_LOGE(TAG, "Error al inicializar el MCP23017");
00396         ESP_ERROR_CHECK(ESP_FAIL);
00397     }
00398
00399     gpio_init_interrupts();
00400     freq_0_10V_output();
00401     set_freq_output(0);
00402
00403     xTaskCreate(MCP23017_buzzer_control, "MCP23017_buzzer_control", 2048, NULL, 10, NULL);
00404     xTaskCreate(MCP23017_relay_control, "MCP23017_relay_control", 2048, NULL, 10, NULL);
00405     xTaskCreate(MCP23017_keyboard_control, "MCP23017_keyboard_control", 2048, NULL, 10, NULL);
00406
00407     vTaskDelay(pdMS_TO_TICKS(100));
00408
00409     for (;;) {
00410         system_status_t s_e;
00411         get_status(&s_e);
00412         set_freq_output(s_e.frequency);
00413         if (xQueueReceive(GPIO_evt_queue, &io_num, pdMS_TO_TICKS(50))) {
00414             if (io_num == INT_A_PIN) {
00415                 ESP_ERROR_CHECK_WITHOUT_ABORT( mcp_interrupt_flag(PORTA, &scratch_data) );
00416                 ESP_ERROR_CHECK_WITHOUT_ABORT( mcp_read_port(PORTA, &mcp_porta) );
00417
00418                 if (scratch_data & TERMO_SW_MASK) {
00419                     if (mcp_porta & TERMO_SW_MASK) {
00420                         last_interrupt = TERMO_SW_RELEASED;
00421                     } else {
00422                         last_interrupt = TERMO_SW_PRESSED;
00423                     }
00424                 } else if (scratch_data & FREQ_SEL_MASK) {
00425                     mcp_porta = ~mcp_porta;
00426                     speed_selector = mcp_porta & FREQ_SEL_MASK;
00427
00428 // 00ss ss00
00429 // 0000 ssss
00430                     speed_selector = (speed_selector » (FREQ_SEL_1_PIN));
00431
00432                     last_interrupt = SPEED_SELECTOR_0 + speed_selector;
00433                 } else if (scratch_data & STOP_SW_MASK) {
00434                     if (mcp_porta & STOP_SW_MASK) {
00435                         last_interrupt = STOP_RELEASED;
00436                     } else {
00437                         last_interrupt = STOP_PRESSED;
00438                     }
00439
00440                 } else if (io_num == INT_B_PIN) {
00441                     ESP_ERROR_CHECK_WITHOUT_ABORT( mcp_read_port(PORTB, &mcp_portb) );
00442                     switch (mcp_portb) {
00443                         case SWITCH_BUTTON_MENU:
00444                             last_interrupt = BUTTON_MENU;
00445                             break;
00446                         case SWITCH_BUTTON_DOWN:
00447                             last_interrupt = BUTTON_DOWN;
00448                             break;
00449                         case SWITCH_BUTTON_RIGHT:
00450                             last_interrupt = BUTTON_RIGHT;
00451                             break;
00452                         case SWITCH_BUTTON_LEFT:
00453                             last_interrupt = BUTTON_LEFT;
00454                             break;
00455                         case SWITCH_BUTTON_UP:
00456                             last_interrupt = BUTTON_UP;
00457                             break;
00458                         case SWITCH_BUTTON_OK:
00459                             last_interrupt = BUTTON_OK;
00460                             break;
00461                         case SWITCH_BUTTON_SAVE:
00462                             last_interrupt = BUTTON_SAVE;
00463                             break;
00464                         case SWITCH_BUTTON_BACK:
00465                             last_interrupt = BUTTON_BACK;
00466                             break;
00467
00468                     }
00469                 } else if (io_num == STOP_BUTTON_PIN) {
00470                     if (gpio_get_level(io_num)) {
00471                         last_interrupt = EMERGENCI_STOP_PRESSED;
00472                     } else {
00473                         last_interrupt = EMERGENCI_STOP_RELEASED;
00474                     }
00475                 } else if (io_num == START_BUTTON_PIN) {
00476                     if (!gpio_get_level(io_num)) {
00477                         last_interrupt = START_PRESSED;
00478                     } else {
00479                         last_interrupt = START_RELEASED;
00480                     }
00481                 }
00482             }
00483         }
00484     }
00485
00486     vTaskDelete(NULL);
00487
00488     return NULL;
00489 }

```

```
00475         }
00476     }
00477     continue;
00478 }
00479 mcp_read_port (PORTA, &mcp_porta);
00480 mcp_read_port (PORTB, &mcp_portb);
00481
00482 switch (last_interrupt) {
00483     case BUTTON_MENU:
00484         if ( mcp_portb == SWITCH_BUTTON_MENU ) {
00485             DisplayEventPost (last_interrupt);
00486             BuzzerEventPost ( 50 );
00487         }
00488         break;
00489     case BUTTON_DOWN:
00490         if ( mcp_portb == SWITCH_BUTTON_DOWN ) {
00491             DisplayEventPost (last_interrupt);
00492             BuzzerEventPost ( 50 );
00493         }
00494         break;
00495     case BUTTON_RIGHT:
00496         if ( mcp_portb == SWITCH_BUTTON_RIGHT ) {
00497             DisplayEventPost (last_interrupt);
00498             BuzzerEventPost ( 50 );
00499         }
00500         break;
00501     case BUTTON_LEFT:
00502         if ( mcp_portb == SWITCH_BUTTON_LEFT ) {
00503             DisplayEventPost (last_interrupt);
00504             BuzzerEventPost ( 50 );
00505         }
00506         break;
00507     case BUTTON_UP:
00508         if ( mcp_portb == SWITCH_BUTTON_UP ) {
00509             DisplayEventPost (last_interrupt);
00510             BuzzerEventPost ( 50 );
00511         }
00512         break;
00513     case BUTTON_OK:
00514         if ( mcp_portb == SWITCH_BUTTON_OK ) {
00515             DisplayEventPost (last_interrupt);
00516             BuzzerEventPost ( 50 );
00517         }
00518         break;
00519     case BUTTON_SAVE:
00520         if ( mcp_portb == SWITCH_BUTTON_SAVE ) {
00521             DisplayEventPost (last_interrupt);
00522             BuzzerEventPost ( 50 );
00523         }
00524         break;
00525     case BUTTON_BACK:
00526         if ( mcp_portb == SWITCH_BUTTON_BACK ) {
00527             DisplayEventPost (last_interrupt);
00528             BuzzerEventPost ( 50 );
00529         }
00530         break;
00531     case TERMO_SW_PRESSED:
00532         if ( !(mcp_porta & TERMO_SW_MASK) ) {
00533             SystemEventPost (TERMO_SW_PRESSED);
00534         }
00535         break;
00536     case TERMO_SW_RELEASED:
00537         if ( mcp_porta & TERMO_SW_MASK ) {
00538             SystemEventPost (TERMO_SW_RELEASED);
00539         }
00540         break;
00541     case STOP_PRESSED:
00542         if ( !(mcp_porta & STOP_SW_MASK) ) {
00543             SystemEventPost (STOP_PRESSED);
00544         }
00545         break;
00546     case STOP_RELEASED:
00547         if ( mcp_porta & STOP_SW_MASK ) {
00548             SystemEventPost (STOP_RELEASED);
00549         }
00550         break;
00551     case SPEED_SELECTOR_0:
00552     case SPEED_SELECTOR_1:
00553     case SPEED_SELECTOR_2:
00554     case SPEED_SELECTOR_3:
00555     case SPEED_SELECTOR_4:
00556     case SPEED_SELECTOR_5:
00557     case SPEED_SELECTOR_6:
00558     case SPEED_SELECTOR_7:
00559     case SPEED_SELECTOR_8:
00560     case SPEED_SELECTOR_9:
00561         mcp_porta = ~mcp_porta;
```

```

00562     speed_selector = mcp_porta & FREQ_SEL_MASK;
00563 // 00ss ss00
00564 // 0000 ssss
00565     speed_selector = ( speed_selector » ( FREQ_SEL_1_PIN ) );
00566
00567     if ( speed_selector + SPEED_SELECTOR_0 == last_interrupt ) {
00568         SystemEventPost(last_interrupt);
00569     }
00570     break;
00571 case EMERGENCI_STOP_PRESSED:
00572     if ( gpio_get_level(STOP_BUTTON_PIN) ) {
00573         SystemEventPost(EMERGENCI_STOP_PRESSED);
00574     }
00575     break;
00576 case EMERGENCI_STOP_RELEASED:
00577     if ( !gpio_get_level(STOP_BUTTON_PIN) ) {
00578         SystemEventPost(EMERGENCI_STOP_RELEASED);
00579     }
00580     break;
00581 case START_PRESSED:
00582     if ( !gpio_get_level(START_BUTTON_PIN) ) {
00583         SystemEventPost(START_PRESSED);
00584     }
00585     break;
00586 case START_RELEASED:
00587     if ( gpio_get_level(START_BUTTON_PIN) ) {
00588         SystemEventPost(START_RELEASED);
00589     }
00590     break;
00591 }
00592
00593 esp_err_t RelayEventPost( uint8_t relay_event ) {
00594     uint8_t event = relay_event;
00595     if ( event == 1 || event == 0 ) {
00596         return xQueueSend(relay_evt_queue, &event, 0);
00597     }
00598     return ESP_FAIL;
00599 }
00600
00601 esp_err_t BuzzerEventPost( uint32_t buzzer_time_on ) {
00602     if ( buzzer_time_on > 3000 ) {
00603         buzzer_time_on = 3000;
00604     }
00605     return xQueueSend(buzzer_evt_queue, &buzzer_time_on, 0);
00606 }

```

6.19. Referencia del archivo main/io_control/io_control.h

Declaración de funciones de control del relay, buzzer y tarea del uso del expansor de entradas y salidas y entradas y salidas aisladas directas al ESP32.

Funciones

- void **GPIO_interrupt_attendance_task** (void *arg)

Tarea que en función de las interrupciones recibidas permite obtener cual de los pulsadores del teclado, entradas aisladas, termo switch o entradas del MCP2307 fue presionado.
- esp_err_t **set_freq_output** (uint16_t freq)

Convierte la frecuencia en Hz que está girando el motor en el duty necesario para poder operar el PWM de la salida 0-10V.
- esp_err_t **RelayEventPost** (uint8_t relay_event)

Crea un evento para que el relay se activo o desactive de acuerdo a la variable relay_event.
- esp_err_t **BuzzerEventPost** (uint32_t buzzer_time_on)

Envía una señal para hacer sonar el buzzer durante buzzer_time_on milisegundos.

6.19.1. Descripción detallada

Declaración de funciones de control del relay, buzzer y tarea del uso del expansor de entradas y salidas y entradas y salidas aisladas directas al ESP32.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo [io_control.h](#).

6.19.2. Documentación de funciones

6.19.2.1. BuzzerEventPost()

```
esp_err_t BuzzerEventPost (
    uint32_t buzzer_time_on)
```

Envía una señal para hacer sonar el buzzer durante `buzzer_time_on` mili segundos.

Parámetros

in	<code>buzzer_time_on</code>	Tiempo en mili segundos durante el que el buzzer sonará.
----	-----------------------------	--

Devuelve

- `ESP_OK` Evento creado exitosamente
- `ESP_FAIL` Error al crear el evento

Definición en la línea 601 del archivo [io_control.c](#).

6.19.2.2. GPIO_interrupt_attendance_task()

```
void GPIO_interrupt_attendance_task (
    void * arg)
```

Tarea que en función de las interrupciones recibidas permite obtener cual de los pulsadores del teclado, entradas aisladas, termo switch o entradas del MCP2307 fue presionado.

Trabaja con un antirrebote de 200ms. Cuando una entrada cambia su estado, la tarea vuelve a leer las entradas luego de 200ms para asegurarse el valor leído y recién ahí envía la señal correspondiente

Definición en la línea 386 del archivo [io_control.c](#).

6.19.2.3. RelayEventPost()

```
esp_err_t RelayEventPost (
    uint8_t relay_event)
```

Crea un evento para que el relay se active o desactive de acuerdo a la variable `relay_event`.

Con `relay_event` en 1 activa el relay, con 0 desactiva el relay. Cualquier otro valor retorna `ESP_FAIL`.

Parámetros

in	<i>relay_event</i>	Estado del relay
----	--------------------	------------------

Devuelve

- ESP_OK Evento creado exitosamente
- ESP_FAIL Error al crear el evento

Definición en la línea 593 del archivo [io_control.c](#).

6.19.2.4. set_freq_output()

```
esp_err_t set_freq_output (
    uint16_t freq)
```

Convierte la frecuencia en Hz que está girando el motor en el duty necesario para poder operar el PWM de la salida 0-10V.

Permite configurar el duty del PWM desde 0 a 450Hz siendo que, para 0Hz la salida será 0V y para 450HZ serán 10V.

Parámetros

in	<i>freq</i>	Es la frecuencia que está girando el motor.
----	-------------	---

Devuelve

- ESP_OK PWM configurado correctamente
- ESP_ERR_INVALID_ARG Error al configurar el duty del PWM

Definición en la línea 359 del archivo [io_control.c](#).

6.20. io_control.h

[Ir a la documentación de este archivo.](#)

```
00001
00009 #ifndef __IO_CONTROL_H__
00010
00011 #define __IO_CONTROL_H__
00012
00021 void GPIO_interrupt_attendance_task(void* arg);
00022
00037 esp_err_t set_freq_output(uint16_t freq);
00038
00053 esp_err_t RelayEventPost( uint8_t relay_event );
00054
00067 esp_err_t BuzzerEventPost( uint32_t buzzer_time_on );
00068
00069 #endif
```

6.21. Referencia del archivo main/io_control/MCP23017.c

Declaración de funciones de inicialización, lectura y escritura a través del puerto I2C el dispositivo MCP23017.

```
#include "MCP23017.h"
#include "esp_log.h"
#include "esp_err.h"
#include "driver/i2c.h"
#include "freertos/sempwr.h"
```

defines

- #define I2C_MASTER_TIMEOUT_MS 100
- #define I2C_MASTER_SCL_IO 15
- #define I2C_MASTER_SDA_IO 2
- #define I2C_IO_MASTER_NUM I2C_NUM_1
- #define I2C_MASTER_FREQ_HZ 100000
- #define I2C_MASTER_TX_BUF_DISABLE 0
- #define I2C_MASTER_RX_BUF_DISABLE 0

Funciones

- static esp_err_t **i2c_master_init** (void)

Función que inicializa el puerto I2C y el mutex para acceder ordenadamente al hardware.
- static bool **i2c_is_hardware_free** ()

Función que pide recursos de hardware para realizar una operación de lectura o escritura.
- static void **i2c_release_hardware** ()

Función que devuelve recursos de hardware luego de realizar una operación para que pueda realizarse una nueva lectura o escritura.
- static esp_err_t **mcp_register_read** (uint8_t register_address, uint8_t *data)

Función que lee register_address del MCP23017. La información obtenida se almacena en data.
- static esp_err_t **mcp_register_write** (uint8_t register_address, uint8_t data)

Función que escribe en register_address del MCP23017 el dato data.
- esp_err_t **MCP23017_INIT** (void)

Función dedicada a inicializar y configurar el MCP23017. El chip utiliza i2C.
- esp_err_t **mcp_get_on_interrupt_input** (enum **mcp_port_e** port, uint8_t *data)

Función que lee el registro INTCAP del MCP23017.
- esp_err_t **mcp_write_output_pin** (enum **mcp_port_e** port, uint8_t pin, bool state)

Función que lee el registro GPIO, escribe state en el pin del port escribe el registro OLAT del MCP23017.
- esp_err_t **mcp_write_output_port** (enum **mcp_port_e** port, uint8_t data)

Función que escribe data en el escribir el registro OLAT del port del MCP23017.
- esp_err_t **mcp_read_port** (enum **mcp_port_e** port, uint8_t *data)

Función que lee el registro GPIO del port del MCP23017 y lo devuelve en data.
- esp_err_t **mcp_interrupt_flag** (enum **mcp_port_e** port, uint8_t *data)

Función que lee el registro INTF del MCP23017.

Variables

- static const char * **TAG** = "MCP23017"
- SemaphoreHandle_t **xI2C_Mutex** = NULL

6.21.1. Descripción detallada

Declaración de funciones de inicialización, lectura y escritura a través del puerto I2C el dispositivo MCP23017.

Autor

Andrenacci - Carra

Este chip permite usar sus dos puertos de 8 bits por separado o como uno solo de 16 bits. Tiene la función de reportar una interrupción configurable por puerto a través de 2 pines, uno por puerto, que informar si hubo algún cambio en las entradas si así lo deseó; estos pines de interrupción pueden ser configurados también para que actúen en conjunto o separado. Para su funcionamiento de direccionamiento tiene dos modos: "Byte mode" o " \leftarrow Sequencial mode" que se configuran en el registro IOCON que, aunque en el mapa de registros esté duplicado, es compartido entre ambos puerto, por lo que acceder por una dirección u otra es indiferente:

- Byte mode: no incrementa el puntero interno de direcciones en cada comando enviado, sino que en cada escritura/lectura se le debe indicar a qué dirección de memoria se desea acceder si no se desea acceder a la última dirección de memoria accedida. Esto permite hacer polling en el mismo registro, con la intención de estar monitoreando continuamente la/las entradas de interés. Utilizando el modo ICON.BANK = 0, habilita una función especial que permite tooglear entre los registros del puerto A y B secuencialmente en cada acceso al chip.
- Sequenctial mode: El puntero de dirección interno del chip se incrementa a cada byte de datos enviado. Cuando llega al último registro del mapa de registro hace un rollover al registro 0x00.

La secuencia de escritura se define de la siguiente manera: S 0100AAA0 ADDR DIN P

Versión

2.0

Fecha

2025-11-06

Definición en el archivo [MCP23017.c](#).

6.21.2. Documentación de «define»

6.21.2.1. I2C_IO_MASTER_NUM

```
#define I2C_IO_MASTER_NUM I2C_NUM_1
```

Número de puerto I2C de la comunicación con el MCP23017

Definición en la línea 106 del archivo [MCP23017.c](#).

6.21.2.2. I2C_MASTER_FREQ_HZ

```
#define I2C_MASTER_FREQ_HZ 100000
```

Frecuencia estándar de operación del I2C para comunicación con el MCP23017 en 100000MHz

Definición en la línea 107 del archivo [MCP23017.c](#).

6.21.2.3. I2C_MASTER_RX_BUF_DISABLE

```
#define I2C_MASTER_RX_BUF_DISABLE 0
```

Largo de buffer I2C en 0 porque no es utilizado al ser un dispositivo maestro el ESP32

Definición en la línea 109 del archivo [MCP23017.c](#).

6.21.2.4. I2C_MASTER_SCL_IO

```
#define I2C_MASTER_SCL_IO 15
```

GPIO para SCL de la comunicación con el MCP23017

Definición en la línea 104 del archivo [MCP23017.c](#).

6.21.2.5. I2C_MASTER_SDA_IO

```
#define I2C_MASTER_SDA_IO 2
```

GPIO para SDA de la comunicación con el MCP23017

Definición en la línea 105 del archivo [MCP23017.c](#).

6.21.2.6. I2C_MASTER_TIMEOUT_MS

```
#define I2C_MASTER_TIMEOUT_MS 100
```

Tiempo durante el cual se espera alguna respuesta del esclavo antes de cortar la comunicación

Definición en la línea 103 del archivo [MCP23017.c](#).

6.21.2.7. I2C_MASTER_TX_BUF_DISABLE

```
#define I2C_MASTER_TX_BUF_DISABLE 0
```

Largo de buffer I2C en 0 porque no es utilizado al ser un dispositivo maestro el ESP32

Definición en la línea 108 del archivo [MCP23017.c](#).

6.21.3. Documentación de funciones

6.21.3.1. i2c_is_hardware_free()

```
bool i2c_is_hardware_free () [static]
```

Función que pide recursos de hardware para realizar una operación de lectura o escritura.

En caso que los recursos de hardware estuviesen siendo usados, duerme la app durante 100ms

Atención

Función bloqueante

Definición en la línea 134 del archivo [MCP23017.c](#).

6.21.3.2. i2c_master_init()

```
esp_err_t i2c_master_init (
    void ) [static]
```

Función que inicializa el puerto I2C y el mutex para acceder ordenadamente al hardware.

Parámetros

in	<i>register_address</i>	Es la dirección de memoria del registro que se quiere acceder en el MCP23017.
in	<i>data</i>	Información que quiere escribirse en <i>register_address</i>

Devuelve

- ESP_OK en caso de éxito.
- ESP_ERR_INVALID_ARG Error de parámetro
- ESP_FAIL Error al enviar comando, el esclavo no envió ACK a la transferencia.
- ESP_ERR_INVALID_STATE I2C El driver no fue inicializado o no está en modo master.
- ESP_ERR_TIMEOUT Timout de la operación porque el bus está ocupado.

Definición en la línea 111 del archivo [MCP23017.c](#).

6.21.3.3. i2c_release_hardware()

```
void i2c_release_hardware () [static]
```

Función que devuelve recursos de hardware luego de realizar una operación para que pueda realizarse una nueva lectura o escritura.

Definición en la línea 142 del archivo [MCP23017.c](#).

6.21.3.4. MCP23017_INIT()

```
esp_err_t MCP23017_INIT (
    void )
```

Función dedicada a inicializar y configurar el MCP23017. El chip utiliza i2C.

Inicializa el MCP32017. Configura las entradas IO2, IO3, IO4, IO5 y IO6 del puerto A (Todas ellas sin pull up activo) y las entradas IO0, IO1, IO2 y IO3 del puerto B (Todas ellas con pull up activo); configura como salidas IO0, IO1 y IO7 del puerto A y IO4, IO5, IO6 y IO7 del puerto B. Todas las entradas generan una interrupción que se informa por los pines INTA e INTB del chip por separado (Las interrupciones del puerto A genera un cambio en el pin INTA; misma explicación para INTB). Las interrupciones se generarán siempre cuando haya un cambio en la entrada (Ya sea un flanco positivo o negativo). Los pines INTA e INTB son salidas activas que se pondrán en alto en caso de tener una interrupción.

Devuelve

- ESP_OK en caso de éxito.
- ESP_ERR_INVALID_ARG Error de parámetro
- ESP_FAIL Error al enviar comando, el esclavo no envió ACK a la transferencia.
- ESP_ERR_INVALID_STATE I2C El driver no fue inicializado o no está en modo master.
- ESP_ERR_TIMEOUT Timout de la operación porque el bus está ocupado.

Definición en la línea 174 del archivo [MCP23017.c](#).

6.21.3.5. mcp_get_on_interrupt_input()

```
esp_err_t mcp_get_on_interrupt_input (
    enum mcp_port_e port,
    uint8_t * data)
```

Función que lee el registro INTCAP del MCP23017.

El registro INTCAP retiene el valor de las entradas al generarse una interrupción.

Parámetros

in	<i>port</i>	Puede tomar los valores PORTA o PORTB, dependiendo del puerto que se quiera acceder.
out	<i>data</i>	Puntero donde se almacenará la lectura del registro INTCAP.

Devuelve

- ESP_OK en caso de éxito.
- ESP_ERR_INVALID_ARG Error de parámetro
- ESP_FAIL Error al enviar comando, el esclavo no envió ACK a la transferencia.
- ESP_ERR_INVALID_STATE I2C El driver no fue inicializado o no está en modo master.
- ESP_ERR_TIMEOUT Timout de la operación porque el bus está ocupado.

Definición en la línea 312 del archivo [MCP23017.c](#).

6.21.3.6. `mcp_interrupt_flag()`

```
esp_err_t mcp_interrupt_flag (
    enum mcp_port_e port,
    uint8_t * data)
```

Función que lee el registro INTF del MCP23017.

El registro INTF retiene el valor de la entrada que generó la interrupción. Luego de haber leído este registro, el pin INTx vuelve a su estado de reposo.

Parámetros

in	<i>port</i>	Puede tomar los valores PORTA o PORTB, dependiendo del puerto que se quiera acceder.
out	<i>data</i>	Puntero donde se almacenará la lectura del registro INTF.

Devuelve

- ESP_OK en caso de éxito.
- ESP_ERR_INVALID_ARG Error de parámetro
- ESP_FAIL Error al enviar comando, el esclavo no envió ACK a la transferencia.
- ESP_ERR_INVALID_STATE I2C El driver no fue inicializado o no está en modo master.
- ESP_ERR_TIMEOUT Timout de la operación porque el bus está ocupado.

Definición en la línea 387 del archivo [MCP23017.c](#).

6.21.3.7. `mcp_read_port()`

```
esp_err_t mcp_read_port (
    enum mcp_port_e port,
    uint8_t * data)
```

Función que lee el registro GPIO del *port* del MCP23017 y lo devuelve en *data*.

El registro GPIO es el que representa la exteriorización de los estados de los etados lógicos altos o bajos de los pines.

Parámetros

in	<i>port</i>	Puede tomar los valores PORTA o PORTB, dependiendo del puerto que se quiera escribir.
out	<i>data</i>	Puntero donde se almacena el estado del puerto del MCP23017.

Devuelve

- ESP_OK en caso de éxito.
- ESP_ERR_INVALID_ARG Error de parámetro
- ESP_FAIL Error al enviar comando, el esclavo no envió ACK a la transferencia.
- ESP_ERR_INVALID_STATE I2C El driver no fue inicializado o no está en modo master.
- ESP_ERR_TIMEOUT Timout de la operación porque el bus está ocupado.

Definición en la línea 370 del archivo [MCP23017.c](#).

6.21.3.8. mcp_register_read()

```
esp_err_t mcp_register_read (
    uint8_t register_address,
    uint8_t * data) [static]
```

Función que lee `register_address` del MCP23017. La información obtenida se almacena en `data`.

Parámetros

in	<i>register_address</i>	Es la dirección de memoria del registro que se quiere acceder en el MCP23017.
out	<i>data</i>	Información que leída de <code>register_address</code>

Devuelve

- ESP_OK en caso de éxito.
- ESP_ERR_INVALID_ARG Error de parámetro
- ESP_FAIL Error al enviar comando, el esclavo no envió ACK a la transferencia.
- ESP_ERR_INVALID_STATE I2C El driver no fue inicializado o no está en modo master.
- ESP_ERR_TIMEOUT Timout de la operación porque el bus está ocupado.

Definición en la línea 146 del archivo [MCP23017.c](#).

6.21.3.9. mcp_register_write()

```
esp_err_t mcp_register_write (
    uint8_t register_address,
    uint8_t data) [static]
```

Función que escribe en `register_address` del MCP23017 el dato `data`.

Parámetros

in	<i>register_address</i>	Es la dirección de memoria del registro que se quiere acceder en el MCP23017.
in	<i>data</i>	Información que quiere escribirse en <i>register_address</i>

Devuelve

- ESP_OK en caso de éxito.
- ESP_ERR_INVALID_ARG Error de parámetro
- ESP_FAIL Error al enviar comando, el esclavo no envió ACK a la transferencia.
- ESP_ERR_INVALID_STATE I2C El driver no fue inicializado o no está en modo master.
- ESP_ERR_TIMEOUT Timout de la operación porque el bus está ocupado.

Definición en la línea 161 del archivo [MCP23017.c](#).

6.21.3.10. `mcp_write_output_pin()`

```
esp_err_t mcp_write_output_pin (
    enum mcp_port_e port,
    uint8_t pin,
    bool state)
```

Función que lee el registro GPIO, escribe `state` en el `pin` del `port` escribe el registro OLAT del MCP23017.

El registro OLAT es el que exterioriza el estado del pin en estados lógicos altos o bajos.

Parámetros

in	<i>port</i>	Puede tomar los valores PORTA o PORTB, dependiendo del puerto que se quiera escribir.
in	<i>pin</i>	Pin físico que quiere ser modificado su estado
in	<i>state</i>	Estado True o False que puede tomar el pin que quiere ser modificado

Devuelve

- ESP_OK en caso de éxito.
- ESP_ERR_INVALID_ARG Error de parámetro
- ESP_FAIL Error al enviar comando, el esclavo no envió ACK a la transferencia.
- ESP_ERR_INVALID_STATE I2C El driver no fue inicializado o no está en modo master.
- ESP_ERR_TIMEOUT Timout de la operación porque el bus está ocupado.

Definición en la línea 328 del archivo [MCP23017.c](#).

6.21.3.11. `mcp_write_output_port()`

```
esp_err_t mcp_write_output_port (
    enum mcp_port_e port,
    uint8_t data)
```

Función que escribe `data` en el escribe el registro OLAT del `port` del MCP23017.

Escribe `data` entero, no pin a pin.

Parámetros

in	<i>port</i>	Puede tomar los valores PORTA o PORTB, dependiendo del puerto que se quiera escribir.
in	<i>data</i>	Estado True o False que puede tomar los pines del <i>port</i>

Devuelve

- ESP_OK en caso de éxito.
- ESP_ERR_INVALID_ARG Error de parámetro
- ESP_FAIL Error al enviar comando, el esclavo no envió ACK a la transferencia.
- ESP_ERR_INVALID_STATE I2C El driver no fue inicializado o no está en modo master.
- ESP_ERR_TIMEOUT Timout de la operación porque el bus está ocupado.

Definición en la línea 353 del archivo [MCP23017.c](#).

6.21.4. Documentación de variables

6.21.4.1. TAG

```
const char* TAG = "MCP23017" [static]
```

Variable que se imprime en los ESP_LOG

Definición en la línea 100 del archivo [MCP23017.c](#).

6.21.4.2. xI2C_Mutex

```
SemaphoreHandle_t xI2C_Mutex = NULL
```

Semáforo (Mutex) que es utilizado para que dos comandos sean ejecutados al mismo tiempo y evitar que uno superponga al otro, trayendo datos incorrectos del MCP23017

Definición en la línea 101 del archivo [MCP23017.c](#).

6.22. MCP23017.c

[Ir a la documentación de este archivo.](#)

```
00001
00015
00016 #include "MCP23017.h"
00017 #include "esp_log.h"
00018 #include "esp_err.h"
00019 #include "driver/i2c.h"
00020 #include "freertos/semphr.h"
00021
00040 static esp_err_t i2c_master_init(void);
00041
00051 static bool i2c_is_hardware_free();
00052
00058 static void i2c_release_hardware();
00059
00078 static esp_err_t mcp_register_read(uint8_t register_address, uint8_t *data);
00079
00098 static esp_err_t mcp_register_write(uint8_t register_address, uint8_t data);
00099
```

```

00100 static const char *TAG = "MCP23017";
00101 SemaphoreHandle_t xI2C_Mutex = NULL;
00102
00103 #define I2C_MASTER_TIMEOUT_MS          100
00104 #define I2C_MASTER_SCL_IO            15
00105 #define I2C_MASTER_SDA_IO            2
00106 #define I2C_IO_MASTER_NUM           I2C_NUM_1
00107 #define I2C_MASTER_FREQ_HZ          100000
00108 #define I2C_MASTER_TX_BUF_DISABLE   0
00109 #define I2C_MASTER_RX_BUF_DISABLE   0
00110
00111 static esp_err_t i2c_master_init(void) {
00112     i2c_config_t conf = {
00113         .mode = I2C_MODE_MASTER,
00114         .sda_io_num = I2C_MASTER_SDA_IO,
00115         .scl_io_num = I2C_MASTER_SCL_IO,
00116         .sda_pullup_en = GPIO_PULLUP_ENABLE,
00117         .scl_pullup_en = GPIO_PULLUP_ENABLE,
00118         .master.clk_speed = I2C_MASTER_FREQ_HZ,
00119     };
00120     ESP_ERROR_CHECK(i2c_param_config(I2C_IO_MASTER_NUM, &conf));
00121     ESP_ERROR_CHECK(i2c_driver_install(I2C_IO_MASTER_NUM, conf.mode,
00122                                         I2C_MASTER_RX_BUF_DISABLE,
00123                                         I2C_MASTER_TX_BUF_DISABLE, 0));
00124     if (xI2C_Mutex == NULL) {
00125         xI2C_Mutex = xSemaphoreCreateMutex();
00126         if (xI2C_Mutex == NULL) {
00127             ESP_LOGE(TAG, "No se pudo crear el mutex de I2C");
00128         }
00129     }
00130
00131     return ESP_OK;
00132 }
00133
00134 static bool i2c_is_hardware_free() {
00135     while (xSemaphoreTake(xI2C_Mutex, pdMS_TO_TICKS(100)) != pdTRUE) {
00136         ESP_LOGI(TAG, "Espero recursos de hardware I2C...");
00137         return false;
00138     }
00139     return true;
00140 }
00141
00142 static void i2c_release_hardware() {
00143     xSemaphoreGive(xI2C_Mutex);
00144 }
00145
00146 static esp_err_t mcp_register_read(uint8_t register_address, uint8_t *data) {
00147     esp_err_t ret = ESP_OK;
00148     i2c_cmd_handle_t cmd = i2c_cmd_link_create();
00149     i2c_master_start(cmd);
00150     i2c_master_write_byte(cmd, __MCP23017_WRITE_OPCODE__, 0);
00151     i2c_master_write_byte(cmd, register_address, true);
00152     i2c_master_start(cmd);
00153     i2c_master_write_byte(cmd, __MCP23017_READ_OPCODE__, 0);
00154     i2c_master_read_byte(cmd, data, I2C_MASTER_NACK);
00155     i2c_master_stop(cmd);
00156     ret = i2c_master_cmd_begin(I2C_IO_MASTER_NUM, cmd, pdMS_TO_TICKS(I2C_MASTER_TIMEOUT_MS));
00157     i2c_cmd_link_delete(cmd);
00158     return ret;
00159 }
00160
00161 static esp_err_t mcp_register_write(uint8_t register_address, uint8_t data) {
00162     esp_err_t ret = ESP_OK;
00163     i2c_cmd_handle_t cmd = i2c_cmd_link_create();
00164     i2c_master_start(cmd);
00165     i2c_master_write_byte(cmd, __MCP23017_WRITE_OPCODE__, 0);
00166     i2c_master_write_byte(cmd, register_address, true);
00167     i2c_master_write_byte(cmd, data, true);
00168     i2c_master_stop(cmd);
00169     ret = i2c_master_cmd_begin(I2C_IO_MASTER_NUM, cmd, pdMS_TO_TICKS(I2C_MASTER_TIMEOUT_MS));
00170     i2c_cmd_link_delete(cmd);
00171     return ret;
00172 }
00173
00174 esp_err_t MCP23017_INIT( void ) {
00175     esp_err_t ret = ESP_OK;
00176     ESP_LOGI(TAG, "Iniciando modulo");
00177     if ( i2c_master_init() == ESP_OK ) {
00178         ESP_LOGI(TAG, "I2C Inicializado correctamente");
00179     } else {
00180         ESP_LOGE(TAG, "Error al inicializar I2C");
00181         return ESP_FAIL;
00182     }
00183     MCP23017_IOCON_t iocon;
00184     MCP23017_GPPU_t gppua, gppub;
00185     MCP23017_IODIR_t iodira, iodirb;
00186     MCP23017_GPINTEN_t gpintena, gpintenb;

```

```

00187     MCP23017_DEFVAL_t defvala, defvalb;
00188     MCP23017_INTCON_t intccona, intconb;
00189
00190     iocon.all = 0x00;
00191     iocon.bits.INTPOL = __MCP23017_INTPOL_POLARITY_HIGH__; // Los pines de interrupción se
00192     ponen en 1 cuando hay una interrupción
00193     iocon.bits.ODR = __MCP23017_ODR_ACTIVE_DRIVER_OUTPUT__; // Los pines de interrupción
00194     configuran como salidas activas
00195     iocon.bits.DISSLW = __MCP23017_DISSLW_ENABLED__; // Activo el Slew Rate en el SDA
00196     para una mejor comunicación
00197     iocon.bits.SEQOP = __MCP23017_SEQOP_DISABLED__; // No activo el incremento del
00198     puntero de direcciones para poder leer constantemente el mismo registro si quiero
00199     iocon.bits.MIRROR = __MCP23017_MIRROR_UNCONNECTED__; // Los pines de interrupción A y B
00200     se controlan por separado
00201     iocon.bits.BANK = __MCP23017_FUNC_MODE__; // Uso el modo Byte para tratar
00202     los puertos como si giesen de 8 bits y no de 16
00203
00204     gppua.all = __MCP23017_GPPU_PULL_UP_DISABLE__;
00205
00206     gppub.all = __MCP23017_GPPU_PULL_UP_DISABLE__;
00207     gppub.bits.PU0 = __MCP23017_GPPU_PULL_UP_ENABLE__;
00208     gppub.bits.PU1 = __MCP23017_GPPU_PULL_UP_ENABLE__;
00209     gppub.bits.PU2 = __MCP23017_GPPU_PULL_UP_ENABLE__;
00210     gppub.bits.PU3 = __MCP23017_GPPU_PULL_UP_ENABLE__;
00211
00212     iodira.all = 0x00;
00213     iodira.bits.IO0 = __MCP23017_IODIR_OUTPUT__;
00214     iodira.bits.IO1 = __MCP23017_IODIR_OUTPUT__;
00215     iodira.bits.IO2 = __MCP23017_IODIR_INPUT__;
00216     iodira.bits.IO3 = __MCP23017_IODIR_INPUT__;
00217     iodira.bits.IO4 = __MCP23017_IODIR_INPUT__;
00218     iodira.bits.IO5 = __MCP23017_IODIR_INPUT__;
00219     iodira.bits.IO6 = __MCP23017_IODIR_INPUT__;
00220     iodira.bits.IO7 = __MCP23017_IODIR_OUTPUT__;
00221
00222     iodirb.all = 0x00;
00223     iodirb.bits.IO0 = __MCP23017_IODIR_INPUT__;
00224     iodirb.bits.IO1 = __MCP23017_IODIR_INPUT__;
00225     iodirb.bits.IO2 = __MCP23017_IODIR_INPUT__;
00226     iodirb.bits.IO3 = __MCP23017_IODIR_INPUT__;
00227     iodirb.bits.IO4 = __MCP23017_IODIR_OUTPUT__;
00228     iodirb.bits.IO5 = __MCP23017_IODIR_OUTPUT__;
00229     iodirb.bits.IO6 = __MCP23017_IODIR_OUTPUT__;
00230     iodirb.bits.IO7 = __MCP23017_IODIR_OUTPUT__;
00231
00232     gpintena.all = 0x00;
00233     gpintena.bits.GPINT2 = __MCP23017_GPINTEN_ENABLE__;
00234     gpintena.bits.GPINT3 = __MCP23017_GPINTEN_ENABLE__;
00235     gpintena.bits.GPINT4 = __MCP23017_GPINTEN_ENABLE__;
00236     gpintena.bits.GPINT5 = __MCP23017_GPINTEN_ENABLE__;
00237     gpintena.bits.GPINT6 = __MCP23017_GPINTEN_ENABLE__;
00238
00239     defvala.all = 0x00;
00240     defvala.bits.DEF2 = 1;
00241     defvala.bits.DEF3 = 1;
00242     defvala.bits.DEF4 = 1;
00243     defvala.bits.DEF5 = 1;
00244     defvala.bits.DEF6 = 0;
00245
00246     defvalb.all = 0x0F;
00247
00248     intccona.all = __MCP23017_INTCON_CHANGE__;
00249
00250     intconb.all = __MCP23017_INTCON_CHANGE__;
00251
00252     ret = mcp_register_write( (uint8_t) MCP23017_IOCON_REGISTER , iocon.all ); //
00253     if ( ret != ESP_OK ) {
00254         ESP_LOGE(TAG, "Error al escribir en el registro IOCON");
00255         return ret;
00256     }
00257     ret = mcp_register_write( (uint8_t) MCP23017_IODIRA_REGISTER , iodira.all ); //
00258     if ( ret != ESP_OK ) {
00259         ESP_LOGE(TAG, "Error al escribir en el registro IODIRA");
00260         return ret;
00261     }
00262     ret = mcp_register_write( (uint8_t) MCP23017_IODIRB_REGISTER , iodirb.all ); //
00263     if ( ret != ESP_OK ) {
00264         ESP_LOGE(TAG, "Error al escribir en el registro IODIRB");
00265         return ret;
00266     }
00267 // ret = mcp_register_write( (uint8_t) MCP23017_IPOLB_REGISTER , iodira.all ); // Mantienen

```

```

    reset_val
00268     ret = mcp_register_write( uint8_t MCP23017_GPINTENA_REGISTER , gpintena.all ); // 
00269     if ( ret != ESP_OK ) {
00270         ESP_LOGE(TAG, "Error al escribir en el registro GPINTENA");
00271         return ret;
00272     }
00273     ret = mcp_register_write( uint8_t MCP23017_GPINTENB_REGISTER , gpintenb.all ); // 
00274     if ( ret != ESP_OK ) {
00275         ESP_LOGE(TAG, "Error al escribir en el registro GPINTENB");
00276         return ret;
00277     }
00278     ret = mcp_register_write( uint8_t MCP23017_DEFVALA_REGISTER , defvala.all ); // 
00279     if ( ret != ESP_OK ) {
00280         ESP_LOGE(TAG, "Error al escribir en el registro DEFVALA");
00281         return ret;
00282     }
00283     ret = mcp_register_write( uint8_t MCP23017_DEFVALB_REGISTER , defvalb.all ); // 
00284     if ( ret != ESP_OK ) {
00285         ESP_LOGE(TAG, "Error al escribir en el registro DEFVALB");
00286         return ret;
00287     }
00288     ret = mcp_register_write( uint8_t MCP23017_INTCONA_REGISTER , intcona.all ); // 
00289     if ( ret != ESP_OK ) {
00290         ESP_LOGE(TAG, "Error al escribir en el registro DEFVALB");
00291         return ret;
00292     }
00293     ret = mcp_register_write( uint8_t MCP23017_INTCONB_REGISTER , intconb.all ); // 
00294     if ( ret != ESP_OK ) {
00295         ESP_LOGE(TAG, "Error al escribir en el registro DEFVALB");
00296         return ret;
00297     }
00298     ret = mcp_register_write( uint8_t MCP23017_GPPUA_REGISTER , gppua.all ); // 
00299     if ( ret != ESP_OK ) {
00300         ESP_LOGE(TAG, "Error al escribir en el registro DEFVALB");
00301         return ret;
00302     }
00303     ret = mcp_register_write( uint8_t MCP23017_GPPUB_REGISTER , gppub.all ); // 
00304     if ( ret != ESP_OK ) {
00305         ESP_LOGE(TAG, "Error al escribir en el registro DEFVALB");
00306         return ret;
00307     }
00308
00309     return ret;
00310 }
00311
00312 esp_err_t mcp_get_on_interrupt_input(enum mcp_port_e port, uint8_t *data) {
00313     while ( i2c_is_hardware_free() != pdTRUE ) {
00314         ESP_LOGI(TAG, "Espero recursos de hardware I2C...");
00315         vTaskDelay(pdMS_TO_TICKS(10));
00316     }
00317
00318     esp_err_t esp_err;
00319     if ( port == PORTA || port == PORTB ) {
00320         esp_err = mcp_register_read(MCP23017_INTCAPA_REGISTER + port, data);
00321     } else {
00322         esp_err = ESP_ERR_INVALID_ARG;
00323     }
00324     i2c_release_hardware();
00325     return esp_err;
00326 }
00327
00328 esp_err_t mcp_write_output_pin(enum mcp_port_e port, uint8_t pin, bool state) {
00329     esp_err_t esp_err;
00330     uint8_t data;
00331
00332     while ( i2c_is_hardware_free() != pdTRUE ) {
00333         ESP_LOGI(TAG, "Espero recursos de hardware I2C...");
00334         vTaskDelay(pdMS_TO_TICKS(10));
00335     }
00336
00337     if ( port == PORTA || port == PORTB ) {
00338         if ( !(esp_err = mcp_register_read(MCP23017_GPIOA_REGISTER + port, &data) ) ) {
00339             if (state) {
00340                 data |= (1 << pin);
00341             } else {
00342                 data &= ~ (1 << pin);
00343             }
00344             esp_err = mcp_register_write(MCP23017_OLATA_REGISTER + port, data);
00345         }
00346     } else {
00347         esp_err = ESP_ERR_INVALID_ARG;
00348     }
00349     i2c_release_hardware();
00350     return esp_err;
00351 }
00352
00353 esp_err_t mcp_write_output_port(enum mcp_port_e port, uint8_t data) {

```

```

00354     esp_err_t esp_err;
00355
00356     while ( i2c_is_hardware_free() != pdTRUE ) {
00357         ESP_LOGI(TAG, "Espero recursos de hardware I2C...");
00358         vTaskDelay(pdMS_TO_TICKS(10));
00359     }
00360
00361     if ( port == PORTA || port == PORTB ) {
00362         esp_err = mcp_register_write(MCP23017_OLATA_REGISTER + port, data);
00363     } else {
00364         esp_err = ESP_ERR_INVALID_ARG;
00365     }
00366     i2c_release_hardware();
00367     return esp_err;
00368 }
00369
00370 esp_err_t mcp_read_port(enum mcp_port_e port, uint8_t *data) {
00371     esp_err_t esp_err;
00372
00373     while ( i2c_is_hardware_free() != pdTRUE ) {
00374         ESP_LOGI(TAG, "Espero recursos de hardware I2C...");
00375         vTaskDelay(pdMS_TO_TICKS(10));
00376     }
00377
00378     if ( port == PORTA || port == PORTB ) {
00379         esp_err = mcp_register_read(MCP23017_GPIOA_REGISTER + port, data);
00380     } else {
00381         esp_err = ESP_ERR_INVALID_ARG;
00382     }
00383     i2c_release_hardware();
00384     return esp_err;
00385 }
00386
00387 esp_err_t mcp_interrupt_flag(enum mcp_port_e port, uint8_t *data) {
00388     esp_err_t esp_err;
00389
00390     while ( i2c_is_hardware_free() != pdTRUE ) {
00391         ESP_LOGI(TAG, "Espero recursos de hardware I2C...");
00392         vTaskDelay(pdMS_TO_TICKS(10));
00393     }
00394
00395     if ( port == PORTA || port == PORTB ) {
00396         esp_err = mcp_register_read(MCP23017_INTFA_REGISTER + port, data);
00397     } else {
00398         esp_err = ESP_ERR_INVALID_ARG;
00399     }
00400     i2c_release_hardware();
00401     return esp_err;
00402 }

```

6.23. Referencia del archivo main/io_control/MCP23017.h

Declaración de funciones de inicialización, lectura y escritura a través del puerto I2C el dispositivo MCP23017.

```
#include "esp_mac.h"
#include "mcp23017_defs.h"
#include <inttypes.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
```

Funciones

- **esp_err_t MCP23017_INIT (void)**
Función dedicada a inicializar y configurar el MCP23017. El chip utiliza I2C.
- **esp_err_t mcp_get_on_interrupt_input (enum mcp_port_e port, uint8_t *data)**
Función que lee el registro INTCAP del MCP23017.
- **esp_err_t mcp_interrupt_flag (enum mcp_port_e port, uint8_t *data)**
Función que lee el registro INTF del MCP23017.

- `esp_err_t mcp_write_output_pin` (enum `mcp_port_e` port, uint8_t pin, bool state)
Función que lee el registro GPIO, escribe state en el pin del port escribe el registro OLAT del MCP23017.
- `esp_err_t mcp_read_port` (enum `mcp_port_e` port, uint8_t *data)
Función que lee el registro GPIO del port del MCP23017 y lo devuelve en data.
- `esp_err_t mcp_write_output_port` (enum `mcp_port_e` port, uint8_t data)
Función que escribe data en el escribe el registro OLAT del port del MCP23017.

6.23.1. Descripción detallada

Declaración de funciones de inicialización, lectura y escritura a través del puerto I2C el dispositivo MCP23017.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo [MCP23017.h](#).

6.23.2. Documentación de funciones

6.23.2.1. MCP23017_INIT()

```
esp_err_t MCP23017_INIT (
    void )
```

Función dedicada a inicializar y configurar el MCP23017. El chip utiliza i2C.

Inicializa el MCP32017. Configura las entradas IO2, IO3, IO4, IO5 y IO6 del puerto A (Todas ellas sin pull up activo) y las entradas IO0, IO1, IO2 y IO3 del puerto B (Todas ellas con pull up activo); configura como salidas IO0, IO1 y IO7 del puerto A y IO4, IO5, IO6 y IO7 del puerto B. Todas las entradas generan una interrupción que se informa por los pines INTA e INTB del chip por separado (Las interrupciones del puerto A genera un cambio en el pin INTA; misma explicación para INTB). Las interrupciones se generarán siempre cuando haya un cambio en la entrada (Ya sea un flanco positivo o negativo). Los pines INTA e INTB son salidas activas que se pondrán en alto en caso de tener una interrupción.

Devuelve

- ESP_OK en caso de éxito.
- ESP_ERR_INVALID_ARG Error de parámetro
- ESP_FAIL Error al enviar comando, el esclavo no envió ACK a la transferencia.
- ESP_ERR_INVALID_STATE I2C El driver no fue inicializado o no está en modo master.
- ESP_ERR_TIMEOUT Timout de la operación porque el bus está ocupado.

Definición en la línea 174 del archivo [MCP23017.c](#).

6.23.2.2. mcp_get_on_interrupt_input()

```
esp_err_t mcp_get_on_interrupt_input (
    enum mcp_port_e port,
    uint8_t * data)
```

Función que lee el registro INTCAP del MCP23017.

El registro INTCAP retiene el valor de las entradas al generarse una interrupción.

Parámetros

in	<i>port</i>	Puede tomar los valores PORTA o PORTB, dependiendo del puerto que se quiera acceder.
out	<i>data</i>	Puntero donde se almacenará la lectura del registro INTCAP.

Devuelve

- ESP_OK en caso de éxito.
- ESP_ERR_INVALID_ARG Error de parámetro
- ESP_FAIL Error al enviar comando, el esclavo no envió ACK a la transferencia.
- ESP_ERR_INVALID_STATE I2C El driver no fue inicializado o no está en modo master.
- ESP_ERR_TIMEOUT Timout de la operación porque el bus está ocupado.

Definición en la línea 312 del archivo [MCP23017.c](#).

6.23.2.3. `mcp_interrupt_flag()`

```
esp_err_t mcp_interrupt_flag (
    enum mcp_port_e port,
    uint8_t * data)
```

Función que lee el registro INTF del MCP23017.

El registro INTF retiene el valor de la entrada que generó la interrupción. Luego de haber leído este registro, el pin INTx vuelve a su estado de reposo.

Parámetros

in	<i>port</i>	Puede tomar los valores PORTA o PORTB, dependiendo del puerto que se quiera acceder.
out	<i>data</i>	Puntero donde se almacenará la lectura del registro INTF.

Devuelve

- ESP_OK en caso de éxito.
- ESP_ERR_INVALID_ARG Error de parámetro
- ESP_FAIL Error al enviar comando, el esclavo no envió ACK a la transferencia.
- ESP_ERR_INVALID_STATE I2C El driver no fue inicializado o no está en modo master.
- ESP_ERR_TIMEOUT Timout de la operación porque el bus está ocupado.

Definición en la línea 387 del archivo [MCP23017.c](#).

6.23.2.4. `mcp_read_port()`

```
esp_err_t mcp_read_port (
    enum mcp_port_e port,
    uint8_t * data)
```

Función que lee el registro GPIO del *port* del MCP23017 y lo devuelve en *data*.

El registro GPIO es el que representa la exteriorización de los estados de los etados lógicos altos o bajos de los pines.

Parámetros

in	<i>port</i>	Puede tomar los valores PORTA o PORTB, dependiendo del puerto que se quiera escribir.
out	<i>data</i>	Puntero donde se almacena el estado del puerto del MCP23017.

Devuelve

- ESP_OK en caso de éxito.
- ESP_ERR_INVALID_ARG Error de parámetro
- ESP_FAIL Error al enviar comando, el esclavo no envió ACK a la transferencia.
- ESP_ERR_INVALID_STATE I2C El driver no fue inicializado o no está en modo master.
- ESP_ERR_TIMEOUT Timout de la operación porque el bus está ocupado.

Definición en la línea 370 del archivo [MCP23017.c](#).

6.23.2.5. `mcp_write_output_pin()`

```
esp_err_t mcp_write_output_pin (
    enum mcp_port_e port,
    uint8_t pin,
    bool state)
```

Función que lee el registro GPIO, escribe `state` en el `pin` del `port` escribe el registro OLAT del MCP23017.

El registro OLAT es el que exterioriza el estado del pin en estados lógicos altos o bajos.

Parámetros

in	<i>port</i>	Puede tomar los valores PORTA o PORTB, dependiendo del puerto que se quiera escribir.
in	<i>pin</i>	Pin físico que quiere ser modificado su estado
in	<i>state</i>	Estado True o False que puede tomar el pin que quiere ser modificado

Devuelve

- ESP_OK en caso de éxito.
- ESP_ERR_INVALID_ARG Error de parámetro
- ESP_FAIL Error al enviar comando, el esclavo no envió ACK a la transferencia.
- ESP_ERR_INVALID_STATE I2C El driver no fue inicializado o no está en modo master.
- ESP_ERR_TIMEOUT Timout de la operación porque el bus está ocupado.

Definición en la línea 328 del archivo [MCP23017.c](#).

6.23.2.6. `mcp_write_output_port()`

```
esp_err_t mcp_write_output_port (
    enum mcp_port_e port,
    uint8_t data)
```

Función que escribe `data` en el escribe el registro OLAT del `port` del MCP23017.

Escribe `data` entero, no pin a pin.

Parámetros

in	<i>port</i>	Puede tomar los valores PORTA o PORTB, dependiendo del puerto que se quiera escribir.
in	<i>data</i>	Estado True o False que puede tomar los pines del <i>port</i>

Devuelve

- ESP_OK en caso de éxito.
- ESP_ERR_INVALID_ARG Error de parámetro
- ESP_FAIL Error al enviar comando, el esclavo no envió ACK a la transferencia.
- ESP_ERR_INVALID_STATE I2C El driver no fue inicializado o no está en modo master.
- ESP_ERR_TIMEOUT Timout de la operación porque el bus está ocupado.

Definición en la línea 353 del archivo [MCP23017.c](#).

6.24. MCP23017.h

[Ir a la documentación de este archivo.](#)

```

00001
00009
00010 #ifndef __MCP23017_H__
00011 #define __MCP23017_H__
00012
00013 #include "esp_mac.h"
00014 #include "mcp23017_defs.h"
00015 #include <inttypes.h>
00016 #include <stdio.h>
00017 #include <stdlib.h>
00018 #include <stdbool.h>
00019 #include <stdbool.h>
00020
00021
00036 esp_err_t MCP23017_INIT( void );
00037
00058 esp_err_t mcp_get_on_interrupt_input(enum mcp_port_e port, uint8_t *data);
00059
00080 esp_err_t mcp_interrupt_flag(enum mcp_port_e port, uint8_t *data);
00081
00105 esp_err_t mcp_write_output_pin(enum mcp_port_e port, uint8_t pin, bool state);
00106
00127 esp_err_t mcp_read_port(enum mcp_port_e port, uint8_t *data);
00128
00149 esp_err_t mcp_write_output_port(enum mcp_port_e port, uint8_t data);
00150
00151 #endif

```

6.25. Referencia del archivo main/io_control/mcp23017_defs.h

```

#include <inttypes.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

```

Estructuras de datos

- union `MCP23017_IODIR_t`
- union `MCP23017_IPOL_t`
- union `MCP23017_GPINTEN_t`
- union `MCP23017_GPPU_t`
- union `MCP23017_GPIO_t`
- union `MCP23017_OLAT_t`
- union `MCP23017_DEFVAL_t`
- union `MCP23017_INTCON_t`
- union `MCP23017_IOCON_t`
- union `MCP23017_INTF_t`
- union `MCP23017_INTCAP_t`

defines

- `#define __MCP23017_BANK_SEQUENTIAL__ 0`
- `#define __MCP23017_BANK_SPLIT__ 1`
- `#define __MCP23017_MIRROR_UNCONNECTED__ 0`
- `#define __MCP23017_MIRROR_CONNECTED__ 1`
- `#define __MCP23017_SEQOP_ENABLED__ 0`
- `#define __MCP23017_SEQOP_DISABLED__ 1`
- `#define __MCP23017_DISSLW_DISABLED__ 0`
- `#define __MCP23017_DISSLW_ENABLED__ 1`
- `#define __MCP23017_ODR_OPEN_DRAIN_OUTPUT__ 1`
- `#define __MCP23017_ODR_ACTIVE_DRIVER_OUTPUT__ 0`
- `#define __MCP23017_INTPOL_POLARITY_LOW__ 1`
- `#define __MCP23017_INTPOL_POLARITY_HIGH__ 0`
- `#define __MCP23017_GPPU_PULL_UP_ENABLE__ 1`
- `#define __MCP23017_GPPU_PULL_UP_DISABLE__ 0`
- `#define __MCP23017_IODIR_INPUT__ 1`
- `#define __MCP23017_IODIR_OUTPUT__ 0`
- `#define __MCP23017_GPINTEN_ENABLE__ 1`
- `#define __MCP23017_GPINTEN_DISABLE__ 0`
- `#define __MCP23017_INTCON_DEFVAL__ 1`
- `#define __MCP23017_INTCON_CHANGE__ 0`
- `#define __MCP23017_WRITE__ 0`
- `#define __MCP23017_READ__ 1`
- `#define __MCP23017_POSITIVE_LOGIC__ 0`
- `#define __MCP23017_OPPOSITE_LOGIC__ 1`
- `#define __MCP23017_INTERRUPT_ENABLE__ 0`
- `#define __MCP23017_INTERRUPT_DISABLE__ 1`
- `#define __MCP23017_ADDRESS__ 0b010`
- `#define __MCP23017_READ_OPCODE__ 0b0100 << 4 | __MCP23017_ADDRESS__ << 1 | __MCP23017_READ__`
- `#define __MCP23017_WRITE_OPCODE__ 0b0100 << 4 | __MCP23017_ADDRESS__ << 1 | __MCP23017_WRITE__`
- `#define __MCP23017_FUNC_MODE__ __MCP23017_BANK_SEQUENTIAL__`
- `#define MCP23017_IODIRA_REGISTER 0x00`
- `#define MCP23017_IODIRB_REGISTER 0x01`
- `#define MCP23017_IPOLA_REGISTER 0x02`
- `#define MCP23017_IPOLB_REGISTER 0x03`
- `#define MCP23017_GPINTENA_REGISTER 0x04`
- `#define MCP23017_GPINTENB_REGISTER 0x05`

- #define MCP23017_DEFVALA_REGISTER 0x06
- #define MCP23017_DEFVALB_REGISTER 0x07
- #define MCP23017_INTCONA_REGISTER 0x08
- #define MCP23017_INTCONB_REGISTER 0x09
- #define MCP23017_IOCON_REGISTER 0x0A
- #define MCP23017_GPPUA_REGISTER 0x0C
- #define MCP23017_GPPUB_REGISTER 0x0D
- #define MCP23017_INTFA_REGISTER 0x0E
- #define MCP23017_INTFB_REGISTER 0x0F
- #define MCP23017_INTCAPA_REGISTER 0x10
- #define MCP23017_INTCAPB_REGISTER 0x11
- #define MCP23017_GPIOA_REGISTER 0x12
- #define MCP23017_GPIOB_REGISTER 0x13
- #define MCP23017_OLATA_REGISTER 0x14
- #define MCP23017_OLATB_REGISTER 0x15

Enumeraciones

- enum mcp_port_e { PORTA = 0 , PORTB = 1 }

Selector de puertos del MCP23017. Puede ser PORTA (0) o PORTB (1). De esta manera el usuario se abrae en conocer como se llaman los registros del chip, solo con las funciones específicas de lectura y escritura, datos, pines y esta definición de puerto.

6.25.1. Documentación de «define»

6.25.1.1. __MCP23017_ADDRESS__

```
#define __MCP23017_ADDRESS__ 0b010
```

CONFIGURADO POR HARDWARE: Address del MCP23017 = 0b010 ----> OPCODE: 0100AAA0 -> 01000000 |
MCP23017_ADDRESS << 1 | R/W

Definición en la línea 50 del archivo [mcp23017_defs.h](#).

6.25.1.2. __MCP23017_BANK_SEQUENTIAL__

```
#define __MCP23017_BANK_SEQUENTIAL__ 0
```

Modo de banqueo de memoria separado por puertos

Definición en la línea 11 del archivo [mcp23017_defs.h](#).

6.25.1.3. __MCP23017_BANK_SPLIT__

```
#define __MCP23017_BANK_SPLIT__ 1
```

Modo de banqueo de memoria entrelazando puertos para manejo con entreos de 16 bits

Definición en la línea 12 del archivo [mcp23017_defs.h](#).

6.25.1.4. **__MCP23017_DISSLW_DISABLED__**

```
#define __MCP23017_DISSLW_DISABLED__ 0
```

Slew rate deshabilitado

Definición en la línea 20 del archivo [mcp23017_defs.h](#).

6.25.1.5. **__MCP23017_DISSLW_ENABLED__**

```
#define __MCP23017_DISSLW_ENABLED__ 1
```

Slew rate habilitado

Definición en la línea 21 del archivo [mcp23017_defs.h](#).

6.25.1.6. **__MCP23017_FUNC_MODE__**

```
#define __MCP23017_FUNC_MODE__ __MCP23017_BANK_SEQUENTIAL__
```

Modo de funcionamiento del MCP23017 que funcionará en el sistema

Definición en la línea 54 del archivo [mcp23017_defs.h](#).

6.25.1.7. **__MCP23017_GPINTEN_DISABLE__**

```
#define __MCP23017_GPINTEN_DISABLE__ 0
```

Interrupción de GPIO deshabilitada

Definición en la línea 36 del archivo [mcp23017_defs.h](#).

6.25.1.8. **__MCP23017_GPINTEN_ENABLE__**

```
#define __MCP23017_GPINTEN_ENABLE__ 1
```

Interrupción de GPIO habilitada

Definición en la línea 35 del archivo [mcp23017_defs.h](#).

6.25.1.9. **__MCP23017_GPPU_PULL_UP_DISABLE__**

```
#define __MCP23017_GPPU_PULL_UP_DISABLE__ 0
```

DESEactivación de pull up de GPIO

Definición en la línea 30 del archivo [mcp23017_defs.h](#).

6.25.1.10. __MCP23017_GPPU_PULL_UP_ENABLE__

```
#define __MCP23017_GPPU_PULL_UP_ENABLE__ 1
```

Activación de pull up de GPIO

Definición en la línea 29 del archivo [mcp23017_defs.h](#).

6.25.1.11. __MCP23017_INTCON_CHANGE__

```
#define __MCP23017_INTCON_CHANGE__ 0
```

Interrupción dispara por diferencia con último valor

Definición en la línea 39 del archivo [mcp23017_defs.h](#).

6.25.1.12. __MCP23017_INTCON_DEFVAL__

```
#define __MCP23017_INTCON_DEFVAL__ 1
```

Interrupción dispara por diferencia con valor default

Definición en la línea 38 del archivo [mcp23017_defs.h](#).

6.25.1.13. __MCP23017_INTERRUPT_DISABLE__

```
#define __MCP23017_INTERRUPT_DISABLE__ 1
```

Habilitación de la interrupción del GPIO

Definición en la línea 48 del archivo [mcp23017_defs.h](#).

6.25.1.14. __MCP23017_INTERRUPT_ENABLE__

```
#define __MCP23017_INTERRUPT_ENABLE__ 0
```

Deshabilitación de la interrupción del GPIO

Definición en la línea 47 del archivo [mcp23017_defs.h](#).

6.25.1.15. __MCP23017_INTPOL_POLARITY_HIGH__

```
#define __MCP23017_INTPOL_POLARITY_HIGH__ 0
```

Flags de interrupción de los puerto A y B en 1 significa una nueva interrupción

Definición en la línea 27 del archivo [mcp23017_defs.h](#).

6.25.1.16. __MCP23017_INTPOL_POLARITY_LOW__

```
#define __MCP23017_INTPOL_POLARITY_LOW__ 1
```

Flags de interrupción de los puerto A y B en 0 significa una nueva interrupción

Definición en la línea 26 del archivo [mcp23017_defs.h](#).

6.25.1.17. __MCP23017_IODIR_INPUT__

```
#define __MCP23017_IODIR_INPUT__ 1
```

GPIO configurada como entrada

Definición en la línea 32 del archivo [mcp23017_defs.h](#).

6.25.1.18. __MCP23017_IODIR_OUTPUT__

```
#define __MCP23017_IODIR_OUTPUT__ 0
```

GPIO configurada como salida

Definición en la línea 33 del archivo [mcp23017_defs.h](#).

6.25.1.19. __MCP23017_MIRROR_CONNECTED__

```
#define __MCP23017_MIRROR_CONNECTED__ 1
```

Los pines de interrupción están internamente conectados

Definición en la línea 15 del archivo [mcp23017_defs.h](#).

6.25.1.20. __MCP23017_MIRROR_UNCONNECTED__

```
#define __MCP23017_MIRROR_UNCONNECTED__ 0
```

Los pines de interrupción no están internamente conectados

Definición en la línea 14 del archivo [mcp23017_defs.h](#).

6.25.1.21. __MCP23017_ODR_ACTIVE_DRIVER_OUTPUT__

```
#define __MCP23017_ODR_ACTIVE_DRIVER_OUTPUT__ 0
```

Flags de interrupción de los puerto A y B configuradas como salidas activas

Definición en la línea 24 del archivo [mcp23017_defs.h](#).

6.25.1.22. __MCP23017_ODR_OPEN_DRAIN_OUTPUT__

```
#define __MCP23017_ODR_OPEN_DRAIN_OUTPUT__ 1
```

Flags de interrupción de los puerto A y B configuradas open drain

Definición en la línea 23 del archivo [mcp23017_defs.h](#).

6.25.1.23. __MCP23017_OPPOSITE_LOGIC__

```
#define __MCP23017_OPPOSITE_LOGIC__ 1
```

Configuración de la lógica de los GPIO como negativa

Definición en la línea 45 del archivo [mcp23017_defs.h](#).

6.25.1.24. __MCP23017_POSITIVE_LOGIC__

```
#define __MCP23017_POSITIVE_LOGIC__ 0
```

Configuración de la lógica de los GPIO como positiva

Definición en la línea 44 del archivo [mcp23017_defs.h](#).

6.25.1.25. __MCP23017_READ__

```
#define __MCP23017_READ__ 1
```

Función de lectura del MCP23017 - Se inserta en el opcode

Definición en la línea 42 del archivo [mcp23017_defs.h](#).

6.25.1.26. __MCP23017_READ_OPCODE__

```
#define __MCP23017_READ_OPCODE__ 0b0100 << 4 | __MCP23017_ADDRESS__ << 1 | __MCP23017_READ__
```

Código de operación para lectura en el MCP23017

Definición en la línea 52 del archivo [mcp23017_defs.h](#).

6.25.1.27. __MCP23017_SEQOP_DISABLED__

```
#define __MCP23017_SEQOP_DISABLED__ 1
```

El puntero de direccionamiento de memoria no incrementa en cada acceso

Definición en la línea 18 del archivo [mcp23017_defs.h](#).

6.25.1.28. __MCP23017_SEQOP_ENABLED__

```
#define __MCP23017_SEQOP_ENABLED__ 0
```

El puntero de direccionamiento de memoria incrementa en cada acceso

Definición en la línea 17 del archivo [mcp23017_defs.h](#).

6.25.1.29. __MCP23017_WRITE__

```
#define __MCP23017_WRITE__ 0
```

Función de escritura del MCP23017 - Se inserta en el opcode

Definición en la línea 41 del archivo [mcp23017_defs.h](#).

6.25.1.30. __MCP23017_WRITE_OPCODE__

```
#define __MCP23017_WRITE_OPCODE__ 0b0100 << 4 | __MCP23017_ADDRESS__ << 1 | __MCP23017_WRITE__
```

Código de operación para escritura en el MCP23017

Definición en la línea 53 del archivo [mcp23017_defs.h](#).

6.25.1.31. MCP23017_DEFVALA_REGISTER

```
#define MCP23017_DEFVALA_REGISTER 0x06
```

Dirección de memoria del registro DEFVALA

Definición en la línea 63 del archivo [mcp23017_defs.h](#).

6.25.1.32. MCP23017_DEFVALB_REGISTER

```
#define MCP23017_DEFVALB_REGISTER 0x07
```

Dirección de memoria del registro DEFVALB

Definición en la línea 64 del archivo [mcp23017_defs.h](#).

6.25.1.33. MCP23017_GPINTENA_REGISTER

```
#define MCP23017_GPINTENA_REGISTER 0x04
```

Dirección de memoria del registro GPINTENA

Definición en la línea 61 del archivo [mcp23017_defs.h](#).

6.25.1.34. MCP23017_GPINTENB_REGISTER

```
#define MCP23017_GPINTENB_REGISTER 0x05
```

Dirección de memoria del registro GPINTENB

Definición en la línea 62 del archivo [mcp23017_defs.h](#).

6.25.1.35. MCP23017_GPIOA_REGISTER

```
#define MCP23017_GPIOA_REGISTER 0x12
```

Dirección de memoria del registro GPIOA

Definición en la línea 75 del archivo [mcp23017_defs.h](#).

6.25.1.36. MCP23017_GPIOB_REGISTER

```
#define MCP23017_GPIOB_REGISTER 0x13
```

Dirección de memoria del registro GPIOB

Definición en la línea 76 del archivo [mcp23017_defs.h](#).

6.25.1.37. MCP23017_GPPUA_REGISTER

```
#define MCP23017_GPPUA_REGISTER 0x0C
```

Dirección de memoria del registro GPPUA

Definición en la línea 69 del archivo [mcp23017_defs.h](#).

6.25.1.38. MCP23017_GPPUB_REGISTER

```
#define MCP23017_GPPUB_REGISTER 0x0D
```

Dirección de memoria del registro GPPUB

Definición en la línea 70 del archivo [mcp23017_defs.h](#).

6.25.1.39. MCP23017_INTCAPA_REGISTER

```
#define MCP23017_INTCAPA_REGISTER 0x10
```

Dirección de memoria del registro INTCAPA

Definición en la línea 73 del archivo [mcp23017_defs.h](#).

6.25.1.40. MCP23017_INTCAPB_REGISTER

```
#define MCP23017_INTCAPB_REGISTER 0x11
```

Dirección de memoria del registro INTCAPB

Definición en la línea 74 del archivo [mcp23017_defs.h](#).

6.25.1.41. MCP23017_INTCONA_REGISTER

```
#define MCP23017_INTCONA_REGISTER 0x08
```

Dirección de memoria del registro INTCONA

Definición en la línea 65 del archivo [mcp23017_defs.h](#).

6.25.1.42. MCP23017_INTCONB_REGISTER

```
#define MCP23017_INTCONB_REGISTER 0x09
```

Dirección de memoria del registro INTCONB

Definición en la línea 66 del archivo [mcp23017_defs.h](#).

6.25.1.43. MCP23017_INTFA_REGISTER

```
#define MCP23017_INTFA_REGISTER 0x0E
```

Dirección de memoria del registro INTFA

Definición en la línea 71 del archivo [mcp23017_defs.h](#).

6.25.1.44. MCP23017_INTFB_REGISTER

```
#define MCP23017_INTFB_REGISTER 0x0F
```

Dirección de memoria del registro INTFB

Definición en la línea 72 del archivo [mcp23017_defs.h](#).

6.25.1.45. MCP23017_IOCON_REGISTER

```
#define MCP23017_IOCON_REGISTER 0x0A
```

Dirección de memoria del registro IOCON

Definición en la línea 67 del archivo [mcp23017_defs.h](#).

6.25.1.46. MCP23017_IODIRA_REGISTER

```
#define MCP23017_IODIRA_REGISTER 0x00
```

Dirección de memoria del registro IODIRA

Definición en la línea [57](#) del archivo [mcp23017_defs.h](#).

6.25.1.47. MCP23017_IODIRB_REGISTER

```
#define MCP23017_IODIRB_REGISTER 0x01
```

Dirección de memoria del registro IODIRB

Definición en la línea [58](#) del archivo [mcp23017_defs.h](#).

6.25.1.48. MCP23017_IPOLA_REGISTER

```
#define MCP23017_IPOLA_REGISTER 0x02
```

Dirección de memoria del registro IPOLA

Definición en la línea [59](#) del archivo [mcp23017_defs.h](#).

6.25.1.49. MCP23017_IPOLB_REGISTER

```
#define MCP23017_IPOLB_REGISTER 0x03
```

Dirección de memoria del registro IPOLB

Definición en la línea [60](#) del archivo [mcp23017_defs.h](#).

6.25.1.50. MCP23017_OLATA_REGISTER

```
#define MCP23017_OLATA_REGISTER 0x14
```

Dirección de memoria del registro OLATA

Definición en la línea [77](#) del archivo [mcp23017_defs.h](#).

6.25.1.51. MCP23017_OLATB_REGISTER

```
#define MCP23017_OLATB_REGISTER 0x15
```

Dirección de memoria del registro OLATB

Definición en la línea [78](#) del archivo [mcp23017_defs.h](#).

6.25.2. Documentación de enumeraciones

6.25.2.1. mcp_port_e

```
enum mcp_port_e
```

Selector de puertos del MCP23017. Puede ser PORTA (0) o PORTB (1). De esta manera el usuario se abraza en conocer como se llaman los registros del chip, solo con las funciones específicas de lectura y escritura, datos, pines y esta definición de puerto.

Valores de enumeraciones

PORTA	Puerto A del MCP23017
PORTB	Puerto B del MCP23017

Definición en la línea 340 del archivo [mcp23017_defs.h](#).

6.26. mcp23017_defs.h

[Ir a la documentación de este archivo.](#)

```

00001 #ifndef __MCP23017_DEFS_H__
00002
00003 #define __MCP23017_DEFS_H__
00004
00005 #include <inttypes.h>
00006 #include <stdio.h>
00007 #include <stdlib.h>
00008 #include <stdbool.h>
00009 #include <stdbool.h>
00010
00011 #define __MCP23017_BANK_SEQUENTIAL__ 0
00012 #define __MCP23017_BANK_SPLIT__ 1
00013
00014 #define __MCP23017_MIRROR_UNCONNECTED__ 0
00015 #define __MCP23017_MIRROR_CONNECTED__ 1
00016
00017 #define __MCP23017_SEQOP_ENABLED__ 0
00018 #define __MCP23017_SEQOP_DISABLED__ 1
00019
00020 #define __MCP23017_DISSLLW_DISABLED__ 0
00021 #define __MCP23017_DISSLLW_ENABLED__ 1
00022
00023 #define __MCP23017_ODR_OPEN_DRAIN_OUTPUT__ 1
00024 #define __MCP23017_ODR_ACTIVE_DRIVER_OUTPUT__ 0
00025
00026 #define __MCP23017_INTPOL_POLARITY_LOW__ 1
00027 #define __MCP23017_INTPOL_POLARITY_HIGH__ 0
00028
00029 #define __MCP23017_GPPU_PULL_UP_ENABLE__ 1
00030 #define __MCP23017_GPPU_PULL_UP_DISABLE__ 0
00031
00032 #define __MCP23017_IODIR_INPUT__ 1
00033 #define __MCP23017_IODIR_OUTPUT__ 0
00034
00035 #define __MCP23017_GPINTEN_ENABLE__ 1
00036 #define __MCP23017_GPINTEN_DISABLE__ 0
00037
00038 #define __MCP23017_INTCON_DEFVAL__ 1
00039 #define __MCP23017_INTCON_CHANGE__ 0
00040
00041 #define __MCP23017_WRITE__ 0
00042 #define __MCP23017_READ__ 1
00043
00044 #define __MCP23017_POSITIVE_LOGIC__ 0
00045 #define __MCP23017_OPPOSITE_LOGIC__ 1
00046
00047 #define __MCP23017_INTERRUPT_ENABLE__ 0
00048 #define __MCP23017_INTERRUPT_DISABLE__ 1
00049
00050 #define __MCP23017_ADDRESS__ 0b010
00051
00052 #define __MCP23017_READ_OPCODE__ 0b0100 « 4 | __MCP23017_ADDRESS__ « 1 | __MCP23017_READ__
00053 #define __MCP23017_WRITE_OPCODE__ 0b0100 « 4 | __MCP23017_ADDRESS__ « 1 | __MCP23017_WRITE__
00054 #define __MCP23017_FUNC_MODE__ __MCP23017_BANK_SEQUENTIAL__

00055
00056 #if __MCP23017_FUNC_MODE__ == __MCP23017_SEQUENTIAL_MODE__
00057     #define MCP23017_IODIRA_REGISTER 0x00
00058     #define MCP23017_IODIRB_REGISTER 0x01
00059     #define MCP23017_IPOLA_REGISTER 0x02
00060     #define MCP23017_IPOLB_REGISTER 0x03
00061     #define MCP23017_GPINTENA_REGISTER 0x04
00062     #define MCP23017_GPINTENB_REGISTER 0x05
00063     #define MCP23017_DEFVALA_REGISTER 0x06
00064     #define MCP23017_DEFVALB_REGISTER 0x07
00065     #define MCP23017_INTCONA_REGISTER 0x08

```

```

00066     #define MCP23017_INTCONB_REGISTER          0x09
00067     #define MCP23017_IOCON_REGISTER           0x0A
00068 // #define MCP23017__IOCON_REGISTER          0x0B
00069     memoria del registro IOCON */
00070     #define MCP23017_GPPUA_REGISTER           0x0C
00071     #define MCP23017_GPPUB_REGISTER           0x0D
00072     #define MCP23017_INTFB_REGISTER           0x0E
00073     #define MCP23017_INTCAPA_REGISTER          0x0F
00074     #define MCP23017_INTCAPB_REGISTER          0x10
00075     #define MCP23017_GPIOA_REGISTER           0x11
00076     #define MCP23017_GPIOB_REGISTER           0x12
00077     #define MCP23017_OLATA_REGISTER           0x13
00078     #define MCP23017_OLATB_REGISTER           0x14
00079 #elif __MCP23017_FUNC_MODE__ == __MCP23017_BYTE_MODE__
00080     #define MCP23017_IODIRA_REGISTER          0x00
00081     #define MCP23017_IODIRB_REGISTER          0x10
00082     #define MCP23017_IPOLA_REGISTER           0x01
00083     #define MCP23017_IPOLB_REGISTER           0x11
00084     #define MCP23017_GPINTENA_REGISTER         0x02
00085     #define MCP23017_GPINTENB_REGISTER         0x12
00086     #define MCP23017_DEFVALA_REGISTER          0x03
00087     #define MCP23017_DEFVALB_REGISTER          0x13
00088     #define MCP23017_INTCONA_REGISTER          0x04
00089     #define MCP23017_INTCONB_REGISTER          0x14
00090     #define MCP23017_IOCON_REGISTER           0x05
00091 // #define MCP23017__IOCON_REGISTER          0x15
00092     memoria del registro IOCON */
00093     #define MCP23017_GPPUA_REGISTER           0x06
00094     #define MCP23017_GPPUB_REGISTER           0x16
00095     #define MCP23017_INTFB_REGISTER           0x07
00096     #define MCP23017_INTCAPA_REGISTER          0x17
00097     #define MCP23017_INTCAPB_REGISTER          0x08
00098     #define MCP23017_GPIOA_REGISTER           0x18
00099     #define MCP23017_GPIOB_REGISTER           0x09
00100     #define MCP23017_OLATA_REGISTER           0x19
00101     #define MCP23017_OLATB_REGISTER           0x0A
00102 #endif
00103
00111 typedef union {
00112     uint8_t all;
00113     struct {
00114         uint8_t IO0 : 1;
00115         uint8_t IO1 : 1;
00116         uint8_t IO2 : 1;
00117         uint8_t IO3 : 1;
00118         uint8_t IO4 : 1;
00119         uint8_t IO5 : 1;
00120         uint8_t IO6 : 1;
00121         uint8_t IO7 : 1;
00122     } bits;
00123 } MCP23017_IODIR_t;
00124
00132 typedef union {
00133     uint8_t all;
00134     struct {
00135         uint8_t IP0 : 1;
00136         uint8_t IP1 : 1;
00137         uint8_t IP2 : 1;
00138         uint8_t IP3 : 1;
00139         uint8_t IP4 : 1;
00140         uint8_t IP5 : 1;
00141         uint8_t IP6 : 1;
00142         uint8_t IP7 : 1;
00143     } bits;
00144 } MCP23017_IPOL_t;
00145
00153 typedef union {
00154     uint8_t all;
00155     struct {
00156         uint8_t GPINT0 : 1;
00157         uint8_t GPINT1 : 1;
00158         uint8_t GPINT2 : 1;
00159         uint8_t GPINT3 : 1;
00160         uint8_t GPINT4 : 1;
00161         uint8_t GPINT5 : 1;
00162         uint8_t GPINT6 : 1;
00163         uint8_t GPINT7 : 1;
00164     } bits;
00165 } MCP23017_GPINTEN_t;
00166
00174 typedef union {
00175     uint8_t all;
00176     struct {
00177         uint8_t PU0 : 1;
00178         uint8_t PU1 : 1;

```

```
00179     uint8_t PU2 : 1;
00180     uint8_t PU3 : 1;
00181     uint8_t PU4 : 1;
00182     uint8_t PU5 : 1;
00183     uint8_t PU6 : 1;
00184     uint8_t PU7 : 1;
00185 } bits;
00186 } MCP23017_GPPU_t;
00187
00195 typedef union {
00196     uint8_t all;
00197     struct {
00198         uint8_t GP0 : 1;
00199         uint8_t GP1 : 1;
00200         uint8_t GP2 : 1;
00201         uint8_t GP3 : 1;
00202         uint8_t GP4 : 1;
00203         uint8_t GP5 : 1;
00204         uint8_t GP6 : 1;
00205         uint8_t GP7 : 1;
00206     } bits;
00207 } MCP23017_GPIO_t;
00208
00216 typedef union {
00217     uint8_t all;
00218     struct {
00219         uint8_t OLO : 1;
00220         uint8_t OLL : 1;
00221         uint8_t OL2 : 1;
00222         uint8_t OL3 : 1;
00223         uint8_t OL4 : 1;
00224         uint8_t OL5 : 1;
00225         uint8_t OL6 : 1;
00226         uint8_t OL7 : 1;
00227     } bits;
00228 } MCP23017_OLAT_t;
00229
00237 typedef union {
00238     uint8_t all;
00239     struct {
00240         uint8_t DEF0 : 1;
00241         uint8_t DEF1 : 1;
00242         uint8_t DEF2 : 1;
00243         uint8_t DEF3 : 1;
00244         uint8_t DEF4 : 1;
00245         uint8_t DEF5 : 1;
00246         uint8_t DEF6 : 1;
00247         uint8_t DEF7 : 1;
00248     } bits;
00249 } MCP23017_DEFVAL_t;
00250
00258 typedef union {
00259     uint8_t all;
00260     struct {
00261         uint8_t IOC0 : 1;
00262         uint8_t IOC1 : 1;
00263         uint8_t IOC2 : 1;
00264         uint8_t IOC3 : 1;
00265         uint8_t IOC4 : 1;
00266         uint8_t IOC5 : 1;
00267         uint8_t IOC6 : 1;
00268         uint8_t IOC7 : 1;
00269     } bits;
00270 } MCP23017_INTCON_t;
00271
00279 typedef union {
00280     uint8_t all;
00281     struct {
00282         uint8_t reserved : 1;
00283         uint8_t INTPOL : 1;
00284         uint8_t ODR : 1;
00285         uint8_t reserved_2 : 1;
00286         uint8_t DISSLW : 1;
00287         uint8_t SEQOP : 1;
00288         uint8_t MIRROR : 1;
00289         uint8_t BANK : 1;
00290     } bits;
00291 } MCP23017_IOCON_t;
00292
00300 typedef union {
00301     uint8_t all;
00302     struct {
00303         uint8_t INT0 : 1;
00304         uint8_t INT1 : 1;
00305         uint8_t INT2 : 1;
00306         uint8_t INT3 : 1;
00307         uint8_t INT4 : 1;
```

```

00308     uint8_t INT5 : 1;
00309     uint8_t INT6 : 1;
00310     uint8_t INT7 : 1;
00311 } bits;
00312 } MCP23017_INTF_t;
00313
00321 typedef union {
00322     uint8_t all;
00323     struct {
00324         uint8_t ICP0 : 1;
00325         uint8_t ICP1 : 1;
00326         uint8_t ICP2 : 1;
00327         uint8_t ICP3 : 1;
00328         uint8_t ICP4 : 1;
00329         uint8_t ICP5 : 1;
00330         uint8_t ICP6 : 1;
00331         uint8_t ICP7 : 1;
00332     } bits;
00333 } MCP23017_INTCAP_t;
00334
00340 enum mcp_port_e {
00341     PORTA = 0,
00342     PORTB = 1
00343 };
00344
00345 #endif

```

6.27. Referencia del archivo main/LVFV_system.h

Declaraciones de estructuras, variables, definiciones y estados generales del sistema utilizados en varios de los archivos.

```
#include <inttypes.h>
#include <time.h>
```

Estructuras de datos

- struct `frequency_settings_t`
- struct `time_settings_t`
- struct `seecurity_settings_t`
- struct `system_status_t`

Estructura con las variables necesarias para establecer las condiciones de trabajo del motor.

- struct `frequency_settings_SH1106_t`
- struct `time_settings_SH1106_t`
- struct `seecurity_settings_SH1106_t`

defines

- `#define LINE_INCREMENT 9`
- `#define VARIABLE_FIRST 20`
- `#define VARIABLE_SECOND 29`
- `#define VARIABLE_THIRD 38`
- `#define VARIABLE_FOURTH 47`
- `#define VARIABLE_FIFTH 56`
- `#define VARIABLE_SIXTH 65`
- `#define VARIABLE_SEVENTH 74`
- `#define VARIABLE_EIGTH 83`
- `#define SH1106_SIZE_1 0`
- `#define SH1106_SIZE_2 1`

typedefs

- `typedef enum system_status_e system_status_e`
- `typedef enum sh1106_variable_lines_e sh1106_variable_lines_e`
- `typedef struct frequency_settings_t frequency_settings_t`
- `typedef struct time_settings_t time_settings_t`
- `typedef struct security_settings_t security_settings_t`
- `typedef struct system_status_t system_status_t`
- `typedef struct frequency_settings_SH1106_t frequency_settings_SH1106_t`
- `typedef struct time_settings_SH1106_t time_settings_SH1106_t`
- `typedef struct security_settings_SH1106_t security_settings_SH1106_t`

Enumeraciones

- `enum systemSignal_e {`
`BUTTON_MENU = 1, BUTTON_OK, BUTTON_BACK, BUTTON_UP,`
`BUTTON_DOWN, BUTTON_LEFT, BUTTON_RIGHT, BUTTON_SAVE,`
`EMERGENCI_STOP_PRESSED, EMERGENCI_STOP_RELEASED, START_PRESSED, START_RELEASED`
,
- `enum system_status_e {`
`SYSTEM_IDLE, SYSTEM_ACCL_DESACCEL, SYSTEM_REGIME, SYSTEM_BREAKING,`
`SYSTEM_EMERGENCY, SYSTEM_EMERGENCY_OK}`
Posibles estados general que puede tomar el sistema.
- `enum sh1106_variable_lines_e {`
`first = VARIABLE_FIRST, second = VARIABLE_SECOND, third = VARIABLE_THIRD, fourth = VARIABLE_`
`_FOURTH,`
`fifth = VARIABLE_FIFTH, sixth = VARIABLE_SIXTH, seventh = VARIABLE_SEVENTH }`

6.27.1. Descripción detallada

Declaraciones de estructuras, variables, definiciones y estados generales del sistema utilizados en varios de los archivos.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo [LVFV_system.h](#).

6.27.2. Documentación de «define»

6.27.2.1. LINE_INCREMENT

```
#define LINE_INCREMENT 9
```

Definición en la línea 15 del archivo [LVFV_system.h](#).

6.27.2.2. SH1106_SIZE_1

```
#define SH1106_SIZE_1 0
```

Definición en la línea 124 del archivo [LVFV_system.h](#).

6.27.2.3. SH1106_SIZE_2

```
#define SH1106_SIZE_2 1
```

Definición en la línea 125 del archivo [LVFV_system.h](#).

6.27.2.4. VARIABLE_EIGHTH

```
#define VARIABLE_EIGHTH 83
```

Definición en la línea 24 del archivo [LVFV_system.h](#).

6.27.2.5. VARIABLE_FIFTH

```
#define VARIABLE_FIFTH 56
```

Definición en la línea 21 del archivo [LVFV_system.h](#).

6.27.2.6. VARIABLE_FIRST

```
#define VARIABLE_FIRST 20
```

Definición en la línea 17 del archivo [LVFV_system.h](#).

6.27.2.7. VARIABLE_FOURTH

```
#define VARIABLE_FOURTH 47
```

Definición en la línea 20 del archivo [LVFV_system.h](#).

6.27.2.8. VARIABLE_SECOND

```
#define VARIABLE_SECOND 29
```

Definición en la línea 18 del archivo [LVFV_system.h](#).

6.27.2.9. VARIABLE_SEVENTH

```
#define VARIABLE_SEVENTH 74
```

Definición en la línea 23 del archivo [LVFV_system.h](#).

6.27.2.10. VARIABLE_SIXTH

```
#define VARIABLE_SIXTH 65
```

Definición en la línea 22 del archivo [LVFV_system.h](#).

6.27.2.11. VARIABLE_THIRD

```
#define VARIABLE_THIRD 38
```

Definición en la línea 19 del archivo [LVFV_system.h](#).

6.27.3. Documentación de «typedef»

6.27.3.1. frequency_settings_SH1106_t

```
typedef struct frequency_settings_SH1106_t frequency_settings_SH1106_t
```

6.27.3.2. frequency_settings_t

```
typedef struct frequency_settings_t frequency_settings_t
```

6.27.3.3. seccurity_settings_SH1106_t

```
typedef struct seccurity_settings_SH1106_t seccurity_settings_SH1106_t
```

6.27.3.4. seccurity_settings_t

```
typedef struct seccurity_settings_t seccurity_settings_t
```

6.27.3.5. sh1106_variable_lines_e

```
typedef enum sh1106_variable_lines_e sh1106_variable_lines_e
```

6.27.3.6. system_status_e

```
typedef enum system_status_e system_status_e
```

6.27.3.7. system_status_t

```
typedef struct system_status_t system_status_t
```

6.27.3.8. time_settings_SH1106_t

```
typedef struct time_settings_SH1106_t time_settings_SH1106_t
```

6.27.3.9. time_settings_t

```
typedef struct time_settings_t time_settings_t
```

6.27.4. Documentación de enumeraciones

6.27.4.1. sh1106_variable_lines_e

```
enum sh1106_variable_lines_e
```

Valores de enumeraciones

first	
second	
third	
fourth	
fifth	
sixth	
seventh	

Definición en la línea 76 del archivo [LVFV_system.h](#).

6.27.4.2. system_status_e

```
enum system_status_e
```

Posibles estados general que puede tomar el sistema.

Valores de enumeraciones

SYSTEM_IDLE	
SYSTEM_ACCEL_DESACCEL	
SYSTEM_REGIME	
SYSTEM_BREAKING	
SYSTEM_EMERGENCY	
SYSTEM_EMERGENCY_OK	

Definición en la línea 67 del archivo [LVFV_system.h](#).

6.27.4.3. systemSignal_e

enum [systemSignal_e](#)

Variable dedicada a la identificación del tipo de señal que es recibida por las entradas digitales y analógicas del sistema para que se tome una decisión a nivel de sistema.

Valores de enumeraciones

BUTTON_MENU	
BUTTON_OK	
BUTTON_BACK	
BUTTON_UP	
BUTTON_DOWN	
BUTTON_LEFT	
BUTTON_RIGHT	
BUTTON_SAVE	
EMERGENCI_STOP_PRESSED	
EMERGENCI_STOP_RELEASED	
START_PRESSED	
START_RELEASED	
TERMO_SW_PRESSED	
TERMO_SW_RELEASED	
STOP_PRESSED	
STOP_RELEASED	
SPEED_SELECTOR_0	
SPEED_SELECTOR_1	
SPEED_SELECTOR_2	
SPEED_SELECTOR_3	
SPEED_SELECTOR_4	

SPEED_SELECTOR_5	
SPEED_SELECTOR_6	
SPEED_SELECTOR_7	
SPEED_SELECTOR_8	
SPEED_SELECTOR_9	
SECURITY_EXCEEDED	
SECURITY_OK	

Definición en la línea 31 del archivo [LVFV_system.h](#).

6.28. LVFV_system.h

[Ir a la documentación de este archivo.](#)

```

00001
00009 #ifndef LVFV_SYSTEM_H
00010 #define LVFV_SYSTEM_H
00011
00012 #include <inttypes.h>
00013 #include <time.h>
00014
00015 #define LINE_INCREMENT 9
00016
00017 #define VARIABLE_FIRST 20
00018 #define VARIABLE_SECOND 29
00019 #define VARIABLE_THIRD 38
00020 #define VARIABLE_FOURTH 47
00021 #define VARIABLE_FIFTH 56
00022 #define VARIABLE_SIXTH 65
00023 #define VARIABLE_SEVENTH 74
00024 #define VARIABLE_EIGHTH 83
00025
00031 typedef enum {
00032     BUTTON_MENU = 1,
00033     BUTTON_OK,
00034     BUTTON_BACK,
00035     BUTTON_UP,
00036     BUTTON_DOWN,
00037     BUTTON_LEFT,
00038     BUTTON_RIGHT,
00039     BUTTON_SAVE,
00040     EMERGENCY_STOP_PRESSED,
00041     EMERGENCY_STOP_RELEASED,
00042     START_PRESSED,
00043     START_RELEASED,
00044     TERMO_SW_PRESSED,
00045     TERMO_SW_RELEASED,
00046     STOP_PRESSED,
00047     STOP_RELEASED,
00048     SPEED_SELECTOR_0,
00049     SPEED_SELECTOR_1,
00050     SPEED_SELECTOR_2,
00051     SPEED_SELECTOR_3,
00052     SPEED_SELECTOR_4,
00053     SPEED_SELECTOR_5,
00054     SPEED_SELECTOR_6,
00055     SPEED_SELECTOR_7,
00056     SPEED_SELECTOR_8,
00057     SPEED_SELECTOR_9,
00058     SECURITY_EXCEEDED,
00059     SECURITY_OK
00060 } systemSignal_e;
00061
00067 typedef enum system_status_e {

```

```

00068     SYSTEM_IDLE,
00069     SYSTEM_ACCEL_DESACCEL,
00070     SYSTEM_REGIME,
00071     SYSTEM_BREAKING,
00072     SYSTEM_EMERGENCY,
00073     SYSTEM_EMERGENCY_OK
00074 } system_status_e;
00075
00076 typedef enum sh1106_variable_lines_e{
00077     first = VARIABLE_FIRST,
00078     second = VARIABLE_SECOND,
00079     third = VARIABLE_THIRD,
00080     fourth = VARIABLE_FOURTH,
00081     fifth = VARIABLE_FIFTH,
00082     sixth = VARIABLE_SIXTH,
00083     seventh = VARIABLE_SEVENTH
00084 } sh1106_variable_lines_e;
00085
00086 typedef struct frequency_settings_t {
00087     uint16_t freq_regime;
00088     uint16_t acceleration;
00089     uint16_t desacceleration;
00090     uint16_t input_variable;
00091 } frequency_settings_t;
00092
00093 typedef struct time_settings_t {
00094     struct tm *time_system;
00095     struct tm *time_start;
00096     struct tm *time_stop;
00097 } time_settings_t;
00098
00099 typedef struct security_settings_t {
00100     uint16_t vbus_min;
00101     uint16_t ibus_max;
00102 } security_settings_t;
00103
00104 typedef struct system_status_t {
00105     uint16_t frequency;                                // Frecuencia actual de giro del motor
00106     uint16_t frequency_destiny;                         // Frecuencia a la que terminará
00107     uint16_t vbus_min;                                  // Tensión mínima del bus de continua en
00108     la que será una condición buena                  // Corriente máxima del bus de continua en
00109     uint16_t ibus_max;                                // Velocidad de aceleración del motor
00110     la que será una condición buena                  // Velocidad de desaceleración del motor
00111     uint16_t acceleration;                            // Índice de entrada aislada seleccionado
00112     configurada por el usuario                      // Estado general del sistema
00113     uint16_t desacceleration;                         // Estado actual de las señales de
00114     configurada por el usuario                      emergencia b0: Entrada emergencia; b1: Security; b3: Termo-switch
00115     uint8_t inputs_status;                           00116
00116     system_status_e status;                          // Estado general del sistema
00117     uint8_t emergency_signals;                      // Estado actual de las señales de
00118     uint8_t emergency_b0;                           emergencia b0: Entrada emergencia; b1: Security; b3: Termo-switch
00119 } system_status_t;
00120
00121 /*
00122  * Definiciones para display
00123 */
00124 #define SH1106_SIZE_1          0
00125 #define SH1106_SIZE_2          1
00126
00127 typedef struct frequency_settings_SH1106_t {
00128     frequency_settings_t frequency_settings;
00129     sh1106_variable_lines_e *edit;
00130     uint8_t edit_variable;
00131     uint8_t *edit_flag;
00132     uint8_t *multiplier;
00133 } frequency_settings_SH1106_t;
00134
00135 typedef struct time_settings_SH1106_t {
00136     time_settings_t time_settings;
00137     sh1106_variable_lines_e *edit;
00138     uint8_t edit_variable;
00139     uint8_t *edit_flag;
00140     uint8_t *multiplier;
00141 } time_settings_SH1106_t;
00142
00143 typedef struct security_settings_SH1106_t {
00144     security_settings_t security_settings;
00145     sh1106_variable_lines_e *edit;
00146     uint8_t edit_variable;
00147     uint8_t *edit_flag;
00148     uint8_t *multiplier;
00149 } security_settings_SH1106_t;
00150
00151 #endif

```

6.29. Referencia del archivo main/main.c

Inicialización de tareas y periféricos que no tienen una tarea específica. El resto de los periféricos se inicializan en las tareas que son utilizados.

```
#include <inttypes.h>
#include <time.h>
#include <sys/time.h>
#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_chip_info.h"
#include "esp_flash.h"
#include "esp_system.h"
#include "esp_log.h"
#include "esp_err.h"
#include "esp_sleep.h"
#include "esp_clk_tree.h"
#include "driver/gpio.h"
#include "./display/display.h"
#include "./io_control/io_control.h"
#include "./System/SysControl.h"
#include "./adc/adc.h"
#include "./nvs/nvs.h"
#include "./rtc/rtc.h"
#include "./wifi/wifi.h"
#include "LVFV_system.h"
```

Funciones

- void [app_main](#) (void)
Tag de logs del sistema.

Variables

- static const char * [TAG](#) = "UTN-CA-PF2025"

6.29.1. Descripción detallada

Inicialización de tareas y periféricos que no tienen una tarea específica. El resto de los periféricos se inicializan en las tareas que son utilizados.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo [main.c](#).

6.29.2. Documentación de funciones

6.29.2.1. app_main()

```
void app_main (
    void )
```

Tag de logs del sistema.

Función principal de la aplicación

Definición en la línea 43 del archivo [main.c](#).

6.29.3. Documentación de variables

6.29.3.1. TAG

```
const char* TAG = "UTN-CA-PF2025" [static]
```

Definición en la línea 37 del archivo [main.c](#).

6.30. main.c

[Ir a la documentación de este archivo.](#)

```
00001
00009 #include <inttypes.h>
00010 #include <time.h>
00011 #include <sys/time.h>
00012
00013 #include "sdkconfig.h"
00014 #include "freertos/FreeRTOS.h"
00015 #include "freertos/task.h"
00016
00017 #include "esp_chip_info.h"
00018 #include "esp_flash.h"
00019 #include "esp_system.h"
00020 #include "esp_log.h"
00021 #include "esp_err.h"
00022 #include "esp_sleep.h"
00023 #include "esp_clk_tree.h"
00024
00025 #include "driver/gpio.h"
00026
00027 #include "./display/display.h"
00028 #include "./io_control/io_control.h"
00029 #include "./System/SysControl.h"
00030 #include "./adc/adc.h"
00031 #include "./nvs/nvs.h"
00032 #include "./rtc/rtc.h"
00033 #include "./wifi/wifi.h"
00034
00035 #include "LVFV_system.h"
00036
00037 static const char *TAG = "UTN-CA-PF2025";
00038
00043 void app_main(void) {
00044
00045     nvs_init_once();
00046     if ( initRTC() == ESP_OK ) {
00047         ESP_LOGI(TAG, "RTC inicializado correctamente");
00048     } else {
00049         ESP_LOGE(TAG, "Error al inicializar el RTC");
00050         ESP_ERROR_CHECK(ESP_FAIL);
00051     }
00052     xTaskCreatePinnedToCore( task_display,
                                "Tarea Display",
                                NULL, 10, NULL, 1);
```

4096,

```

00053
00054     xTaskCreatePinnedToCore(adc_task, "adc_task", 4096,
00055         NULL, 9, NULL, 0);
00056     xTaskCreatePinnedToCore(SPI_communication, "SPI_communication", 4096,
00057         NULL, 9, NULL, 0);
00058     xTaskCreatePinnedToCore(GPIO_interrupt_attendance_task, "GPIO_interrupt_attendance_task", 4096,
00059         NULL, 10, NULL, 0);
00060
00061     wifi_init_softap();
00062     // start_mdns();
00063     start_webserver();
00064
00065 }

```

6.31. Referencia del archivo main/nvs/nvs.c

Funciones de lectura y escritura de variables no volátiles. Carga en sistema además las variables de seguridad al momento de ser guardadas en la memoria.

```

#include "nvs_flash.h"
#include "nvs.h"
#include "esp_log.h"
#include "../LVFV_system.h"

```

defines

- #define **save_frequency**(freq_op)
- #define **save_acceleration**(freq_acceleration)
- #define **save_desacceleration**(freq_desacceleration)
- #define **save_input_variable**(freq_input_variable)
- #define **save_vbus_min**(vbus_min)
- #define **save_ibus_max**(ibus_max)
- #define **save_hour_ini**(hour_ini)
- #define **save_min_ini**(min_ini)
- #define **save_hour_fin**(hour_fin)
- #define **save_min_fin**(min_fin)
- #define **load_frequency**(freq_op)
- #define **load_acceleration**(freq_acceleration)
- #define **load_desacceleration**(freq_desacceleration)
- #define **load_input_variable**(freq_input_variable)
- #define **load_vbus_min**(vbus_min)
- #define **load_ibus_max**(ibus_max)
- #define **load_hour_ini**(hour_ini)
- #define **load_min_ini**(min_ini)
- #define **load_hour_fin**(hour_fin)
- #define **load_min_fin**(min_fin)

Funciones

- static esp_err_t [save_16](#) (const char *var_tag, int16_t value)
Accede a la memoria NVS para guardar un dato de 16 bits en memoria no volatil.
- static esp_err_t [load_16](#) (const char *var_tag, int16_t *value, int16_t defval)
Accede a la memoria NVS para obtener un dato de 16 bits guardado en memoria no volatil.
- esp_err_t [nvs_init_once](#) (void)
Inicializa la memoria NVS una única vez.
- esp_err_t [load_variables](#) ([frequency_settings_t](#) *frequency_settings, [time_settings_t](#) *time_settings, [security_settings_t](#) *security_settings)
Carga desde NVS los parámetros de frecuencia, seguridad y ventanas horarias.
- esp_err_t [save_variables](#) ([frequency_settings_t](#) *frequency_settings, [time_settings_t](#) *time_settings, [security_settings_t](#) *security_settings)
Guarda en NVS los parámetros de frecuencia, seguridad y ventanas horarias.

Variables

- static const char * [TAG](#) = "NVS"

6.31.1. Descripción detallada

Funciones de lectura y escritura de variables no volátiles. Carga en sistema además las variables de seguridad al momento de ser guardadas en la memoria.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo [nvs.c](#).

6.31.2. Documentación de «define»

6.31.2.1. load_acceleration

```
#define load_acceleration(  
    freq_acceleration)
```

Valor:

```
load_16("freq_acce", (freq_acceleration), 5 )
```

Definición en la línea [28](#) del archivo [nvs.c](#).

6.31.2.2. **load_desacceleration**

```
#define load_desacceleration(  
    freq_desacceleration)
```

Valor:

```
load_16("freq_desa", (freq_desacceleration), 3 )
```

Definición en la línea 29 del archivo [nvs.c](#).

6.31.2.3. **load_frequency**

```
#define load_frequency(  
    freq_op)
```

Valor:

```
load_16("freq_freq", (freq_op), 50 )
```

Definición en la línea 27 del archivo [nvs.c](#).

6.31.2.4. **load_hour_fin**

```
#define load_hour_fin(  
    hour_fin)
```

Valor:

```
load_16("hour_fin", (hour_fin), 20 )
```

Definición en la línea 35 del archivo [nvs.c](#).

6.31.2.5. **load_hour_ini**

```
#define load_hour_ini(  
    hour_ini)
```

Valor:

```
load_16("hour_ini", (hour_ini), 20 )
```

Definición en la línea 33 del archivo [nvs.c](#).

6.31.2.6. **load_ibus_max**

```
#define load_ibus_max(  
    ibus_max)
```

Valor:

```
load_16("ibus_max", (ibus_max), 2000 )
```

Definición en la línea 32 del archivo [nvs.c](#).

6.31.2.7. `load_input_variable`

```
#define load_input_variable(  
    freq_input_variable)
```

Valor:

```
load_16("freq_input", (freq_input_variable), 1 )
```

Definición en la línea 30 del archivo [nvs.c](#).

6.31.2.8. `load_min_fin`

```
#define load_min_fin(  
    min_fin)
```

Valor:

```
load_16("min_fin", (min_fin), 35 )
```

Definición en la línea 36 del archivo [nvs.c](#).

6.31.2.9. `load_min_ini`

```
#define load_min_ini(  
    min_ini)
```

Valor:

```
load_16("min_ini", (min_ini), 30 )
```

Definición en la línea 34 del archivo [nvs.c](#).

6.31.2.10. `load_vbus_min`

```
#define load_vbus_min(  
    vbus_min)
```

Valor:

```
load_16("vbus_min", (vbus_min), 310 )
```

Definición en la línea 31 del archivo [nvs.c](#).

6.31.2.11. `save_acceleration`

```
#define save_acceleration(  
    freq_acceleration)
```

Valor:

```
save_16("freq_acce", (freq_acceleration) )
```

Definición en la línea 17 del archivo [nvs.c](#).

6.31.2.12. save_desacceleration

```
#define save_desacceleration(  
    freq_desacceleration)
```

Valor:

```
save_16("freq_desa", (freq_desacceleration) )
```

Definición en la línea 18 del archivo [nvs.c](#).

6.31.2.13. save_frequency

```
#define save_frequency(  
    freq_op)
```

Valor:

```
save_16("freq_freq", (freq_op) )
```

Definición en la línea 16 del archivo [nvs.c](#).

6.31.2.14. save_hour_fin

```
#define save_hour_fin(  
    hour_fin)
```

Valor:

```
save_16("hour_fin", (hour_fin) )
```

Definición en la línea 24 del archivo [nvs.c](#).

6.31.2.15. save_hour_ini

```
#define save_hour_ini(  
    hour_ini)
```

Valor:

```
save_16("hour_ini", (hour_ini) )
```

Definición en la línea 22 del archivo [nvs.c](#).

6.31.2.16. save_ibus_max

```
#define save_ibus_max(  
    ibus_max)
```

Valor:

```
save_16("ibus_max", (ibus_max) )
```

Definición en la línea 21 del archivo [nvs.c](#).

6.31.2.17. `save_input_variable`

```
#define save_input_variable(
    freq_input_variable)
```

Valor:

```
save_16("freq_input", (freq_input_variable) )
```

Definición en la línea 19 del archivo [nvs.c](#).

6.31.2.18. `save_min_fin`

```
#define save_min_fin(
    min_fin)
```

Valor:

```
save_16("min_fin", (min_fin) )
```

Definición en la línea 25 del archivo [nvs.c](#).

6.31.2.19. `save_min_ini`

```
#define save_min_ini(
    min_ini)
```

Valor:

```
save_16("min_ini", (min_ini) )
```

Definición en la línea 23 del archivo [nvs.c](#).

6.31.2.20. `save_vbus_min`

```
#define save_vbus_min(
    vbus_min)
```

Valor:

```
save_16("vbus_min", (vbus_min) )
```

Definición en la línea 20 del archivo [nvs.c](#).

6.31.3. Documentación de funciones

6.31.3.1. `load_16()`

```
esp_err_t load_16 (
    const char * var_tag,
    int16_t * value,
    int16_t defval) [static]
```

Accede a la memoria NVS para obtener un dato de 16 bits guardado en memoria no volatil.

Abre la memoria NVS con el namespace "storage" y intenta leer la variable cuyo namespace es `var_tag` para guardarla en `value`. En caso de error, guarda `defval` como valor default en `value`.

Parámetros

in	<i>var_tag</i>	Nombre del namespace donde se almacena la variable que se está intentando leer. Es un string.
out	<i>value</i>	Puntero donde se guardará el dato que se desea leer de la memoria.
in	<i>defval</i>	Valor default que se utiliza en caso que nunca se haya creado la variable o si existe algún problema al abrir la NVS.

Valores devueltos

- ESP_OK: Si la lectura de la variable fue exitosa
 - ESP_FAIL: si hay un interno error; generalmente ocurre cuando la memoria tiene una partición corrupta.
 - ESP_ERR_NVS_NOT_INITIALIZED: si la memoria no fue inicializada.
 - ESP_ERR_NVS_PART_NOT_FOUND: si la partición con la etiqueta "nvs" no se encuentra.
 - ESP_ERR_NVS_NOT_FOUND: id namespace doesn't exist yet and mode is NVS_READONLY.
 - ESP_ERR_NVS_INVALID_NAME: si el nombre del namespace no cumple con las restricciones.
 - ESP_ERR_NO_MEM: en caso que la memoria no pueda ser guardada por la estructura interna.
 - ESP_ERR_NVS_NOT_ENOUGH_SPACE: Si no hay espacio para una nueva entrada o si hay demasiados namespaces diferentes (max 254).
 - ESP_ERR_NOT_ALLOWED: Si la partición es solo lectura y se abrió en modo NVS_READWRITE.
 - ESP_ERR_INVALID_ARG: Si el handler de la NVS es NULL.

Definición en la línea [203](#) del archivo [nvs.c](#).

6.31.3.2. `load_variables()`

```
esp_err_t load_variables (
    frequency_settings_t * frequency_settings,
    time_settings_t * time_settings,
    security_settings_t * security_settings)
```

Carga desde NVS los parámetros de frecuencia, seguridad y ventanas horarias.

Utiliza las funciones internas de lectura (`load_16`) para obtener cada parámetro persistido en NVS. Si alguna lectura falla, retorna el primer error encontrado. En caso de no existir una clave, se aplica el valor por defecto definido en cada macro de carga.

Parámetros

out	<i>frequency_settings</i>	Estructura de parámetros de frecuencia a cargar.
out	<i>time_settings</i>	Estructura de parámetros de tiempo (ventanas start/stop) a cargar.
out	<i>security_settings</i>	Estructura de parámetros de seguridad (vbus/ibus) a cargar.

Valores devueltos

- ESP_OK: Si todas las lecturas fueron exitosas (o claves ausentes con default aplicado).
 - ESP_FAIL: si hay un error interno.
 - ESP_ERR_NVS_NOT_INITIALIZED: si la NVS no está inicializada.
 - ESP_ERR_NVS_PART_NOT_FOUND: si la partición "nvs" no se encuentra.
 - ESP_ERR_NVS_INVALID_NAME: si alguna clave no cumple restricciones.
 - ESP_ERR_NO_MEM / ESP_ERR_NVS_NOT_ENOUGH_SPACE: errores internos de la NVS.
 - ESP_ERR_NOT_ALLOWED: si la partición es solo lectura.
 - ESP_ERR_INVALID_ARG: si el handler interno de NVS es NULL.

Definición en la línea [245](#) del archivo [nvs.c](#).

6.31.3.3. nvs_init_once()

```
esp_err_t nvs_init_once (
    void )
```

Inicializa la memoria NVS una única vez.

Intenta inicializar la NVS mediante `nvs_flash_init()`. Si la memoria está llena (`ESP_ERR_NVS_NO_FREE_PAGES`) o se detecta una nueva versión de NVS (`ESP_ERR_NVS_NEW_VERSION_FOUND`), se borra la partición NVS y se vuelve a inicializar. No realiza inicialización repetida si ya fue hecha por el sistema.

Valores devueltos

- ESP_OK: Si la inicialización fue exitosa.
 - `ESP_ERR_NVS_NO_FREE_PAGES`: Sin páginas libres; requiere borrado y reinicialización.
 - `ESP_ERR_NVS_NEW_VERSION_FOUND`: Versión de NVS incompatible; requiere borrado.
 - `ESP_ERR_NVS_NOT_INITIALIZED`: si la memoria no pudo inicializarse.
 - `ESP_ERR_NVS_PART_NOT_FOUND`: si la partición con la etiqueta "nvs" no se encuentra.
 - `ESP_ERR_NOT_ALLOWED`: Si la partición es solo lectura.
 - Otros códigos de error propagados por `nvs_flash_init()` o `nvs_flash_erase()`.

Definición en la línea [236](#) del archivo [nvs.c](#).

6.31.3.4. save_16()

```
esp_err_t save_16 (
    const char * var_tag,
    int16_t value) [static]
```

Accede a la memoria NVS para guardar un dato de 16 bits en memoria no volatil.

Abre la memoria NVS con el namespace "storage" y guarda la variable cuyo namespace es `var_tag` para guardarla en `value`.

Parámetros

in	<i>var_tag</i>	Nombre del namespace donde se almacenará la variable que se está intentando guardar. Es un string.
out	<i>value</i>	Puntero donde se guardará el dato que se desea guardar de la memoria.

Valores devueltos

- ESP_OK: Si la escritura de la variable fue exitosa
 - ESP_FAIL: si hay un interno error; generalmente ocurre cuando la memoria tiene una partición corrupta.
 - ESP_ERR_NVS_NOT_INITIALIZED: si la memoria no fue inicializada.
 - ESP_ERR_NVS_PART_NOT_FOUND: si la partición con la etiqueta "nvs" no se encuentra.
 - ESP_ERR_NVS_NOT_FOUND: id namespace doesn't exist yet and mode is NVS_READONLY.
 - ESP_ERR_NVS_INVALID_NAME: si el nombre del namespace no cumple con las restricciones.
 - ESP_ERR_NO_MEM: en caso que la memoria no puda ser guardada por la estructura interna.
 - ESP_ERR_NVS_NOT_ENOUGH_SPACE: Si no hay espacio para una nueva entrada o si hay demasiados namespaces diferentes (max 254).
 - ESP_ERR_NOT_ALLOWED: Si la partición es solo lectura y se abrió en modo NVS_READWRITE.
 - ESP_ERR_INVALID_ARG: Si el handler de la NVS es NULL.

Definición en la línea 104 del archivo [nvs.c](#).

6.31.3.5. save_variables()

```
esp_err_t save_variables (
    frequency_settings_t * frequency_settings,
    time_settings_t * time_settings,
    security_settings_t * security_settings)
```

Guarda en NVS los parámetros de frecuencia, seguridad y ventanas horarias.

Utiliza las funciones internas de guardado (save_16) para persistir cada parámetro en la NVS. Si alguna escritura falla, retorna el primer error encontrado.

Parámetros

in	<i>frequency_settings</i>	Estructura con parámetros de frecuencia a guardar.
in	<i>time_settings</i>	Estructura con parámetros de tiempo (ventanas start/stop) a guardar.
in	<i>security_settings</i>	Estructura con parámetros de seguridad (vbus/ibus) a guardar.

Valores devueltos

- | ESP_OK: Si todas las escrituras fueron exitosas.
 - ESP_FAIL: si hay un error interno.
 - ESP_ERR_NVS_NOT_INITIALIZED: si la NVS no está inicializada.
 - ESP_ERR_NVS_PART_NOT_FOUND: si la partición "nvs" no se encuentra.
 - ESP_ERR_NVS_INVALID_NAME: si alguna clave no cumple restricciones.
 - ESP_ERR_NO_MEM / ESP_ERR_NVS_NOT_ENOUGH_SPACE: errores internos de la NVS o espacio insuficiente.
 - ESP_ERR_NOT_ALLOWED: si la partición es solo lectura.
 - ESP_ERR_INVALID_ARG: si el handler interno de NVS es NULL.

Definición en la línea 309 del archivo [nvs.c](#).

6.31.4. Documentación de variables

6.31.4.1. TAG

```
const char* TAG = "NVS" [static]
```

Definición en la línea 38 del archivo [nvs.c](#).

6.32. nvs.c

[Ir a la documentación de este archivo.](#)

```

00001
00009 #include "nvs_flash.h"
0010 #include "nvs.h"
0011 #include "esp_log.h"
0012
0013 #include "../LVFV_system.h"
0014 #include "./nvs.h"
0015
0016 #define save_frequency(freq_op)
0017 #define save_acceleration(freq_acceleration)
0018 #define save_desacceleration(freq_desacceleration)
0019 #define save_input_variable(freq_input_variable)
0020 #define save_vbus_min(vbus_min)
0021 #define save_ibus_max(ibus_max)
0022 #define save_hour_ini(hour_ini)
0023 #define save_min_ini(min_ini)
0024 #define save_hour_fin(hour_fin)
0025 #define save_min_fin(min_fin)
0026
0027 #define load_frequency(freq_op)
0028 #define load_acceleration(freq_acceleration)
0029 #define load_desacceleration(freq_desacceleration)
0030 #define load_input_variable(freq_input_variable)
0031 #define load_vbus_min(vbus_min)
0032 #define load_ibus_max(ibus_max)
0033 #define load_hour_ini(hour_ini)
0034 #define load_min_ini(min_ini)
0035 #define load_hour_fin(hour_fin)

          save_16("freq_freq", (freq_op) )
          save_16("freq_acce",
          save_16("freq_desa",
          save_16("freq_input",
          save_16("vbus_min", (vbus_min) )
          save_16("ibus_max", (ibus_max) )
          save_16("hour_ini", (hour_ini) )
          save_16("min_ini", (min_ini) )
          save_16("hour_fin", (hour_fin) )
          save_16("min_fin", (min_fin) )

          load_16("freq_freq", (freq_op), 50 )
          load_16("freq_acce",
          load_16("freq_desa",
          load_16("freq_input",
          load_16("vbus_min", (vbus_min), 310 )
          load_16("ibus_max", (ibus_max), 2000 )
          load_16("hour_ini", (hour_ini), 20 )
          load_16("min_ini", (min_ini), 30 )
          load_16("hour_fin", (hour_fin), 20 )

```

```

00036 #define load_min_fin(min_fin)
00037
00038 static const char *TAG = "NVS";
00039
00040 // /**
00041 // * @fn static esp_err_t save_32 (const char *var_tag, int32_t value);
00042 // *
00043 // * @brief Accede a la memoria NVS para guardar un dato de 32 bits en memoria no volatil.
00044 // *
00045 // * @details Abre la memoria NVS con el namespace "storage" y guarda la variable cuyo namespace es
00046 // * @p var_tag para guardarla en @p value.
00047 // *
00048 // *      Nombre del namespace donde se almacenará la variable que se está intentando guardar. Es un
00049 // *      string.
00050 // *      @param[in] var_tag
00051 // *      Puntero donde se guardará el dato que se desea guardar de la memoria.
00052 // *
00053 // *      @retval
00054 // *          - ESP_OK: Si la escritura de la variable fue exitosa
00055 // *          - ESP_FAIL: si hay un interno error; generalmente ocurre cuando la memoria tiene una
00056 // *              partición corrupta.
00057 // *          - ESP_ERR_NVS_NOT_INITIALIZED: si la memoria no fue inicializada.
00058 // *          - ESP_ERR_NVS_PART_NOT_FOUND: si la partición con la etiqueta "nvs" no se encuentra.
00059 // *          - ESP_ERR_NVS_NOT_FOUND: id namespace doesn't exist yet and mode is NVS_READONLY.
00060 // *          - ESP_ERR_NO_MEM: en caso que la memoria no puda ser guardada por la estructura interna.
00061 // *          - ESP_ERR_NVS_NOT_ENOUGH_SPACE: Si no hay espacio para una nueva entrada o si hay demasiados
00062 // *              namespaces diferentes (max 254).
00063 // *          - ESP_ERR_NOT_ALLOWED: Si la partición es solo lectura y se abrió en modo NVS_READWRITE.
00064 // *          - ESP_ERR_INVALID_ARG: Si el handler de la NVS es NULL.
00065 // *      static esp_err_t save_32 (const char *var_tag, int32_t value) {
00066 //     nvs_handle_t h;
00067 //     esp_err_t err;
00068 //     err = nvs_open("storage", NVS_READWRITE, &h);
00069 //     if (err != ESP_OK) {
00070 //         return err;
00071 //     }
00072 //     ESP_LOGI(TAG, "Guardando %s = %ld", var_tag, (long)value);
00073 //     err = nvs_set_i32(h, var_tag, value); // <- i32 correcto
00074 //     if (err == ESP_OK) err = nvs_commit(h);
00075 //     nvs_close(h);
00076 //     return err;
00077 // }
00078
00104 static esp_err_t save_16 (const char *var_tag, int16_t value) {
00105     nvs_handle_t h;
00106     esp_err_t err;
00107
00108     err = nvs_open("storage", NVS_READWRITE, &h);
00109     if (err != ESP_OK) {
00110         return err;
00111     }
00112
00113     ESP_LOGI(TAG, "Guardando %s = %d", var_tag, (int)value);
00114     err = nvs_set_i16(h, var_tag, value);
00115     if (err == ESP_OK) err = nvs_commit(h);
00116
00117     nvs_close(h);
00118     return err;
00119 }
00120
00121 // /**
00122 // * @fn static esp_err_t load_32(const char *var_tag, int32_t *value, int32_t defval);
00123 // *
00124 // * @brief Accede a la memoria NVS para obtener un dato de 32 bits guardado en memoria no volatil.
00125 // *
00126 // * @details Abre la memoria NVS con el namespace "storage" y intenta leer la variable cuyo
00127 // * namespace es @p var_tag para guardarla en @p value. En caso de error, guarda @p defval como valor
00128 // * default en @p value.
00129 // *      @param[in] var_tag
00130 // *      Nombre del namespace donde se almacena la variable que se está intentando leer. Es un
00131 // *      string.
00132 // *      @param[out] value
00133 // *      Puntero donde se guardará el dato que se desea leer de la memoria.
00134 // *      @param[in] defval
00135 // *      Valor default que se utiliza en caso que nunca se haya creado la variable o si existe algún
00136 // *      problema al abrir la NVS.
00137 // *      @retval
00138 // *          - ESP_OK: Si la lectura de la variable fue exitosa
00139 // *          - ESP_FAIL: si hay un interno error; generalmente ocurre cuando la memoria tiene una

```

```

oartición corrupta.
00140 // *      - ESP_ERR_NVS_NOT_INITIALIZED: si la memoria no fue inicializada.
00141 // *      - ESP_ERR_NVS_PART_NOT_FOUND: si la partición con la etiqueta "nvs" no se encuentra.
00142 // *      - ESP_ERR_NVS_NOT_FOUND: id namespace doesn't exist yet and mode is NVS_READONLY.
00143 // *      - ESP_ERR_NVS_INVALID_NAME: si el nombre del namespace no cumple con las restricciones.
00144 // *      - ESP_ERR_NO_MEM: en caso que la memoria no puda ser guardada por la estructura interna.
00145 // *      - ESP_ERR_NVS_NOT_ENOUGH_SPACE: Si no hay espacio para una nueva entrada o si hay demasiados
    namespaces diferentes (max 254).
00146 // *      - ESP_ERR_NOT_ALLOWED: Si la partición es solo lectura y se abrió en modo NVS_READWRITE.
00147 // *      - ESP_ERR_INVALID_ARG: Si el handler de la NVS es NULL.
00148 // */
00149 // static esp_err_t load_32(const char *var_tag, int32_t *value, int32_t defval) {
00150 //     nvs_handle_t h;
00151 //     esp_err_t err;
00152 //     if ( var_tag == NULL ) {
00153 //         return ESP_FAIL;
00154 //     }
00155 //     err = nvs_open("storage", NVS_READWRITE, &h);
00156 //     if (err != ESP_OK) {
00157 //         return err;
00158 //     }
00159 //     if (err != ESP_OK) {
00160 //         *value = defval;
00161 //         return err;
00162 //     }
00163 //     err = nvs_get_i32(h, var_tag, value);
00164 //     if (err == ESP_ERR_NVS_NOT_FOUND) {
00165 //         *value = defval;
00166 //         ESP_LOGW(TAG, "Clave %s no encontrada, usando default=%ld", var_tag, (long)defval);
00167 //         err = ESP_OK;
00168 //     } else if (err == ESP_OK) {
00169 //         ESP_LOGI(TAG, "Cargado %s = %ld", var_tag, (long)*value); // <-- *value
00170 //     }
00171 //     nvs_close(h);
00172 //     return err;
00173 // }
00174
00203 static esp_err_t load_16(const char *var_tag, int16_t *value, int16_t defval) {
00204     nvs_handle_t h;
00205
00206     esp_err_t err;
00207
00208     if ( var_tag == NULL ) {
00209         return ESP_FAIL;
00210     }
00211
00212     err = nvs_open("storage", NVS_READWRITE, &h);
00213
00214     if (err != ESP_OK) {
00215         return err;
00216     }
00217
00218     if (err != ESP_OK) {
00219         *value = defval;
00220         return err;
00221     }
00222
00223     err = nvs_get_i16(h, var_tag, value);
00224     if (err == ESP_ERR_NVS_NOT_FOUND) {
00225         *value = defval;
00226         ESP_LOGW(TAG, "Clave %s no encontrada, usando default=%d", var_tag, (int)defval);
00227         err = ESP_OK;
00228     } else if (err == ESP_OK) {
00229         ESP_LOGI(TAG, "Cargado %s = %d", var_tag, (int)*value); // <-- *value
00230     }
00231
00232     nvs_close(h);
00233     return err;
00234 }
00235
00236 esp_err_t nvs_init_once(void) {
00237     esp_err_t err = nvs_flash_init();
00238     if (err == ESP_ERR_NVS_NO_FREE_PAGES || err == ESP_ERR_NVS_NEW_VERSION_FOUND) {
00239         ESP_ERROR_CHECK(nvs_flash_erase());
00240         err = nvs_flash_init();
00241     }
00242     return err;
00243 }
00244
00245 esp_err_t load_variables(frequency_settings_t *frequency_settings, time_settings_t *time_settings,
    security_settings_t *security_settings) {
00246     esp_err_t retval;
00247     retval = load_frequency( (int16_t*) &(frequency_settings->freq_regime) );
00248     if ( retval != ESP_OK ){
00249         return retval;
00250     }
00251 }
```

```

00252
00253     retval = load_acceleration( (int16_t*) &(frequency_settings->acceleration) );
00254     if ( retval != ESP_OK ){
00255         return retval;
00256     }
00257
00258
00259     retval = load_desacceleration( (int16_t*) &(frequency_settings->desacceleration) );
00260     if ( retval != ESP_OK ){
00261         return retval;
00262     }
00263
00264
00265     retval = load_input_variable( (int16_t*) &(frequency_settings->input_variable) );
00266     if ( retval != ESP_OK ){
00267         return retval;
00268     }
00269
00270
00271     retval = load_vbus_min( (int16_t*) &(seccurity_settings->vbus_min) );
00272     if ( retval != ESP_OK ){
00273         return retval;
00274     }
00275
00276
00277     retval = load_ibus_max( (int16_t*) &(seccurity_settings->ibus_max) );
00278     if ( retval != ESP_OK ){
00279         return retval;
00280     }
00281
00282
00283     retval = load_hour_ini( (int16_t*) &(time_settings->time_start->tm_hour) );
00284     if ( retval != ESP_OK ){
00285         return retval;
00286     }
00287
00288
00289     retval = load_min_ini( (int16_t*) &(time_settings->time_start->tm_min) );
00290     if ( retval != ESP_OK ){
00291         return retval;
00292     }
00293
00294
00295     retval = load_hour_fin( (int16_t*) &(time_settings->time_stop->tm_hour) );
00296     if ( retval != ESP_OK ){
00297         return retval;
00298     }
00299
00300
00301     retval = load_min_fin( (int16_t*) &(time_settings->time_stop->tm_min) );
00302     if ( retval != ESP_OK ){
00303         return retval;
00304     }
00305
00306     return retval;
00307 }
00308
00309 esp_err_t save_variables(frequency_settings_t *frequency_settings, time_settings_t *time_settings,
00310                           seccurity_settings_t *seccurity_settings) {
00311     esp_err_t retval;
00312     retval = save_frequency( (int16_t) frequency_settings->freq_regime);
00313     if ( retval != ESP_OK ){
00314         return retval;
00315     }
00316
00317     retval = save_acceleration( (int16_t) frequency_settings->acceleration);
00318     if ( retval != ESP_OK ){
00319         return retval;
00320     }
00321
00322     retval = save_desacceleration( (int16_t) frequency_settings->desacceleration);
00323     if ( retval != ESP_OK ){
00324         return retval;
00325     }
00326
00327     retval = save_input_variable( (int16_t) frequency_settings->input_variable);
00328     if ( retval != ESP_OK ){
00329         return retval;
00330     }
00331
00332     retval = save_vbus_min( (int16_t) seccurity_settings->vbus_min);
00333     if ( retval != ESP_OK ){
00334         return retval;
00335     }
00336
00337     retval = save_ibus_max( (int16_t) seccurity_settings->ibus_max);
00338     if ( retval != ESP_OK ){

```

```

00338     return retval;
00339 }
00340
00341     retval = save_hour_ini( (int16_t) time_settings->time_start->tm_hour);
00342     if ( retval != ESP_OK ){
00343         return retval;
00344     }
00345
00346     retval = save_min_ini( (int16_t) time_settings->time_start->tm_min);
00347     if ( retval != ESP_OK ){
00348         return retval;
00349     }
00350
00351     retval = save_hour_fin( (int16_t) time_settings->time_stop->tm_hour);
00352     if ( retval != ESP_OK ){
00353         return retval;
00354     }
00355
00356     retval = save_min_fin( (int16_t) time_settings->time_stop->tm_min);
00357     if ( retval != ESP_OK ){
00358         return retval;
00359     }
00360
00361     return retval;
00362 }
```

6.33. Referencia del archivo main/nvs/nvs.h

Declaración de funciones de lectura y escritura de memoria no volátil. Útiles para los reset y no tener que reconfigurar siempre las variables básicas del sistema.

```
#include <stdint.h>
#include "../LVFV_system.h"
```

Funciones

- **esp_err_t nvs_init_once (void)**
Inicializa la memoria NVS una única vez.
- **esp_err_t load_variables (frequency_settings_t *frequency_settings, time_settings_t *time_settings, security_settings_t *seccurity_settings)**
Carga desde NVS los parámetros de frecuencia, seguridad y ventanas horarias.
- **esp_err_t save_variables (frequency_settings_t *frequency_settings, time_settings_t *time_settings, security_settings_t *seccurity_settings)**
Guarda en NVS los parámetros de frecuencia, seguridad y ventanas horarias.

6.33.1. Descripción detallada

Declaración de funciones de lectura y escritura de memoria no volátil. Útiles para los reset y no tener que reconfigurar siempre las variables básicas del sistema.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo [nvs.h](#).

6.33.2. Documentación de funciones

6.33.2.1. load_variables()

```
esp_err_t load_variables (
    frequency_settings_t * frequency_settings,
    time_settings_t * time_settings,
    security_settings_t * seccurity_settings)
```

Carga desde NVS los parámetros de frecuencia, seguridad y ventanas horarias.

Utiliza las funciones internas de lectura (load_16) para obtener cada parámetro persistido en NVS. Si alguna lectura falla, retorna el primer error encontrado. En caso de no existir una clave, se aplica el valor por defecto definido en cada macro de carga.

Parámetros

out	<i>frequency_settings</i>	Estructura de parámetros de frecuencia a cargar.
out	<i>time_settings</i>	Estructura de parámetros de tiempo (ventanas start/stop) a cargar.
out	<i>security_settings</i>	Estructura de parámetros de seguridad (vbus/ibus) a cargar.

Valores devueltos

- ESP_OK: Si todas las lecturas fueron exitosas (o claves ausentes con default aplicado).
 - ESP_FAIL: si hay un error interno.
 - ESP_ERR_NVS_NOT_INITIALIZED: si la NVS no está inicializada.
 - ESP_ERR_NVS_PART_NOT_FOUND: si la partición "nvs" no se encuentra.
 - ESP_ERR_NVS_INVALID_NAME: si alguna clave no cumple restricciones.
 - ESP_ERR_NO_MEM / ESP_ERR_NVS_NOT_ENOUGH_SPACE: errores internos de la NVS.
 - ESP_ERR_NOT_ALLOWED: si la partición es solo lectura.
 - ESP_ERR_INVALID_ARG: si el handler interno de NVS es NULL.

Definición en la línea [245](#) del archivo [nvs.c](#).

6.33.2.2. nvs_init_once()

```
esp_err_t nvs_init_once (
    void )
```

Inicializa la memoria NVS una única vez.

Intenta inicializar la NVS mediante `nvs_flash_init()`. Si la memoria está llena (`ESP_ERR_NVS_NO_FREE_PAGES`) o se detecta una nueva versión de NVS (`ESP_ERR_NVS_NEW_VERSION_FOUND`), se borra la partición NVS y se vuelve a inicializar. No realiza inicialización repetida si ya fue hecha por el sistema.

Valores devueltos

- ESP_OK: Si la inicialización fue exitosa.
 - ESP_ERR_NVS_NO_FREE_PAGES: Sin páginas libres; requiere borrado y reinicialización.
 - ESP_ERR_NVS_NEW_VERSION_FOUND: Versión de NVS incompatible; requiere borrado.
 - ESP_ERR_NVS_NOT_INITIALIZED: si la memoria no pudo inicializarse.
 - ESP_ERR_NVS_PART_NOT_FOUND: si la partición con la etiqueta "nvs" no se encuentra.
 - ESP_ERR_NOT_ALLOWED: Si la partición es solo lectura.
 - Otros códigos de error propagados por nvs_flash_init() o nvs_flash_erase().

Definición en la línea 236 del archivo [nvs.c](#).

6.33.2.3. save_variables()

```
esp_err_t save_variables (
    frequency_settings_t * frequency_settings,
    time_settings_t * time_settings,
    security_settings_t * security_settings)
```

Guarda en NVS los parámetros de frecuencia, seguridad y ventanas horarias.

Utiliza las funciones internas de guardado (save_16) para persistir cada parámetro en la NVS. Si alguna escritura falla, retorna el primer error encontrado.

Parámetros

in	<i>frequency_settings</i>	Estructura con parámetros de frecuencia a guardar.
in	<i>time_settings</i>	Estructura con parámetros de tiempo (ventanas start/stop) a guardar.
in	<i>security_settings</i>	Estructura con parámetros de seguridad (vbus/ibus) a guardar.

Valores devueltos

- ESP_OK: Si todas las escrituras fueron exitosas.
 - ESP_FAIL: si hay un error interno.
 - ESP_ERR_NVS_NOT_INITIALIZED: si la NVS no está inicializada.
 - ESP_ERR_NVS_PART_NOT_FOUND: si la partición "nvs" no se encuentra.
 - ESP_ERR_NVS_INVALID_NAME: si alguna clave no cumple restricciones.
 - ESP_ERR_NO_MEM / ESP_ERR_NVS_NOT_ENOUGH_SPACE: errores internos de la NVS o espacio insuficiente.
 - ESP_ERR_NOT_ALLOWED: si la partición es solo lectura.
 - ESP_ERR_INVALID_ARG: si el handler interno de NVS es NULL.

Definición en la línea 309 del archivo [nvs.c](#).

6.34. nvs.h

[Ir a la documentación de este archivo.](#)

```

00001
00009 #ifndef VARIABLE_ADMIN_H
00010 #define VARIABLE_ADMIN_H
00011
00012 #include <stdint.h>
00013 #include "../LVFV_system.h"
00033 esp_err_t nvs_init_once(void);
00034
00064 esp_err_t load_variables(frequency_settings_t *frequency_settings, time_settings_t *time_settings,
    security_settings_t *seccurity_settings);
00065
00093 esp_err_t save_variables(frequency_settings_t *frequency_settings, time_settings_t *time_settings,
    security_settings_t *seccurity_settings);
00094
00095 #endif

```

6.35. Referencia del archivo main/rtc/rtc.c

Funciones de lectura y escritura del RTC y alarmas.

```

#include <sys/time.h>
#include "sdkconfig.h"
#include "esp_err.h"
#include "esp_timer.h"
#include "esp_log.h"
#include "../system/sysControl.h"
#include "RTC.h"

```

Estructuras de datos

- struct `rtc_alarms_t`

Funciones

- static esp_err_t `program_alarm` (esp_timer_handle_t *timer_handle, void(*callback)(void *), int hour, int minute, const char *name)

Función dedicada a programar la alarma según hour y minute.
- static void `alarm_start_cb` (void *arg)

Función programada como alarma para iniciar el funcionamiento del motor.
- static void `alarm_stop_cb` (void *arg)

Función programada como alarma para finalizar el funcionamiento del motor.
- esp_err_t `initRTC` ()

Inicializa el RTC en la hora 00:00:00.
- esp_err_t `setTime` (struct tm *setting_time)

Inicializa el RTC en la hora cargada en setting_time.
- esp_err_t `getTime` (struct tm *current_timeinfo)

Carga la hora actual desde el RTC del ESP32.
- esp_err_t `rtc_schedule_alarms` (time_settings_t *time_settings)

Inicializa las alarmas de inicio y fin de funcionamiento del motor.

Variables

- static const char * TAG = "RTC"
- static rtc_alarms_t rtc_alarms
- static time_settings_t alarm_settings
- static struct tm time_start
- static struct tm time_stop

6.35.1. Descripción detallada

Funciones de lectura y escritura del RTC y alarmas.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo [rtc.c](#).

6.35.2. Documentación de funciones

6.35.2.1. alarm_start_cb()

```
void alarm_start_cb (
    void * arg) [static]
```

Función programada como alarma para iniciar el funcionamiento del motor.

Envía la señal al sistema para poder iniciar el funcionamiento del motor según la frecuencia de régimen, aceleración configuradas.

Parámetros

in	arg	Sin uso.
----	-----	----------

Definición en la línea [212](#) del archivo [rtc.c](#).

6.35.2.2. alarm_stop_cb()

```
void alarm_stop_cb (
    void * arg) [static]
```

Función programada como alarma para finalizar el funcionamiento del motor.

Envía la señal al sistema para poder finalizar el funcionamiento del motor según la desaceleración configurada.

Parámetros

in	<i>arg</i>	Sin uso.
----	------------	----------

Definición en la línea 233 del archivo [rtc.c](#).

6.35.2.3. `getTime()`

```
esp_err_t getTime (
    struct tm * current_timeinfo)
```

Carga la hora actuak desde el RTC del ESP32.

Lee la hora del RTC del ESP32 expresada en unix time y la convierte en un formato entendible por el usuario en la estructura `current_timeinfo`

Parámetros

in	<i>current_timeinfo</i>	Esturctura tm en donde se cargará la fecha y hora del sistema.
----	-------------------------	--

Devuelve

- `ESP_OK` si carga la hora exitosamente.
- `ESP_ERR_INVALID_ARG` si `current_timeinfo` es `NULL`

Definición en la línea 124 del archivo [rtc.c](#).

6.35.2.4. `initRTC()`

```
esp_err_t initRTC ()
```

Inicializa el RTC en la hora 00:00:00.

No es importante el día configurado para el sistema, pero carga la hora 0 de UNIX_TIME.

Devuelve

- `ESP_OK` siempre.

Definición en la línea 92 del archivo [rtc.c](#).

6.35.2.5. `program_alarm()`

```
esp_err_t program_alarm (
    esp_timer_handle_t * timer_handle,
    void(* callback )(void *),
    int hour,
    int minute,
    const char * name) [static]
```

Función dedicada a programar la alarma según hour y minute.

No distingue de inicio o fin de marcha, para ello se le pasa el puntero a función 'callback' para la función de inicio o fin de marcha, el handler 'timer_handler' y un nombre descriptivo. Los handlers de los timers son creados como tareas y no como interrupciones para poder utilizar las funciones .

Parámetros

out	<i>timer_handle</i>	Puntero de salida donde se devuelve el handle del timer creado. Debe ser no-NULL. El caller es responsable de detener y destruir el timer cuando ya no se utilice (<code>esp_timer_stop</code> / <code>esp_timer_delete</code>).
in	<i>callback</i>	Puntero a función de tipo <code>void (*) (void *)</code> que se invocará cuando la alarma dispare. Se ejecuta en el contexto de la tarea interna de <code>esp_timer</code> (no en ISR). Puede usar APIs no-ISR (colas, logs, etc.).
in	<i>hour</i>	Hora en formato 24 h. Rango válido: 0–23.
in	<i>minute</i>	Minutos. Rango válido: 0–59.
in	<i>name</i>	Nombre descriptivo del timer (aparece en logs/diagnóstico). Debe apuntar a memoria de duración estática (p. ej., literal o <code>static const</code>).

Devuelve

- `ESP_OK` en caso de éxito.
- `ESP_ERR_INVALID_ARG` si `timer_handle` es `NULL`, `callback` es `NULL` o `hour/minute` están fuera de rango.
- `ESP_ERR_INVALID_STATE` Si el RTC aún no fue inicializado
- `ESP_ERR_NO_MEM` si laallocación de memoria para los handlers falla

Definición en la línea 171 del archivo [rtc.c](#).

6.35.2.6. `rtc_schedule_alarms()`

```
esp_err_t rtc_schedule_alarms (
    time_settings_t * time_settings)
```

Inicializa las alarmas de inicio y fin de funcionamiento del motor.

La hora y minuto cargada en `time_start` es la que dará inicio al funcionamiento del motor, mientras que la cargada en `time_stop` será la que finalizará el mismo.

Parámetros

in	<i>time_settings</i>	Esturctura time_settings_t que contiene los horarios de alarma de inicio y fin. Además contiene la hora actual, una variable que no es utilizada por la función.
----	----------------------	--

Devuelve

- `ESP_OK` si Configura las alarmas exitosamente.
- `ESP_FAIL` si falla en la creación de las alarmas.

Definición en la línea 137 del archivo [rtc.c](#).

6.35.2.7. `setTime()`

```
esp_err_t setTime (
    struct tm * setting_time)
```

Inicializa el RTC en la hora cargada en `setting_time`.

No es importante el día configurado para el sistema, pero carga la hora 0 de `UNIX_TIME`.

Parámetros

in	<i>setting_time</i>	Esturctura tm que es cargada en el RTC del ESP32 según su año, mes, día, hora, minuto y segundo.
----	---------------------	--

Devuelve

- ESP_OK si carga el horario correctamente.
- ESP_ERR_INVALID_ARG en caso de recibir un puntero NULL

Definición en la línea 107 del archivo [rtc.c](#).

6.35.3. Documentación de variables

6.35.3.1. alarm_settings

```
time_settings_t alarm_settings [static]
```

Definición en la línea 31 del archivo [rtc.c](#).

6.35.3.2. rtc_alarms

```
rtc_alarms_t rtc_alarms [static]
```

Definición en la línea 30 del archivo [rtc.c](#).

6.35.3.3. TAG

```
const char* TAG = "RTC" [static]
```

Definición en la línea 29 del archivo [rtc.c](#).

6.35.3.4. time_start

```
struct tm time_start [static]
```

Definición en la línea 33 del archivo [rtc.c](#).

6.35.3.5. time_stop

```
struct tm time_stop [static]
```

Definición en la línea 34 del archivo [rtc.c](#).

6.36. rtc.c

[Ir a la documentación de este archivo.](#)

```

00001
00009 #include <sys/time.h>
00010
00011 #include "sdkconfig.h"
00012 #include "esp_err.h"
00013 #include "esp_timer.h"
00014 #include "esp_log.h"
00015
00016 #include "../system/sysControl.h"
00017 #include "RTC.h"
00018
00024 typedef struct {
00025     esp_timer_handle_t start_timer;
00026     esp_timer_handle_t stop_timer;
00027 } rtc_alarms_t;
00028
00029 static const char *TAG = "RTC";
00030 static mensajes de logs
mensajes de logs
00030 static rtc_alarms_t rtc_alarms;
00031     de alarma para inicio y fin de marcha
00031 static time_settings_t alarm_settings;
00032     // Settings de horarios de inicio y fin de
00032     // marcha de motor
00033 static struct tm time_start;
00034 static struct tm time_stop;
00035
00066 static esp_err_t program_alarm(esp_timer_handle_t *timer_handle, void (*callback)(void *), int hour,
00066     int minute, const char *name);
00067
00078 static void alarm_start_cb(void *arg);
00079
00090 static void alarm_stop_cb(void *arg);
00091
00092 esp_err_t initRTC() {
00093     struct timeval tv = {
00094         .tv_sec = 0,
00095         .tv_usec = 0
00096     };
00097
00098     settimeofday(&tv, NULL); // Cargar en RTC interno
00099
00100     alarm_settings.time_start = &time_start;
00101     alarm_settings.time_stop = &time_stop;
00102
00104     return ESP_OK;
00105 }
00106
00107 esp_err_t setTime(struct tm *setting_time) {
00108     if ( setting_time == NULL ) {
00109         return ESP_ERR_INVALID_ARG;
00110     }
00112
00113     time_t now = mktime(setting_time); // Convierte a timestamp UNIX
00114
00115     struct timeval tv = {
00116         .tv_sec = now,
00117         .tv_usec = 0
00118     };
00119
00120     settimeofday(&tv, NULL); // Cargar en RTC interno
00121     return ESP_OK;
00122 }
00123
00124 esp_err_t getTime(struct tm *current_timeinfo) {
00125     time_t now;
00126
00127     if ( current_timeinfo == NULL ) {
00128         return ESP_ERR_INVALID_ARG;
00129     }
00130
00131     time(&now);
00132     localtime_r(&now, current_timeinfo);
00133
00134     return ESP_OK;
00135 }
00136
00137 esp_err_t rtc_schedule_alarms(time_settings_t *time_settings) {
00138
00139     if ( time_settings != NULL ) {
00140         alarm_settings.time_start->tm_hour = time_settings->time_start->tm_hour;
00140         alarm_settings.time_stop->tm_hour = time_settings->time_stop->tm_hour;
00141     }
00142
00143     alarm_start_cb(alarm_settings.time_start);
00144
00145     esp_timer_start_once(alarm_settings.start_timer, alarm_settings.time_start->tm_sec);
00146
00147     alarm_stop_cb(alarm_settings.time_stop);
00148
00149     esp_timer_start_once(alarm_settings.stop_timer, alarm_settings.time_stop->tm_sec);
00150
00151     return ESP_OK;
00152 }
00153
00154
00155
00156
00157
00158
00159
00160
00161
00162
00163
00164
00165
00166
00167
00168
00169
00170
00171
00172
00173
00174
00175
00176
00177
00178
00179
00180
00181
00182
00183
00184
00185
00186
00187
00188
00189
00190
00191
00192
00193
00194
00195
00196
00197
00198
00199
00200
00201
00202
00203
00204
00205
00206
00207
00208
00209
00210
00211
00212
00213
00214
00215
00216
00217
00218
00219
00220
00221
00222
00223
00224
00225
00226
00227
00228
00229
00230
00231
00232
00233
00234
00235
00236
00237
00238
00239
00240
00241
00242
00243
00244
00245
00246
00247
00248
00249
00250
00251
00252
00253
00254
00255
00256
00257
00258
00259
00260
00261
00262
00263
00264
00265
00266
00267
00268
00269
00270
00271
00272
00273
00274
00275
00276
00277
00278
00279
00280
00281
00282
00283
00284
00285
00286
00287
00288
00289
00290
00291
00292
00293
00294
00295
00296
00297
00298
00299
00300
00301
00302
00303
00304
00305
00306
00307
00308
00309
00310
00311
00312
00313
00314
00315
00316
00317
00318
00319
00320
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01839
01840
01841
01842
01843
01844
01845
01846
01847
01848
01849
01849
01850
01851
01852
01853
01854
01855
01856
01857
01858
01859
01859
01860
01861
01862
01863
01864
01865
01866
01867
01868
01869
01869
01870
01871
01872
01873
01874
01875
01876
01877
01878
01879
01879
01880
01881
01882
01883
01884
01885
01886
01887
01888
01889
01889
01890
01891
01892
01893
01894
01895
01896
01897
01898
01899
01899
01900
01901
01902
01903
01904
01905
01906
01907
01908
01909
01909
01910
01911
01912
01
```

```

00141     alarm_settings.time_start->tm_min = time_settings->time_start->tm_min;
00142     alarm_settings.time_stop->tm_hour = time_settings->time_stop->tm_hour;
00143     alarm_settings.time_stop->tm_min = time_settings->time_stop->tm_min;
00144 }
00145
00146 // Cancelo las anteriores si existían
00147 if (rtc_alarms.start_timer) {
00148     esp_timer_stop(rtc_alarms.start_timer);
00149 }
00150 // Programo las nuevas
00151 esp_err_t err1 = program_alarm(&rtc_alarms.start_timer, alarm_start_cb,
alarm_settings.time_start->tm_hour, alarm_settings.time_start->tm_min, "start_alarm");
00152
00153 // Cancelo las anteriores si existían
00154 if (rtc_alarms.stop_timer) {
00155     esp_timer_stop(rtc_alarms.stop_timer);
00156 }
00157 // Programo las nuevas
00158 esp_err_t err2 = program_alarm(&rtc_alarms.stop_timer, alarm_stop_cb,
alarm_settings.time_stop->tm_hour, alarm_settings.time_stop->tm_min, "stop_alarm");
00159
00160 if (err1 == ESP_OK && err2 == ESP_OK){
00161     ESP_LOGI(TAG, "Configurando alarmas: INICIO%02d: %02d", alarm_settings.time_start->tm_hour,
alarm_settings.time_start->tm_min);
00162     ESP_LOGI(TAG, " : FIN %02d: %02d", alarm_settings.time_stop->tm_hour,
alarm_settings.time_stop->tm_min);
00163     return ESP_OK;
00164 } else {
00165     ESP_LOGE(TAG, "ERROR AL CONFIGURAR LAS ALARMAS");
00166     return ESP_FAIL;
00167 }
00168 }
00169 }
00170
00171 static esp_err_t program_alarm(esp_timer_handle_t *timer_handle, void (*callback)(void *), int hour,
int minute, const char *name) {
00172     time_t now;
00173     struct tm timeinfo_alarm;
00174     time(&now);
00175     localtime_r(&now, &timeinfo_alarm);
00176
00177     struct tm alarm_tm = timeinfo_alarm;
00178     alarm_tm.tm_hour = hour;
00179     alarm_tm.tm_min = minute;
00180     alarm_tm.tm_sec = 0;
00181
00182     time_t alarm_epoch = mktime(&alarm_tm);
00183     if (alarm_epoch <= now) {
00184         alarm_epoch += 24 * 3600;
00185     }
00186
00187     int64_t usec_until_alarm = (alarm_epoch - now) * 1000000LL;
00188
00189     esp_timer_create_args_t timer_args = {
00190         .callback = callback,
00191         .arg = NULL,
00192         .dispatch_method = ESP_TIMER_TASK,
00193         .name = name
00194     };
00195
00196     esp_err_t err = esp_timer_create(&timer_args, timer_handle);
00197     if (err != ESP_OK) {
00198         return err;
00199     }
00200
00201     err = esp_timer_start_once(*timer_handle, usec_until_alarm);
00202     if (err != ESP_OK) {
00203         return err;
00204     }
00205
00206     ESP_LOGI(TAG, "Programada %s para %02d: %02d (en %lld segundos)", name, hour, minute, (alarm_epoch -
now));
00207
00208     return ESP_OK;
00209 }
00210
00211 // Callback para inicio de tarea
00212 static void alarm_start_cb(void *arg) {
00213     ESP_LOGI(TAG, "Alarma de INICIO disparada");
00214
00215     // Cancelo las anteriores si existían
00216     if (rtc_alarms.start_timer) {
00217         esp_timer_stop(rtc_alarms.start_timer);
00218     }
00219     // Programo las nuevas
00220     ESP_ERROR_CHECK(program_alarm(&rtc_alarms.start_timer, alarm_start_cb,
alarm_settings.time_start->tm_hour, alarm_settings.time_start->tm_min, "start_alarm")));

```

```

00221     // Cancelo las anteriores si existían
00222     if (rtc_alarms.stop_timer) {
00223         esp_timer_stop(rtc_alarms.stop_timer);
00224     }
00225     // Programo las nuevas
00226     ESP_ERROR_CHECK(program_alarm(&rtc_alarms.stop_timer, alarm_stop_cb,
00227     alarm_settings.time_stop->tm_hour, alarm_settings.time_stop->tm_min, "stop_alarm"));
00228
00229     SystemEventPost(START_PRESSED);
00230 }
00231
00232 // Callback para fin de tarea
00233 static void alarm_stop_cb(void *arg) {
00234     ESP_LOGI(TAG, "Alarma de FIN disparada");
00235
00236     // Cancelo las anteriores si existían
00237     if (rtc_alarms.start_timer) {
00238         esp_timer_stop(rtc_alarms.start_timer);
00239     }
00240     // Programo las nuevas
00241     ESP_ERROR_CHECK(program_alarm(&rtc_alarms.start_timer, alarm_start_cb,
00242     alarm_settings.time_start->tm_hour, alarm_settings.time_start->tm_min, "start_alarm"));
00243
00244     // Cancelo las anteriores si existían
00245     if (rtc_alarms.stop_timer) {
00246         esp_timer_stop(rtc_alarms.stop_timer);
00247     }
00248     // Programo las nuevas
00249     ESP_ERROR_CHECK(program_alarm(&rtc_alarms.stop_timer, alarm_stop_cb,
00250     alarm_settings.time_stop->tm_hour, alarm_settings.time_stop->tm_min, "stop_alarm"));
00251 }

```

6.37. Referencia del archivo main/rtc/rtc.h

Header con prototipos de funciones básicas para el lectura y escritura del RTC y alarmas.

```
#include "../LVFV_system.h"
#include "esp_err.h"
```

Funciones

- **esp_err_t initRTC ()**
Inicializa el RTC en la hora 00:00:00.
- **esp_err_t setTime (struct tm *setting_time)**
Inicializa el RTC en la hora cargada en setting_time.
- **esp_err_t rtc_schedule_alarms (time_settings_t *time_settings)**
Inicializa las alarmas de inicio y fin de funcionamiento del motor.
- **esp_err_t getTime (struct tm *current_timeinfo)**
Carga la hora actual desde el RTC del ESP32.

6.37.1. Descripción detallada

Header con prototipos de funciones básicas para el lectura y escritura del RTC y alarmas.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo [rtc.h](#).

6.37.2. Documentación de funciones

6.37.2.1. getTime()

```
esp_err_t getTime (
    struct tm * current_timeinfo)
```

Carga la hora actuak desde el RTC del ESP32.

Lee la hora del RTC del ESP32 expresada en unix time y la convierte en un formato entendible por el usuario en la estructura `current_timeinfo`

Parámetros

in	<code>current_timeinfo</code>	Esturctura tm en donde se cargará la fecha y hora del sistema.
----	-------------------------------	--

Devuelve

- `ESP_OK` si carga la hora exitosamente.
- `ESP_ERR_INVALID_ARG` si `current_timeinfo` es `NULL`

Definición en la línea 124 del archivo [rtc.c](#).

6.37.2.2. initRTC()

```
esp_err_t initRTC ()
```

Inicializa el RTC en la hora 00:00:00.

No es importante el día configurado para el sistema, pero carga la hora 0 de UNIX_TIME.

Devuelve

- `ESP_OK` siempre.

Definición en la línea 92 del archivo [rtc.c](#).

6.37.2.3. rtc_schedule_alarms()

```
esp_err_t rtc_schedule_alarms (
    time_settings_t * time_settings)
```

Inicializa las alarmas de inicio y fin de funcionamiento del motor.

La hora y minuto cargada en `time_start` es la que dará inicio al funcionamiento del motor, mientas que la cargada en `time_stop` será la que finalizará el mismo.

Parámetros

in	<i>time_settings</i>	Esturctura <code>time_settings_t</code> que contiene los horarios de alarma de inicio y fin. Además contiene la hora actual, una variable que no es utilizada por la función.
----	----------------------	---

Devuelve

- `ESP_OK` si Configura las alarmas exitosamente.
- `ESP_FAIL` si falla en la creación de las alarmas.

Definición en la línea 137 del archivo [rtc.c](#).

6.37.2.4. `setTime()`

```
esp_err_t setTime (
    struct tm * setting_time)
```

Inicializa el RTC en la hora cargada en `setting_time`.

No es importante el día configurado para el sistema, pero carga la hora 0 de `UNIX_TIME`.

Parámetros

in	<i>setting_time</i>	Esturctura <code>tm</code> que es cargada en el RTC del ESP32 según su año, mes, día, hora, minuto y segundo.
----	---------------------	---

Devuelve

- `ESP_OK` si carga el horario correctamente.
- `ESP_ERR_INVALID_ARG` en caso de recibir un puntero NULL

Definición en la línea 107 del archivo [rtc.c](#).

6.38. rtc.h

[Ir a la documentación de este archivo.](#)

```
00001
00009 #ifndef __RTC_H__
00010
00011 #define __RTC_H__
00012
00013 #include "../LVFV_system.h"
00014 #include "esp_err.h"
00015
00026 esp_err_t initRTC();
00027
00042 esp_err_t setTime(struct tm *setting_time);
00043
00058 esp_err_t rtc_schedule_alarms(time_settings_t *time_settings);
00059
00074 esp_err_t getTime(struct tm *current_timeinfo);
00075
00076 #endif
```

6.39. Referencia del archivo main/system/sysAdmin.c

Archivo de funciones que permiten controlar las variables fundamentales del sistema que controla el motor, solo para poder ser representadas en el display. Las variables reales de sistema se encuentran en el STM32.

```
#include "esp_log.h"
#include "esp_err.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "esp_system.h"
#include "SysAdmin.h"
#include "SysControl.h"
#include "../display/display.h"
```

Funciones

- static void **accelerating** (void *pvParameters)

Tarea dedicada a incrementar la frecuencia que es impresa en el display de acuerdo a la aceleración y hasta la frecuencia de régimen.
- static void **desaccelerating** (void *pvParameters)

Tarea dedicada a decrementar la frecuencia que es impresa en el display de acuerdo a la desaceleración y hasta la frecuencia de régimen o 0Hz.
- uint16_t **engine_start** ()

Función que setea la frecuencia de régimen de acuerdo a lo configurado por el usuario y ejecuta la tarea accelerating.
- uint16_t **engine_stop** ()

Función que setea la frecuencia de régimen de acuerdo a lo configurado por el usuario y ejecuta la tarea desaccelerating.
- bool **engine_emergency_stop_release** (uint8_t signal)

Pasa a estado SYSTEM_EMERGENCY_OK si todas las señales de emergencia son 0 para que el usuario pueda enviar la señal de stop al STM32 necesaria para poder arrancar nuevamente el motor.
- void **engine_emergency_stop** (uint8_t signal)

Pasa la frecuencia de régimen a 0Hz y pasa a estado SYSTEM_EMERGENCY.
- uint16_t **change_frequency** (uint8_t speed_selector)

Cambia la frecuencias de régimen del motor de acuerdo a la entrada aislada que haya ingresado. Ejecuta accelerating o desaccelerating, de acuerdo a lo necesario.
- esp_err_t **get_status** (system_status_t *s_e)

Obtiene una réplica del status del sistema.
- system_status_t **update_meas** (uint16_t vbus_meas, uint16_t ibus_meas)

Actualiza las mediciones de tensión y corriente del bus de continua.
- void **set_frequency_table** (uint16_t input_variable, uint16_t freq_regime)

Carga en el vector de velocidades configurables por entradas, las diferentes frecuencias de trabajo.
- void **set_system_settings** (frequency_settings_t *f_s, security_settings_t *s_s)

Actualiza las variables de seguridad del sistema.

Variables

- static const char * **TAG** = "sysAdmin"
- static system_status_t **system_status**
- static security_settings_t **system_security_settings**
- static uint16_t **frequency_table** [8]
- static TaskHandle_t **accelerating_handle** = NULL
- static TaskHandle_t **desaccelerating_handle** = NULL

6.39.1. Descripción detallada

Archivo de funciones que permiten controlar las variables fundamentales del sistema que controla el motor, solo para poder ser representadas en el display. Las variables reales de sistema se encuentran en el STM32.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo [sysAdmin.c](#).

6.39.2. Documentación de funciones

6.39.2.1. accelerating()

```
void accelerating (
    void * pvParameters) [static]
```

Tareas dedicada a incrementar la frecuencia que es impresa en el display de acuerdo a la aceleración y hasta la frecuencia de regimen.

Es una tarea que debe ser creada cada vez que se necesita incrementar la frecuencia impresa ya que, al terminar de llegar a régimen, termina su propia ejecución. Al terminar su ejecución pasa de SYSTEM_ACCLERATE a SYSTEM_REGIME

Parámetros

in	<i>pvParameters</i>	Sin uso
----	---------------------	---------

Definición en la línea [53](#) del archivo [sysAdmin.c](#).

6.39.2.2. change_frequency()

```
uint16_t change_frequency (
    uint8_t speed_selector)
```

Cambia la frecuencias de régimen del motor de acuerdo a la entrada aislada que haya ingresado. Ejecuta accelerating o desaccelerating, de acuerdo a lo necesario.

De acuerdo a la frecuencia de régimen elegida y el tipo de variación, la frecuencia de destino tendrá diferentes valores, siempre menores a las de régimen.

Parámetros

in	<i>speed_selector</i>	Índice dentro del array con frecuencias de trabajo
----	-----------------------	--

Devuelve

0Hz si *speed_selector* fue mal pasada como argumento, sino la frecuencia de destino

Definición en la línea 151 del archivo [sysAdmin.c](#).

6.39.2.3. desaccelerating()

```
void desaccelerating (
    void * pvParameters) [static]
```

Tareas dedicada a decrementar la frecuencia que es impresa en el display de acuerdo a la desaceleración y hasta la frecuencia de regimen o 0Hz.

Es una tarea que debe ser creada cada vez que se necesita decrementar la frecuencia impresa ya que, al terminar de llegar a régimen, termina su propia ejecución. Al terminar su ejecución pasa de SYSTEM_BREAKING a SYSTEM_REGIME o de SYSTEM_ACCEL DESACCEL a SYSTEM_REGIME. En caso de estar en SYSTEM_EMERGENCY no cambia el status

Parámetros

in	<i>pvParameters</i>	Sin uso
----	---------------------	---------

Definición en la línea 69 del archivo [sysAdmin.c](#).

6.39.2.4. engine_emergency_stop()

```
void engine_emergency_stop (
    uint8_t signal)
```

Pasa la frecuencia de regimen a 0Hz y pasa a estado SYSTEM_EMERGENCY.

Termina las tareas accelerating y desaccelerating y guarda la señal que dispara la emergencia para contabilizar todos las señales vigentes

Parámetros

in	<i>signal</i>	Señal que genera el estado de emergencia
----	---------------	--

Definición en la línea 136 del archivo [sysAdmin.c](#).

6.39.2.5. engine_emergency_stop_release()

```
bool engine_emergency_stop_release (
    uint8_t signal)
```

Pasa a estado SYSTEM_EMERGENCY_OK si todas las señales de emergencia son 0 para que el usuario pueda enviar la señal de stop al STM32 necesaria para poder arrancar nuevamente el motor.

Parámetros

in	<i>signal</i>	Señal que normaliza el estado de emergencia
----	---------------	---

Valores devueltos

true	si cambia a SYSTEM_EMERGENCY_OK; false si existe alguna señal aún activa
------	--

Definición en la línea 124 del archivo [sysAdmin.c](#).

6.39.2.6. engine_start()

```
uint16_t engine_start ()
```

Función que setea la frecuencia de régimen de acuerdo a lo configurado por el usuario y ejecuta la tarea accelerating.

En caso de que desaccelerating esté corriendo, antes de ejecutar accelerating, la cierra para no generar un efecto de subida y bajada de frecuencia. El status pasará a SYSTEM_ACCEL_DESACCEL

Devuelve

Frecuencia de destino configurada

Definición en la línea 93 del archivo [sysAdmin.c](#).

6.39.2.7. engine_stop()

```
uint16_t engine_stop ()
```

Función que setea la frecuencia de régimen de acuerdo a lo configurado por el usuario y ejecuta la tarea desaccelerating.

En caso de que accelerating esté corriendo, antes de ejecutar desaccelerating, la cierra para no generar un efecto de subida y bajada de frecuencia. El status pasará a SYSTEM_ACCEL_DESACCEL o SYSTEM_BREAKING de acuerdo a la acción que haya ocurrido

Devuelve

Frecuencia de destino configurada

Definición en la línea 106 del archivo [sysAdmin.c](#).

6.39.2.8. get_status()

```
esp_err_t get_status (
    system_status_t * s_e)
```

Obtiene una réplica del status del sistema.

Parámetros

<code>out</code>	<code>s_e</code>	Puntero a la variable donde se devolverá la información de status
------------------	------------------	---

Devuelve

- `ESP_ERR_NOT_ALLOWED`: En caso de que `s_e` sea un puntero a NULL
- `ESP_OK`: Si la copia fue exitosa

Definición en la línea 185 del archivo [sysAdmin.c](#).

6.39.2.9. `set_frequency_table()`

```
void set_frequency_table (
    uint16_t input_variable,
    uint16_t freq_regime)
```

Carga en el vector de velocidades configurables por entradas, las diferentes frecuencias de trabajo.

La posición 0 del buffer es la frecuencia de regimen, la posición 7 es la frecuencia más baja distinta de cero

Parámetros

<code>in</code>	<code>input_variable</code>	Tipo de variación entre las diferentes entradas. 1 para variación lineal; 2 para variación cuadrática.
<code>in</code>	<code>freq_regime</code>	Frecuencia de regimen ingresada por el usuario

Definición en la línea 229 del archivo [sysAdmin.c](#).

6.39.2.10. `set_system_settings()`

```
void set_system_settings (
    frequency_settings_t * f_s,
    security_settings_t * s_s)
```

Actualiza las variables de seguridad del sistema.

Parámetros

<code>in</code>	<code>f_s</code>	Puntero a la estructura con las variables de frecuencia a actualizar
<code>in</code>	<code>s_s</code>	Puntero a la estructura con las variables de seguridad a actualizar

Definición en la línea 251 del archivo [sysAdmin.c](#).

6.39.2.11. update_meas()

```
system_status_e update_meas (
    uint16_t vbus_meas,
    uint16_t ibus_meas)
```

Actualiza las mediciones de tensión y corriente del bus de continua.

En caso que superar los límites configurados, entra en SYSTEM_EMERGENCY

Parámetros

in	<i>vbus_meas</i>	<ul style="list-style-type: none"> Tensión de bus de continua leído por el ADC
in	<i>ibus_meas</i>	<ul style="list-style-type: none"> Corriente de bus de continua leído por el ADC

Devuelve

- SYSTEM_IDLE: El sistema se encuentra en estado de reposo
- SYSTEM_ACCEL_DESACCEL: El sistema se encuentra acelerando o desacelerando
- SYSTEM_REGIME: El sistema está con el motor girando a régimen
- SYSTEM_BREAKING: El sistema está frenando
- SYSTEM_EMERGENCY: El sistema entra por primera vez en estado de emergencia

Definición en la línea [197](#) del archivo [sysAdmin.c](#).

6.39.3. Documentación de variables

6.39.3.1. accelerating_handle

```
TaskHandle_t accelerating_handle = NULL [static]
```

Definición en la línea [26](#) del archivo [sysAdmin.c](#).

6.39.3.2. desaccelerating_handle

```
TaskHandle_t desaccelerating_handle = NULL [static]
```

Definición en la línea [27](#) del archivo [sysAdmin.c](#).

6.39.3.3. frequency_table

```
uint16_t frequency_table[8] [static]
```

Definición en la línea [25](#) del archivo [sysAdmin.c](#).

6.39.3.4. system_seccurity_settings

```
seccurity_settings_t system_seccurity_settings [static]
```

Definición en la línea [24](#) del archivo [sysAdmin.c](#).

6.39.3.5. system_status

```
system_status_t system_status [static]
```

Definición en la línea 23 del archivo [sysAdmin.c](#).

6.39.3.6. TAG

```
const char* TAG = "sysAdmin" [static]
```

Definición en la línea 22 del archivo [sysAdmin.c](#).

6.40. sysAdmin.c

[Ir a la documentación de este archivo.](#)

```
00001
00009 #include "esp_log.h"
00010 #include "esp_err.h"
00011
00012 #include "freertos/FreeRTOS.h"
00013 #include "freertos/task.h"
00014 #include "freertos/queue.h"
00015
00016 #include "esp_system.h"
00017
00018 #include "SysAdmin.h"
00019 #include "SysControl.h"
00020 #include "../display/display.h"
00021
00022 static const char *TAG = "sysAdmin";
00023 static system_status_t system_status;
00024 static security_settings_t system_seccurity_settings;
00025 static uint16_t frequency_table[8];
00026 static TaskHandle_t accelerating_handle = NULL;
00027 static TaskHandle_t desaccelerating_handle = NULL;
00028
00029 static void accelerating(void *pvParameters );
00040
00051 static void desaccelerating( void *pvParameters );
00052
00053 static void accelerating(void *pvParameters ) {
00054     // uint16_t acceleration = *((uint16_t*)pvParameters);
00055     vTaskDelay(pdMS_TO_TICKS(200));
00056     while( system_status.frequency != system_status.frequency_destiny ) {
00057         if ( (system_status.frequency + system_status.acceleration) > system_status.frequency_destiny
00058             ) {
00059             system_status.frequency = system_status.frequency_destiny;
00060         } else {
00061             system_status.frequency += system_status.acceleration;
00062         }
00063         vTaskDelay(pdMS_TO_TICKS(1000));
00064     }
00065     system_status.status = SYSTEM_REGIME;
00066     accelerating_handle = NULL;
00067     vTaskDelete(NULL);
00068
00069 static void desaccelerating( void *pvParameters ) {
00070     // uint16_t desacceleration = *((uint16_t*)pvParameters);
00071     vTaskDelay(pdMS_TO_TICKS(200));
00072     while( system_status.frequency != system_status.frequency_destiny ) {
00073         if ( system_status.frequency - system_status.desacceleration < system_status.frequency_destiny
00074             ) {
00075             system_status.frequency = system_status.frequency_destiny;
00076         } else if ( system_status.frequency > system_status.desacceleration ) {
00077             system_status.frequency -= system_status.desacceleration;
00078         } else {
00079             system_status.frequency = 0;
00080         }
00081         vTaskDelay(pdMS_TO_TICKS(1000));
00082     }
00082     if ( system_status.status != SYSTEM_EMERGENCY ) {
```

```

00083     if( system_status.status != SYSTEM_BREAKING ) {
00084         system_status.status = SYSTEM_REGIME;
00085     } else {
00086         system_status.status = SYSTEM_IDLE;
00087     }
00088 }
00089 desaccelerating_handle = NULL;
00090 vTaskDelete(NULL);
00091 }
00092
00093 uint16_t engine_start() {
00094     if ( system_status.status != SYSTEM_EMERGENCY ) {
00095         system_status.frequency_destiny = frequency_table[system_status.inputs_status];
00096         system_status.status = SYSTEM_ACCEL_DESACCEL;
00097         if ( desaccelerating_handle != NULL ) {
00098             vTaskDelete( desaccelerating_handle );
00099             desaccelerating_handle = NULL;
00100         }
00101         xTaskCreatePinnedToCore(accelerating, "accelerating", 1024, (void*)
00102             &system_status.acceleration, 9, &accelerating_handle, 1);
00103     }
00104     return system_status.frequency_destiny;
00105 }
00106 uint16_t engine_stop() {
00107     if ( system_status.status == SYSTEM_EMERGENCY ) {
00108     } else if ( system_status.status == SYSTEM_EMERGENCY_OK ) {
00109         system_status.status = SYSTEM_IDLE;
00110     } else {
00111         system_status.acceleration = get_system_acceleration();
00112         system_status.desacceleration = get_system_desacceleration();
00113         system_status.frequency_destiny = 0;
00114         if ( accelerating_handle != NULL ) {
00115             vTaskDelete( accelerating_handle );
00116             accelerating_handle = NULL;
00117         }
00118         system_status.status = SYSTEM_BREAKING;
00119         xTaskCreatePinnedToCore(desaccelerating, "desaccelerating", 1024, (void*)
00120             &system_status.desacceleration, 9, &desaccelerating_handle, 1);
00121     }
00122     return system_status.frequency_destiny;
00123 }
00124 bool engine_emergency_stop_release(uint8_t signal) {
00125     bool retval = false;
00126     system_status.emergency_signals &= ~signal;
00127     if ( system_status.emergency_signals == 0 ) {
00128         system_status.status = SYSTEM_EMERGENCY_OK;
00129         retval = true;
00130     } else {
00131         ESP_LOGI(TAG, "Estado de las señales de emergencia:%d", system_status.emergency_signals );
00132     }
00133     return retval;
00134 }
00135
00136 void engine_emergency_stop(uint8_t signal) {
00137     if ( accelerating_handle != NULL ) {
00138         vTaskDelete( accelerating_handle );
00139         accelerating_handle = NULL;
00140     }
00141     if ( desaccelerating_handle != NULL ) {
00142         vTaskDelete( desaccelerating_handle );
00143         desaccelerating_handle = NULL;
00144     }
00145     system_status.frequency_destiny = 0;
00146     system_status.frequency = 0;
00147     system_status.status = SYSTEM_EMERGENCY;
00148     system_status.emergency_signals |= signal;
00149 }
00150
00151 uint16_t change_frequency(uint8_t speed_slector) {
00152     if (speed_slector > 8 ) {
00153         ESP_LOGE(TAG, "El numero del indice del selector de velocidad es muy grande%d",
00154             speed_slector);
00155         return 0;
00156     }
00157     system_status.inputs_status = speed_slector;
00158     if ( system_status.frequency > frequency_table[system_status.inputs_status] ) {
00159         if ( accelerating_handle != NULL ) {
00160             vTaskDelete( accelerating_handle );
00161             accelerating_handle = NULL;
00162         }
00163         uint16_t desacceleration = get_system_desacceleration();
00164         system_status.frequency_destiny = frequency_table[system_status.inputs_status];
00165         if ( desaccelerating_handle == NULL ) {
00166             xTaskCreatePinnedToCore(desaccelerating, "desaccelerating", 1024, (void*)

```

```

        &desacceleration, 9, &desaccelerating_handle, 1);
00167    }
00168    ESP_LOGI( TAG, "Cambiando la frecuencia de regimen a %d. Desacelerando",
00169    system_status.frequency_destiny);
00170    } else if ( system_status.frequency < frequency_table[system_status.inputs_status] ) {
00171        if ( desaccelerating_handle != NULL ) {
00172            vTaskDelete( desaccelerating_handle );
00173            desaccelerating_handle = NULL;
00174        }
00175        uint16_t acceleration = get_system_acceleration();
00176        system_status.frequency_destiny = frequency_table[system_status.inputs_status];
00177        if ( accelerating_handle == NULL ) {
00178            xTaskCreatePinnedToCore(accelerating, "accelerating", 1024, (void*) &(acceleration), 9,
00179            &accelerating_handle, 1);
00180        }
00181        ESP_LOGI( TAG, "Cambiando la frecuencia de regimen a %d. Acelerando",
00182        system_status.frequency_destiny);
00183    }
00184
00185 esp_err_t get_status(system_status_t *s_e) {
00186    if (s_e == NULL)
00187        return ESP_ERR_NOT_ALLOWED;
00188    s_e->frequency = system_status.frequency;
00189    s_e->frequency_destiny = system_status.frequency_destiny;
00190    s_e->vbus_min = system_status.vbus_min;
00191    s_e->ibus_max = system_status.ibus_max;
00192    s_e->inputs_status = system_status.inputs_status;
00193    s_e->status = system_status.status;
00194    return ESP_OK;
00195 }
00196
00197 system_status_t update_meas(uint16_t vbus_meas, uint16_t ibus_meas) {
00198    bool in_emergency = false;
00199    system_status.vbus_min = vbus_meas;
00200    system_status.ibus_max = ibus_meas;
00201
00202    if ( system_status.ibus_max > system_security_settings.ibus_max ) {
00203        in_emergency = true;
00204        if ( system_status.status != SYSTEM_EMERGENCY ) {
00205            ESP_LOGE( TAG, "Disparo de emergencia por sobre corriente.");
00206        }
00207    }
00208
00209    if ( system_status.vbus_min < system_security_settings.vbus_min ) {
00210        in_emergency = true;
00211        if ( system_status.status != SYSTEM_EMERGENCY ) {
00212            ESP_LOGE( TAG, "Disparo de emergencia por baja tensión.");
00213        }
00214    }
00215
00216    if ( in_emergency == true ) {
00217        if ( system_status.status != SYSTEM_EMERGENCY ) {
00218            SystemEventPost(SECURITY_EXCEEDED);
00219            ESP_LOGI( TAG, "Mando senal.");
00220        }
00221    } else if ( system_status.status == SYSTEM_EMERGENCY && ( system_status.emergency_signals & 0b010
00222 ) ) {
00223        SystemEventPost(SECURITY_OK);
00224        ESP_LOGI( TAG, "Salgo de emergencia por seguridad.");
00225    }
00226
00227    return system_status.status;
00228 }
00229 void set_frequency_table( uint16_t input_variable, uint16_t freq_regime ) {
00230    if ( input_variable == 1 ) { //lineal
00231        uint16_t frequency_gap = freq_regime / 8;
00232        frequency_table[7] = frequency_gap;
00233        for (uint8_t i = 6; i > 0; i-- ) {
00234            frequency_table[i] = frequency_table[i + 1] + frequency_gap;
00235            ESP_LOGI(TAG, "frequency_table[%d] = %d", i, frequency_table[i]);
00236        }
00237        frequency_table[0] = freq_regime;
00238        ESP_LOGI(TAG, "frequency_table[0] = %d", frequency_table[0]);
00239    } else if ( input_variable == 2 ) { //Cuadratica
00240        for (uint8_t i = 0; i < 8; i++ ) {
00241            uint32_t num = (uint32_t)freq_regime * (uint32_t)(i + 1) * (uint32_t)(i + 1);
00242            uint16_t fi = (uint16_t)((num + 32) / 64);
00243            if (fi == 0)
00244                fi = 1;
00245            frequency_table[7 - i] = fi;
00246            ESP_LOGI(TAG, "frequency_table[%d] = %d", 7 - i, frequency_table[7 - i]);
00247        }
00248    }

```

```

00249 }
00250
00251 void set_system_settings( frequency_settings_t *f_s, security_settings_t *s_s ) {
00252     system_status.acceleration = f_s->acceleration;
00253     system_status.desacceleration = f_s->desacceleration;
00254     system_seccurity_settings.vbus_min = s_s->vbus_min;
00255     system_seccurity_settings.ibus_max = s_s->ibus_max;
00256     set_frequency_table( f_s->input_variable, f_s->freq_regime );
00257 }
```

6.41. Referencia del archivo main/system/sysAdmin.h

Header con las funciones de funcionamiento del sistema.

```
#include "../LVFV_system.h"
```

Funciones

- **esp_err_t get_status (system_status_t *s_e)**
Obtiene una réplica del status del sistema.
- **uint16_t engine_start ()**
Función que setea la frecuencia de régimen de acuerdo a lo configurado por el usuario y ejecuta la tarea accelerating.
- **uint16_t engine_stop ()**
Función que setea la frecuencia de régimen de acuerdo a lo configurado por el usuario y ejecuta la tarea desaccelerating.
- **bool engine_emergency_stop_release (uint8_t signal)**
Pasa a estado SYSTEM_EMERGENCY_OK si todas las señales de emergencia son 0 para que el usuario pueda enviar la señal de stop al STM32 necesaria para poder arrancar nuevamente el motor.
- **void engine_emergency_stop (uint8_t signal)**
Pasa la frecuencia de regimen a 0Hz y pasa a estado SYSTEM_EMERGENCY.
- **uint16_t change_frequency (uint8_t speed_slector)**
Cambia la frecuencias de régimen del motor de acuerdo a la entrada aislada que haya ingresado. Ejecuta accelerating o desaccelerating, de acuerdo a lo necesario.
- **system_status_e update_meas (uint16_t vbus_meas, uint16_t ibus_meas)**
Actualiza las mediciones de tensión y corriente del bus de continua.
- **void set_frequency_table (uint16_t input_variable, uint16_t freq_regime)**
Carga en el vector de velocidades configurables por entradas, las diferentes frecuencias de trabajo.
- **void set_system_settings (frequency_settings_t *f_s, security_settings_t *s_s)**
Actualiza las variables de seguridad del sistema.

6.41.1. Descripción detallada

Header con las funciones de funcionamiento del sistema.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo [sysAdmin.h](#).

6.41.2. Documentación de funciones

6.41.2.1. change_frequency()

```
uint16_t change_frequency (
    uint8_t speed_selector)
```

Cambia la frecuencias de régimen del motor de acuerdo a la entrada aislada que haya ingresado. Ejecuta accelerating o desaccelerating, de acuerdo a lo necesario.

De acuerdo a la frecuencia de régimen elegida y el tipo de variación, la frecuencia de destino tendrá diferentes valores, siempre menores a las de régimen.

Parámetros

in	<i>speed_selector</i>	Índice dentro del array con frecuencias de trabajo
----	-----------------------	--

Devuelve

0Hz si *speed_selector* fue mal pasada como argumento, sino la frecuencia de destino

Definición en la línea 151 del archivo [sysAdmin.c](#).

6.41.2.2. engine_emergency_stop()

```
void engine_emergency_stop (
    uint8_t signal)
```

Pasa la frecuencia de regimen a 0Hz y pasa a estado SYSTEM_EMERGENCY.

Termina las tareas accelerating y desaccelerating y guarda la señal que dispara la emergencia para contabilizar todos las señales vigentes

Parámetros

in	<i>signal</i>	Señal que genera el estado de emergencia
----	---------------	--

Definición en la línea 136 del archivo [sysAdmin.c](#).

6.41.2.3. engine_emergency_stop_release()

```
bool engine_emergency_stop_release (
    uint8_t signal)
```

Pasa a estado SYSTEM_EMERGENCY_OK si todas las señales de emergencia son 0 para que el usuario pueda enviar la señal de stop al STM32 necesaria para poder arrancar nuevamente el motor.

Parámetros

<code>in</code>	<code>signal</code>	Señal que normaliza el estado de emergencia
-----------------	---------------------	---

Valores devueltos

<code>true</code>	si cambia a SYSTEM_EMERGENCY_OK; false si existe alguna señal aún activa
-------------------	--

Definición en la línea 124 del archivo [sysAdmin.c](#).

6.41.2.4. engine_start()

```
uint16_t engine_start ()
```

Función que setea la frecuencia de régimen de acuerdo a lo configurado por el usuario y ejecuta la tarea accelerating.

En caso de que desaccelerating esté corriendo, antes de ejecutar accelerating, la cierra para no generar un efecto de subida y bajada de frecuencia. El status pasará a SYSTEM_ACCEL_DESACCEL

Devuelve

Frecuencia de destino configurada

Definición en la línea 93 del archivo [sysAdmin.c](#).

6.41.2.5. engine_stop()

```
uint16_t engine_stop ()
```

Función que setea la frecuencia de régimen de acuerdo a lo configurado por el usuario y ejecuta la tarea desaccelerating.

En caso de que accelerating esté corriendo, antes de ejecutar desaccelerating, la cierra para no generar un efecto de subida y bajada de frecuencia. El status pasará a SYSTEM_ACCEL_DESACCEL o SYSTEM_BREAKING de acuerdo a la acción que haya ocurrido

Devuelve

Frecuencia de destino configurada

Definición en la línea 106 del archivo [sysAdmin.c](#).

6.41.2.6. get_status()

```
esp_err_t get_status (
    system_status_t * s_e)
```

Obtiene una réplica del status del sistema.

Parámetros

out	$s \leftarrow$	Puntero a la variable donde se devolverá la información de status
	$_e$	

Devuelve

- ESP_ERR_NOT_ALLOWED: En caso de que s_e sea un puntero a NULL
- ESP_OK: Si la copia fue exitosa

Definición en la línea 185 del archivo [sysAdmin.c](#).

6.41.2.7. set_frequency_table()

```
void set_frequency_table (
    uint16_t input_variable,
    uint16_t freq_regime)
```

Carga en el vector de velocidades configurables por entradas, las diferentes frecuencias de trabajo.

La posición 0 del buffer es la frecuencia de regimen, la posición 7 es la frecuencia más baja distinta de cero

Parámetros

in	<i>input_variable</i>	Tipo de variación entre las diferentes entradas. 1 para variación lineal; 2 para variación cuadrática.
in	<i>freq_regime</i>	Frecuencia de regimen ingresada por el usuario

Definición en la línea 229 del archivo [sysAdmin.c](#).

6.41.2.8. set_system_settings()

```
void set_system_settings (
    frequency_settings_t * f_s,
    security_settings_t * s_s)
```

Actualiza las variables de seguridad del sistema.

Parámetros

in	$f \leftarrow$	Puntero a la estructura con las variables de frecuencia a actualizar
in	$s \leftarrow$	Puntero a la estructura con las variables de seguridad a actualizar

Definición en la línea 251 del archivo [sysAdmin.c](#).

6.41.2.9. update_meas()

```
system_status_e update_meas (
    uint16_t vbus_meas,
    uint16_t ibus_meas)
```

Actualiza las mediciones de tensión y corriente del bus de continua.

En caso que superar los límites configurados, entra en SYSTEM_EMERGENCY

Parámetros

in	<i>vbus_meas</i>	<ul style="list-style-type: none"> Tensión de bus de continua leído por el ADC
in	<i>ibus_meas</i>	<ul style="list-style-type: none"> Corriente de bus de continua leído por el ADC

Devuelve

- SYSTEM_IDLE: El sistema se encuentra en estado de reposo
- SYSTEM_ACCEL_DESACCEL: El sistema se encuentra acelerando o desacelerando
- SYSTEM_REGIME: El sistema está con el motor girando a régimen
- SYSTEM_BREAKING: El sistema está frenando
- SYSTEM_EMERGENCY: El sistema entra por primera vez en estado de emergencia

Definición en la línea 197 del archivo [sysAdmin.c](#).

6.42. sysAdmin.h

[Ir a la documentación de este archivo.](#)

```

00001
00009 #ifndef __SYS_ADMIN_H__
00010
00011 #define __SYS_ADMIN_H__
00012
00013 #include "../LVFV_system.h"
00014
00027 esp_err_t get_status(system_status_t *s_e);
00028
00038 uint16_t engine_start();
00039
00049 uint16_t engine_stop();
00050
00060 bool engine_emergency_stop_release(uint8_t signal);
00061
00071 void engine_emergency_stop(uint8_t signal);
00072
00085 uint16_t change_frequency(uint8_t speed_selector);
00086
00107 system_status_e update_meas(uint16_t vbus_meas, uint16_t ibus_meas);
00108
00122 void set_frequency_table( uint16_t input_variable, uint16_t freq_regime );
00123
00135 void set_system_settings( frequency_settings_t *f_s, security_settings_t *s_s );
00136
00137 #endif

```

6.43. Referencia del archivo main/system/sysControl.c

Archivo de funciones que controlan el sistema físico a través del puerto SPI contra el STM32. En función de las respuestas del STM32, es que se controlan las variables internas de sistema del ESP32.

```

#include <stdio.h>
#include <string.h>
#include "esp_log.h"
#include "esp_err.h"

```

```
#include "freertos/Freertos.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "driver/spi_master.h"
#include "driver/gpio.h"
#include "./io_control/io_control.h"
#include "../LVFV_system.h"
#include "./display/display.h"
#include "./adc/adc.h"
#include "./SysAdmin.h"
#include "./SysControl.h"
```

defines

- #define PIN_NUM_CS 12 /* @def PIN_NUM_CS @brief Chip select del SPI para comunicación con el STM32. IO12 */
- #define PIN_NUM_CLK 14 /* @def PIN_NUM_CLK @brief Clock del SPI para comunicación con el STM32. IO14 */
- #define PIN_NUM_MISO 26 /* @def PIN_NUM_MISO @brief master input slave output del SPI para comunicación con el STM32. IO26 */
- #define PIN_NUM_MOSI 25 /* @def PIN_NUM_MOSI @brief master output slave input del SPI para comunicación con el STM32. IO25 */
- #define SPI_HOST_USED SPI2_HOST /* @def SPI_HOST_USED @brief don't know */
- #define SPI_CLOCK_HZ 1*1000*1000 /* @def SPI_CLOCK_HZ @brief Velocidad de clock: 1 MHz */
- #define SPI_QUEUE_TX_DEPTH 1 /* @def SPI_QUEUE_TX_DEPTH @brief Profundidad de comandos para el puerto SPI */

Funciones

- static SPI_Response SPI_SendRequest (spi_cmd_item_t *spi_cmd_item)

Envía el comando configurado en spi_cmd_item->request con el argumento spi_cmd_item->setValue en caso de ser necesario y obtiene los datos que responde el STM32 en spi_cmd_item->getValue si corresponde.
- esp_err_t SPI_Init (void)

Inicializa el módulo SPI del ESP32.
- void SPI_communication (void *arg)

Tarea que controla en arranque, parada y emergencia del sistema.
- esp_err_t SystemEventPost (systemSignal_e event)

Encola comandos en la cola system_event_queue.

Variables

- static const char * TAG = "sysControl"
- QueueHandle_t system_event_queue = NULL
- static spi_device_handle_t spi_handle = NULL

6.43.1. Descripción detallada

Archivo de funciones que controlan el sistema físico a través del puerto SPI contra el STM32. En función de las respuestas del STM32, es que se controlan las variables internas de sistema del ESP32.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo [sysControl.c](#).

6.43.2. Documentación de «define»

6.43.2.1. PIN_NUM_CLK

```
#define PIN_NUM_CLK 14 /** @def PIN_NUM_CLK @brief Clock del SPI para comunicación con el  
STM32. IO14 */
```

Definición en la línea [31](#) del archivo [sysControl.c](#).

6.43.2.2. PIN_NUM_CS

```
#define PIN_NUM_CS 12 /** @def PIN_NUM_CS @brief Chip select del SPI para comunicación con el  
STM32. IO12 */
```

Definición en la línea [30](#) del archivo [sysControl.c](#).

6.43.2.3. PIN_NUM_MISO

```
#define PIN_NUM_MISO 26 /** @def PIN_NUM_MISO @brief master input slave output del SPI para  
comunicación con el STM32. IO26 */
```

Definición en la línea [32](#) del archivo [sysControl.c](#).

6.43.2.4. PIN_NUM_MOSI

```
#define PIN_NUM_MOSI 25 /** @def PIN_NUM_MOSI @brief master output slave input del SPI para  
comunicación con el STM32. IO25 */
```

Definición en la línea [33](#) del archivo [sysControl.c](#).

6.43.2.5. SPI_CLOCK_HZ

```
#define SPI_CLOCK_HZ 1*1000*1000 /** @def SPI_CLOCK_HZ @brief Velocidad de clock: 1 MHz */
```

Definición en la línea 37 del archivo [sysControl.c](#).

6.43.2.6. SPI_HOST_USED

```
#define SPI_HOST_USED SPI2_HOST /** @def SPI_HOST_USED @brief don't know */
```

Definición en la línea 36 del archivo [sysControl.c](#).

6.43.2.7. SPI_QUEUE_TX_DEPTH

```
#define SPI_QUEUE_TX_DEPTH 1 /** @def SPI_QUEUE_TX_DEPTH @brief Profundidad de comandos para el puerto SPI */
```

Definición en la línea 38 del archivo [sysControl.c](#).

6.43.3. Documentación de funciones

6.43.3.1. SPI_communication()

```
void SPI_communication (
    void * arg)
```

Tarea que controla en arranque, parada y emergencia del sistema.

Espera comandos a través de la cola system_event_queue para evaluar si enviar comandos de start, stop, emergencia, cambios de velocidad, etc. Hace indirectamente el polling de las mediciones del ADC para evaluar si está en estado de emergencia o no.

Parámetros

in, out	arg	Sin uso
---------	-----	---------

Definición en la línea 186 del archivo [sysControl.c](#).

6.43.3.2. SPI_Init()

```
esp_err_t SPI_Init (
    void )
```

Inicializa el módulo SPI del ESP32.

Con figura los pines PIN_NUM_CS en el pin IO12, PIN_NUM_CLK en el pin IO14, PIN_NUM_MISO en el pin IO26 y PIN_NUM_MOSI en el pin IO25; además configura el clock del SPI en 1MHz y tendrá solo un comando para encolar

Devuelve

- **ESP_ERR_INVALID_ARG** Si la configuración es inválida. La combinación de los parámetros puede resultar inválida.
- **ESP_ERR_INVALID_STATE** Si el host ya se encontraba en uso, si la fuente de clock es inviable o si el SPI no fue inicializado correctamente.
- **ESP_ERR_NOT_FOUND** if there is no available DMA channel
- **ESP_ERR_NO_MEM** Si no hay memoria suficiente para inicializar el módulo
- **ESP_OK** si la configuración fue exitosa

Definición en la línea 155 del archivo [sysControl.c](#).

6.43.3.3. SPI_SendRequest()

```
SPI_Response SPI_SendRequest (
    spi_cmd_item_t * spi_cmd_item) [static]
```

Envía el comando configurado en `spi_cmd_item->request` con el argumento `spi_cmd_item->setValue` en caso de ser necesario y obtiene los datos que responde el STM32 en `spi_cmd_item->getValue` si corresponde.

El ESP32 debe enviar comandos en dos instancias. La primera envía el comando deseado por el usuario, se detiene la ejecución durante 200 mili segundos y luego se envía un nuevo comando para consultarle al STM32 si el comando enviado por el usuario fue recibido correctamente o no.

Parámetros

in, out	<code>spi_cdm_item</code>	Estructura de datos con los parámetros necesarios para llevar a cabo la comunicación. Allí se almacena el comando, valores a enviar y valores recibidos.
---------	---------------------------	--

Devuelve

- `SPI_RESPONSE_ERR`: Error en la transmisión del comando por problemas de inicialización del handler. El comando no sale del ESP32
- `SPI_RESPONSE_ERR_CMD_UNKNOWN`: El comando enviado no está dentro los posibles
- `SPI_RESPONSE_ERR_MOVING`: Se está intentando enviar un comando de start con el motor en movimiento
- `SPI_RESPONSE_ERR_NOT_MOVING`: Se está intentando enviar un comando de stop con el motor detenido
- `SPI_RESPONSE_ERR_DATA_MISSING`: Faltó enviar un dato dentro de la trama
- `SPI_RESPONSE_ERR_DATA_INVALID`: Los datos enviados como argumento dentro del comando están fuera de rango
- `SPI_RESPONSE_OK`: La comunicación entre ESP32 y STM32 fue exitosa

Definición en la línea [67](#) del archivo `sysControl.c`.

6.43.3.4. SystemEventPost()

```
esp_err_t SystemEventPost (
    systemSignal_e event)
```

Encola comandos en la cola `system_event_queue`.

Parámetros

in	<code>event</code>	Evento que se desea encolar en el sistema
----	--------------------	---

Devuelve

- pdTRUE Si el comando se posteó correctamente
- Cualquier otra respuesta implica que la cola está llena

Definición en la línea [385](#) del archivo `sysControl.c`.

6.43.4. Documentación de variables

6.43.4.1. spi_handle

```
spi_device_handle_t spi_handle = NULL [static]
```

Definición en la línea 44 del archivo [sysControl.c](#).

6.43.4.2. system_event_queue

```
QueueHandle_t system_event_queue = NULL
```

Definición en la línea 42 del archivo [sysControl.c](#).

6.43.4.3. TAG

```
const char* TAG = "sysControl" [static]
```

Definición en la línea 40 del archivo [sysControl.c](#).

6.44. sysControl.c

[Ir a la documentación de este archivo.](#)

```
00001
00009 #include <stdio.h>
00010 #include <string.h>
00011
00012 #include "esp_log.h"
00013 #include "esp_err.h"
00014
00015 #include "freertos/FreeRTOS.h"
00016 #include "freertos/task.h"
00017 #include "freertos/queue.h"
00018
00019 #include "driver/spi_master.h"
00020 #include "driver/gpio.h"
00021
00022 #include "./io_control/io_control.h"
00023 #include "../LVFV_system.h"
00024 #include "./display/display.h"
00025 #include "./adc/adc.h"
00026 #include "./SysAdmin.h"
00027 #include "./SysControl.h"
00028
00029 // ===== Pines =====
00030 #define PIN_NUM_CS 12
00031 #define PIN_NUM_CLK 14
00032 #define PIN_NUM_MISO 26
00033 #define PIN_NUM_MOSI 25
00034
00035 // ===== SPI =====
00036 #define SPI_HOST_USED SPI2_HOST
00037 #define SPI_CLOCK_HZ 1*1000*1000
00038 #define SPI_QUEUE_TX_DEPTH 1
00039
00040 static const char *TAG = "sysControl";
00041
00042 QueueHandle_t system_event_queue = NULL;
00043
00044 static spi_device_handle_t spi_handle = NULL; // Handler del puerto SPI para
comunicación con el STM32. Inicializado en esp_err_t SPI_Init(void)
00045
00065 static SPI_Response SPI_SendRequest(spi_cmd_item_t *spi_cmd_item);
00066
00067 static SPI_Response SPI_SendRequest(spi_cmd_item_t *spi_cmd_item) {
```

```

00068
00069     uint8_t tx_buffer[4];
00070     uint8_t rx_buffer[4];
00071     esp_err_t ret;
00072     int flagDevolverValor;      // Se necesita devolver el valor por parametro
00073
00074     ESP_LOGI( TAG, "[SPI Module] Request %d", spi_cmd_item->request);
00075
00076     // Chequeo de errores fuera de rango y comando desconocido
00077     if(spi_cmd_item->setValue > 512 || spi_cmd_item->setValue < 0) {
00078         return SPI_RESPONSE_ERR_DATA_INVALID;
00079     }
00080
00081     if(spi_cmd_item->request < SPI_REQUEST_START || spi_cmd_item->request > SPI_REQUEST_EMERGENCY) {
00082         ESP_LOGI( TAG, "[SPI Module] Comando desconocido\n");
00083         return SPI_RESPONSE_ERR_CMD_UNKNOWN;
00084     }
00085
00086     // Es de los comandos que deben devolver un valor por parametro?
00087     if(spi_cmd_item->request >= SPI_REQUEST_GET_FREC && spi_cmd_item->request <= SPI_REQUEST_IS_STOP)
00088     {
00089         flagDevolverValor = 1;
00090     }else {
00091         flagDevolverValor = 0;
00092     }
00093
00094     // Se arma la cadena a enviar
00095     tx_buffer[0] = spi_cmd_item->request;
00096
00097     if(spi_cmd_item->setValue > 255) {
00098         tx_buffer[1] = 255;
00099         tx_buffer[2] = spi_cmd_item->setValue - 255;
00100         tx_buffer[3] = ';';
00101     }else {
00102         tx_buffer[1] = spi_cmd_item->setValue;
00103         tx_buffer[2] = 0;
00104         tx_buffer[3] = ';';
00105     }
00106
00107     // Limpiamos el buffer de recepcion
00108     memset(rx_buffer, 0, sizeof(rx_buffer));
00109
00110     // Siempre envio y recibo 4 bytes
00111     spi_transaction_t t = {
00112         .length = 8 * 4,
00113         .tx_buffer = tx_buffer,
00114         .rx_buffer = rx_buffer,
00115         .rxlength = 8 * 4,
00116     };
00117
00118     ret = spi_device_transmit(spi_handle, &t);
00119     if (ret != ESP_OK) {
00120         ESP_LOGI( TAG, "[ESP32] Error en SPI transmit: %d", ret);
00121         return SPI_RESPONSE_ERR;
00122     }
00123
00124     tx_buffer[0] = SPI_REQUEST_RESPONSE;
00125     tx_buffer[1] = ';';
00126
00127     t.length = 8 * 4;
00128     t.tx_buffer = tx_buffer;
00129     t.rx_buffer = rx_buffer;
00130     t.rxlength = 8 * 4;
00131
00132     vTaskDelay(pdMS_TO_TICKS(400));
00133
00134     ret = spi_device_transmit(spi_handle, &t);
00135     if (ret != ESP_OK) {
00136         ESP_LOGI( TAG, "[ESP32] Error en SPI transmit: %d", ret);
00137         return SPI_RESPONSE_ERR;
00138     }
00139
00140
00141
00142     ESP_LOGI( TAG, "rx_buffer[0]:%d", rx_buffer[0]);
00143     if(flagDevolverValor && rx_buffer[0] == SPI_RESPONSE_OK) {
00144
00145         spi_cmd_item->getValue = rx_buffer[1] + rx_buffer[2];
00146         ESP_LOGI( TAG, "RxValores:%d - %d", rx_buffer[1], rx_buffer[2]);
00147
00148     }else {
00149         spi_cmd_item->getValue = 0;
00150     }
00151
00152     return rx_buffer[0];
00153 }
```

```

00154
00155 esp_err_t SPI_Init(void) {
00156     spi_bus_config_t buscfg = {
00157         .miso_io_num = PIN_NUM_MISO,
00158         .mosi_io_num = PIN_NUM_MOSI,
00159         .sclk_io_num = PIN_NUM_CLK,
00160         .quadwp_io_num = -1,
00161         .quadhd_io_num = -1,
00162         .max_transfer_sz = 32
00163     };
00164
00165     spi_device_interface_config_t devcfg = {
00166         .clock_speed_hz = SPI_CLOCK_HZ,
00167         .mode = 0,                                     // SPI mode 0
00168         .spics_io_num = PIN_NUM_CS,                     // CS pin
00169         .queue_size = 1,                                // Sin dummy bits
00170         .flags = SPI_DEVICE_NO_DUMMY
00171     };
00172
00173     // Inicializar bus SPI
00174     esp_err_t ret = spi_bus_initialize(VSPI_HOST, &buscfg, 1);
00175     if (ret != ESP_OK) {
00176         return ret;
00177     }
00178     ESP_ERROR_CHECK(ret);
00179
00180     // Añadir dispositivo
00181     ret = spi_bus_add_device(VSPI_HOST, &devcfg, &spi_handle);
00182     ESP_ERROR_CHECK(ret);
00183     return ESP_OK;
00184 }
00185
00186 void SPI_communication(void *arg) {
00187     spi_cmd_item_t item;
00188
00189     systemSignal_e new_button;
00190
00191     if (SPI_Init() != ESP_OK) {
00192         ESP_LOGE(TAG, "SPI_Init falló; no creo SPI_communication");
00193         ESP_ERROR_CHECK(ESP_FAIL);
00194     }
00195
00196     if (system_event_queue == NULL) {
00197         system_event_queue = xQueueCreate(1, sizeof(systemSignal_e));
00198         if (system_event_queue == NULL) {
00199             ESP_LOGE(TAG, "No se pudo crear la cola de interrupciones");
00200             ESP_ERROR_CHECK(ESP_FAIL);
00201         }
00202     }
00203
00204     vTaskDelay(pdMS_TO_TICKS(4000));
00205
00206     do {
00207         item.request = SPI_REQUEST_STOP;
00208         item.setValue = 0;
00209         item.getValue = 0;
00210         vTaskDelay(pdMS_TO_TICKS(100));
00211     } while (SPI_SendRequest(&item) != SPI_RESPONSE_ERR_NOT_MOVING);
00212
00213     for (uint32_t i = 0; i++ ) {
00214         if (readADC()) {
00215             while( xQueueReceive( system_event_queue, &new_button, pdMS_TO_TICKS(0) ) );
00216             SystemEventPost(STOP_PRESSED);
00217             break;
00218         }
00219         vTaskDelay(pdMS_TO_TICKS(100));
00220     }
00221
00222     SystemEventPost(STOP_PRESSED);
00223
00224     ESP_LOGI(TAG, "SPI_communication lista. Esperando comandos...");
00225
00226     while (1) {
00227         system_status_t s_e;
00228         get_status( &s_e );
00229         readADC();
00230         if (xQueueReceive( system_event_queue, &new_button, pdMS_TO_TICKS(20) ) ) {
00231             switch ( new_button ) {
00232                 case EMERGENCI_STOP_PRESSED:
00233                     if ( s_e.status != SYSTEM_EMERGENCY ) {
00234                         ESP_LOGI(TAG, "Botón de EMERGENCIA presionado");
00235                         item.request = SPI_REQUEST_EMERGENCY;
00236                         item.setValue = 0;
00237                         item.getValue = 0;
00238                         SPI_SendRequest(&item);
00239                         RelayEventPost( 1 );
00240                     } else {
00241
00242
00243
00244
00245
00246
00247
00248
00249
00250
00251
00252
00253
00254
00255
00256
00257
00258
00259
00260
00261
00262
00263
00264
00265
00266
00267
00268
00269
00270
00271
00272
00273
00274
00275
00276
00277
00278
00279
00280
00281
00282
00283
00284
00285
00286
00287
00288
00289
00290
00291
00292
00293
00294
00295
00296
00297
00298
00299
00300
00301
00302
00303
00304
00305
00306
00307
00308
00309
00310
00311
00312
00313
00314
00315
00316
00317
00318
00319
00320
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01840
01841
01842
01843
01844
01845
01846
01847
01848
01849
01850
01851
01852
01853
01854
01855
01856
01857
01858
01859
01860
01861
01862
01863
01864
01865
01866
01867
01868
01869
01870
01871
01872
01873
01874
01875
01876
01877
01878
01879
01880
01881
01882
01883
01884
01885
01886
01887
01888
01889
01890
01891
01892
01893
01894
018
```

```

00241             ESP_LOGI(TAG, "El sistema ya se encontraba en emergencia");
00242         }
00243         engine_emergency_stop(0b001);
00244         break;
00245     case SECURITY_EXCEEDED:
00246         if ( s_e.status != SYSTEM_EMERGENCY ) {
00247             ESP_LOGI(TAG, "Corriente elevada o tensión reducida");
00248             item.request = SPI_REQUEST_EMERGENCY;
00249             item.setValue = 0;
00250             item.getValue = 0;
00251             SPI_SendRequest(&item);
00252             RelayEventPost( 1 );
00253         } else {
00254             ESP_LOGI(TAG, "El sistema ya se encontraba en emergencia");
00255         }
00256         engine_emergency_stop(0b010);
00257         break;
00258     case TERMO_SW_PRESSED:
00259         if ( s_e.status != SYSTEM_EMERGENCY ) {
00260             ESP_LOGI(TAG, "Termoswitch activo");
00261             item.request = SPI_REQUEST_EMERGENCY;
00262             item.setValue = 0;
00263             item.getValue = 0;
00264             SPI_SendRequest(&item);
00265             RelayEventPost( 1 );
00266         } else {
00267             ESP_LOGI(TAG, "El sistema ya se encontraba en emergencia");
00268         }
00269         engine_emergency_stop(0b100);
00270         break;
00271     case EMERGENCI_STOP_RELEASED:
00272         ESP_LOGI(TAG, "Botón de EMERGENCIA liberado");
00273         engine_emergency_stop_release(0b001);
00274         break;
00275     case SECURITY_OK:
00276         ESP_LOGI(TAG, "Tensión y corriente normalizadas");
00277         engine_emergency_stop_release(0b010);
00278         break;
00279     case TERMO_SW_RELEASED:
00280         ESP_LOGI(TAG, "Termoswitch desactivado");
00281         engine_emergency_stop_release(0b100);
00282         break;
00283     case STOP_PRESSED:
00284         if ( s_e.status == SYSTEM_EMERGENCY ) {
00285             ESP_LOGI(TAG,"Primero salir de estado de emergencia");
00286         } else if ( s_e.status == SYSTEM_ACCEL_DESACCEL || s_e.status == SYSTEM_REGIME ) {
00287             engine_stop();
00288             ESP_LOGI(TAG, "Botón de Parada presionado");
00289             item.request = SPI_REQUEST_STOP;
00290             item.setValue = 0;
00291             item.getValue = 0;
00292             SPI_SendRequest(&item);
00293         } else if ( s_e.status == SYSTEM_EMERGENCY_OK ) {
00294             engine_stop();
00295             ESP_LOGI(TAG, "Botón de Parada presionado - Liberando estado de emergencia");
00296             item.request = SPI_REQUEST_STOP;
00297             item.setValue = 0;
00298             item.getValue = 0;
00299             SPI_SendRequest(&item);
00300             RelayEventPost( 0 );
00301         }
00302         break;
00303     case START_PRESSED:
00304         if ( s_e.status == SYSTEM_IDLE ) {
00305             ESP_LOGI(TAG, "Botón de Inicio presionado");
00306             SPI_Response SPI_Commando_response;
00307
00308             item.request = SPI_REQUEST_SET_FREQ;
00309             item.setValue = get_system_frequency();
00310             item.getValue = 0;
00311             SPI_Commando_response = SPI_SendRequest(&item);
00312             if ( SPI_Commando_response != SPI_RESPONSE_OK ) {
00313                 ESP_LOGE(TAG,"Error cargando la frecuencia");
00314                 break;
00315             }
00316
00317             item.request = SPI_REQUEST_SET_ACCEL;
00318             item.setValue = get_system_acceleration();
00319             item.getValue = 0;
00320             SPI_Commando_response = SPI_SendRequest(&item);
00321             if ( SPI_Commando_response != SPI_RESPONSE_OK ) {
00322                 ESP_LOGE(TAG,"Error cargando la aceleracion");
00323                 break;
00324             }
00325
00326             item.request = SPI_REQUEST_SET_DESACCEL;
00327             item.setValue = get_system_desacceleration();

```

```

00328         item.getValue = 0;
00329         SPI_Commando_response = SPI_SendRequest(&item);
00330         if ( SPI_Commando_response != SPI_RESPONSE_OK ) {
00331             ESP_LOGE(TAG,"Error cargando la desaceleracion");
00332             break;
00333         }
00334
00335         item.request = SPI_REQUEST_START;
00336         item.setValue = 0;
00337         item.getValue = 0;
00338         SPI_Commando_response = SPI_SendRequest(&item);
00339         if ( SPI_Commando_response != SPI_RESPONSE_OK ) {
00340             ESP_LOGE(TAG,"Error arrancando el motor");
00341             break;
00342         }
00343         uint16_t dest = engine_start();
00344         ESP_LOGI(TAG, "Motor arrancado. Frecuencia destino:%d", dest);
00345     } else {
00346         ESP_LOGE(TAG,"El motor debe estar detenido para poder arrancarlo");
00347     }
00348     break;
00349 case START_RELEASED:
00350     ESP_LOGI(TAG, "Botón de Inicio liberado");
00351     break;
00352 case SPEED_SELECTOR_0:
00353 case SPEED_SELECTOR_1:
00354 case SPEED_SELECTOR_2:
00355 case SPEED_SELECTOR_3:
00356 case SPEED_SELECTOR_4:
00357 case SPEED_SELECTOR_5:
00358 case SPEED_SELECTOR_6:
00359 case SPEED_SELECTOR_7:
00360 case SPEED_SELECTOR_8:
00361 case SPEED_SELECTOR_9:
00362
00363     if ( s_e.status == SYSTEM_REGIME || s_e.status == SYSTEM_ACCEL_DESACCEL ) {
00364         ESP_LOGI(TAG, "Cambio de velocidad:%d", new_button - SPEED_SELECTOR_0);
00365         uint8_t old_input_status = s_e.inputs_status;
00366         item.request = SPI_REQUEST_SET_FREQ;
00367         item.setValue = change_frequency( new_button - SPEED_SELECTOR_0 );
00368         item.getValue = 0;
00369         if ( SPI_SendRequest(&item) != SPI_RESPONSE_OK ) {
00370             change_frequency( old_input_status );
00371             ESP_LOGE(TAG, "El STM32 no respondio correctamente");
00372             xQueueSend(system_event_queue, &new_button, pdMS_TO_TICKS(1000));
00373         }
00374     } else {
00375         ESP_LOGI(TAG, "No puede cambiar la velocidad, status=%d", s_e.status);
00376     }
00377     break;
00378 default:
00379     break;
00380 }
00381 }
00382 }
00383 }
00384
00385 esp_err_t SystemEventPost(systemSignal_e event) {
00386     if (system_event_queue == NULL) {
00387         return ESP_FAIL;
00388     }
00389     return xQueueSend(system_event_queue, &event, 0);
00390 }

```

6.45. Referencia del archivo main/system/sysControl.h

Declaraciones de funciones que controlan el sistema del STM32. Además se encontrarán los comandos y respuestas que admite el STM32.

```
#include "../LVFV_system.h"
```

Estructuras de datos

- struct `spi_cmd_item_t`

Estructura de datos que contiene un comando a enviar por SPI, un argumento que se desea enviar al STM32 y un dato que pueda recibirse como respuesta en consecuencia.

Enumeraciones

- enum SPI_Request {

 SPI_REQUEST_START = 10, SPI_REQUEST_STOP, SPI_REQUEST_SET_FREQ, SPI_REQUEST_SET_ACCEL

 ,

 SPI_REQUEST_SET_DESACCEL, SPI_REQUEST_SET_DIR, SPI_REQUEST_GET_FREQ, SPI_REQUEST_GET_ACCEL

 ,

 SPI_REQUEST_GET_DESACCEL, SPI_REQUEST_GET_DIR, SPI_REQUEST_IS_STOP, SPI_REQUEST_EMERGENCY

 ,

 SPI_REQUEST_RESPONSE = 0x50 }

 Posibles comandos que puede enviar el STM32.
- enum SPI_Response {

 SPI_RESPONSE_ERR = 0xA0, SPI_RESPONSE_ERR_CMD_UNKNOWN, SPI_RESPONSE_ERR_NO_COMMAND

 ,

 SPI_RESPONSE_ERR_MOVING ,

 SPI_RESPONSE_ERR_NOT_MOVING, SPI_RESPONSE_ERR_DATA_MISSING, SPI_RESPONSE_ERR_DATA_INVALID

 ,

 SPI_RESPONSE_ERR_DATA_OUT_RANGE ,

 SPI_RESPONSE_OK = 0xFF }

Posibles respuesta que puede enviar el STM32 en consecuencia por los request enviados.

Funciones

- esp_err_t SPI_Init (void)

 Inicializa el módulo SPI del ESP32.
- void SPI_communication (void *arg)

 Tarea que controla en arranque, parada y emergencia del sistema.
- esp_err_t SystemEventPost (systemSignal_e event)

 Encola comandos en la cola system_event_queue.

6.45.1. Descripción detallada

Declaraciones de funciones que controlan el sistema del STM32. Además se encontrarán los comandos y respuestas que admite el STM32.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo [sysControl.h](#).

6.45.2. Documentación de enumeraciones

6.45.2.1. SPI_Request

enum SPI_Request

Posibles comandos que puede enviar el STM32.

Valores de enumeraciones

SPI_REQUEST_START	
SPI_REQUEST_STOP	
SPI_REQUEST_SET_FREC	
SPI_REQUEST_SET_ACCEL	
SPI_REQUEST_SET_DESACEL	
SPI_REQUEST_SET_DIR	
SPI_REQUEST_GET_FREC	
SPI_REQUEST_GET_ACCEL	
SPI_REQUEST_GET_DESACEL	
SPI_REQUEST_GET_DIR	
SPI_REQUEST_IS_STOP	
SPI_REQUEST_EMERGENCY	
SPI_REQUEST_RESPONSE	

Definición en la línea 19 del archivo [sysControl.h](#).

6.45.2.2. SPI_Response

enum [SPI_Response](#)

Posibles respuesta que puede enviar el STM32 en consecuencia por los request enviados.

Valores de enumeraciones

SPI_RESPONSE_ERR	
SPI_RESPONSE_ERR_CMD_UNKNOWN	
SPI_RESPONSE_ERR_NO_COMMAND	
SPI_RESPONSE_ERR_MOVING	
SPI_RESPONSE_ERR_NOT_MOVING	
SPI_RESPONSE_ERR_DATA_MISSING	
SPI_RESPONSE_ERR_DATA_INVALID	
SPI_RESPONSE_ERR_DATA_OUT_RANGE	
SPI_RESPONSE_OK	

Definición en la línea 40 del archivo [sysControl.h](#).

6.45.3. Documentación de funciones

6.45.3.1. SPI_communication()

```
void SPI_communication (
    void * arg)
```

Tarea que controla en arranque, parada y emergencia del sistema.

Espera comandos a través de la cola system_event_queue para evaluar si enviar comandos de start, stop, emergencia, cambios de velocidad, etc. Hace indirectamente el polling de las mediciones del ADC para evaluar si está en estado de emergencia o no.

Parámetros

in, out	<i>arg</i>	Sin uso
---------	------------	---------

Definición en la línea 186 del archivo [sysControl.c](#).

6.45.3.2. SPI_Init()

```
esp_err_t SPI_Init (
    void )
```

Inicializa el módulo SPI del ESP32.

Con figura los pines PIN_NUM_CS en el pin IO12, PIN_NUM_CLK en el pin IO14, PIN_NUM_MISO en el pin IO26 y PIN_NUM_MOSI en el pin IO25; además configura el clock del SPI en 1MHz y tendrá solo un comando para encolar

Devuelve

- `ESP_ERR_INVALID_ARG` Si la configuración es inválida. La combinación de los parámetros puede resultar inválida.
- `ESP_ERR_INVALID_STATE` Si el host ya se encontraba en uso, si la fuente de clock es inviable o si el SPI no fue inicializado correctamente.
- `ESP_ERR_NOT_FOUND` if there is no available DMA channel
- `ESP_ERR_NO_MEM` Si no hay memoria suficiente para inicializar el módulo
- `ESP_OK` si la configuración fue exitosa

Definición en la línea 155 del archivo [sysControl.c](#).

6.45.3.3. SystemEventPost()

```
esp_err_t SystemEventPost (
    systemSignal_e event)
```

Encola comandos en la cola `system_event_queue`.

Parámetros

in	<i>event</i>	Evento que se desea encolar en el sistema
----	--------------	---

Devuelve

- pdTRUE Si el comando se posteó correctamente
- Cualquier otra respuesta implica que la cola está llena

Definición en la línea 385 del archivo [sysControl.c](#).

6.46. sysControl.h

[Ir a la documentación de este archivo.](#)

```

00001
00009 #ifndef SPI_MODULE_H
0010 #define SPI_MODULE_H
0011
0012 #include "../LVFV_system.h"
0013
0014 typedef enum {
0015     SPI_REQUEST_START = 10,                                // 10 - Comando de arranque de motor
0016     SPI_REQUEST_STOP,                                     // 11 - Comando de parada de motor
0017     SPI_REQUEST_SET_FREC,                                 // 12 - Comando para configurar la frecuencia de
0018         régimen
0019     SPI_REQUEST_SET_ACCEL,                               // 13 - Comando para setear la aceleración hasta la
0020         frecuencia de régimen
0021     SPI_REQUEST_SET_DESACEL,                            // 14 - Comando para setear la desaceleración a 0Hz o
0022         frecuencia de régimen
0023     SPI_REQUEST_SET_DIR,                                // 15 - Comando para setear la dirección - TODO
0024     SPI_REQUEST_GET_FREC,                               // 16 - Comando de consulta de frecuencia de régimen
0025     SPI_REQUEST_GET_ACCEL,                             // 17 - Comando de consulta de aceleración
0026     SPI_REQUEST_GET_DESACEL,                           // 18 - Comando de consulta de desaceleración
0027     SPI_REQUEST_GET_DIR,                               // 19 - Comando de consulta de dirección de giro
0028     SPI_REQUEST_IS_STOP,                              // 20 - Comando de consulta para saber si el motor
0029         está parado o en movimiento
0030     SPI_REQUEST_EMERGENCY,                            // 21 - Comando para entrar en estado de emergencia -
0031         deja de conmutar la salida
0032     SPI_REQUEST_RESPONSE = 0x50                      // 80 - Comando para pedirle al STM32 la respuesta al
0033 } SPI_Request;
0034
0035 typedef enum {
0036     SPI_RESPONSE_ERR = 0xA0,                            // 160 - Error en la transmisión del comando por
0037         problemas de inicialización del handler. El comando no sale del ESP32
0038     SPI_RESPONSE_ERR_CMD_UNKNOWN,                     // 161 - El comando enviado no está dentro los
0039         posibles
0040     SPI_RESPONSE_ERR_NO_COMMAND,                   // 162 - Llegó mensaje pero sin comando
0041     SPI_RESPONSE_ERR_MOVING,                        // 163 - Se está intentando enviar un comando de start
0042         con el motor en movimiento
0043     SPI_RESPONSE_ERR_NOT_MOVING,                  // 164 - Se está intentando enviar un comando de stop
0044         con el motor detenido
0045     SPI_RESPONSE_ERR_DATA_MISSING,                // 165 - Faltó enviar un dato dentro de la trama
0046     SPI_RESPONSE_ERR_DATA_INVALID,               // 166 - Los datos enviados como argumento dentro del
0047         comando están fuera de rango
0048     SPI_RESPONSE_ERR_DATA_OUT_RANGE,             // 167 - Comando con datos fuera de rango permitido
0049     SPI_RESPONSE_OK = 0xFF,                         // 255 - La comunicación entre ESP32 y STM32 fue
0050         exitosa
0051 } SPI_Response;
0052
0053 typedef struct {
0054     SPI_Request request;                            // Comando a enviar al STM32
0055     int getValue;                                  // Variable devuelta por el STM32 en caso de ser un
0056         comando get
0057     int setValue;                                // Dato enviado con los comandos set
0058 } spi_cmd_item_t;
0059
0060 esp_err_t SPI_Init(void);
0061 void SPI_communication(void *arg);
0062
0063 esp_err_t SystemEventPost(systemSignal_e event);
0064
0065 #endif // SPI_MODULE_H

```

6.47. Referencia del archivo main/wifi/form.c

Variables

- const char * [HTML_FORM](#)

6.47.1. Documentación de variables

6.47.1.1. HTML_FORM

```
const char* HTML_FORM
```

Definición en la línea 1 del archivo `form.c`.

6.48. form.c

[Ir a la documentación de este archivo.](#)

```
00001 const char *HTML_FORM =
00002 "<!DOCTYPE html>"
00003 "<html lang=\"es\">"
00004 "<head>"
00005 "    <meta charset=\"UTF-8\">"
00006 "    <meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">"
00007 "    <title>Configuración del Variador</title>"
00008 "    <style>"
00009 "        body{font-family:system-ui,-apple-system,Segoe
UI,Roboto,sans-serif;background:#f8f9fa;margin:0;}"
00010 "        .container{max-width:960px;margin:0 auto;padding:1rem;}"
00011 "        h1{font-size:1.8rem;margin-bottom:1.5rem;}"
00012 "        .row{display:flex;flex-wrap:wrap;margin:-0.5rem;}"
00013 "        .col-md-4{flex:0 0 100%;max-width:100%;padding:0.5rem;}"
00014 "        @media (min-width:768px){.col-md-4{flex:0 0 33.3333%;max-width:33.3333%;}}"
00015 "        .form-label{display:block;font-weight:600;margin-bottom:0.25rem;font-size:0.95rem;}"
00016 "        .form-control,.form-select{display:block;width:100%;padding:0.375rem 0.75rem;}"
00017 "            font-size:1rem;line-height:1.5;color:#212529;background-color:#fff;}"
00018 "            border:1px solid #ced4da;border-radius:0.25rem;box-sizing:border-box;}"
00019 "            .form-control:focus,.form-select:focus{outline:2px solid #0d6efd33;outline-offset:1px;}"
00020 "            .btn{display:inline-block;font-weight:400;line-height:1.5;text-align:center;}"
00021 "                text-decoration:none;border:1px solid transparent;padding:0.375rem 0.75rem;}"
00022 "                font-size:1rem;border-radius:0.25rem;cursor:pointer;margin:1.5rem;}"
00023 "                .btn-primary{color:#fff;background-color:#0d6efd;border-color:#0d6efd;width:100%;}"
00024 "                .btn-primary:hover{background-color:#0b5ed7;border-color:#0a58ca;}"
00025 "                .btn-success{color:#fff;background-color:#198754;border-color:#198754;}"
00026 "                .btn-danger{color:#fff;background-color:#dc3545;border-color:#dc3545;}"
00027 "                .btn-success:hover{background-color:#157347;border-color:#146c43;}"
00028 "                .btn-danger:hover{background-color:#bb2d3b;border-color:#b02a37;}"
00029 "                .mb-4{margin-bottom:1.5rem;}"
00030 "                .mt-4{margin-top:1.5rem;}"
00031 "                .py-4{padding-top:1.5rem;padding-bottom:1.5rem;}"
00032 "                .me-2{margin-right:0.5rem;}"
00033 "            </style>"
00034 "</head>"
00035 "<body class=\"bg-light\">"
00036 "<div class=\"container py-4\">"
00037 "    <h1 class=\"mb-4\">Configuración del Variador</h1>"
00038 "    <form id=\"config-form\" method=\"POST\" action=\"/save\" class=\"row g-3\">"
00039
00040 "        <div class=\"col-md-4\">
00041 "            <label class=\"form-label\">Frecuencia de operación (Hz)</label>
00042 "            <input type=\"number\" class=\"form-control\" name=\"frequency\" min=\"1\" max=\"150\" required>
00043 "        </div>
00044
00045 "        <div class=\"col-md-4\">
00046 "            <label class=\"form-label\">Aceleración (Hz/s)</label>
00047 "            <input type=\"number\" class=\"form-control\" name=\"accel\" min=\"1\" max=\"50\" required>
00048 "        </div>
00049
00050 "        <div class=\"col-md-4\">
00051 "            <label class=\"form-label\">Desaceleración (Hz/s)</label>
00052 "            <input type=\"number\" class=\"form-control\" name=\"decel\" min=\"1\" max=\"50\" required>
00053 "        </div>
00054
00055 "        <div class=\"col-md-4\">
00056 "            <label class=\"form-label\">Variación de las entradas</label>
00057 "            <select class=\"form-select\" name=\"variation\" required>
00058 "                <option value=\"1\">Lineal</option>
00059 "                <option value=\"2\">Cuadrática</option>
00060 "            </select>
00061 "        </div>
00062
00063 "        <div class=\"col-md-4\">
```

```

00064 "      <label class=\"form-label\">Corriente máx. bus DC (mA)</label>
00065 "      <input type=\"number\" class=\"form-control\" name=\"imax\" min=\"500\" max=\"2000\" required>
00066 "    </div>
00067
00068 "    <div class=\"col-md-4\">
00069 "      <label class=\"form-label\">Tensión máx. bus DC (V)</label>
00070 "      <input type=\"number\" class=\"form-control\" name=\"vmin\" min=\"250\" max=\"350\" required>
00071 "    </div>
00072
00073 "    <div class=\"col-md-4\">
00074 "      <label class=\"form-label\">Hora del sistema (HH:mm:ss)</label>
00075 "      <input type=\"text\" class=\"form-control\" name=\"sys_time\" placeholder=\"12:34:56\" required>
00076 "    </div>
00077
00078 "    <div class=\"col-md-4\">
00079 "      <label class=\"form-label\">Horario de arranque (HH:mm)</label>
00080 "      <input type=\"text\" class=\"form-control\" name=\"start_time\" placeholder=\"08:00\" required>
00081 "    </div>
00082
00083 "    <div class=\"col-md-4\">
00084 "      <label class=\"form-label\">Horario de parada (HH:mm)</label>
00085 "      <input type=\"text\" class=\"form-control\" name=\"stop_time\" placeholder=\"18:00\" required>
00086 "    </div>
00087
00088 "    <div class=\"col-12 mt-4\">
00089 "      <button type=\"submit\" class=\"btn btn-primary me-2\" name=\"cmd\" value=\"save\">Guardar configuración</button>
00090 "      <button type=\"submit\" class=\"btn btn-success me-2\" name=\"cmd\" value=\"start\">Arrancar motor</button>
00091 "      <button type=\"submit\" class=\"btn btn-danger\" name=\"cmd\" value=\"stop\">Parar motor</button>
00092 "    </div>
00093 "  </form>
00094 "</div>
00095
00096 "<script>
00097 "  const formId = 'config-form';
00098 "  const storageKey = 'variador_form_state';
00099 "  function saveFormState() {
00100 "    const form = document.getElementById(formId);
00101 "    if (!form) return;
00102 "    const data = {};
00103 "    Array.from(form.elements).forEach(el => {
00104 "      if (!el.name) return;
00105 "      if (el.type === 'checkbox' || el.type === 'radio') {
00106 "        data[el.name] = el.checked;
00107 "      } else {
00108 "        data[el.name] = el.value;
00109 "      }
00110 "    });
00111 "    try {
00112 "      localStorage.setItem(storageKey, JSON.stringify(data));
00113 "    } catch (e) {
00114 "      console.log('No se pudo guardar estado:', e);
00115 "    }
00116 "  }
00117 "  function loadFormState() {
00118 "    const form = document.getElementById(formId);
00119 "    if (!form) return;
00120 "    const raw = localStorage.getItem(storageKey);
00121 "    if (!raw) return;
00122 "    try {
00123 "      const data = JSON.parse(raw);
00124 "      Array.from(form.elements).forEach(el => {
00125 "        if (!el.name || !(el.name in data)) return;
00126 "        if (el.name === 'cmd') return;
00127 "        if (el.type === 'checkbox' || el.type === 'radio') {
00128 "          el.checked = !!data[el.name];
00129 "        } else {
00130 "          el.value = data[el.name];
00131 "        }
00132 "      });
00133 "    } catch (e) {
00134 "      console.log('No se pudo leer estado:', e);
00135 "    }
00136 "  }
00137 "  window.addEventListener('load', () => {
00138 "    loadFormState();
00139 "    const form = document.getElementById(formId);
00140 "    if (!form) return;
00141 "    form.addEventListener('input', saveFormState);
00142 "  });
00143 "</script>
00144
00145 "</body>

```

```
00146 "</html>";
```

6.49. Referencia del archivo main/wifi/form.h

Variables

- const char * [HTML_FORM](#)

6.49.1. Documentación de variables

6.49.1.1. HTML_FORM

```
const char* HTML_FORM [extern]
```

Definición en la línea 1 del archivo [form.c](#).

6.50. form.h

[Ir a la documentación de este archivo.](#)

```
00001
00002 #ifndef __FORM_H__
00003 #define __FORM_H__
00004
00005 extern const char *HTML_FORM; // definido antes
00006
00007 #endif
```

6.51. Referencia del archivo main/wifi/wifi.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_event.h"
#include "esp_log.h"
#include "nvs_flash.h"
#include "esp_netif.h"
#include "../LVFV_system.h"
#include "../system/sysControl.h"
#include "../display/display.h"
#include "./wifi.h"
#include "form.h"
```

Funciones

- static int `hex2int` (char c)
Convierte un dígito hex a entero (0..15).
- static void `url_decode` (char *dst, const char *src)
Decodifica URL (reemplaza '+' por espacio y HH por byte).
- static bool `get_param_value` (const char *body, const char *key, char *dest, size_t dest_len)
Obtiene el valor de un parámetro key=val del body x-www-form-urlencoded.
- static esp_err_t `root_get_handler` (httpd_req_t *req)
Handler GET "/" – devuelve el formulario HTML.
- static esp_err_t `save_post_handler` (httpd_req_t *req)
Handler POST "/save" – parsea el formulario y actúa.
- httpd_handle_t `start_webserver` (void)
Arranca el servidor HTTP y registra endpoints.
- void `wifi_init_softap` (void)
Configura ESP-NETIF + Wi-Fi en modo AP y arranca SSID/clave. (Canal 1, máx 1 cliente, WPA/WPA2).

Variables

- static const char * `TAG` = "WEB_CFG"

6.51.1. Documentación de funciones

6.51.1.1. `get_param_value()`

```
bool get_param_value (
    const char * body,
    const char * key,
    char * dest,
    size_t dest_len) [static]
```

Obtiene el valor de un parámetro key=val del body x-www-form-urlencoded.

Parámetros

<code>body</code>	cuerpo completo del POST (NUL-terminated).
<code>key</code>	nombre del parámetro (sin '=').
<code>dest</code>	buffer destino para el valor decodificado.
<code>dest_len</code>	tamaño de dest, incluye NUL.

Devuelve

true si se encontró la clave; false si no.

Nota

Devuelve el valor decodificado (URL-decoding).

Definición en la línea 115 del archivo [wifi.c](#).

6.51.1.2. hex2int()

```
int hex2int (
    char c) [static]
```

Convierte un dígito hex a entero (0..15).

Definición en la línea 91 del archivo [wifi.c](#).

6.51.1.3. root_get_handler()

```
esp_err_t root_get_handler (
    httpd_req_t * req) [static]
```

Handler GET "/" – devuelve el formulario HTML.

Definición en la línea 152 del archivo [wifi.c](#).

6.51.1.4. save_post_handler()

```
esp_err_t save_post_handler (
    httpd_req_t * req) [static]
```

Handler POST "/save" – parsea el formulario y actúa.

Flujo: 1) Lee el body (x-www-form-urlencoded) a 'body'. 2) Extrae parámetros: frequency, accel, decel, imax, vmin, variation_str (linear/cuadratica), sys_time, start_time, stop_time. 3) Llena estructuras frequency_edit, seccurity_edit, time_edit. 4) Si cmd=save → system_variables_save(...). Si cmd=start → SystemEventPost([START_PRESSED](#)). Si cmd=stop → SystemEventPost([STOP_PRESSED](#)). 5) Responde nuevamente con el formulario.

Atención

Los campos time_* se cargan como punteros a variables locales (stack). Si [system_variables_save\(\)](#) no copia por valor de inmediato y almacena los punteros, quedarían colgando. Ver "Posibles issues" más abajo.

Definición en la línea 158 del archivo [wifi.c](#).

6.51.1.5. start_webserver()

```
httpd_handle_t start_webserver (
    void )
```

Arranca el servidor HTTP y registra endpoints.

- GET "/" → formulario
- POST "/save" → procesa y responde

Definición en la línea 282 del archivo [wifi.c](#).

6.51.1.6. url_decode()

```
void url_decode (
    char * dst,
    const char * src) [static]
```

Decodifica URL (reemplaza '+' por espacio y HH por byte).

Parámetros

<i>dst</i>	buffer destino (puede ser mismo que src si hay espacio).
<i>src</i>	cadena codificada.

Definición en la línea 98 del archivo [wifi.c](#).

6.51.1.7. wifi_init_softap()

```
void wifi_init_softap (
    void )
```

Configura ESP-NETIF + Wi-Fi en modo AP y arranca SSID/clave. (Canal 1, máx 1 cliente, WPA/WPA2).

Definición en la línea 307 del archivo [wifi.c](#).

6.51.2. Documentación de variables

6.51.2.1. TAG

```
const char* TAG = "WEB_CFG" [static]
```

Definición en la línea 31 del archivo [wifi.c](#).

6.52. wifi.c

[Ir a la documentación de este archivo.](#)

```
00001
00009 #include <stdio.h>
00010 #include <string.h>
00011 #include <stdlib.h>
00012 #include <ctype.h>
00013
00014 #include "freertos/FreeRTOS.h"
00015 #include "freertos/task.h"
00016
00017 #include "esp_event.h"
00018 #include "esp_log.h"
00019 #include "nvs_flash.h"
00020 // #include "mdns.h"
00021
00022 #include "esp_netif.h"
00023
00024 // Tus headers con las structs y funciones:
00025 #include "../LVFV_system.h"
00026 #include "../system/sysControl.h"
00027 #include "../display/display.h"
00028 #include "../wifi.h"
00029 #include "form.h"
00030
00031 static const char *TAG = "WEB_CFG";
00032
00033 /* ----- Helpers para parsear POST x-www-form-urlencoded ----- */
00034
00040 static int hex2int(char c);
00041
00049 static void url_decode(char *dst, const char *src);
00050
00063 static bool get_param_value(const char *body, const char *key, char *dest, size_t dest_len);
00064
00070 static esp_err_t root_get_handler(httpd_req_t *req);
00071
00089 static esp_err_t save_post_handler(httpd_req_t *req);
```

```

00090
00091 static int hex2int(char c) {
00092     if (c >= '0' && c <= '9') return c - '0';
00093     if (c >= 'a' && c <= 'f') return 10 + (c - 'a');
00094     if (c >= 'A' && c <= 'F') return 10 + (c - 'A');
00095     return 0;
00096 }
00097
00098 static void url_decode(char *dst, const char *src) {
00099     while (*src) {
00100         if (*src == '+') {
00101             *dst++ = ' ';
00102             src++;
00103         } else if (*src == '%' && isxdigit((unsigned char)src[1]) && isxdigit((unsigned char)src[2])) {
00104             int hi = hex2int(src[1]);
00105             int lo = hex2int(src[2]);
00106             *dst++ = (char)((hi << 4) | lo);
00107             src += 3;
00108         } else {
00109             *dst++ = *src++;
00110         }
00111     }
00112     *dst = '\0';
00113 }
00114
00115 static bool get_param_value(const char *body, const char *key, char *dest, size_t dest_len) {
00116     size_t key_len = strlen(key);
00117     const char *p = body;
00118
00119     while (p && *p) {
00120         const char *eq = strchr(p, '=');
00121         if (!eq) break;
00122
00123         // Nombre de parámetro
00124         size_t this_key_len = (size_t)(eq - p);
00125
00126         if (this_key_len == key_len && strncmp(p, key, key_len) == 0) {
00127             // Encontramos la clave. Buscar fin del valor (& o final de cadena)
00128             const char *val_start = eq + 1;
00129             const char *amp = strchr(val_start, '&');
00130             size_t val_len = amp ? (size_t)(amp - val_start) : strlen(val_start);
00131             if (val_len >= dest_len) val_len = dest_len - 1;
00132
00133             char encoded[256];
00134             if (val_len >= sizeof(encoded)) val_len = sizeof(encoded) - 1;
00135
00136             memcpy(encoded, val_start, val_len);
00137             encoded[val_len] = '\0';
00138
00139             url_decode(dest, encoded);
00140             return true;
00141         }
00142
00143         // Ir al siguiente par key=value
00144         const char *amp = strchr(eq + 1, '&');
00145         if (!amp) break;
00146         p = amp + 1;
00147     }
00148
00149     return false;
00150 }
00151
00152 static esp_err_t root_get_handler(httpd_req_t *req) {
00153     httpd_resp_set_type(req, "text/html");
00154     httpd_resp_send(req, HTML_FORM, HTTPD_RESP_USE_STRLEN);
00155     return ESP_OK;
00156 }
00157
00158 static esp_err_t save_post_handler(httpd_req_t *req) {
00159     char body[512];
00160     char cmd[16] = {0};
00161
00162     time_settings_SH1106_t time_edit;
00163     security_settings_SH1106_t seccurity_edit;
00164     frequency_settings_SH1106_t frequency_edit;
00165
00166     if (req->content_len >= sizeof(body)) {
00167         ESP_LOGW(TAG, "Body demasiado grande (%d)", req->content_len);
00168         httpd_resp_send_err(req, HTTPD_500_INTERNAL_SERVER_ERROR, "Body too large");
00169         return ESP_FAIL;
00170     }
00171
00172     int received = httpd_req_recv(req, body, req->content_len);
00173     if (received <= 0) {
00174         ESP_LOGW(TAG, "Error recibiendo body");
00175         httpd_resp_send_err(req, HTTPD_500_INTERNAL_SERVER_ERROR, "Receive error");
}

```

```

00176     return ESP_FAIL;
00177 }
00178 body[received] = '\0';
00179
00180 ESP_LOGI(TAG, "Body: %s", body);
00181
00182 char buf[64];
00183
00184 /* ----- Leer parámetros ----- */
00185
00186 int frequency = 0, accel = 0, decel = 0, variation = 0;
00187 int imax = 0, vmin = 0;
00188 char sys_time_str[16] = {0};
00189 char start_time_str[16] = {0};
00190 char stop_time_str[16] = {0};
00191
00192 if (get_param_value(body, "frequency", buf, sizeof(buf))) {
00193     frequency = atoi(buf);
00194     ESP_LOGI(TAG, "Frecuencia: %d", frequency);
00195 }
00196 if (get_param_value(body, "accel", buf, sizeof(buf))) {
00197     accel = atoi(buf);
00198     ESP_LOGI(TAG, "Aceleración: %d", accel);
00199 }
00200 if (get_param_value(body, "decel", buf, sizeof(buf))) {
00201     decel = atoi(buf);
00202     ESP_LOGI(TAG, "Desaceleración: %d", decel);
00203 }
00204 if (get_param_value(body, "imax", buf, sizeof(buf))) {
00205     imax = atoi(buf);
00206     ESP_LOGI(TAG, "Corriente maxima: %d", imax);
00207 }
00208 if (get_param_value(body, "vmin", buf, sizeof(buf))) {
00209     vmin = atoi(buf);
00210     ESP_LOGI(TAG, "Tension minima: %d", vmin);
00211 }
00212 if (get_param_value(body, "variation", buf, sizeof(buf))) {
00213     variation = atoi(buf);
00214     ESP_LOGI(TAG, "Variacion de entradas %d", variation);
00215 }
00216 get_param_value(body, "sys_time", sys_time_str, sizeof(sys_time_str));
00217 get_param_value(body, "start_time", start_time_str, sizeof(start_time_str));
00218 get_param_value(body, "stop_time", stop_time_str, sizeof(stop_time_str));
00219
00220 /* ----- Parsear horas ----- */
00221
00222 int sys_h=0, sys_m=0, sys_s=0;
00223 int start_h=0, start_m=0;
00224 int stop_h=0, stop_m=0;
00225
00226 if (sscanf(sys_time_str, "%2d:%2d:%2d", &sys_h, &sys_m, &sys_s) != 3) {
00227     ESP_LOGW(TAG, "Hora de sistema inválida: %s", sys_time_str);
00228 }
00229 if (sscanf(start_time_str, "%2d:%2d", &start_h, &start_m) != 2) {
00230     ESP_LOGW(TAG, "Hora de arranque inválida: %s", start_time_str);
00231 }
00232 if (sscanf(stop_time_str, "%2d:%2d", &stop_h, &stop_m) != 2) {
00233     ESP_LOGW(TAG, "Hora de parada inválida: %s", stop_time_str);
00234 }
00235 ESP_LOGI(TAG, "Hora de sistema: %02d:%02d:%02d", sys_h, sys_m, sys_s);
00236 ESP_LOGI(TAG, "Hora de arranque: %02d:%02d", start_h, start_m);
00237 ESP_LOGI(TAG, "Hora de parada: %02d:%02d", stop_h, stop_m);
00238
00239 struct tm time_system;
00240 time_system.tm_hour = sys_h;
00241 time_system.tm_min = sys_m;
00242 time_system.tm_sec = sys_s;
00243 struct tm time_start;
00244 time_start.tm_hour = start_h;
00245 time_start.tm_min = start_m;
00246 struct tm time_stop;
00247 time_stop.tm_hour = stop_h;
00248 time_stop.tm_min = stop_m;
00249
00250 frequency_edit.frequency_settings.freq_regime = frequency;
00251 frequency_edit.frequency_settings.acceleration = accel;
00252 frequency_edit.frequency_settings.desacceleration = decel;
00253 frequency_edit.frequency_settings.input_variable = variation;
00254
00255 security_edit.security_settings.ibus_max = imax;
00256 security_edit.security_settings.vbus_min = vmin;
00257
00258 time_edit.time_settings.time_system = &time_system;
00259 time_edit.time_settings.time_start = &time_start;
00260 time_edit.time_settings.time_stop = &time_stop;
00261
00262

```

```

00263     if (get_param_value(body, "cmd", cmd, sizeof(cmd))) {
00264         ESP_LOGI(TAG, "Comando: %s", cmd);
00265         if (strcmp(cmd, "save") == 0) {
00266             system_variables_save(&frequency_edit, &time_edit, &security_edit);
00267         } else if (strcmp(cmd, "start") == 0) {
00268             SystemEventPost(START_PRESSED);
00269         } else if (strcmp(cmd, "stop") == 0) {
00270             SystemEventPost(STOP_PRESSED);
00271         }
00272     }
00273
00274     /* ----- Respuesta al navegador ----- */
00275
00276     httpd_resp_set_type(req, "text/html");
00277     httpd_resp_send(req, HTML_FORM, HTTPD_RESP_USE_STRLEN);
00278
00279     return ESP_OK;
00280 }
00281
00282 httpd_handle_t start_webserver(void) {
00283     httpd_config_t config = HTTPD_DEFAULT_CONFIG();
00284     config.server_port = 80;
00285
00286     httpd_handle_t server = NULL;
00287     if (httpd_start(&server, &config) == ESP_OK) {
00288         httpd_uri_t root_uri = {
00289             .uri      = "/",
00290             .method   = HTTP_GET,
00291             .handler  = root_get_handler,
00292             .user_ctx = NULL
00293         };
00294         httpd_register_uri_handler(server, &root_uri);
00295
00296         httpd_uri_t save_uri = {
00297             .uri      = "/save",
00298             .method   = HTTP_POST,
00299             .handler  = save_post_handler,
00300             .user_ctx = NULL
00301         };
00302         httpd_register_uri_handler(server, &save_uri);
00303     }
00304     return server;
00305 }
00306
00307 void wifi_init_softap(void) {
00308     ESP_ERROR_CHECK(esp_netif_init());
00309     ESP_ERROR_CHECK(esp_event_loop_create_default());
00310     esp_netif_create_default_wifi_ap();
00311
00312     wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
00313     ESP_ERROR_CHECK(esp_wifi_init(&cfg));
00314
00315     wifi_config_t wifi_config = {
00316         .ap = {
00317             .ssid = "LVFV_2025",
00318             .ssid_len = 0,
00319             .channel = 1,
00320             .password = "LVFV_2025",
00321             .max_connection = 1,
00322             .authmode = WIFI_AUTH_WPA_WPA2_PSK
00323         },
00324     };
00325     if (strlen((char *)wifi_config.ap.password) == 0) {
00326         wifi_config.ap.authmode = WIFI_AUTH_OPEN;
00327     }
00328
00329     ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_AP));
00330     ESP_ERROR_CHECK(esp_wifi_set_config(WIFI_IF_AP, &wifi_config));
00331     ESP_ERROR_CHECK(esp_wifi_start());
00332
00333     ESP_LOGI(TAG, "AP iniciado. SSID:%s, pass:%s", wifi_config.ap.ssid, wifi_config.ap.password);
00334 }

```

6.53. Referencia del archivo main/wifi/wifi.h

```
#include "esp_wifi.h"
#include "esp_http_server.h"
```

Funciones

- `httpd_handle_t start_webserver (void)`

Arranca el servidor HTTP y registra endpoints.

- `void wifi_init_softap (void)`

Configura ESP-NETIF + Wi-Fi en modo AP y arranca SSID/clave. (Canal 1, máx 1 cliente, WPA/WPA2).

6.53.1. Documentación de funciones

6.53.1.1. `start_webserver()`

```
httpd_handle_t start_webserver (
    void )
```

Arranca el servidor HTTP y registra endpoints.

- GET "/" → formulario
- POST "/save" → procesa y responde

Definición en la línea [282](#) del archivo `wifi.c`.

6.53.1.2. `wifi_init_softap()`

```
void wifi_init_softap (
    void )
```

Configura ESP-NETIF + Wi-Fi en modo AP y arranca SSID/clave. (Canal 1, máx 1 cliente, WPA/WPA2).

Definición en la línea [307](#) del archivo `wifi.c`.

6.54. wifi.h

[Ir a la documentación de este archivo.](#)

```
00001
00002 #ifndef __MAIN_WIFI_WIFI_H__
00003 #define __MAIN_WIFI_WIFI_H__
00004
00005 #include "esp_wifi.h"
00006 #include "esp_http_server.h"
00007
00015 httpd_handle_t start_webserver(void);
00016
00022 void wifi_init_softap(void);
00023 // void start_mdns(void);
00024
00025 #endif
```


Índice alfabético

__MCP23017_ADDRESS__
 mcp23017_defs.h, 150
__MCP23017_BANK_SEQUENTIAL__
 mcp23017_defs.h, 150
__MCP23017_BANK_SPLIT__
 mcp23017_defs.h, 150
__MCP23017_DISSLW_DISABLED__
 mcp23017_defs.h, 150
__MCP23017_DISSLW_ENABLED__
 mcp23017_defs.h, 151
__MCP23017_FUNC_MODE__
 mcp23017_defs.h, 151
__MCP23017_GPINTEN_DISABLE__
 mcp23017_defs.h, 151
__MCP23017_GPINTEN_ENABLE__
 mcp23017_defs.h, 151
__MCP23017_GPPU_PULL_UP_DISABLE__
 mcp23017_defs.h, 151
__MCP23017_GPPU_PULL_UP_ENABLE__
 mcp23017_defs.h, 151
__MCP23017_INTCON_CHANGE__
 mcp23017_defs.h, 152
__MCP23017_INTCON_DEFVAL__
 mcp23017_defs.h, 152
__MCP23017_INTERRUPT_DISABLE__
 mcp23017_defs.h, 152
__MCP23017_INTERRUPT_ENABLE__
 mcp23017_defs.h, 152
__MCP23017_INTPOL_POLARITY_HIGH__
 mcp23017_defs.h, 152
__MCP23017_INTPOL_POLARITY_LOW__
 mcp23017_defs.h, 152
__MCP23017_IODIR_INPUT__
 mcp23017_defs.h, 153
__MCP23017_IODIR_OUTPUT__
 mcp23017_defs.h, 153
__MCP23017_MIRROR_CONNECTED__
 mcp23017_defs.h, 153
__MCP23017_MIRROR_UNCONNECTED__
 mcp23017_defs.h, 153
__MCP23017_ODR_ACTIVE_DRIVER_OUTPUT__
 mcp23017_defs.h, 153
__MCP23017_ODR_OPEN_DRAIN_OUTPUT__
 mcp23017_defs.h, 153
__MCP23017_OPPOSITE_LOGIC__
 mcp23017_defs.h, 154
__MCP23017_POSITIVE_LOGIC__
 mcp23017_defs.h, 154
__MCP23017_READ_OPCODE__
 mcp23017_defs.h, 154
__MCP23017_READ__
 mcp23017_defs.h, 154
__MCP23017_SEQOP_DISABLED__
 mcp23017_defs.h, 154
__MCP23017_SEQOP_ENABLED__
 mcp23017_defs.h, 154
__MCP23017_WRITE_OPCODE__
 mcp23017_defs.h, 155
__MCP23017_WRITE__
 mcp23017_defs.h, 155

accelerating
 sysAdmin.c, 199

accelerating_handle
 sysAdmin.c, 204

acceleration
 frequency_settings_t, 14
 system_status_t, 42

adc.c
 ADC_ATTEN_USED, 48
 adc_cali_try_init, 50
 ADC_CH_GPIO34, 48
 ADC_CH_GPIO35, 48
 ADC_CH_GPIO36, 49
 ADC_CH_GPIO39, 49
 adc_init, 51
 ADC_PERIOD_MS, 49
 adc_task, 51
 ADC_UNIT_USED, 49
 bus_meas_evt_queue, 53
 calibration_3V3_source, 53
 calibration_3V3_source_ok, 53
 calibration_5V_source, 53
 calibration_5V_source_ok, 53
 calibration_bus_voltage, 53
 calibration_bus_voltage_ok, 53
 calibration_current, 54
 calibration_current_ok, 54
 readADC, 51
 s_adc, 54
 TAG, 54

adc.h
 adc_init, 58
 adc_task, 58
 readADC, 58

ADC_ATTEN_USED
 adc.c, 48

adc_cali_try_init

adc.c, 50
 ADC_CH_GPIO34
 adc.c, 48
 ADC_CH_GPIO35
 adc.c, 48
 ADC_CH_GPIO36
 adc.c, 49
 ADC_CH_GPIO39
 adc.c, 49
 adc_init
 adc.c, 51
 adc.h, 58
 ADC_PERIOD_MS
 adc.c, 49
 adc_task
 adc.c, 51
 adc.h, 58
 ADC_UNIT_USED
 adc.c, 49
 alarm_settings
 rtc.c, 192
 alarm_start_cb
 rtc.c, 189
 alarm_stop_cb
 rtc.c, 189
 all
 MCP23017_DEFVAL_t, 15
 MCP23017_GPINTEN_t, 17
 MCP23017_GPIO_t, 19
 MCP23017_GPPU_t, 21
 MCP23017_INTCAP_t, 24
 MCP23017_INTCON_t, 26
 MCP23017_INTF_t, 28
 MCP23017_IOCON_t, 30
 MCP23017_IODIR_t, 32
 MCP23017_IPOL_t, 34
 MCP23017_OLAT_t, 36
 app_main
 main.c, 171
 arrow_bmp
 sh1106_graphics.c, 89

 BANK
 MCP23017_IOCON_t, 30
 bitmap
 bitmap_t, 11
 bitmap_t, 11
 bitmap, 11
 h, 11
 sh1106_graphics.c, 87
 w, 12
 x, 12
 y, 12
 bits
 MCP23017_DEFVAL_t, 15
 MCP23017_GPINTEN_t, 17
 MCP23017_GPIO_t, 19
 MCP23017_GPPU_t, 21
 MCP23017_INTCAP_t, 24

 MCP23017_INTCON_t, 26
 MCP23017_INTF_t, 28
 MCP23017_IOCON_t, 30
 MCP23017_IODIR_t, 32
 MCP23017_IPOL_t, 34
 MCP23017_OLAT_t, 36

 blink
 display.c, 70
 buffer
 sh1106_t, 40
 bus_meas_evt_queue
 adc.c, 53
 BUTTON_BACK
 LVFV_system.h, 167
 BUTTON_DOWN
 LVFV_system.h, 167
 button_evt_queue
 display.c, 70
 BUTTON_LEFT
 LVFV_system.h, 167
 BUTTON_MENU
 LVFV_system.h, 167
 BUTTON_OK
 LVFV_system.h, 167
 BUTTON_RIGHT
 LVFV_system.h, 167
 BUTTON_SAVE
 LVFV_system.h, 167
 BUTTON_UP
 LVFV_system.h, 167
 buzzer_evt_queue
 io_control.c, 120
 BUZZER_PIN
 io_control.c, 111
 BuzzerEventPost
 io_control.c, 117
 io_control.h, 129

 calibration_3V3_source
 adc.c, 53
 calibration_3V3_source_ok
 adc.c, 53
 calibration_5V_source
 adc.c, 53
 calibration_5V_source_ok
 adc.c, 53
 calibration_bus_voltage
 adc.c, 53
 calibration_bus_voltage_ok
 adc.c, 53
 calibration_current
 adc.c, 54
 calibration_current_ok
 adc.c, 54
 change_frequency
 sysAdmin.c, 199
 sysAdmin.h, 209
 CMD_CONTROL
 sh1106_i2c.c, 105

DATA_CONTROL
 sh1106_i2c.c, 105

DEF0
 MCP23017_DEFVAL_t, 15

DEF1
 MCP23017_DEFVAL_t, 15

DEF2
 MCP23017_DEFVAL_t, 15

DEF3
 MCP23017_DEFVAL_t, 16

DEF4
 MCP23017_DEFVAL_t, 16

DEF5
 MCP23017_DEFVAL_t, 16

DEF6
 MCP23017_DEFVAL_t, 16

DEF7
 MCP23017_DEFVAL_t, 16

desaccelerating
 sysAdmin.c, 200

desaccelerating_handle
 sysAdmin.c, 204

desacceleration
 frequency_settings_t, 14
 system_status_t, 42

display.c
 blink, 70
 button_evt_queue, 70
 DisplayEventPost, 65
 EDIT_ACCELERATION_INDX, 61
 EDIT_DESACCELERATION_INDX, 61
 EDIT_FREQUENCY_INDX, 62
 EDIT_HFIN_LINE_INDX, 62
 EDIT_HINI_LINE_INDX, 62
 EDIT_IBUS_LINE_INDX, 62
 EDIT_INPUT_VARIATION_INDX, 62
 EDIT_TIME_LINE_INDX, 62
 EDIT_VBUS_LINE_INDX, 62
 FREC_LINE_INDX, 62
 FREQUENCY CUADRATIC VARIABLE, 63
 FREQUENCY LINEAR VARIABLE, 63
 get_system_acceleration, 65
 get_system_desacceleration, 65
 get_system_frequency, 65
 HFIN_LINE_INDX, 63
 HINI_LINE_INDX, 63
 I2C DISPLAY MASTER_NUM, 63
 I2C_MASTER_FREQ_HZ, 63
 i2c_master_init, 66
 I2C_MASTER_RX_BUF_DISABLE, 63
 I2C_MASTER_SCL_IO, 63
 I2C_MASTER_SDA_IO, 64
 I2C_MASTER_TX_BUF_DISABLE, 64
 IBUS_LINE_INDX, 64
 init_seq, 70
 oled, 70
 screen_displayed, 70
 SCREEN_FREQUENCY_EDIT, 65
 SCREEN_MAIN, 64
 SCREEN_SECURITY_EDIT, 64
 SCREEN_SELECT_VARIABLE, 64
 screen_selected_e, 64
 SCREEN_TIME_EDIT, 64
 sh1106_clear_buffer, 66
 sh1106_frequency_edit_variables, 66
 sh1106_init, 66
 sh1106_main_screen, 66
 sh1106_print_emergency, 67
 sh1106_print_frame, 67
 sh1106_refresh, 67
 sh1106_security_edit_variables, 67
 sh1106_select_edit_variables, 68
 sh1106_splash_screen, 68
 sh1106_time_edit_variables, 68
 system_frequency_settings, 70
 system_seccurity_settings, 71
 system_time_settings, 71
 system_variables_save, 68
 TAG, 71
 task_display, 69
 VBUS_LINE_INDX, 64

display.h
 DisplayEventPost, 82
 get_system_acceleration, 83
 get_system_desacceleration, 83
 get_system_frequency, 83
 sh1106_init, 83
 system_variables_save, 84
 task_display, 84

DisplayEventPost
 display.c, 65
 display.h, 82

DISSLW
 MCP23017_IOCON_t, 30

DUTY_MAX_PCT
 io_control.c, 111

DUTY_MIN_PCT
 io_control.c, 111

edit
 frequency_settings_SH1106_t, 13
 seccurity_settings_SH1106_t, 38
 time_settings_SH1106_t, 44

EDIT_ACCELERATION_INDX
 display.c, 61

EDIT_DESACCELERATION_INDX
 display.c, 61

edit_flag
 frequency_settings_SH1106_t, 13
 seccurity_settings_SH1106_t, 38
 time_settings_SH1106_t, 44

EDIT_FREQUENCY_INDX
 display.c, 62

EDIT_HFIN_LINE_INDX
 display.c, 62

EDIT_HINI_LINE_INDX
 display.c, 62

EDIT_IBUS_LINE_INDX
 display.c, 62

EDIT_INPUT_VARIATION_INDX
 display.c, 62

EDIT_TIME_LINE_INDX
 display.c, 62

edit_variable
 frequency_settings_SH1106_t, 13
 seccurity_settings_SH1106_t, 39
 time_settings_SH1106_t, 44

EDIT_VBUS_LINE_INDX
 display.c, 62

EMERGENCI_STOP_PRESSED
 LVFV_system.h, 167

EMERGENCI_STOP_RELEASED
 LVFV_system.h, 167

emergency_signals
 system_status_t, 43

engine_emergency_stop
 sysAdmin.c, 200
 sysAdmin.h, 209

engine_emergency_stop_release
 sysAdmin.c, 200
 sysAdmin.h, 209

engine_start
 sysAdmin.c, 201
 sysAdmin.h, 210

engine_stop
 sysAdmin.c, 201
 sysAdmin.h, 210

fail_bmp
 sh1106_graphics.c, 89

fifth
 LVFV_system.h, 166

first
 LVFV_system.h, 166

font5x7_close_excl
 sh1106_graphics.c, 89

font5x7_close_quest
 sh1106_graphics.c, 89

font5x7_comma
 sh1106_graphics.c, 89

font5x7_dot
 sh1106_graphics.c, 89

font5x7_double_dot
 sh1106_graphics.c, 89

font5x7_minus
 sh1106_graphics.c, 90

font5x7_rowmajor
 sh1106_graphics.c, 90

font5x7_rowminor
 sh1106_graphics.c, 90

font5x7_rownumber
 sh1106_graphics.c, 91

font5x7_space
 sh1106_graphics.c, 92

font8x14_close_excl
 sh1106_graphics.c, 92

font8x14_close_quest
 sh1106_graphics.c, 92

font8x14_comma
 sh1106_graphics.c, 92

font8x14_dot
 sh1106_graphics.c, 92

font8x14_double_dot
 sh1106_graphics.c, 92

font8x14_minus
 sh1106_graphics.c, 93

font8x14_rowmajor
 sh1106_graphics.c, 93

font8x14_rowminor
 sh1106_graphics.c, 93

font8x14_rownumber
 sh1106_graphics.c, 94

font8x14_space
 sh1106_graphics.c, 95

form.c
 HTML_FORM, 228

form.h
 HTML_FORM, 230

fourth
 LVFV_system.h, 166

FREC_LINE_INDX
 display.c, 62

freq_0_10V_output
 io_control.c, 118

freq_regime
 frequency_settings_t, 14

FREQ_SEL_1_PIN
 io_control.c, 111

FREQ_SEL_2_PIN
 io_control.c, 112

FREQ_SEL_3_PIN
 io_control.c, 112

FREQ_SEL_MASK
 io_control.c, 112

frequency
 system_status_t, 43

FREQUENCY CUADRATIC VARIABLE
 display.c, 63

frequency_destiny
 system_status_t, 43

FREQUENCY_LINEAR_VARIABLE
 display.c, 63

frequency_settings
 frequency_settings_SH1106_t, 13

frequency_settings_SH1106_t, 12
 edit, 13
 edit_flag, 13
 edit_variable, 13
 frequency_settings, 13
 LVFV_system.h, 165
 multiplier, 13

frequency_settings_t, 13
 acceleration, 14
 desacceleration, 14

freq_regime, 14
input_variable, 14
LVFV_system.h, 165
frequency_table
 sysAdmin.c, 204

get_param_value
 wifi.c, 231
get_status
 sysAdmin.c, 201
 sysAdmin.h, 210
get_system_acceleration
 display.c, 65
 display.h, 83
get_system_desacceleration
 display.c, 65
 display.h, 83
get_system_frequency
 display.c, 65
 display.h, 83
getTime
 rtc.c, 190
 rtc.h, 196
getValue
 spi_cmd_item_t, 41
GP0
 MCP23017_GPIO_t, 19
GP1
 MCP23017_GPIO_t, 20
GP2
 MCP23017_GPIO_t, 20
GP3
 MCP23017_GPIO_t, 20
GP4
 MCP23017_GPIO_t, 20
GP5
 MCP23017_GPIO_t, 20
GP6
 MCP23017_GPIO_t, 20
GP7
 MCP23017_GPIO_t, 21
GPINT0
 MCP23017_GPINTEN_t, 17
GPINT1
 MCP23017_GPINTEN_t, 17
GPINT2
 MCP23017_GPINTEN_t, 18
GPINT3
 MCP23017_GPINTEN_t, 18
GPINT4
 MCP23017_GPINTEN_t, 18
GPINT5
 MCP23017_GPINTEN_t, 18
GPINT6
 MCP23017_GPINTEN_t, 18
GPINT7
 MCP23017_GPINTEN_t, 18
GPIO_evt_queue
 io_control.c, 120

gpio_init_interrupts
 io_control.c, 118
GPIO_interrupt_attendance_task
 io_control.c, 118
 io_control.h, 129
gpio_isr_handler
 io_control.c, 119

h
 bitmap_t, 11
height
 sh1106_t, 40
hex2int
 wifi.c, 231
HFIN_LINE_INDX
 display.c, 63
HINI_LINE_INDX
 display.c, 63
HTML_FORM
 form.c, 228
 form.h, 230

I2C_DISPLAY
 sh1106_i2c.c, 105
I2C_DISPLAY_MASTER_NUM
 display.c, 63
I2C_IO_MASTER_NUM
 MCP23017.c, 133
i2c_is_hardware_free
 MCP23017.c, 134
I2C_MASTER_FREQ_HZ
 display.c, 63
 MCP23017.c, 133
i2c_master_init
 display.c, 66
 MCP23017.c, 134
I2C_MASTER_RX_BUF_DISABLE
 display.c, 63
 MCP23017.c, 133
I2C_MASTER_SCL_IO
 display.c, 63
 MCP23017.c, 134
I2C_MASTER_SDA_IO
 display.c, 64
 MCP23017.c, 134
I2C_MASTER_TIMEOUT_MS
 MCP23017.c, 134
I2C_MASTER_TX_BUF_DISABLE
 display.c, 64
 MCP23017.c, 134
i2c_release_hardware
 MCP23017.c, 135
IBUS_LINE_INDX
 display.c, 64
ibus_max
 seccurity_settings_t, 40
 system_status_t, 43
ICP0
 MCP23017_INTCAP_t, 24

ICP1
 MCP23017_INTCAP_t, 24
 ICP2
 MCP23017_INTCAP_t, 24
 ICP3
 MCP23017_INTCAP_t, 24
 ICP4
 MCP23017_INTCAP_t, 24
 ICP5
 MCP23017_INTCAP_t, 25
 ICP6
 MCP23017_INTCAP_t, 25
 ICP7
 MCP23017_INTCAP_t, 25
 init_seq
 display.c, 70
 initRTC
 rtc.c, 190
 rtc.h, 196
 input_variable
 frequency_settings_t, 14
 inputs_status
 system_status_t, 43
 INT0
 MCP23017_INTF_t, 28
 INT1
 MCP23017_INTF_t, 28
 INT2
 MCP23017_INTF_t, 28
 INT3
 MCP23017_INTF_t, 28
 INT4
 MCP23017_INTF_t, 28
 INT5
 MCP23017_INTF_t, 29
 INT6
 MCP23017_INTF_t, 29
 INT7
 MCP23017_INTF_t, 29
 INT_A_PIN
 io_control.c, 112
 INT_B_PIN
 io_control.c, 112
 INTPOL
 MCP23017_IOCON_t, 30
 IO0
 MCP23017_IODIR_t, 32
 IO1
 MCP23017_IODIR_t, 32
 IO2
 MCP23017_IODIR_t, 32
 IO3
 MCP23017_IODIR_t, 32
 IO4
 MCP23017_IODIR_t, 32
 IO5
 MCP23017_IODIR_t, 33
 IO6
 MCP23017_IODIR_t, 33
 MCP23017_IODIR_t, 33
 IO7
 MCP23017_IODIR_t, 33
 io_control.c
 buzzer_evt_queue, 120
 BUZZER_PIN, 111
 BuzzerEventPost, 117
 DUTY_MAX_PCT, 111
 DUTY_MIN_PCT, 111
 freq_0_10V_output, 118
 FREQ_SEL_1_PIN, 111
 FREQ_SEL_2_PIN, 112
 FREQ_SEL_3_PIN, 112
 FREQ_SEL_MASK, 112
 GPIO_evt_queue, 120
 gpio_init_interrupts, 118
 GPIO_interrupt_attendance_task, 118
 gpio_isr_handler, 119
 INT_A_PIN, 112
 INT_B_PIN, 112
 MATRIX_SW1, 112
 MATRIX_SW2, 113
 MATRIX_SW3, 113
 MATRIX_SW4, 113
 MATRIX_SW5, 113
 MATRIX_SW6, 113
 MATRIX_SW7, 113
 MATRIX_SW8, 114
 MCP23017_buzzer_control, 119
 MCP23017_keyboard_control, 119
 MCP23017_relay_control, 119
 mcp_porta, 120
 mcp_portb, 121
 PWM_CHANNEL, 114
 PWM_FREQ_HZ, 114
 PWM_GPIO, 114
 PWM_MODE, 114
 PWM_RES, 114
 PWM_STEP_MS, 115
 PWM_TIMER, 115
 relay_evt_queue, 121
 RELAY_PIN, 115
 RelayEventPost, 119
 set_freq_output, 120
 START_BUTTON_PIN, 115
 STOP_BUTTON_PIN, 115
 STOP_PIN, 115
 STOP_SW_MASK, 116
 SWITCH_BUTTON_BACK, 116
 SWITCH_BUTTON_DOWN, 116
 SWITCH_BUTTON_LEFT, 116
 SWITCH_BUTTON_MENU, 116
 SWITCH_BUTTON_OK, 116
 SWITCH_BUTTON_RIGHT, 117
 SWITCH_BUTTON_SAVE, 117
 SWITCH_BUTTON_UP, 117
 TAG, 121
 TERMO_SW_MASK, 117

TERMO_SW_PIN, 117
io_control.h
 BuzzerEventPost, 129
 GPIO_interrupt_attendance_task, 129
 RelayEventPost, 129
 set_freq_output, 131
IOC0
 MCP23017_INTCON_t, 26
IOC1
 MCP23017_INTCON_t, 26
IOC2
 MCP23017_INTCON_t, 26
IOC3
 MCP23017_INTCON_t, 26
IOC4
 MCP23017_INTCON_t, 26
IOC5
 MCP23017_INTCON_t, 27
IOC6
 MCP23017_INTCON_t, 27
IOC7
 MCP23017_INTCON_t, 27
IP0
 MCP23017_IPOL_t, 34
IP1
 MCP23017_IPOL_t, 34
IP2
 MCP23017_IPOL_t, 34
IP3
 MCP23017_IPOL_t, 34
IP4
 MCP23017_IPOL_t, 34
IP5
 MCP23017_IPOL_t, 35
IP6
 MCP23017_IPOL_t, 35
IP7
 MCP23017_IPOL_t, 35

line_bmp
 sh1106_graphics.c, 95
LINE_INCREMENT
 LVFV_system.h, 164
load_16
 nvs.c, 177
load_acceleration
 nvs.c, 173
load_desacceleration
 nvs.c, 173
load_frequency
 nvs.c, 174
load_hour_fin
 nvs.c, 174
load_hour_ini
 nvs.c, 174
load_ibus_max
 nvs.c, 174
load_input_variable
 nvs.c, 174

load_min_fin
 nvs.c, 175
load_min_ini
 nvs.c, 175
load_variables
 nvs.c, 178
 nvs.h, 186
load_vbus_min
 nvs.c, 175
logo16_utn bmp
 sh1106_graphics.c, 95
LVFV_system.h
 BUTTON_BACK, 167
 BUTTON_DOWN, 167
 BUTTON_LEFT, 167
 BUTTON_MENU, 167
 BUTTON_OK, 167
 BUTTON_RIGHT, 167
 BUTTON_SAVE, 167
 BUTTON_UP, 167
 EMERGENCI_STOP_PRESSED, 167
 EMERGENCI_STOP_RELEASED, 167
fifth, 166
first, 166
fourth, 166
frequency_settings_SH1106_t, 165
frequency_settings_t, 165
LINE_INCREMENT, 164
seccurity_settings_SH1106_t, 165
seccurity_settings_t, 165
second, 166
SECURITY_EXCEEDED, 168
SECURITY_OK, 168
seventh, 166
SH1106_SIZE_1, 164
SH1106_SIZE_2, 164
sh1106_variable_lines_e, 165, 166
sixth, 166
SPEED_SELECTOR_0, 167
SPEED_SELECTOR_1, 167
SPEED_SELECTOR_2, 167
SPEED_SELECTOR_3, 167
SPEED_SELECTOR_4, 167
SPEED_SELECTOR_5, 168
SPEED_SELECTOR_6, 168
SPEED_SELECTOR_7, 168
SPEED_SELECTOR_8, 168
SPEED_SELECTOR_9, 168
START_PRESSED, 167
START_RELEASED, 167
STOP_PRESSED, 167
STOP_RELEASED, 167
SYSTEM_ACCEL_DESACCEL, 167
SYSTEM_BREAKING, 167
SYSTEM_EMERGENCY, 167
SYSTEM_EMERGENCY_OK, 167
SYSTEM_IDLE, 167
SYSTEM_REGIME, 167

system_status_e, 166
 system_status_t, 166
 systemSignal_e, 167
 TERMO_SW_PRESSED, 167
 TERMO_SW_RELEASED, 167
 third, 166
 time_settings_SH1106_t, 166
 time_settings_t, 166
 VARIABLE_EIGHT, 164
 VARIABLE_FIFTH, 164
 VARIABLE_FIRST, 164
 VARIABLE_FOURTH, 164
 VARIABLE_SECOND, 164
 VARIABLE_SEVENTH, 165
 VARIABLE_SIXTH, 165
 VARIABLE_THIRD, 165

main.c
 app_main, 171
 TAG, 171

main/adc/adc.c, 47, 54
main/adc/adc.h, 57, 59

main/display/display.c, 59, 71
main/display/display.h, 81, 85
main/display/sh1106_graphics.c, 85, 96
main/display/sh1106_graphics.h, 101, 104

main/display/sh1106_i2c.c, 104, 106
main/display/sh1106_i2c.h, 107, 108

main/io_control/io_control.c, 109, 121
main/io_control/io_control.h, 128, 131
main/io_control/MCP23017.c, 132, 139
main/io_control/MCP23017.h, 143, 148
main/io_control/mcp23017_defs.h, 148, 159

main/LVVF_system.h, 162, 168

main/main.c, 170, 171

main/nvs/nvs.c, 172, 181
main/nvs/nvs.h, 185, 188

main/rtc/rtc.c, 188, 193
main/rtc/rtc.h, 195, 197

main/system/sysAdmin.c, 198, 205
main/system/sysAdmin.h, 208, 213

main/system/sysControl.c, 213, 218
main/system/sysControl.h, 222, 227

main/wifi/form.c, 227, 228
main/wifi/form.h, 230

main/wifi/wifi.c, 230, 233
main/wifi/wifi.h, 236, 237

MATRIX_SW1
 io_control.c, 112

MATRIX_SW2
 io_control.c, 113

MATRIX_SW3
 io_control.c, 113

MATRIX_SW4
 io_control.c, 113

MATRIX_SW5
 io_control.c, 113

MATRIX_SW6
 io_control.c, 113

MATRIX_SW7
 io_control.c, 113

MATRIX_SW8
 io_control.c, 114

MCP23017.c
 I2C_IO_MASTER_NUM, 133
 i2c_is_hardware_free, 134
 I2C_MASTER_FREQ_HZ, 133
 i2c_master_init, 134
 I2C_MASTER_RX_BUF_DISABLE, 133
 I2C_MASTER_SCL_IO, 134
 I2C_MASTER_SDA_IO, 134
 I2C_MASTER_TIMEOUT_MS, 134
 I2C_MASTER_TX_BUF_DISABLE, 134
 i2c_release_hardware, 135
MCP23017_INIT, 135
 mcp_get_on_interrupt_input, 135
 mcp_interrupt_flag, 136
 mcp_read_port, 136
 mcp_register_read, 137
 mcp_register_write, 137
 mcp_write_output_pin, 138
 mcp_write_output_port, 138
 TAG, 139
 xI2C_Mutex, 139

MCP23017.h
 MCP23017_INIT, 144
 mcp_get_on_interrupt_input, 144
 mcp_interrupt_flag, 146
 mcp_read_port, 146
 mcp_write_output_pin, 147
 mcp_write_output_port, 147

MCP23017_buzzer_control
 io_control.c, 119

mcp23017_defs.h
 __MCP23017_ADDRESS__, 150
 __MCP23017_BANK_SEQUENTIAL__, 150
 __MCP23017_BANK_SPLIT__, 150
 __MCP23017_DISSLW_DISABLED__, 150
 __MCP23017_DISSLW_ENABLED__, 151
 __MCP23017_FUNC_MODE__, 151
 __MCP23017_GPINTEN_DISABLE__, 151
 __MCP23017_GPINTEN_ENABLE__, 151
 __MCP23017_GPPU_PULL_UP_DISABLE__, 151
 __MCP23017_GPPU_PULL_UP_ENABLE__, 151
 __MCP23017_INTCON_CHANGE__, 152
 __MCP23017_INTCON_DEFVAL__, 152
 __MCP23017_INTERRUPT_DISABLE__, 152
 __MCP23017_INTERRUPT_ENABLE__, 152
 __MCP23017_INTPOL_POLARITY_HIGH__, 152
 __MCP23017_INTPOL_POLARITY_LOW__, 152
 __MCP23017_IODIR_INPUT__, 153
 __MCP23017_IODIR_OUTPUT__, 153
 __MCP23017_MIRROR_CONNECTED__, 153
 __MCP23017_MIRROR_UNCONNECTED__, 153
 __MCP23017_ODR_ACTIVE_DRIVER_OUTPUT__, 153

__MCP23017_ODR_OPEN_DRAIN_OUTPUT__,
 153
__MCP23017_OPPOSITE_LOGIC__, 154
__MCP23017_POSITIVE_LOGIC__, 154
__MCP23017_READ_OPCODE__, 154
__MCP23017_READ__, 154
__MCP23017_SEQOP_DISABLED__, 154
__MCP23017_SEQOP_ENABLED__, 154
__MCP23017_WRITE_OPCODE__, 155
__MCP23017_WRITE__, 155
MCP23017_DEFVALA_REGISTER, 155
MCP23017_DEFVALB_REGISTER, 155
MCP23017_GPINTENA_REGISTER, 155
MCP23017_GPINTENB_REGISTER, 155
MCP23017_GPIOA_REGISTER, 156
MCP23017_GPIOB_REGISTER, 156
MCP23017_GPPUA_REGISTER, 156
MCP23017_GPPUB_REGISTER, 156
MCP23017_INTCAPA_REGISTER, 156
MCP23017_INTCAPB_REGISTER, 156
MCP23017_INTCONA_REGISTER, 157
MCP23017_INTCONB_REGISTER, 157
MCP23017_INTFA_REGISTER, 157
MCP23017_INTFB_REGISTER, 157
MCP23017_IOCON_REGISTER, 157
MCP23017_IODIRA_REGISTER, 157
MCP23017_IODIRB_REGISTER, 158
MCP23017_IPOLA_REGISTER, 158
MCP23017_IPOLB_REGISTER, 158
MCP23017_OLATA_REGISTER, 158
MCP23017_OLATB_REGISTER, 158
mcp_port_e, 158
PORTA, 159
PORTB, 159
MCP23017_DEFVAL_t, 14
 all, 15
 bits, 15
 DEF0, 15
 DEF1, 15
 DEF2, 15
 DEF3, 16
 DEF4, 16
 DEF5, 16
 DEF6, 16
 DEF7, 16
MCP23017_DEFVALA_REGISTER
 mcp23017_defs.h, 155
MCP23017_DEFVALB_REGISTER
 mcp23017_defs.h, 155
MCP23017_GPINTEN_t, 17
 all, 17
 bits, 17
 GPINT0, 17
 GPINT1, 17
 GPINT2, 18
 GPINT3, 18
 GPINT4, 18
 GPINT5, 18
 GPINT6, 18
 GPINT7, 18
MCP23017_GPINTENA_REGISTER
 mcp23017_defs.h, 155
MCP23017_GPINTENB_REGISTER
 mcp23017_defs.h, 155
MCP23017_GPIO_t, 19
 all, 19
 bits, 19
 GP0, 19
 GP1, 20
 GP2, 20
 GP3, 20
 GP4, 20
 GP5, 20
 GP6, 20
 GP7, 21
MCP23017_GPIOA_REGISTER
 mcp23017_defs.h, 156
MCP23017_GPIOB_REGISTER
 mcp23017_defs.h, 156
MCP23017_GPPU_t, 21
 all, 21
 bits, 21
 PU0, 22
 PU1, 22
 PU2, 22
 PU3, 22
 PU4, 22
 PU5, 22
 PU6, 23
 PU7, 23
MCP23017_GPPUA_REGISTER
 mcp23017_defs.h, 156
MCP23017_GPPUB_REGISTER
 mcp23017_defs.h, 156
MCP23017_INIT
 MCP23017.c, 135
 MCP23017.h, 144
MCP23017_INTCAP_t, 23
 all, 24
 bits, 24
 ICP0, 24
 ICP1, 24
 ICP2, 24
 ICP3, 24
 ICP4, 24
 ICP5, 25
 ICP6, 25
 ICP7, 25
MCP23017_INTCAPA_REGISTER
 mcp23017_defs.h, 156
MCP23017_INTCAPB_REGISTER
 mcp23017_defs.h, 156
MCP23017_INTCON_t, 25
 all, 26
 bits, 26
 IOC0, 26

IOC1, 26
 IOC2, 26
 IOC3, 26
 IOC4, 26
 IOC5, 27
 IOC6, 27
 IOC7, 27
 MCP23017_INTCNA_REGISTER
 mcp23017_defs.h, 157
 MCP23017_INTCNB_REGISTER
 mcp23017_defs.h, 157
 MCP23017_INTF_t, 27
 all, 28
 bits, 28
 INT0, 28
 INT1, 28
 INT2, 28
 INT3, 28
 INT4, 28
 INT5, 29
 INT6, 29
 INT7, 29
 MCP23017_INTFA_REGISTER
 mcp23017_defs.h, 157
 MCP23017_INTFB_REGISTER
 mcp23017_defs.h, 157
 MCP23017_IOCON_REGISTER
 mcp23017_defs.h, 157
 MCP23017_IOCON_t, 29
 all, 30
 BANK, 30
 bits, 30
 DISSLW, 30
 INTPOL, 30
 MIRROR, 30
 ODR, 30
 reserved, 31
 reserved_2, 31
 SEQOP, 31
 MCP23017_IODIR_t, 31
 all, 32
 bits, 32
 IO0, 32
 IO1, 32
 IO2, 32
 IO3, 32
 IO4, 32
 IO5, 33
 IO6, 33
 IO7, 33
 MCP23017_IODIRA_REGISTER
 mcp23017_defs.h, 157
 MCP23017_IODIRB_REGISTER
 mcp23017_defs.h, 158
 MCP23017_IPOL_t, 33
 all, 34
 bits, 34
 IP0, 34
 IP1, 34
 IP2, 34
 IP3, 34
 IP4, 34
 IP5, 35
 IP6, 35
 IP7, 35
 MCP23017_IPOLA_REGISTER
 mcp23017_defs.h, 158
 MCP23017_IPOLB_REGISTER
 mcp23017_defs.h, 158
 MCP23017_keyboard_control
 io_control.c, 119
 MCP23017_OLAT_t, 35
 all, 36
 bits, 36
 OL0, 36
 OL1, 36
 OL2, 36
 OL3, 36
 OL4, 36
 OL5, 37
 OL6, 37
 OL7, 37
 MCP23017_OLATA_REGISTER
 mcp23017_defs.h, 158
 MCP23017_OLATB_REGISTER
 mcp23017_defs.h, 158
 MCP23017_relay_control
 io_control.c, 119
 mcp_get_on_interrupt_input
 MCP23017.c, 135
 MCP23017.h, 144
 mcp_interrupt_flag
 MCP23017.c, 136
 MCP23017.h, 146
 mcp_port_e
 mcp23017_defs.h, 158
 mcp_porta
 io_control.c, 120
 mcp_portb
 io_control.c, 121
 mcp_read_port
 MCP23017.c, 136
 MCP23017.h, 146
 mcp_register_read
 MCP23017.c, 137
 mcp_register_write
 MCP23017.c, 137
 mcp_write_output_pin
 MCP23017.c, 138
 MCP23017.h, 147
 mcp_write_output_port
 MCP23017.c, 138
 MCP23017.h, 147
 MIRROR
 MCP23017_IOCON_t, 30
 multiplier

frequency_settings_SH1106_t, 13
seccurity_settings_SH1106_t, 39
time_settings_SH1106_t, 44

nvs.c
load_16, 177
load_acceleration, 173
load_desacceleration, 173
load_frequency, 174
load_hour_fin, 174
load_hour_ini, 174
load_ibus_max, 174
load_input_variable, 174
load_min_fin, 175
load_min_ini, 175
load_variables, 178
load_vbus_min, 175
nvs_init_once, 179
save_16, 179
save_acceleration, 175
save_desacceleration, 175
save_frequency, 176
save_hour_fin, 176
save_hour_ini, 176
save_ibus_max, 176
save_input_variable, 176
save_min_fin, 177
save_min_ini, 177
save_variables, 180
save_vbus_min, 177
TAG, 181

nvs.h
load_variables, 186
nvs_init_once, 186
save_variables, 187

nvs_init_once
nvs.c, 179
nvs.h, 186

ODR
MCP23017_IOCON_t, 30

OL0
MCP23017_OLAT_t, 36

OL1
MCP23017_OLAT_t, 36

OL2
MCP23017_OLAT_t, 36

OL3
MCP23017_OLAT_t, 36

OL4
MCP23017_OLAT_t, 36

OL5
MCP23017_OLAT_t, 37

OL6
MCP23017_OLAT_t, 37

OL7
MCP23017_OLAT_t, 37

oled
display.c, 70

PIN_NUM_CLK
sysControl.c, 215

PIN_NUM_CS
sysControl.c, 215

PIN_NUM_MISO
sysControl.c, 215

PIN_NUM_MOSI
sysControl.c, 215

PORTA
mcp23017_defs.h, 159

PORTB
mcp23017_defs.h, 159

program_alarm
rtc.c, 190

PU0
MCP23017_GPPU_t, 22

PU1
MCP23017_GPPU_t, 22

PU2
MCP23017_GPPU_t, 22

PU3
MCP23017_GPPU_t, 22

PU4
MCP23017_GPPU_t, 22

PU5
MCP23017_GPPU_t, 22

PU6
MCP23017_GPPU_t, 23

PU7
MCP23017_GPPU_t, 23

PWM_CHANNEL
io_control.c, 114

PWM_FREQ_HZ
io_control.c, 114

PWM_GPIO
io_control.c, 114

PWM_MODE
io_control.c, 114

PWM_RES
io_control.c, 114

PWM_STEP_MS
io_control.c, 115

PWM_TIMER
io_control.c, 115

readADC
adc.c, 51
adc.h, 58

Referencia del directorio main, 8
Referencia del directorio main/adc, 7
Referencia del directorio main/display, 7
Referencia del directorio main/io_control, 8
Referencia del directorio main/nvs, 8
Referencia del directorio main/rtc, 9
Referencia del directorio main/system, 9
Referencia del directorio main/wifi, 9
relay_evt_queue
io_control.c, 121

RELAY_PIN

io_control.c, 115
 RelayEventPost
 io_control.c, 119
 io_control.h, 129
 request
 spi_cmd_item_t, 41
 reserved
 MCP23017_IOCON_t, 31
 reserved_2
 MCP23017_IOCON_t, 31
 root_get_handler
 wifi.c, 232
 rotation
 sh1106_t, 40
 rtc.c
 alarm_settings, 192
 alarm_start_cb, 189
 alarm_stop_cb, 189
 getTime, 190
 initRTC, 190
 program_alarm, 190
 rtc_alarms, 192
 rtc_schedule_alarms, 191
 setTime, 191
 TAG, 192
 time_start, 192
 time_stop, 192
 rtc.h
 getTime, 196
 initRTC, 196
 rtc_schedule_alarms, 196
 setTime, 197
 rtc_alarms
 rtc.c, 192
 rtc_alarms_t, 37
 start_timer, 38
 stop_timer, 38
 rtc_schedule_alarms
 rtc.c, 191
 rtc.h, 196
 s_adc
 adc.c, 54
 save_16
 nvs.c, 179
 save_acceleration
 nvs.c, 175
 save_desacceleration
 nvs.c, 175
 save_frequency
 nvs.c, 176
 save_hour_fin
 nvs.c, 176
 save_hour_ini
 nvs.c, 176
 save_ibus_max
 nvs.c, 176
 save_input_variable
 nvs.c, 176
 save_min_fin
 nvs.c, 177
 save_min_ini
 nvs.c, 177
 save_post_handler
 wifi.c, 232
 save_variables
 nvs.c, 180
 nvs.h, 187
 save_vbus_min
 nvs.c, 177
 screen_displayed
 display.c, 70
 SCREEN_FREQUENCY_EDIT
 display.c, 65
 SCREEN_MAIN
 display.c, 64
 SCREEN_SECURITY_EDIT
 display.c, 64
 SCREEN_SELECT_VARIABLE
 display.c, 64
 screen_selected_e
 display.c, 64
 SCREEN_TIME_EDIT
 display.c, 64
 seccurity_settings
 seccurity_settings_SH1106_t, 39
 seccurity_settings_SH1106_t, 38
 edit, 38
 edit_flag, 38
 edit_variable, 39
 LVFV_system.h, 165
 multiplier, 39
 seccurity_settings, 39
 seccurity_settings_t, 39
 ibus_max, 40
 LVFV_system.h, 165
 vbus_min, 40
 second
 LVFV_system.h, 166
 SECURITY_EXCEEDED
 LVFV_system.h, 168
 SECURITY_OK
 LVFV_system.h, 168
 SEQOP
 MCP23017_IOCON_t, 31
 set_freq_output
 io_control.c, 120
 io_control.h, 131
 set_frequency_table
 sysAdmin.c, 202
 sysAdmin.h, 211
 set_system_settings
 sysAdmin.c, 202
 sysAdmin.h, 211
 setTime
 rtc.c, 191
 rtc.h, 197

setValue
 spi_cmd_item_t, 42

seventh
 LVFV_system.h, 166

SH1106_ADDR
 sh1106_i2c.c, 106

sh1106_clear_buffer
 display.c, 66

SH1106_COMM_CMD
 sh1106_i2c.h, 108

SH1106_COMM_DATA
 sh1106_i2c.h, 108

sh1106_comm_type_t
 sh1106_i2c.h, 107

sh1106_draw_arrow
 sh1106_graphics.c, 87
 sh1106_graphics.h, 102

sh1106_draw_bitmap
 sh1106_graphics.c, 87

sh1106_draw_fail
 sh1106_graphics.c, 87
 sh1106_graphics.h, 103

sh1106_draw_line
 sh1106_graphics.c, 88
 sh1106_graphics.h, 103

sh1106_draw_text
 sh1106_graphics.c, 88
 sh1106_graphics.h, 103

sh1106_draw_utn_logo
 sh1106_graphics.c, 88
 sh1106_graphics.h, 104

sh1106_frequency_edit_variables
 display.c, 66

sh1106_graphics.c
 arrow_bmp, 89
 bitmap_t, 87
 fail_bmp, 89
 font5x7_close_excl, 89
 font5x7_close_quest, 89
 font5x7_comma, 89
 font5x7_dot, 89
 font5x7_double_dot, 89
 font5x7_minus, 90
 font5x7_rowmajor, 90
 font5x7_rowminor, 90
 font5x7_rownumber, 91
 font5x7_space, 92
 font8x14_close_excl, 92
 font8x14_close_quest, 92
 font8x14_comma, 92
 font8x14_dot, 92
 font8x14_double_dot, 92
 font8x14_minus, 93
 font8x14_rowmajor, 93
 font8x14_rowminor, 93
 font8x14_rownumber, 94
 font8x14_space, 95
 line_bmp, 95

 logo16_utn.bmp, 95
 sh1106_draw_arrow, 87
 sh1106_draw_bitmap, 87
 sh1106_draw_fail, 87
 sh1106_draw_line, 88
 sh1106_draw_text, 88
 sh1106_draw_utn_logo, 88
 slash, 95

 sh1106_graphics.h
 sh1106_draw_arrow, 102
 sh1106_draw_fail, 103
 sh1106_draw_line, 103
 sh1106_draw_text, 103
 sh1106_draw_utn_logo, 104
 sh1106_t, 102

 sh1106_i2c.c
 CMD_CONTROL, 105
 DATA_CONTROL, 105
 I2C_DISPLAY, 105
 SH1106_ADDR, 106
 sh1106_write, 106

 sh1106_i2c.h
 SH1106_COMM_CMD, 108
 SH1106_COMM_DATA, 108
 sh1106_comm_type_t, 107
 sh1106_write, 108

 sh1106_init
 display.c, 66
 display.h, 83

 sh1106_main_screen
 display.c, 66

 sh1106_print_emergency
 display.c, 67

 sh1106_print_frame
 display.c, 67

 sh1106_refresh
 display.c, 67

 sh1106_security_edit_variables
 display.c, 67

 sh1106_select_edit_variables
 display.c, 68

 SH1106_SIZE_1
 LVFV_system.h, 164

 SH1106_SIZE_2
 LVFV_system.h, 164

 sh1106_splash_screen
 display.c, 68

 sh1106_t, 40
 buffer, 40
 height, 40
 rotation, 40
 sh1106_graphics.h, 102
 width, 41

 sh1106_time_edit_variables
 display.c, 68

 sh1106_variable_lines_e
 LVFV_system.h, 165, 166

 sh1106_write

sh1106_i2c.c, 106
 sh1106_i2c.h, 108
 sixth
 LVFV_system.h, 166
 slash
 sh1106_graphics.c, 95
 SPEED_SELECTOR_0
 LVFV_system.h, 167
 SPEED_SELECTOR_1
 LVFV_system.h, 167
 SPEED_SELECTOR_2
 LVFV_system.h, 167
 SPEED_SELECTOR_3
 LVFV_system.h, 167
 SPEED_SELECTOR_4
 LVFV_system.h, 167
 SPEED_SELECTOR_5
 LVFV_system.h, 168
 SPEED_SELECTOR_6
 LVFV_system.h, 168
 SPEED_SELECTOR_7
 LVFV_system.h, 168
 SPEED_SELECTOR_8
 LVFV_system.h, 168
 SPEED_SELECTOR_9
 LVFV_system.h, 168
 SPI_CLOCK_HZ
 sysControl.c, 215
 spi_cmd_item_t, 41
 getValue, 41
 request, 41
 setValue, 42
 SPI_communication
 sysControl.c, 216
 sysControl.h, 224
 spi_handle
 sysControl.c, 218
 SPI_HOST_USED
 sysControl.c, 216
 SPI_Init
 sysControl.c, 216
 sysControl.h, 226
 SPI_QUEUE_TX_DEPTH
 sysControl.c, 216
 SPI_Request
 sysControl.h, 223
 SPI_REQUEST_EMERGENCY
 sysControl.h, 224
 SPI_REQUEST_GET_ACCEL
 sysControl.h, 224
 SPI_REQUEST_GET_DESACCEL
 sysControl.h, 224
 SPI_REQUEST_GET_DIR
 sysControl.h, 224
 SPI_REQUEST_GET_FREC
 sysControl.h, 224
 SPI_REQUEST_IS_STOP
 sysControl.h, 224
 SPI_REQUEST_RESPONSE
 sysControl.h, 224
 SPI_REQUEST_SET_ACCEL
 sysControl.h, 224
 SPI_REQUEST_SET_DESACCEL
 sysControl.h, 224
 SPI_REQUEST_SET_DIR
 sysControl.h, 224
 SPI_REQUEST_SET_FREC
 sysControl.h, 224
 SPI_REQUEST_START
 sysControl.h, 224
 SPI_REQUEST_STOP
 sysControl.h, 224
 SPI_Response
 sysControl.h, 224
 SPI_RESPONSE_ERR
 sysControl.h, 224
 SPI_RESPONSE_ERR_CMD_UNKNOWN
 sysControl.h, 224
 SPI_RESPONSE_ERR_DATA_INVALID
 sysControl.h, 224
 SPI_RESPONSE_ERR_DATA_MISSING
 sysControl.h, 224
 SPI_RESPONSE_ERR_DATA_OUT_RANGE
 sysControl.h, 224
 SPI_RESPONSE_ERR_MOVING
 sysControl.h, 224
 SPI_RESPONSE_ERR_NO_COMMAND
 sysControl.h, 224
 SPI_RESPONSE_ERR_NOT_MOVING
 sysControl.h, 224
 SPI_RESPONSE_OK
 sysControl.h, 224
 SPI_SendRequest
 sysControl.c, 216
 START_BUTTON_PIN
 io_control.c, 115
 START_PRESSED
 LVFV_system.h, 167
 START_RELEASED
 LVFV_system.h, 167
 start_timer
 rtc_alarms_t, 38
 start_webserver
 wifi.c, 232
 wifi.h, 237
 status
 system_status_t, 43
 STOP_BUTTON_PIN
 io_control.c, 115
 STOP_PIN
 io_control.c, 115
 STOP_PRESSED
 LVFV_system.h, 167
 STOP_RELEASED
 LVFV_system.h, 167
 STOP_SW_MASK

io_control.c, 116
stop_timer
 rtc_alarms_t, 38
SWITCH_BUTTON_BACK
 io_control.c, 116
SWITCH_BUTTON_DOWN
 io_control.c, 116
SWITCH_BUTTON_LEFT
 io_control.c, 116
SWITCH_BUTTON_MENU
 io_control.c, 116
SWITCH_BUTTON_OK
 io_control.c, 116
SWITCH_BUTTON_RIGHT
 io_control.c, 117
SWITCH_BUTTON_SAVE
 io_control.c, 117
SWITCH_BUTTON_UP
 io_control.c, 117
sysAdmin.c
 accelerating, 199
 accelerating_handle, 204
 change_frequency, 199
 desaccelerating, 200
 desaccelerating_handle, 204
 engine_emergency_stop, 200
 engine_emergency_stop_release, 200
 engine_start, 201
 engine_stop, 201
 frequency_table, 204
 get_status, 201
 set_frequency_table, 202
 set_system_settings, 202
 system_security_settings, 204
 system_status, 204
 TAG, 205
 update_meas, 202
sysAdmin.h
 change_frequency, 209
 engine_emergency_stop, 209
 engine_emergency_stop_release, 209
 engine_start, 210
 engine_stop, 210
 get_status, 210
 set_frequency_table, 211
 set_system_settings, 211
 update_meas, 211
sysControl.c
 PIN_NUM_CLK, 215
 PIN_NUM_CS, 215
 PIN_NUM_MISO, 215
 PIN_NUM_MOSI, 215
 SPI_CLOCK_HZ, 215
 SPI_communication, 216
 spi_handle, 218
 SPI_HOST_USED, 216
 SPI_Init, 216
 SPI_QUEUE_TX_DEPTH, 216
 SPI_SendRequest, 216
 system_event_queue, 218
 SystemEventPost, 217
 TAG, 218
sysControl.h
 SPI_communication, 224
 SPI_Init, 226
 SPI_Request, 223
 SPI_REQUEST_EMERGENCY, 224
 SPI_REQUEST_GET_ACCEL, 224
 SPI_REQUEST_GET_DESACCEL, 224
 SPI_REQUEST_GET_DIR, 224
 SPI_REQUEST_GET_FREC, 224
 SPI_REQUEST_IS_STOP, 224
 SPI_REQUEST_RESPONSE, 224
 SPI_REQUEST_SET_ACCEL, 224
 SPI_REQUEST_SET_DESACCEL, 224
 SPI_REQUEST_SET_DIR, 224
 SPI_REQUEST_SET_FREC, 224
 SPI_REQUEST_START, 224
 SPI_REQUEST_STOP, 224
 SPI_Response, 224
 SPI_RESPONSE_ERR, 224
 SPI_RESPONSE_ERR_CMD_UNKNOWN, 224
 SPI_RESPONSE_ERR_DATA_INVALID, 224
 SPI_RESPONSE_ERR_DATA_MISSING, 224
 SPI_RESPONSE_ERR_DATA_OUT_RANGE, 224
 SPI_RESPONSE_ERR_MOVING, 224
 SPI_RESPONSE_ERR_NO_COMMAND, 224
 SPI_RESPONSE_ERR_NOT_MOVING, 224
 SPI_RESPONSE_OK, 224
 SystemEventPost, 226
SYSTEM_ACCEL_DESACCEL
 LVFV_system.h, 167
SYSTEM_BREAKING
 LVFV_system.h, 167
SYSTEM_EMERGENCY
 LVFV_system.h, 167
SYSTEM_EMERGENCY_OK
 LVFV_system.h, 167
system_event_queue
 sysControl.c, 218
system_frequency_settings
 display.c, 70
SYSTEM_IDLE
 LVFV_system.h, 167
SYSTEM_REGIME
 LVFV_system.h, 167
system_seccurity_settings
 display.c, 71
 sysAdmin.c, 204
system_status
 sysAdmin.c, 204
system_status_e
 LVFV_system.h, 166
system_status_t, 42
 acceleration, 42
 desacceleration, 42

emergency_signals, 43
 frequency, 43
 frequency_destiny, 43
 ibus_max, 43
 inputs_status, 43
 LVFV_system.h, 166
 status, 43
 vbus_min, 43
 system_time_settings
 display.c, 71
 system_variables_save
 display.c, 68
 display.h, 84
 SystemEventPost
 sysControl.c, 217
 sysControl.h, 226
 systemSignal_e
 LVFV_system.h, 167

TAG
 adc.c, 54
 display.c, 71
 io_control.c, 121
 main.c, 171
 MCP23017.c, 139
 nvs.c, 181
 rtc.c, 192
 sysAdmin.c, 205
 sysControl.c, 218
 wifi.c, 233
 task_display
 display.c, 69
 display.h, 84
 TERMO_SW_MASK
 io_control.c, 117
 TERMO_SW_PIN
 io_control.c, 117
 TERMO_SW_PRESSED
 LVFV_system.h, 167
 TERMO_SW_RELEASED
 LVFV_system.h, 167
 third
 LVFV_system.h, 166
 time_settings
 time_settings_SH1106_t, 45
 time_settings_SH1106_t, 44
 edit, 44
 edit_flag, 44
 edit_variable, 44
 LVFV_system.h, 166
 multiplier, 44
 time_settings, 45
 time_settings_t, 45
 LVFV_system.h, 166
 time_start, 45
 time_stop, 45
 time_system, 45
 time_start
 rtc.c, 192

time_settings_t, 45
 time_stop
 rtc.c, 192
 time_settings_t, 45
 time_system
 time_settings_t, 45

update_meas
 sysAdmin.c, 202
 sysAdmin.h, 211
 url_decode
 wifi.c, 232

VARIABLE_EIGTH
 LVFV_system.h, 164
VARIABLE_FIFTH
 LVFV_system.h, 164
VARIABLE_FIRST
 LVFV_system.h, 164
VARIABLE_FOURTH
 LVFV_system.h, 164
VARIABLE_SECOND
 LVFV_system.h, 164
VARIABLE_SEVENTH
 LVFV_system.h, 165
VARIABLE_SIXTH
 LVFV_system.h, 165
VARIABLE_THIRD
 LVFV_system.h, 165
VBUS_LINE_INDX
 display.c, 64
 vbus_min
 security_settings_t, 40
 system_status_t, 43

w
 bitmap_t, 12

width
 sh1106_t, 41

wifi.c
 get_param_value, 231
 hex2int, 231
 root_get_handler, 232
 save_post_handler, 232
 start_webserver, 232
 TAG, 233
 url_decode, 232
 wifi_init_softap, 233

wifi.h
 start_webserver, 237
 wifi_init_softap, 237

wifi_init_softap
 wifi.c, 233
 wifi.h, 237

x
 bitmap_t, 12

xI2C_Mutex
 MCP23017.c, 139

y

[bitmap_t](#), 12