

Low Voltage Variable Frequency Drive

2.1

Generado por Doxygen 1.15.0

1	Lista de tareas pendientes	1
2	Jerarquía de directorios	3
2.1	Directorios	3
3	Índice de estructuras de datos	5
3.1	Estructuras de datos	5
4	Índice de archivos	7
4.1	Lista de archivos	7
5	Documentación de directorios	9
5.1	Referencia del directorio C:/Users/User/LVFV_ESP32/main/adc	9
5.2	Referencia del directorio C:/Users/User/LVFV_ESP32/main/display	9
5.3	Referencia del directorio C:/Users/User/LVFV_ESP32/main/io_control	10
5.4	Referencia del directorio C:/Users/User/LVFV_ESP32/main	10
5.5	Referencia del directorio C:/Users/User/LVFV_ESP32/main/nvs	10
5.6	Referencia del directorio C:/Users/User/LVFV_ESP32/main/rtc	11
5.7	Referencia del directorio C:/Users/User/LVFV_ESP32/main/system	11
5.8	Referencia del directorio C:/Users/User/LVFV_ESP32/main/wifi	11
6	Documentación de estructuras de datos	13
6.1	Referencia de la estructura bitmap_t	13
6.1.1	Descripción detallada	13
6.1.2	Documentación de campos	13
6.1.2.1	bitmap	13
6.1.2.2	h	14
6.1.2.3	w	14
6.1.2.4	x	14
6.1.2.5	y	14
6.2	Referencia de la estructura frequency_settings_SH1106_t	14
6.2.1	Descripción detallada	15
6.2.2	Documentación de campos	15
6.2.2.1	edit	15
6.2.2.2	edit_flag	15
6.2.2.3	edit_variable	15
6.2.2.4	frequency_settings	15
6.2.2.5	multiplier	15
6.3	Referencia de la estructura frequency_settings_t	15
6.3.1	Descripción detallada	16
6.3.2	Documentación de campos	16
6.3.2.1	acceleration	16
6.3.2.2	desacceleration	16
6.3.2.3	freq_regime	16

6.3.2.4 input_variable	16
6.4 Referencia de la unión MCP23017_DEFVAL_t	16
6.4.1 Descripción detallada	17
6.4.2 Documentación de campos	17
6.4.2.1 all	17
6.4.2.2 [struct]	17
6.4.2.3 DEF0	17
6.4.2.4 DEF1	17
6.4.2.5 DEF2	18
6.4.2.6 DEF3	18
6.4.2.7 DEF4	18
6.4.2.8 DEF5	18
6.4.2.9 DEF6	18
6.4.2.10 DEF7	18
6.5 Referencia de la unión MCP23017_GPINTEN_t	19
6.5.1 Descripción detallada	19
6.5.2 Documentación de campos	19
6.5.2.1 all	19
6.5.2.2 [struct]	19
6.5.2.3 GPINT0	19
6.5.2.4 GPINT1	20
6.5.2.5 GPINT2	20
6.5.2.6 GPINT3	20
6.5.2.7 GPINT4	20
6.5.2.8 GPINT5	20
6.5.2.9 GPINT6	20
6.5.2.10 GPINT7	21
6.6 Referencia de la unión MCP23017_GPIO_t	21
6.6.1 Descripción detallada	21
6.6.2 Documentación de campos	21
6.6.2.1 all	21
6.6.2.2 [struct]	21
6.6.2.3 GP0	22
6.6.2.4 GP1	22
6.6.2.5 GP2	22
6.6.2.6 GP3	22
6.6.2.7 GP4	22
6.6.2.8 GP5	22
6.6.2.9 GP6	23
6.6.2.10 GP7	23
6.7 Referencia de la unión MCP23017_GPPU_t	23
6.7.1 Descripción detallada	23

6.7.2 Documentación de campos	23
6.7.2.1 all	23
6.7.2.2 [struct]	24
6.7.2.3 PU0	24
6.7.2.4 PU1	24
6.7.2.5 PU2	24
6.7.2.6 PU3	24
6.7.2.7 PU4	24
6.7.2.8 PU5	25
6.7.2.9 PU6	25
6.7.2.10 PU7	25
6.8 Referencia de la unión MCP23017_INTCAP_t	25
6.8.1 Descripción detallada	26
6.8.2 Documentación de campos	26
6.8.2.1 all	26
6.8.2.2 [struct]	26
6.8.2.3 ICP0	26
6.8.2.4 ICP1	26
6.8.2.5 ICP2	26
6.8.2.6 ICP3	26
6.8.2.7 ICP4	27
6.8.2.8 ICP5	27
6.8.2.9 ICP6	27
6.8.2.10 ICP7	27
6.9 Referencia de la unión MCP23017_INTCON_t	27
6.9.1 Descripción detallada	28
6.9.2 Documentación de campos	28
6.9.2.1 all	28
6.9.2.2 [struct]	28
6.9.2.3 IOC0	28
6.9.2.4 IOC1	28
6.9.2.5 IOC2	28
6.9.2.6 IOC3	28
6.9.2.7 IOC4	29
6.9.2.8 IOC5	29
6.9.2.9 IOC6	29
6.9.2.10 IOC7	29
6.10 Referencia de la unión MCP23017_INTF_t	29
6.10.1 Descripción detallada	30
6.10.2 Documentación de campos	30
6.10.2.1 all	30
6.10.2.2 [struct]	30

6.10.2.3 INT0	30
6.10.2.4 INT1	30
6.10.2.5 INT2	30
6.10.2.6 INT3	30
6.10.2.7 INT4	31
6.10.2.8 INT5	31
6.10.2.9 INT6	31
6.10.2.10 INT7	31
6.11 Referencia de la unión MCP23017_IOCON_t	31
6.11.1 Descripción detallada	32
6.11.2 Documentación de campos	32
6.11.2.1 all	32
6.11.2.2 BANK	32
6.11.2.3 [struct]	32
6.11.2.4 DISSLW	32
6.11.2.5 INTPOL	32
6.11.2.6 MIRROR	32
6.11.2.7 ODR	33
6.11.2.8 reserved	33
6.11.2.9 reserved_2	33
6.11.2.10 SEQOP	33
6.12 Referencia de la unión MCP23017_IODIR_t	33
6.12.1 Descripción detallada	34
6.12.2 Documentación de campos	34
6.12.2.1 all	34
6.12.2.2 [struct]	34
6.12.2.3 IO0	34
6.12.2.4 IO1	34
6.12.2.5 IO2	34
6.12.2.6 IO3	34
6.12.2.7 IO4	35
6.12.2.8 IO5	35
6.12.2.9 IO6	35
6.12.2.10 IO7	35
6.13 Referencia de la unión MCP23017_IPOL_t	35
6.13.1 Descripción detallada	36
6.13.2 Documentación de campos	36
6.13.2.1 all	36
6.13.2.2 [struct]	36
6.13.2.3 IP0	36
6.13.2.4 IP1	36
6.13.2.5 IP2	36

6.13.2.6 IP3	36
6.13.2.7 IP4	37
6.13.2.8 IP5	37
6.13.2.9 IP6	37
6.13.2.10 IP7	37
6.14 Referencia de la unión MCP23017_OLAT_t	37
6.14.1 Descripción detallada	38
6.14.2 Documentación de campos	38
6.14.2.1 all	38
6.14.2.2 [struct]	38
6.14.2.3 OL0	38
6.14.2.4 OL1	38
6.14.2.5 OL2	38
6.14.2.6 OL3	38
6.14.2.7 OL4	39
6.14.2.8 OL5	39
6.14.2.9 OL6	39
6.14.2.10 OL7	39
6.15 Referencia de la estructura rtc_alarms_t	39
6.15.1 Descripción detallada	39
6.15.2 Documentación de campos	40
6.15.2.1 start_timer	40
6.15.2.2 stop_timer	40
6.16 Referencia de la estructura seecurity_settings_SH1106_t	40
6.16.1 Descripción detallada	40
6.16.2 Documentación de campos	40
6.16.2.1 edit	40
6.16.2.2 edit_flag	41
6.16.2.3 edit_variable	41
6.16.2.4 multiplier	41
6.16.2.5 seecurity_settings	41
6.17 Referencia de la estructura seecurity_settings_t	41
6.17.1 Descripción detallada	41
6.17.2 Documentación de campos	42
6.17.2.1 ibus_max	42
6.17.2.2 vbus_min	42
6.18 Referencia de la estructura sh1106_t	42
6.18.1 Descripción detallada	42
6.18.2 Documentación de campos	42
6.18.2.1 buffer	42
6.18.2.2 height	42
6.18.2.3 rotation	43

6.18.2.4 width	43
6.19 Referencia de la estructura spi_cmd_item_t	43
6.19.1 Descripción detallada	43
6.19.2 Documentación de campos	43
6.19.2.1 getValue	43
6.19.2.2 request	44
6.19.2.3 setValue	44
6.20 Referencia de la estructura system_status_t	44
6.20.1 Descripción detallada	44
6.20.2 Documentación de campos	44
6.20.2.1 acceleration	44
6.20.2.2 desacceleration	45
6.20.2.3 frequency	45
6.20.2.4 frequency_destiny	45
6.20.2.5 ibus_max	45
6.20.2.6 inputs_status	45
6.20.2.7 status	45
6.20.2.8 vbus_min	45
6.21 Referencia de la estructura time_settings_SH1106_t	46
6.21.1 Descripción detallada	46
6.21.2 Documentación de campos	46
6.21.2.1 edit	46
6.21.2.2 edit_flag	46
6.21.2.3 edit_variable	46
6.21.2.4 multiplier	46
6.21.2.5 time_settings	47
6.22 Referencia de la estructura time_settings_t	47
6.22.1 Descripción detallada	47
6.22.2 Documentación de campos	47
6.22.2.1 time_start	47
6.22.2.2 time_stop	47
6.22.2.3 time_system	47
7 Documentación de archivos	49
7.1 Referencia del archivo C:/Users/User/LVFV_ESP32/main/adc/adc.c	49
7.1.1 Descripción detallada	50
7.1.2 Documentación de «define»	50
7.1.2.1 ADC_ATTEN_USED	50
7.1.2.2 ADC_CH_GPIO34	50
7.1.2.3 ADC_CH_GPIO35	51
7.1.2.4 ADC_CH_GPIO36	51
7.1.2.5 ADC_CH_GPIO39	51

7.1.2.6 ADC_PERIOD_MS	51
7.1.2.7 ADC_UNIT_USED	51
7.1.3 Documentación de funciones	52
7.1.3.1 adc_cali_try_init()	52
7.1.3.2 adc_init()	53
7.1.3.3 adc_task()	53
7.1.3.4 readADC()	54
7.1.4 Documentación de variables	55
7.1.4.1 bus_meas_evt_queue	55
7.1.4.2 calibration_3V3_source	55
7.1.4.3 calibration_3V3_source_ok	55
7.1.4.4 calibration_5V_source	55
7.1.4.5 calibration_5V_source_ok	55
7.1.4.6 calibration_bus_voltage	55
7.1.4.7 calibration_bus_voltage_ok	56
7.1.4.8 calibration_current	56
7.1.4.9 calibration_current_ok	56
7.1.4.10 s_adc	56
7.1.4.11 TAG	56
7.2 adc.c	56
7.3 Referencia del archivo C:/Users/User/LVFV_ESP32/main/adc/adc.h	59
7.3.1 Descripción detallada	59
7.3.2 Documentación de funciones	60
7.3.2.1 adc_init()	60
7.3.2.2 adc_task()	60
7.3.2.3 readADC()	60
7.4 adc.h	61
7.5 Referencia del archivo C:/Users/User/LVFV_ESP32/main/display/display.c	61
7.5.1 Descripción detallada	63
7.5.2 Documentación de «define»	63
7.5.2.1 EDIT_ACCELERATION_INDX	63
7.5.2.2 EDIT_DESACCELERATION_INDX	64
7.5.2.3 EDIT_FREQUENCY_INDX	64
7.5.2.4 EDIT_HFIN_LINE_INDX	64
7.5.2.5 EDIT_HINI_LINE_INDX	64
7.5.2.6 EDIT_IBUS_LINE_INDX	64
7.5.2.7 EDIT_INPUT_VARIATION_INDX	64
7.5.2.8 EDIT_TIME_LINE_INDX	64
7.5.2.9 EDIT_VBUS_LINE_INDX	64
7.5.2.10 FREC_LINE_INDX	65
7.5.2.11 FREQUENCY CUADRATIC VARIABLE	65
7.5.2.12 FREQUENCY LINEAR VARIABLE	65

7.5.2.13 HFIN_LINE_INDX	65
7.5.2.14 HINI_LINE_INDX	65
7.5.2.15 I2C_DISPLAY_MASTER_NUM	65
7.5.2.16 I2C_MASTER_FREQ_HZ	65
7.5.2.17 I2C_MASTER_RX_BUF_DISABLE	65
7.5.2.18 I2C_MASTER_SCL_IO	66
7.5.2.19 I2C_MASTER_SDA_IO	66
7.5.2.20 I2C_MASTER_TX_BUF_DISABLE	66
7.5.2.21 IBUS_LINE_INDX	66
7.5.2.22 VBUS_LINE_INDX	66
7.5.3 Documentación de enumeraciones	66
7.5.3.1 screen_selected_e	66
7.5.4 Documentación de funciones	67
7.5.4.1 DisplayEventPost()	67
7.5.4.2 get_system_acceleration()	67
7.5.4.3 get_system_desacceleration()	67
7.5.4.4 get_system_frequency()	67
7.5.4.5 i2c_master_init()	68
7.5.4.6 sh1106_clear_buffer()	68
7.5.4.7 sh1106_frequency_edit_variables()	68
7.5.4.8 sh1106_init()	68
7.5.4.9 sh1106_main_screen()	69
7.5.4.10 sh1106_print_emergency()	69
7.5.4.11 sh1106_print_frame()	69
7.5.4.12 sh1106_refresh()	69
7.5.4.13 sh1106_security_edit_variables()	69
7.5.4.14 sh1106_select_edit_variables()	70
7.5.4.15 sh1106_splash_screen()	70
7.5.4.16 sh1106_time_edit_variables()	70
7.5.4.17 system_variables_save()	71
7.5.4.18 task_display()	71
7.5.5 Documentación de variables	72
7.5.5.1 blink	72
7.5.5.2 button_evt_queue	72
7.5.5.3 init_seq	72
7.5.5.4 oled	72
7.5.5.5 screen_displayed	72
7.5.5.6 system_frequency_settings	73
7.5.5.7 system_seccurity_settings	73
7.5.5.8 system_time_settings	73
7.5.5.9 TAG	73
7.6 display.c	73

7.7 Referencia del archivo C:/Users/User/LVFV_ESP32/main/display/display.h	84
7.7.1 Descripción detallada	84
7.7.2 Documentación de funciones	85
7.7.2.1 DisplayEventPost()	85
7.7.2.2 get_system_acceleration()	86
7.7.2.3 get_system_desacceleration()	86
7.7.2.4 get_system_frequency()	86
7.7.2.5 sh1106_init()	86
7.7.2.6 system_variables_save()	87
7.7.2.7 task_display()	87
7.8 display.h	88
7.9 Referencia del archivo C:/Users/User/LVFV_ESP32/main/display/sh1106_graphics.c	88
7.9.1 Descripción detallada	89
7.9.2 Documentación de «typedef»	90
7.9.2.1 bitmap_t	90
7.9.3 Documentación de funciones	90
7.9.3.1 sh1106_draw_arrow()	90
7.9.3.2 sh1106_draw_bitmap()	90
7.9.3.3 sh1106_draw_fail()	90
7.9.3.4 sh1106_draw_line()	91
7.9.3.5 sh1106_draw_text()	91
7.9.3.6 sh1106_draw_utn_logo()	91
7.9.4 Documentación de variables	92
7.9.4.1 arrow_bmp	92
7.9.4.2 fail_bmp	92
7.9.4.3 font5x7_close_excl	92
7.9.4.4 font5x7_close_quest	92
7.9.4.5 font5x7_comma	92
7.9.4.6 font5x7_dot	92
7.9.4.7 font5x7_double_dot	93
7.9.4.8 font5x7_minus	93
7.9.4.9 font5x7_rowmajor	93
7.9.4.10 font5x7_rowminor	94
7.9.4.11 font5x7_rownumber	94
7.9.4.12 font5x7_space	95
7.9.4.13 font8x14_close_excl	95
7.9.4.14 font8x14_close_quest	95
7.9.4.15 font8x14_comma	95
7.9.4.16 font8x14_dot	95
7.9.4.17 font8x14_double_dot	96
7.9.4.18 font8x14_minus	96
7.9.4.19 font8x14_rowmajor	96

7.9.4.20 font8x14_rowminor	97
7.9.4.21 font8x14_rownumber	97
7.9.4.22 font8x14_space	98
7.9.4.23 line_bmp	98
7.9.4.24 logo16_utn_bmp	98
7.9.4.25 slash	98
7.10 sh1106_graphics.c	99
7.11 Referencia del archivo C:/Users/User/LVFV_ESP32/main/display/sh1106_graphics.h	104
7.11.1 Descripción detallada	105
7.11.2 Documentación de «typedef»	105
7.11.2.1 sh1106_t	105
7.11.3 Documentación de funciones	105
7.11.3.1 sh1106_draw_arrow()	105
7.11.3.2 sh1106_draw_fail()	106
7.11.3.3 sh1106_draw_line()	106
7.11.3.4 sh1106_draw_text()	106
7.11.3.5 sh1106_draw_utn_logo()	107
7.12 sh1106_graphics.h	107
7.13 Referencia del archivo C:/Users/User/LVFV_ESP32/main/display/sh1106_i2c.c	107
7.13.1 Descripción detallada	108
7.13.2 Documentación de «define»	108
7.13.2.1 CMD_CONTROL	108
7.13.2.2 DATA_CONTROL	108
7.13.2.3 I2C_DISPLAY	109
7.13.2.4 SH1106_ADDR	109
7.13.3 Documentación de funciones	109
7.13.3.1 sh1106_write()	109
7.14 sh1106_i2c.c	109
7.15 Referencia del archivo C:/Users/User/LVFV_ESP32/main/display/sh1106_i2c.h	110
7.15.1 Descripción detallada	110
7.15.2 Documentación de enumeraciones	110
7.15.2.1 sh1106_comm_type_t	110
7.15.3 Documentación de funciones	111
7.15.3.1 sh1106_write()	111
7.16 sh1106_i2c.h	111
7.17 Referencia del archivo C:/Users/User/LVFV_ESP32/main/io_control/io_control.c	112
7.17.1 Descripción detallada	114
7.17.2 Documentación de «define»	114
7.17.2.1 BUZZER_PIN	114
7.17.2.2 DUTY_MAX_PCT	114
7.17.2.3 DUTY_MIN_PCT	114
7.17.2.4 FREQ_SEL_1_PIN	115

7.17.2.5 FREQ_SEL_2_PIN	115
7.17.2.6 FREQ_SEL_3_PIN	115
7.17.2.7 FREQ_SEL_MASK	115
7.17.2.8 INT_A_PIN	115
7.17.2.9 INT_B_PIN	115
7.17.2.10 MATRIX_SW1	116
7.17.2.11 MATRIX_SW2	116
7.17.2.12 MATRIX_SW3	116
7.17.2.13 MATRIX_SW4	116
7.17.2.14 MATRIX_SW5	116
7.17.2.15 MATRIX_SW6	116
7.17.2.16 MATRIX_SW7	117
7.17.2.17 MATRIX_SW8	117
7.17.2.18 PWM_CHANNEL	117
7.17.2.19 PWM_FREQ_HZ	117
7.17.2.20 PWM_GPIO	117
7.17.2.21 PWM_MODE	117
7.17.2.22 PWM_RES	118
7.17.2.23 PWM_STEP_MS	118
7.17.2.24 PWM_TIMER	118
7.17.2.25 RELAY_PIN	118
7.17.2.26 START_BUTTON_PIN	118
7.17.2.27 STOP_BUTTON_PIN	118
7.17.2.28 STOP_PIN	119
7.17.2.29 STOP_SW_MASK	119
7.17.2.30 SWITCH_BUTTON_BACK	119
7.17.2.31 SWITCH_BUTTON_DOWN	119
7.17.2.32 SWITCH_BUTTON_LEFT	119
7.17.2.33 SWITCH_BUTTON_MENU	119
7.17.2.34 SWITCH_BUTTON_OK	120
7.17.2.35 SWITCH_BUTTON_RIGHT	120
7.17.2.36 SWITCH_BUTTON_SAVE	120
7.17.2.37 SWITCH_BUTTON_UP	120
7.17.2.38 TERMO_SW_MASK	120
7.17.2.39 TERMO_SW_PIN	120
7.17.3 Documentación de funciones	120
7.17.3.1 BuzzerEventPost()	120
7.17.3.2 freq_0_10V_output()	121
7.17.3.3 gpio_init_interrupts()	121
7.17.3.4 GPIO_interrupt_attendance_task()	122
7.17.3.5 gpio_isr_handler()	122
7.17.3.6 MCP23017_buzzer_control()	122

7.17.3.7 MCP23017_keyboard_control()	122
7.17.3.8 MCP23017_relay_control()	123
7.17.3.9 RelayEventPost()	123
7.17.3.10 set_freq_output()	124
7.17.4 Documentación de variables	124
7.17.4.1 buzzer_evt_queue	124
7.17.4.2 GPIO_evt_queue	124
7.17.4.3 mcp_porta	125
7.17.4.4 mcp_portb	125
7.17.4.5 relay_evt_queue	125
7.17.4.6 TAG	125
7.18 io_control.c	125
7.19 Referencia del archivo C:/Users/User/LVFV_ESP32/main/io_control/io_control.h	132
7.19.1 Descripción detallada	133
7.19.2 Documentación de funciones	133
7.19.2.1 BuzzerEventPost()	133
7.19.2.2 GPIO_interrupt_attendance_task()	133
7.19.2.3 RelayEventPost()	134
7.19.2.4 set_freq_output()	135
7.20 io_control.h	135
7.21 Referencia del archivo C:/Users/User/LVFV_ESP32/main/io_control/MCP23017.c	136
7.21.1 Descripción detallada	137
7.21.2 Documentación de «define»	137
7.21.2.1 I2C_IO_MASTER_NUM	137
7.21.2.2 I2C_MASTER_FREQ_HZ	138
7.21.2.3 I2C_MASTER_RX_BUF_DISABLE	138
7.21.2.4 I2C_MASTER_SCL_IO	138
7.21.2.5 I2C_MASTER_SDA_IO	138
7.21.2.6 I2C_MASTER_TIMEOUT_MS	138
7.21.2.7 I2C_MASTER_TX_BUF_DISABLE	138
7.21.3 Documentación de funciones	139
7.21.3.1 i2c_is_hardware_free()	139
7.21.3.2 i2c_master_init()	139
7.21.3.3 i2c_release_hardware()	139
7.21.3.4 MCP23017_INIT()	140
7.21.3.5 mcp_get_on_interrupt_input()	140
7.21.3.6 mcp_interrupt_flag()	141
7.21.3.7 mcp_read_port()	142
7.21.3.8 mcp_register_read()	142
7.21.3.9 mcp_register_write()	143
7.21.3.10 mcp_write_output_pin()	143
7.21.3.11 mcp_write_output_port()	144

7.21.4 Documentación de variables	144
7.21.4.1 TAG	144
7.21.4.2 xI2C_Mutex	145
7.22 MCP23017.c	145
7.23 Referencia del archivo C:/Users/User/LVFV_ESP32/main/io_control/MCP23017.h	149
7.23.1 Descripción detallada	149
7.23.2 Documentación de funciones	150
7.23.2.1 MCP23017_INIT()	150
7.23.2.2 mcp_get_on_interrupt_input()	150
7.23.2.3 mcp_interrupt_flag()	151
7.23.2.4 mcp_read_port()	151
7.23.2.5 mcp_write_output_pin()	152
7.23.2.6 mcp_write_output_port()	152
7.24 MCP23017.h	153
7.25 Referencia del archivo C:/Users/User/LVFV_ESP32/main/io_control/mcp23017_defs.h	153
7.25.1 Documentación de «define»	155
7.25.1.1 __MCP23017_ADDRESS__	155
7.25.1.2 __MCP23017_BANK_SEQUENTIAL__	155
7.25.1.3 __MCP23017_BANK_SPLIT__	155
7.25.1.4 __MCP23017_DISSLW_DISABLED__	155
7.25.1.5 __MCP23017_DISSLW_ENABLED__	155
7.25.1.6 __MCP23017_FUNC_MODE__	156
7.25.1.7 __MCP23017_GPINTEN_DISABLE__	156
7.25.1.8 __MCP23017_GPINTEN_ENABLE__	156
7.25.1.9 __MCP23017_GPPU_PULL_UP_DISABLE__	156
7.25.1.10 __MCP23017_GPPU_PULL_UP_ENABLE__	156
7.25.1.11 __MCP23017_INTCON_CHANGE__	156
7.25.1.12 __MCP23017_INTCON_DEFVAL__	157
7.25.1.13 __MCP23017_INTERRUPT_DISABLE__	157
7.25.1.14 __MCP23017_INTERRUPT_ENABLE__	157
7.25.1.15 __MCP23017_INTPOL_POLARITY_HIGH__	157
7.25.1.16 __MCP23017_INTPOL_POLARITY_LOW__	157
7.25.1.17 __MCP23017_IODIR_INPUT__	157
7.25.1.18 __MCP23017_IODIR_OUTPUT__	158
7.25.1.19 __MCP23017_MIRROR_CONNECTED__	158
7.25.1.20 __MCP23017_MIRROR_UNCONNECTED__	158
7.25.1.21 __MCP23017_ODR_ACTIVE_DRIVER_OUTPUT__	158
7.25.1.22 __MCP23017_ODR_OPEN_DRAIN_OUTPUT__	158
7.25.1.23 __MCP23017_OPPOSITE_LOGIC__	158
7.25.1.24 __MCP23017_POSITIVE_LOGIC__	159
7.25.1.25 __MCP23017_READ__	159
7.25.1.26 __MCP23017_READ_OPCODE__	159

7.25.1.27 __MCP23017_SEQOP_DISABLED	159
7.25.1.28 __MCP23017_SEQOP_ENABLED	159
7.25.1.29 __MCP23017_WRITE	159
7.25.1.30 __MCP23017_WRITE_OPCODE	160
7.25.1.31 MCP23017_DEFVALA_REGISTER	160
7.25.1.32 MCP23017_DEFVALB_REGISTER	160
7.25.1.33 MCP23017_GPINTENA_REGISTER	160
7.25.1.34 MCP23017_GPINTENB_REGISTER	160
7.25.1.35 MCP23017_GPIOA_REGISTER	160
7.25.1.36 MCP23017_GPIOB_REGISTER	161
7.25.1.37 MCP23017_GPPUA_REGISTER	161
7.25.1.38 MCP23017_GPPUB_REGISTER	161
7.25.1.39 MCP23017_INTCAPA_REGISTER	161
7.25.1.40 MCP23017_INTCAPB_REGISTER	161
7.25.1.41 MCP23017_INTCONA_REGISTER	161
7.25.1.42 MCP23017_INTCONB_REGISTER	162
7.25.1.43 MCP23017_INTFA_REGISTER	162
7.25.1.44 MCP23017_INTFB_REGISTER	162
7.25.1.45 MCP23017_IOCON_REGISTER	162
7.25.1.46 MCP23017_IODIRA_REGISTER	162
7.25.1.47 MCP23017_IODIRB_REGISTER	162
7.25.1.48 MCP23017_IPOLA_REGISTER	163
7.25.1.49 MCP23017_IPOLB_REGISTER	163
7.25.1.50 MCP23017_OLATA_REGISTER	163
7.25.1.51 MCP23017_OLATB_REGISTER	163
7.25.2 Documentación de enumeraciones	163
7.25.2.1 mcp_port_e	163
7.26 mcp23017_defs.h	164
7.27 Referencia del archivo C:/Users/User/LVFV_ESP32/main/LVFV_system.h	167
7.27.1 Descripción detallada	168
7.27.2 Documentación de «define»	168
7.27.2.1 LINE_INCREMENT	168
7.27.2.2 SH1106_SIZE_1	169
7.27.2.3 SH1106_SIZE_2	169
7.27.2.4 VARIABLE_EIGHT	169
7.27.2.5 VARIABLE_FIFTH	169
7.27.2.6 VARIABLE_FIRST	169
7.27.2.7 VARIABLE_FOURTH	169
7.27.2.8 VARIABLE_SECOND	169
7.27.2.9 VARIABLE_SEVENTH	169
7.27.2.10 VARIABLE_SIXTH	170
7.27.2.11 VARIABLE_THIRD	170

7.27.3 Documentación de «typedef»	170
7.27.3.1 frequency_settings_SH1106_t	170
7.27.3.2 frequency_settings_t	170
7.27.3.3 security_settings_SH1106_t	170
7.27.3.4 security_settings_t	170
7.27.3.5 sh1106_variable_lines_e	170
7.27.3.6 system_status_e	170
7.27.3.7 system_status_t	170
7.27.3.8 time_settings_SH1106_t	171
7.27.3.9 time_settings_t	171
7.27.4 Documentación de enumeraciones	171
7.27.4.1 sh1106_variable_lines_e	171
7.27.4.2 system_status_e	171
7.27.4.3 systemSignal_e	172
7.28 LVFV_system.h	173
7.29 Referencia del archivo C:/Users/User/LVFV_ESP32/main/main.c	175
7.29.1 Descripción detallada	176
7.29.2 Documentación de funciones	176
7.29.2.1 app_main()	176
7.29.3 Documentación de variables	177
7.29.3.1 TAG	177
7.30 main.c	177
7.31 Referencia del archivo C:/Users/User/LVFV_ESP32/main/nvs/nvs.c	178
7.31.1 Descripción detallada	179
7.31.2 Documentación de «define»	179
7.31.2.1 load_acceleration	179
7.31.2.2 load_desacceleration	179
7.31.2.3 load_frequency	179
7.31.2.4 load_hour_fin	180
7.31.2.5 load_hour_ini	180
7.31.2.6 load_ibus_max	180
7.31.2.7 load_input_variable	180
7.31.2.8 load_min_fin	180
7.31.2.9 load_min_ini	181
7.31.2.10 load_vbus_min	181
7.31.2.11 save_acceleration	181
7.31.2.12 save_desacceleration	181
7.31.2.13 save_frequency	181
7.31.2.14 save_hour_fin	182
7.31.2.15 save_hour_ini	182
7.31.2.16 save_ibus_max	182
7.31.2.17 save_input_variable	182

7.31.2.18 save_min_fin	182
7.31.2.19 save_min_ini	183
7.31.2.20 save_vbus_min	183
7.31.3 Documentación de funciones	183
7.31.3.1 load_16()	183
7.31.3.2 load_variables()	184
7.31.3.3 nvs_init_once()	185
7.31.3.4 save_16()	185
7.31.3.5 save_variables()	186
7.31.4 Documentación de variables	187
7.31.4.1 TAG	187
7.32 nvs.c	187
7.33 Referencia del archivo C:/Users/User/LVFV_ESP32/main/nvs/nvs.h	191
7.33.1 Descripción detallada	191
7.33.2 Documentación de funciones	192
7.33.2.1 load_variables()	192
7.33.2.2 nvs_init_once()	192
7.33.2.3 save_variables()	193
7.34 nvs.h	194
7.35 Referencia del archivo C:/Users/User/LVFV_ESP32/main/rtc/rtc.c	194
7.35.1 Descripción detallada	195
7.35.2 Documentación de funciones	195
7.35.2.1 alarm_start_cb()	195
7.35.2.2 alarm_stop_cb()	195
7.35.2.3 getTime()	196
7.35.2.4 initRTC()	196
7.35.2.5 program_alarm()	196
7.35.2.6 rtc_schedule_alarms()	197
7.35.2.7 setTime()	197
7.35.3 Documentación de variables	198
7.35.3.1 alarm_settings	198
7.35.3.2 rtc_alarms	198
7.35.3.3 TAG	198
7.35.3.4 time_start	198
7.35.3.5 time_stop	198
7.36 rtc.c	199
7.37 Referencia del archivo C:/Users/User/LVFV_ESP32/main/rtc/rtc.h	201
7.37.1 Descripción detallada	201
7.37.2 Documentación de funciones	202
7.37.2.1 getTime()	202
7.37.2.2 initRTC()	202
7.37.2.3 rtc_schedule_alarms()	202

7.37.2.4 setTime()	203
7.38 rtc.h	203
7.39 Referencia del archivo C:/Users/User/LVFV_ESP32/main/system/sysAdmin.c	204
7.39.1 Descripción detallada	205
7.39.2 Documentación de funciones	205
7.39.2.1 accelerating()	205
7.39.2.2 change_frequency()	205
7.39.2.3 desaccelerating()	206
7.39.2.4 engine_emergency_stop()	206
7.39.2.5 engine_emergency_stop_release()	206
7.39.2.6 engine_start()	207
7.39.2.7 engine_stop()	207
7.39.2.8 get_status()	207
7.39.2.9 set_frequency_table()	208
7.39.2.10 set_system_settings()	209
7.39.2.11 update_meas()	209
7.39.3 Documentación de variables	210
7.39.3.1 accelerating_handle	210
7.39.3.2 desaccelerating_handle	210
7.39.3.3 frequency_table	210
7.39.3.4 system_seccurity_settings	210
7.39.3.5 system_status	210
7.39.3.6 TAG	210
7.40 sysAdmin.c	211
7.41 Referencia del archivo C:/Users/User/LVFV_ESP32/main/system/sysAdmin.h	213
7.41.1 Descripción detallada	214
7.41.2 Documentación de funciones	214
7.41.2.1 change_frequency()	214
7.41.2.2 engine_emergency_stop()	215
7.41.2.3 engine_emergency_stop_release()	215
7.41.2.4 engine_start()	215
7.41.2.5 engine_stop()	216
7.41.2.6 get_status()	216
7.41.2.7 set_frequency_table()	216
7.41.2.8 set_system_settings()	217
7.41.2.9 update_meas()	217
7.42 sysAdmin.h	218
7.43 Referencia del archivo C:/Users/User/LVFV_ESP32/main/system/sysControl.c	218
7.43.1 Descripción detallada	219
7.43.2 Documentación de «define»	220
7.43.2.1 PIN_NUM_CLK	220
7.43.2.2 PIN_NUM_CS	220

7.43.2.3 PIN_NUM_MISO	220
7.43.2.4 PIN_NUM_MOSI	220
7.43.2.5 SPI_CLOCK_HZ	220
7.43.2.6 SPI_HOST_USED	220
7.43.2.7 SPI_QUEUE_TX_DEPTH	220
7.43.3 Documentación de funciones	221
7.43.3.1 SPI_communication()	221
7.43.3.2 SPI_Init()	221
7.43.3.3 SPI_SendRequest()	221
7.43.3.4 SystemEventPost()	222
7.43.4 Documentación de variables	222
7.43.4.1 spi_handle	222
7.43.4.2 system_event_queue	223
7.43.4.3 TAG	223
7.44 sysControl.c	223
7.45 Referencia del archivo C:/Users/User/LVFV_ESP32/main/system/sysControl.h	227
7.45.1 Descripción detallada	228
7.45.2 Documentación de enumeraciones	228
7.45.2.1 SPI_Request	228
7.45.2.2 SPI_Response	229
7.45.3 Documentación de funciones	229
7.45.3.1 SPI_communication()	229
7.45.3.2 SPI_Init()	230
7.45.3.3 SystemEventPost()	230
7.46 sysControl.h	231
7.47 Referencia del archivo C:/Users/User/LVFV_ESP32/main/wifi/form.c	231
7.47.1 Documentación de variables	232
7.47.1.1 HTML_FORM	232
7.48 form.c	232
7.49 Referencia del archivo C:/Users/User/LVFV_ESP32/main/wifi/form.h	234
7.49.1 Documentación de variables	234
7.49.1.1 HTML_FORM	234
7.50 form.h	234
7.51 Referencia del archivo C:/Users/User/LVFV_ESP32/main/wifi/wifi.c	234
7.51.1 Documentación de funciones	235
7.51.1.1 get_param_value()	235
7.51.1.2 hex2int()	236
7.51.1.3 root_get_handler()	236
7.51.1.4 save_post_handler()	236
7.51.1.5 start_webserver()	236
7.51.1.6 url_decode()	236
7.51.1.7 wifi_init_softap()	237

7.51.2 Documentación de variables	237
7.51.2.1 TAG	237
7.52 wifi.c	237
7.53 Referencia del archivo C:/Users/User/LVFV_ESP32/main/wifi/wifi.h	241
7.53.1 Documentación de funciones	241
7.53.1.1 start_webserver()	241
7.53.1.2 wifi_init_softap()	241
7.54 wifi.h	241
Índice alfabético	243

Capítulo 1

Listado de tareas pendientes

Global `GPIO_interrupt_attendance_task(void *arg)`

Revisar correctamente el funcionamiento del filtro antirebote.

Capítulo 2

Jerarquía de directorios

2.1. Directories

adc	9
adc.c	49
adc.h	59
display	9
display.c	61
display.h	84
sh1106_graphics.c	88
sh1106_graphics.h	104
sh1106_i2c.c	107
sh1106_i2c.h	110
io_control	10
io_control.c	112
io_control.h	132
MCP23017.c	136
MCP23017.h	149
mcp23017_defs.h	153
main	10
adc	9
adc.c	49
adc.h	59
display	9
display.c	61
display.h	84
sh1106_graphics.c	88
sh1106_graphics.h	104
sh1106_i2c.c	107
sh1106_i2c.h	110
io_control	10
io_control.c	112
io_control.h	132
MCP23017.c	136
MCP23017.h	149
mcp23017_defs.h	153
nvs	10
nvs.c	178
nvs.h	191
rtc	11

rtc.c	194
rtc.h	201
system	11
sysAdmin.c	204
sysAdmin.h	213
sysControl.c	218
sysControl.h	227
wifi	11
form.c	231
form.h	234
wifi.c	234
wifi.h	241
LVFV_system.h	167
main.c	175
nvs	10
nvs.c	178
nvs.h	191
rtc	11
rtc.c	194
rtc.h	201
system	11
sysAdmin.c	204
sysAdmin.h	213
sysControl.c	218
sysControl.h	227
wifi	11
form.c	231
form.h	234
wifi.c	234
wifi.h	241

Capítulo 3

Índice de estructuras de datos

3.1. 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	13
frequency_settings_SH1106_t	14
frequency_settings_t	15
MCP23017_DEFVAL_t	16
MCP23017_GPINTEN_t	19
MCP23017_GPIO_t	21
MCP23017_GPPU_t	23
MCP23017_INTCAP_t	25
MCP23017_INTCON_t	27
MCP23017_INTF_t	29
MCP23017_IOCON_t	31
MCP23017_IODIR_t	33
MCP23017_IPOL_t	35
MCP23017_OLAT_t	37
rtc_alarms_t	39
seccurity_settings_SH1106_t	40
seccurity_settings_t	41
sh1106_t	42
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	43
system_status_t	Estructura con las variables necesarias para establecer las condiciones de trabajo del motor	44
time_settings_SH1106_t	46
time_settings_t	47

Capítulo 4

Índice de archivos

4.1. Lista de archivos

Lista de todos los archivos con breves descripciones:

C:/Users/User/LVFV_ESP32/main/ LVFV_system.h	Declaraciones de estructuras, variables, definiciones y estados generales del sistema utilizados en varios de los archivos	167
C:/Users/User/LVFV_ESP32/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	175
C:/Users/User/LVFV_ESP32/main/adc/ adc.c	Funcion de inicialización y tarea lectura del ADC	49
C:/Users/User/LVFV_ESP32/main/adc/ adc.h	Declaración de funcion de inicialización y tarea lectura del ADC	59
C:/Users/User/LVFV_ESP32/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	61
C:/Users/User/LVFV_ESP32/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	84
C:/Users/User/LVFV_ESP32/main/display/sh1106_graphics.c	Funciones que permiten operar sobre el display	88
C:/Users/User/LVFV_ESP32/main/display/sh1106_graphics.h	Declaración de funciones que permiten operar sobre el display	104
C:/Users/User/LVFV_ESP32/main/display/sh1106_i2c.c	Funciones de hardware para el puerto I2C	107
C:/Users/User/LVFV_ESP32/main/display/sh1106_i2c.h	Declaración de funciones de hardware para el puerto I2C	110
C:/Users/User/LVFV_ESP32/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	112
C:/Users/User/LVFV_ESP32/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	132
C:/Users/User/LVFV_ESP32/main/io_control/ MCP23017.c	Declaración de funciones de inicialización, lectura y escritura a través del puerto I2C el dispositivo MCP23017	136
C:/Users/User/LVFV_ESP32/main/io_control/ MCP23017.h	Declaración de funciones de inicialización, lectura y escritura a través del puerto I2C el dispositivo MCP23017	149

C:/Users/User/LVFV_ESP32/main/io_control/ mcp23017_defs.h	153
C:/Users/User/LVFV_ESP32/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	178
C:/Users/User/LVFV_ESP32/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	191
C:/Users/User/LVFV_ESP32/main/rtc/ rtc.c Funciones de lectura y escritura del RTC y alarmas	194
C:/Users/User/LVFV_ESP32/main/rtc/ rtc.h Header con prototipos de funciones básicas para el lectura y escritura del RTC y alarmas	201
C:/Users/User/LVFV_ESP32/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	204
C:/Users/User/LVFV_ESP32/main/system/ sysAdmin.h Header con las funciones de funcionamiento del sistema	213
C:/Users/User/LVFV_ESP32/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	218
C:/Users/User/LVFV_ESP32/main/system/ sysControl.h Declaraciones de funciones que controlan el sistema del STM32. Además se encontrará los comandos y respuestas que admite el STM32	227
C:/Users/User/LVFV_ESP32/main/wifi/ form.c	231
C:/Users/User/LVFV_ESP32/main/wifi/ form.h	234
C:/Users/User/LVFV_ESP32/main/wifi/ wifi.c	234
C:/Users/User/LVFV_ESP32/main/wifi/ wifi.h	241

Capítulo 5

Documentación de directorios

5.1. Referencia del directorio C:/Users/User/LVFV_ESP32/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.

5.2. Referencia del directorio C:/Users/User/LVFV_ESP32/main/display

Archivos

- archivo [display.c](#)
Funciones de consulta de los ajustes 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 ajustes 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.

5.3. Referencia del directorio

C:/Users/User/LVFV_ESP32/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](#)

5.4. Referencia del directorio C:/Users/User/LVFV_ESP32/main

Directorios

- 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.

5.5. Referencia del directorio C:/Users/User/LVFV_ESP32/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.

5.6. Referencia del directorio C:/Users/User/LVFV_ESP32/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.

5.7. Referencia del directorio C:/Users/User/LVFV_ESP32/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.

5.8. Referencia del directorio C:/Users/User/LVFV_ESP32/main/wifi

Archivos

- archivo [form.c](#)

- archivo [form.h](#)

- archivo [wifi.c](#)

- archivo [wifi.h](#)

Capítulo 6

Documentación de estructuras de datos

6.1. 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`

6.1.1. 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](#).

6.1.2. Documentación de campos

6.1.2.1. `bitmap`

```
const uint8_t* bitmap
```

Puntero al carácter

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

6.1.2.2. h

uint8_t h

Alto del carácter

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

6.1.2.3. w

uint8_t w

Ancho del carácter

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

6.1.2.4. x

int x

Posición en X del carácter

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

6.1.2.5. 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:

- C:/Users/User/LVVF_ESP32/main/display/[sh1106_graphics.c](#)

6.2. Referencia de la estructura frequency_settings_SH1106_t

```
#include <LVVF_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`

6.2.1. Descripción detallada

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

6.2.2. Documentación de campos

6.2.2.1. edit

```
sh1106_variable_lines_e* edit
```

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

6.2.2.2. edit_flag

```
uint8_t* edit_flag
```

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

6.2.2.3. edit_variable

```
uint8_t edit_variable
```

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

6.2.2.4. frequency_settings

```
frequency_settings_t frequency_settings
```

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

6.2.2.5. multiplier

```
uint8_t* multiplier
```

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

La documentación de esta estructura está generada del siguiente archivo:

- C:/Users/User/LVFV_ESP32/main/[LVFV_system.h](#)

6.3. 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`

6.3.1. Descripción detallada

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

6.3.2. Documentación de campos

6.3.2.1. acceleration

`uint16_t acceleration`

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

6.3.2.2. desacceleration

`uint16_t desacceleration`

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

6.3.2.3. freq_regime

`uint16_t freq_regime`

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

6.3.2.4. input_variable

`uint16_t input_variable`

Definición en la línea 91 del archivo [LVFV_system.h](#).

La documentación de esta estructura está generada del siguiente archivo:

- C:/Users/User/LVFV_ESP32/main/[LVFV_system.h](#)

6.4. 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
```

6.4.1. Descripción detallada

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

6.4.2. Documentación de campos

6.4.2.1. all

```
uint8_t all
```

Unificación de la variable

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

6.4.2.2. [struct]

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

6.4.2.3. DEF0

```
uint8_t DEF0
```

Bit 0 de DEFVAL

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

6.4.2.4. DEF1

```
uint8_t DEF1
```

Bit 1 de DEFVAL

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

6.4.2.5. DEF2

uint8_t DEF2

Bit 2 de DEFVAL

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

6.4.2.6. DEF3

uint8_t DEF3

Bit 3 de DEFVAL

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

6.4.2.7. DEF4

uint8_t DEF4

Bit 4 de DEFVAL

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

6.4.2.8. DEF5

uint8_t DEF5

Bit 5 de DEFVAL

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

6.4.2.9. DEF6

uint8_t DEF6

Bit 6 de DEFVAL

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

6.4.2.10. 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:

- C:/Users/User/LVFV_ESP32/main/io_control/[mcp23017_defs.h](#)

6.5. 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

6.5.1. Descripción detallada

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

6.5.2. Documentación de campos

6.5.2.1. all

uint8_t all

Unificación de la variable

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

6.5.2.2. [struct]

struct { ... } bits

6.5.2.3. GPINT0

uint8_t GPINT0

Bit 0 de GPINTEN

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

6.5.2.4. GPINT1

```
uint8_t GPINT1
```

Bit 1 de GPINTEN

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

6.5.2.5. GPINT2

```
uint8_t GPINT2
```

Bit 2 de GPINTEN

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

6.5.2.6. GPINT3

```
uint8_t GPINT3
```

Bit 3 de GPINTEN

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

6.5.2.7. GPINT4

```
uint8_t GPINT4
```

Bit 4 de GPINTEN

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

6.5.2.8. GPINT5

```
uint8_t GPINT5
```

Bit 5 de GPINTEN

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

6.5.2.9. GPINT6

```
uint8_t GPINT6
```

Bit 6 de GPINTEN

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

6.5.2.10. 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:

- C:/Users/User/LVFV_ESP32/main/io_control/[mcp23017_defs.h](#)

6.6. 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

6.6.1. Descripción detallada

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

6.6.2. Documentación de campos

6.6.2.1. all

```
uint8_t all
```

Unificación de la variable

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

6.6.2.2. [struct]

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

6.6.2.3. GP0

```
uint8_t GP0
```

Bit 0 de GPIO

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

6.6.2.4. GP1

```
uint8_t GP1
```

Bit 1 de GPIO

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

6.6.2.5. GP2

```
uint8_t GP2
```

Bit 2 de GPIO

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

6.6.2.6. GP3

```
uint8_t GP3
```

Bit 3 de GPIO

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

6.6.2.7. GP4

```
uint8_t GP4
```

Bit 4 de GPIO

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

6.6.2.8. GP5

```
uint8_t GP5
```

Bit 5 de GPIO

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

6.6.2.9. GP6

uint8_t GP6

Bit 6 de GPIO

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

6.6.2.10. 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:

- C:/Users/User/LVFV_ESP32/main/io_control/[mcp23017_defs.h](#)

6.7. 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

6.7.1. Descripción detallada

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

6.7.2. Documentación de campos

6.7.2.1. all

uint8_t all

Unificación de la variable

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

6.7.2.2. [struct]

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

6.7.2.3. PU0

uint8_t PU0

Bit 0 de GPPU

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

6.7.2.4. PU1

```
uint8_t PU1
```

Bit 1 de GPPU

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

6.7.2.5. PU2

```
uint8_t PU2
```

Bit 2 de GPPU

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

6.7.2.6. PU3

```
uint8_t PU3
```

Bit 3 de GPPU

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

6.7.2.7. PU4

```
uint8_t PU4
```

Bit 4 de GPPU

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

6.7.2.8. PU5

```
uint8_t PU5
```

Bit 5 de GPPU

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

6.7.2.9. PU6

```
uint8_t PU6
```

Bit 6 de GPPU

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

6.7.2.10. 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:

- C:/Users/User/LVFV_ESP32/main/io_control/[mcp23017_defs.h](#)

6.8. 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

6.8.1. Descripción detallada

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

6.8.2. Documentación de campos

6.8.2.1. all

```
uint8_t all
```

Unificación de la variable

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

6.8.2.2. [struct]

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

6.8.2.3. ICP0

```
uint8_t ICP0
```

Bit 0 de INTCAP

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

6.8.2.4. ICP1

```
uint8_t ICP1
```

Bit 1 de INTCAP

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

6.8.2.5. ICP2

```
uint8_t ICP2
```

Bit 2 de INTCAP

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

6.8.2.6. ICP3

```
uint8_t ICP3
```

Bit 3 de INTCAP

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

6.8.2.7. ICP4

```
uint8_t ICP4
```

Bit 4 de INTCAP

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

6.8.2.8. ICP5

```
uint8_t ICP5
```

Bit 5 de INTCAP

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

6.8.2.9. ICP6

```
uint8_t ICP6
```

Bit 6 de INTCAP

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

6.8.2.10. 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:

- C:/Users/User/LVFV_ESP32/main/io_control/[mcp23017_defs.h](#)

6.9. 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**

6.9.1. Descripción detallada

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

6.9.2. Documentación de campos

6.9.2.1. all

```
uint8_t all
```

Unificación de la variable

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

6.9.2.2. [struct]

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

6.9.2.3. IOC0

```
uint8_t IOC0
```

Bit 0 de INTCON

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

6.9.2.4. IOC1

```
uint8_t IOC1
```

Bit 1 de INTCON

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

6.9.2.5. IOC2

```
uint8_t IOC2
```

Bit 2 de INTCON

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

6.9.2.6. IOC3

```
uint8_t IOC3
```

Bit 3 de INTCON

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

6.9.2.7. IOC4

```
uint8_t IOC4
```

Bit 4 de INTCON

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

6.9.2.8. IOC5

```
uint8_t IOC5
```

Bit 5 de INTCON

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

6.9.2.9. IOC6

```
uint8_t IOC6
```

Bit 6 de INTCON

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

6.9.2.10. 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:

- C:/Users/User/LVVF_ESP32/main/io_control/[mcp23017_defs.h](#)

6.10. 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

6.10.1. Descripción detallada

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

6.10.2. Documentación de campos

6.10.2.1. all

```
uint8_t all
```

Unificación de la variable

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

6.10.2.2. [struct]

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

6.10.2.3. INT0

```
uint8_t INT0
```

Bit 0 de INTF

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

6.10.2.4. INT1

```
uint8_t INT1
```

Bit 1 de INTF

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

6.10.2.5. INT2

```
uint8_t INT2
```

Bit 2 de INTF

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

6.10.2.6. INT3

```
uint8_t INT3
```

Bit 3 de INTF

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

6.10.2.7. INT4

```
uint8_t INT4
```

Bit 4 de INTF

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

6.10.2.8. INT5

```
uint8_t INT5
```

Bit 5 de INTF

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

6.10.2.9. INT6

```
uint8_t INT6
```

Bit 6 de INTF

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

6.10.2.10. 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:

- C:/Users/User/LVFV_ESP32/main/io_control/[mcp23017_defs.h](#)

6.11. 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**

6.11.1. Descripción detallada

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

6.11.2. Documentación de campos

6.11.2.1. all

```
uint8_t all
```

Unificación de la variable

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

6.11.2.2. 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](#).

6.11.2.3. [struct]

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

6.11.2.4. DISSLW

```
uint8_t DISSLW
```

Bit 4 de IOCON - Configuración de velocidad del MCP23017

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

6.11.2.5. INTPOL

```
uint8_t INTPOL
```

Bit 1 de IOCON - Configuración de polaridad de las entradas

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

6.11.2.6. 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](#).

6.11.2.7. 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](#).

6.11.2.8. reserved

```
uint8_t reserved
```

Bit 0 de IOCON - Sin uso

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

6.11.2.9. reserved_2

```
uint8_t reserved_2
```

Bit 3 de IOCON - Sin uso

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

6.11.2.10. 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:

- C:/Users/User/LVFV_ESP32/main/io_control/[mcp23017_defs.h](#)

6.12. 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

6.12.1. Descripción detallada

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

6.12.2. Documentación de campos

6.12.2.1. all

```
uint8_t all
```

Unificación de la variable

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

6.12.2.2. [struct]

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

6.12.2.3. IO0

```
uint8_t IO0
```

Bit 0 de IODIR

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

6.12.2.4. IO1

```
uint8_t IO1
```

Bit 1 de IODIR

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

6.12.2.5. IO2

```
uint8_t IO2
```

Bit 2 de IODIR

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

6.12.2.6. IO3

```
uint8_t IO3
```

Bit 3 de IODIR

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

6.12.2.7. IO4

```
uint8_t IO4
```

Bit 4 de IODIR

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

6.12.2.8. IO5

```
uint8_t IO5
```

Bit 5 de IODIR

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

6.12.2.9. IO6

```
uint8_t IO6
```

Bit 6 de IODIR

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

6.12.2.10. 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:

- C:/Users/User/LVFV_ESP32/main/io_control/[mcp23017_defs.h](#)

6.13. 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

6.13.1. Descripción detallada

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

6.13.2. Documentación de campos

6.13.2.1. all

```
uint8_t all
```

Unificación de la variable

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

6.13.2.2. [struct]

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

6.13.2.3. IPO

```
uint8_t IPO
```

Bit 0 de IPOL

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

6.13.2.4. IP1

```
uint8_t IP1
```

Bit 1 de IPOL

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

6.13.2.5. IP2

```
uint8_t IP2
```

Bit 2 de IPOL

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

6.13.2.6. IP3

```
uint8_t IP3
```

Bit 3 de IPOL

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

6.13.2.7. IP4

```
uint8_t IP4
```

Bit 4 de IPOL

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

6.13.2.8. IP5

```
uint8_t IP5
```

Bit 5 de IPOL

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

6.13.2.9. IP6

```
uint8_t IP6
```

Bit 6 de IPOL

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

6.13.2.10. 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:

- C:/Users/User/LVFV_ESP32/main/io_control/[mcp23017_defs.h](#)

6.14. 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

6.14.1. Descripción detallada

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

6.14.2. Documentación de campos

6.14.2.1. all

```
uint8_t all
```

Unificación de la variable

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

6.14.2.2. [struct]

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

6.14.2.3. OL0

```
uint8_t OL0
```

Bit 0 de OLAT

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

6.14.2.4. OL1

```
uint8_t OL1
```

Bit 1 de OLAT

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

6.14.2.5. OL2

```
uint8_t OL2
```

Bit 2 de OLAT

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

6.14.2.6. OL3

```
uint8_t OL3
```

Bit 3 de OLAT

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

6.14.2.7. OL4

`uint8_t OL4`

Bit 4 de OLAT

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

6.14.2.8. OL5

`uint8_t OL5`

Bit 5 de OLAT

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

6.14.2.9. OL6

`uint8_t OL6`

Bit 6 de OLAT

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

6.14.2.10. 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:

- C:/Users/User/LVFV_ESP32/main/io_control/[mcp23017_defs.h](#)

6.15. Referencia de la estructura `rtc_alarms_t`

Campos de datos

- `esp_timer_handle_t start_timer`
- `esp_timer_handle_t stop_timer`

6.15.1. Descripción detallada

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

6.15.2. Documentación de campos

6.15.2.1. start_timer

```
esp_timer_handle_t start_timer
```

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

6.15.2.2. 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:

- C:/Users/User/LVFV_ESP32/main/rtc/[rtc.c](#)

6.16. 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`

6.16.1. Descripción detallada

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

6.16.2. Documentación de campos

6.16.2.1. edit

```
sh1106_variable_lines_e* edit
```

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

6.16.2.2. `edit_flag`

```
uint8_t* edit_flag
```

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

6.16.2.3. `edit_variable`

```
uint8_t edit_variable
```

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

6.16.2.4. `multiplier`

```
uint8_t* multiplier
```

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

6.16.2.5. `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:

- C:/Users/User/LVFV_ESP32/main/[LVFV_system.h](#)

6.17. Referencia de la estructura `seccurity_settings_t`

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

Campos de datos

- `uint16_t vbus_min`
- `uint16_t ibus_max`

6.17.1. Descripción detallada

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

6.17.2. Documentación de campos

6.17.2.1. ibus_max

```
uint16_t ibus_max
```

Definición en la línea 102 del archivo [LVFV_system.h](#).

6.17.2.2. vbus_min

```
uint16_t vbus_min
```

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

La documentación de esta estructura está generada del siguiente archivo:

- C:/Users/User/LVFV_ESP32/main/[LVFV_system.h](#)

6.18. 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]

6.18.1. Descripción detallada

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

6.18.2. Documentación de campos

6.18.2.1. buffer

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

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

6.18.2.2. height

```
uint8_t height
```

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

6.18.2.3. rotation

```
uint8_t rotation
```

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

6.18.2.4. width

```
uint8_t width
```

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

La documentación de esta estructura está generada del siguiente archivo:

- C:/Users/User/LVFV_ESP32/main/display/[sh1106_graphics.h](#)

6.19. 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](#)

6.19.1. 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](#).

6.19.2. Documentación de campos

6.19.2.1. getValue

```
int getValue
```

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

6.19.2.2. request

```
SPI_Request request
```

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

6.19.2.3. setValue

```
int setValue
```

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

La documentación de esta estructura está generada del siguiente archivo:

- C:/Users/User/LVFV_ESP32/main/system/[sysControl.h](#)

6.20. 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](#)

6.20.1. Descripción detallada

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

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

6.20.2. Documentación de campos

6.20.2.1. acceleration

```
uint16_t acceleration
```

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

6.20.2.2. desacceleration

```
uint16_t desacceleration
```

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

6.20.2.3. frequency

```
uint16_t frequency
```

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

6.20.2.4. frequency_destiny

```
uint16_t frequency_destiny
```

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

6.20.2.5. ibus_max

```
uint16_t ibus_max
```

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

6.20.2.6. inputs_status

```
uint8_t inputs_status
```

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

6.20.2.7. status

```
system_status_e status
```

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

6.20.2.8. vbus_min

```
uint16_t vbus_min
```

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

La documentación de esta estructura está generada del siguiente archivo:

- C:/Users/User/LVFV_ESP32/main/[LVFV_system.h](#)

6.21. 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`

6.21.1. Descripción detallada

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

6.21.2. Documentación de campos

6.21.2.1. edit

```
sh1106_variable_lines_e* edit
```

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

6.21.2.2. edit_flag

```
uint8_t* edit_flag
```

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

6.21.2.3. edit_variable

```
uint8_t edit_variable
```

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

6.21.2.4. multiplier

```
uint8_t* multiplier
```

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

6.21.2.5. 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:

- C:/Users/User/LVFV_ESP32/main/[LVFV_system.h](#)

6.22. 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](#)

6.22.1. Descripción detallada

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

6.22.2. Documentación de campos

6.22.2.1. time_start

```
struct tm* time_start
```

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

6.22.2.2. time_stop

```
struct tm* time_stop
```

Definición en la línea 97 del archivo [LVFV_system.h](#).

6.22.2.3. time_system

```
struct tm* time_system
```

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

La documentación de esta estructura está generada del siguiente archivo:

- C:/Users/User/LVFV_ESP32/main/[LVFV_system.h](#)

Capítulo 7

Documentación de archivos

7.1. Referencia del archivo C:/Users/User/LVFV_ESP32/main/adc/adc.c

Función 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

7.1.1. 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](#).

7.1.2. Documentación de «define»

7.1.2.1. ADC_ATTEN_USED

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

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

7.1.2.2. ADC_CH_GPIO34

```
#define ADC_CH_GPIO34 ADC_CHANNEL_6
```

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

7.1.2.3. ADC_CH_GPIO35

```
#define ADC_CH_GPIO35 ADC_CHANNEL_7
```

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

7.1.2.4. ADC_CH_GPIO36

```
#define ADC_CH_GPIO36 ADC_CHANNEL_0
```

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

7.1.2.5. ADC_CH_GPIO39

```
#define ADC_CH_GPIO39 ADC_CHANNEL_3
```

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

7.1.2.6. 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](#).

7.1.2.7. ADC_UNIT_USED

```
#define ADC_UNIT_USED ADC_UNIT_1
```

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

7.1.3. Documentación de funciones

7.1.3.1. 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](#).

7.1.3.2. 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](#).

7.1.3.3. 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](#).

7.1.3.4. `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](#).

7.1.4. Documentación de variables

7.1.4.1. bus_meas_evt_queue

```
QueueHandle_t bus_meas_evt_queue = NULL
```

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

7.1.4.2. calibration_3V3_source

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

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

7.1.4.3. calibration_3V3_source_ok

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

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

7.1.4.4. calibration_5V_source

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

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

7.1.4.5. calibration_5V_source_ok

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

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

7.1.4.6. calibration_bus_voltage

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

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

7.1.4.7. `calibration_bus_voltage_ok`

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

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

7.1.4.8. `calibration_current`

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

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

7.1.4.9. `calibration_current_ok`

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

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

7.1.4.10. `s_adc`

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

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

7.1.4.11. `TAG`

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

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

7.2. `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/adc_oneshot.h"
00020 #include "esp_adcadc/adc_cali.h"
00021 #include "esp_adcadc/adc_cali_scheme.h"
00022 #include "esp_adcadc/adc_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      } 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 }
```

7.3. Referencia del archivo C:/Users/User/LVFV_ESP32/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.

7.3.1. 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](#).

7.3.2. Documentación de funciones

7.3.2.1. 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](#).

7.3.2.2. 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](#).

7.3.2.3. 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](#).

7.4. 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
```

7.5. Referencia del archivo

[C:/Users/User/LVFV_ESP32/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 contínua.
- 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.

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`

7.5.1. 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](#).

7.5.2. Documentación de «define»

7.5.2.1. EDIT_ACCELERATION_INDX

```
#define EDIT_ACCELERATION_INDX VARIABLE_SECOND
```

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

7.5.2.2. EDIT_DESACCELERATION_INDX

```
#define EDIT_DESACCELERATION_INDX VARIABLE_THIRD
```

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

7.5.2.3. EDIT_FREQUENCY_INDX

```
#define EDIT_FREQUENCY_INDX VARIABLE_FIRST
```

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

7.5.2.4. EDIT_HFIN_LINE_INDX

```
#define EDIT_HFIN_LINE_INDX VARIABLE_FOURTH
```

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

7.5.2.5. EDIT_HINI_LINE_INDX

```
#define EDIT_HINI_LINE_INDX VARIABLE_THIRD
```

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

7.5.2.6. EDIT_IBUS_LINE_INDX

```
#define EDIT_IBUS_LINE_INDX VARIABLE_THIRD
```

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

7.5.2.7. EDIT_INPUT_VARIATION_INDX

```
#define EDIT_INPUT_VARIATION_INDX VARIABLE_FOURTH
```

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

7.5.2.8. EDIT_TIME_LINE_INDX

```
#define EDIT_TIME_LINE_INDX VARIABLE_SECOND
```

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

7.5.2.9. EDIT_VBUS_LINE_INDX

```
#define EDIT_VBUS_LINE_INDX VARIABLE_SECOND
```

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

7.5.2.10. FREC_LINE_INDX

```
#define FREC_LINE_INDX VARIABLE_FIRST
```

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

7.5.2.11. FREQUENCY CUADRATIC VARIABLE

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

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

7.5.2.12. FREQUENCY_LINEAR_VARIABLE

```
#define FREQUENCY_LINEAR_VARIABLE 0
```

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

7.5.2.13. HFIN_LINE_INDX

```
#define HFIN_LINE_INDX VARIABLE_FIFTH
```

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

7.5.2.14. HINI_LINE_INDX

```
#define HINI_LINE_INDX VARIABLE_FOURTH
```

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

7.5.2.15. I2C_DISPLAY_MASTER_NUM

```
#define I2C_DISPLAY_MASTER_NUM I2C_NUM_0
```

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

7.5.2.16. I2C_MASTER_FREQ_HZ

```
#define I2C_MASTER_FREQ_HZ 400000
```

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

7.5.2.17. I2C_MASTER_RX_BUF_DISABLE

```
#define I2C_MASTER_RX_BUF_DISABLE 0
```

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

7.5.2.18. I2C_MASTER_SCL_IO

```
#define I2C_MASTER_SCL_IO 5
```

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

7.5.2.19. I2C_MASTER_SDA_IO

```
#define I2C_MASTER_SDA_IO 18
```

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

7.5.2.20. I2C_MASTER_TX_BUF_DISABLE

```
#define I2C_MASTER_TX_BUF_DISABLE 0
```

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

7.5.2.21. IBUS_LINE_INDX

```
#define IBUS_LINE_INDX VARIABLE_THIRD
```

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

7.5.2.22. VBUS_LINE_INDX

```
#define VBUS_LINE_INDX VARIABLE_SECOND
```

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

7.5.3. Documentación de enumeraciones

7.5.3.1. 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](#).

7.5.4. Documentación de funciones

7.5.4.1. 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 901 del archivo [display.c](#).

7.5.4.2. 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](#).

7.5.4.3. 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](#).

7.5.4.4. 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](#).

7.5.4.5. `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](#).

7.5.4.6. `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](#).

7.5.4.7. `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](#).

7.5.4.8. `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](#).

7.5.4.9. 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](#).

7.5.4.10. sh1106_print_emergency()

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

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

7.5.4.11. 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](#).

7.5.4.12. 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](#).

7.5.4.13. 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](#).

7.5.4.14. 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](#).

7.5.4.15. 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](#).

7.5.4.16. 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](#).

7.5.4.17. 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](#).

7.5.4.18. 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](#).

7.5.5. Documentación de variables

7.5.5.1. blink

```
uint8_t blink [static]
```

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

7.5.5.2. button_evt_queue

```
QueueHandle_t button_evt_queue
```

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

7.5.5.3. 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](#).

7.5.5.4. oled

```
sh1106_t oled
```

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

7.5.5.5. screen_displayed

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

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

7.5.5.6. system_frequency_settings

```
frequency_settings_t system_frequency_settings
```

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

7.5.5.7. system_seccurity_settings

```
seccurity_settings_t system_seccurity_settings
```

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

7.5.5.8. system_time_settings

```
time_settings_t system_time_settings
```

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

7.5.5.9. TAG

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

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

7.6. 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_QUADRATIC_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, security_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->security_settings.vbus_min;
00504     system_seccurity_settings.ibus_max = seccurity_settings->security_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
variable seleccionada
00528     sh1106_variable_lines_e variable_lines = VARIABLE_FIRST;
00529
00530     uint8_t top_multiplier = 100, bottom_multiplier = 1; // Incremental de la
variable seleccionada
00531     uint8_t multiplier = 1;
00532
00533     uint32_t edit_variable_min = 0, edit_variable_max = 0; // Variable
seleccionada
00534     uint16_t *edit_variable = NULL;
00535
00536     uint8_t edit = 0;                                  // Edición o selección
00537
00538     struct tm timeinfo = {
00539         .tm_hour = 0,
00540         .tm_min = 0,
00541         .tm_sec = 0,
00542         .tm_mday = 0,
00543         .tm_mon = 0,
00544         .tm_year = 0
00545     };
00546     struct tm time_start = {
00547         .tm_hour = 0,
00548         .tm_min = 0,
00549         .tm_sec = 0,
00550         .tm_mday = 0,
00551         .tm_mon = 0,
00552         .tm_year = 0
00553     };
00554     struct tm time_stop = {
00555         .tm_hour = 0,
00556         .tm_min = 0,
00557         .tm_sec = 0,
00558         .tm_mday = 0,
00559         .tm_mon = 0,
00560         .tm_year = 0
00561     };
00562
00563     struct tm timestamp_edit = {
00564         .tm_hour = 0,
00565         .tm_min = 0,
00566         .tm_sec = 0,
00567         .tm_mday = 0,
00568         .tm_mon = 0,
00569         .tm_year = 0
00570     };
00571     struct tm time_start_edit = {
00572         .tm_hour = 0,
00573         .tm_min = 0,
00574         .tm_sec = 0,
00575         .tm_mday = 0,
00576         .tm_mon = 0,
00577         .tm_year = 0
00578     };
00579     struct tm time_stop_edit = {
00580         .tm_hour = 0,
00581         .tm_min = 0,
00582         .tm_sec = 0,
00583         .tm_mday = 0,
00584         .tm_mon = 0,
00585         .tm_year = 0
00586     };
00587
00588     if ( sh1106_init() != ESP_OK ) {
00589         ESP_LOGE(TAG, "Error al inicializar el display SH1106");
00590         ESP_ERROR_CHECK(ESP_FAIL);
00591     }
00592     ESP_LOGI(TAG, "Inicialización del display SH1106 exitosa");
00593
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_EIGHTH ) {
00783                         edit_variable = (uint16_t*)
00784                             &(time_edit.time_settings.time_stop->tm_min);
00785                         edit_variable_min = 0;
00786                         edit_variable_max = 59;
00787                     }
00788                     edit = 1;
00789                     multiplier = 1;
00790                     top_multiplier = 10;
00791                     bottom_multiplier = 1;
00792                 } else {
00793                     edit = 0;
00794                     edit_variable = NULL;
00795                     edit_variable_min = 0;
00796                     edit_variable_max = 0;
00797                 }
00798             } else if ( screen_displayed == SCREEN_MAIN ) {
00799                 SystemEventPost(START_PRESSED);
00800             }
00801             break;
00802         case BUTTON_BACK:
00803             if ( screen_displayed == SCREEN_SELECT_VARIABLE ) {
00804                 screen_displayed = SCREEN_MAIN;
00805             } else if ( screen_displayed == SCREEN_SECURITY_EDIT || screen_displayed ==
00806             SCREEN_FREQUENCY_EDIT || screen_displayed == SCREEN_TIME_EDIT ) {
00807                 screen_displayed = SCREEN_SELECT_VARIABLE;
00808                 variable_lines = VARIABLE_SECOND;
00809                 top_variable = VARIABLE_SECOND;
00810                 bottom_variable = VARIABLE_FOURTH;
00811                 edit_variable = NULL;
00812                 edit_variable_min = 0;
00813                 edit_variable_max = 0;
00814             } else if ( screen_displayed == SCREEN_MAIN ) {
00815                 SystemEventPost(STOP_PRESSED);
00816                 RelayEventPost( 0 );
00817             }
00818             break;
00819         case BUTTON_UP:
00820             if ( edit_variable != NULL ) {
00821                 (*edit_variable) += multiplier;
00822                 if ( *edit_variable >= edit_variable_max ) {
00823                     *edit_variable = edit_variable_max;
00824                 }
00825             } else {
00826                 if ( variable_lines == top_variable ) {
00827                     variable_lines = bottom_variable;
00828                 } else {
00829                     variable_lines -= LINE_INCREMENT;
00830                 }
00831             }
00832             break;
00833         case BUTTON_DOWN:
00834             if ( edit_variable != NULL ) {

```

```
00832             (*edit_variable) -= multiplier;
00833             if ( *edit_variable < edit_variable_min || *edit_variable > edit_variable_max
00834                 )
00835                 *edit_variable = edit_variable_min;
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             system_variables_save(&frequency_edit, &time_edit, &seccurity_edit);
00863
00864             screen_displayed = SCREEN_MAIN;
00865             edit = 0;
00866             edit_variable = NULL;
00867             edit_variable_min = 0;
00868             edit_variable_max = 0;
00869             break;
00870         }
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_SENT || s_e.status == SYSTEM_EMERGENCY ||
00879             s_e.status == SYSTEM_EMERGENCY_OK ) && blink < 12 ) {
00880             sh1106_print_emergency();
00881         } else {
00882             sh1106_main_screen();
00883         }
00884         break;
00885     case SCREEN_SELECT_VARIABLE:
00886         sh1106_select_edit_variables(variable_lines);
00887         break;
00888     case SCREEN_TIME_EDIT:
00889         sh1106_time_edit_variables(&time_edit);
00890         break;
00891     case SCREEN_SECURITY_EDIT:
00892         sh1106_security_edit_variables(&seccurity_edit);
00893         break;
00894     case SCREEN_FREQUENCY_EDIT:
00895         sh1106_frequency_edit_variables(&frequency_edit);
00896         break;
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 }
00906 }
```

7.7. Referencia del archivo

C:/Users/User/LVFV_ESP32/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.

7.7.1. 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](#).

7.7.2. Documentación de funciones

7.7.2.1. 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 901 del archivo [display.c](#).

7.7.2.2. `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](#).

7.7.2.3. `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](#).

7.7.2.4. `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](#).

7.7.2.5. `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](#).

7.7.2.6. `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

<i>frequency_settings</i>	Parámetros de frecuencia y rampas (Hz, Hz/s).
<i>time_settings</i>	Tiempos de sistema/start/stop (struct tm vía punteros).
<i>security_settings</i>	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](#).

7.7.2.7. `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](#).

7.8. 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, seccurity_settings_SH1106_t *seccurity_settings);
00097
00098 #endif

```

7.9. Referencia del archivo

[C:/Users/User/LVFV_ESP32/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 de le 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 [] = {0x06,0xE,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 `font8x14_space [14] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}`
- static const uint8_t `font5x7_double_dot [7] = {0x18,0x18,0x00,0x00,0x18,0x18,0x00}`
- static const uint8_t `font8x14_double_dot [14] = {0x00,0x38,0x38,0x38,0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x38,0x00,0x00}`
- static const uint8_t `font5x7_dot [7] = {0x00,0x00,0x00,0x00,0x00,0x18,0x18}`
- static const uint8_t `font8x14_dot [14] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x38}`
- static const uint8_t `font5x7_comma [7] = {0x00,0x00,0x00,0x00,0x18,0x18,0x30}`
- static const uint8_t `font8x14_comma [14] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x70}`
- static const uint8_t `font5x7_close_quest [7] = {0x3C,0x66,0x06,0xC,0x18,0x00,0x18}`
- static const uint8_t `font8x14_close_quest [14] = {0x3C,0x3C,0x66,0x66,0x06,0x06,0x0C,0x0C,0x18,0x18,0x00,0x00,0x18,0x00}`
- static const uint8_t `font5x7_close_excl [7] = {0x18,0x18,0x18,0x18,0x18,0x18,0x18}`
- static const uint8_t `font8x14_close_excl [14] = {0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18}`
- static const uint8_t `font5x7_minus [7] = {0x00,0x00,0x00,0x7E,0x00,0x00,0x00}`
- static const uint8_t `font8x14_minus [14] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}`
- 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]`

7.9.1. 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](#).

7.9.2. Documentación de «typedef»

7.9.2.1. bitmap_t

```
typedef struct bitmap_t bitmap_t
```

7.9.3. Documentación de funciones

7.9.3.1. 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](#).

7.9.3.2. 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](#).

7.9.3.3. 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](#).

7.9.3.4. 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](#).

7.9.3.5. 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](#).

7.9.3.6. sh1106_draw_utn_logo()

```
void sh1106_draw_utn_logo (
    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 diujar el texto de caracteres alfanuméricos
<code>in</code>	<code>y</code>	Coordenada en y donde se desea diujar el texto de caracteres alfanuméricos

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

7.9.4. Documentación de variables

7.9.4.1. 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](#).

7.9.4.2. 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](#).

7.9.4.3. 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](#).

7.9.4.4. 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](#).

7.9.4.5. 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](#).

7.9.4.6. 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](#).

7.9.4.7. 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](#).

7.9.4.8. 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](#).

7.9.4.9. 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},  
    {0x33,0x33,0x33,0x3F,0x3F,0x33},  
    {0x33,0x33,0x1E,0x0C,0x1E,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](#).

7.9.4.10. 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},
    {0x00,0x00,0x66,0x66,0x66,0x3C,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](#).

7.9.4.11. 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](#).

7.9.4.12. **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](#).

7.9.4.13. **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](#).

7.9.4.14. **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](#).

7.9.4.15. **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](#).

7.9.4.16. **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](#).

7.9.4.17. `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](#).

7.9.4.18. `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](#).

7.9.4.19. `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,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,0x33,0x33,0x30,0x30,0x1E,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,0x0C},
    {0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33},
    {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](#).

7.9.4.20. 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,0x30,0x30,0x30,0x30,0x30,0x30,0x30},
    {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},
    {0x00,0x00,0x00,0x00,0x3C,0x66,0x66,0x66,0x66,0x66,0x66,0x3C,0x3C},
    {0x00,0x00,0x7C,0x7C,0x66,0x66,0x66,0x66,0x7C,0x60,0x60,0x60,0x60},
    {0x00,0x00,0x3E,0x3E,0x66,0x66,0x66,0x66,0x3E,0x06,0x06,0x06,0x06},
    {0x00,0x00,0x00,0x00,0x6C,0x6C,0x76,0x76,0x60,0x60,0x60,0x60,0x60,0x60},
    {0x00,0x00,0x00,0x00,0x3E,0x60,0x60,0x3C,0x06,0x06,0x7C,0x7C},
    {0x30,0x30,0x30,0x30,0x7C,0x30,0x30,0x30,0x30,0x30,0x1C,0x1C},
    {0x00,0x00,0x00,0x00,0x66,0x66,0x66,0x66,0x66,0x66,0x66,0x3E,0x3E},
    {0x00,0x00,0x00,0x00,0x66,0x66,0x66,0x66,0x66,0x66,0x66,0x3C,0x18,0x18},
    {0x00,0x00,0x00,0x00,0x66,0x66,0x66,0x66,0x7E,0x7E,0x7E,0x6C,0x6C},
    {0x00,0x00,0x00,0x00,0x66,0x66,0x3C,0x18,0x18,0x3C,0x3C,0x66,0x66},
    {0x00,0x00,0x66,0x66,0x66,0x66,0x3E,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](#).

7.9.4.21. font8x14_rownumber

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

Valor inicial:

```
= {
    {0x3C,0x66,0xC3,0xC3,0xC3,0xC3,0xC3,0xC3,0xC3,0xC3,0x66,0x3C,0x00},
    {0x18,0x38,0x78,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x7E,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](#).

7.9.4.22. 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](#).

7.9.4.23. 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](#).

7.9.4.24. logo16_utn_bmp

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

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

7.9.4.25. slash

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

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

7.10. 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] =
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x38}; // 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] =
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x70}; // 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},  

// M  

00048 {0x33,0x3B,0x3F,0x37,0x33,0x33,0x33},  

// N  

00049 {0x1E,0x33,0x33,0x33,0x33,0x33,0x1E},  

// O  

00050 {0x3E,0x33,0x33,0x3E,0x30,0x30,0x30},  

// P  

00051 {0x1E,0x33,0x33,0x33,0x37,0x36,0x1D},  

// Q  

00052 {0x3E,0x33,0x33,0x3E,0x3C,0x36,0x33},  

// R  

00053 {0x1E,0x33,0x30,0x1E,0x03,0x33,0x1E},  

// S  

00054 {0x3F,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C},  

// T  

00055 {0x33,0x33,0x33,0x33,0x33,0x33,0x1E},  

// U  

00056 {0x33,0x33,0x33,0x33,0x33,0x1E,0x0C},  

// V  

00057 {0x33,0x33,0x33,0x33,0x3F,0x3F,0x33},  

// W  

00058 {0x33,0x33,0x1E,0x0C,0x1E,0x33,0x33},  

// X  

00059 {0x33,0x33,0x33,0x1E,0x0C,0x0C,0x0C},  

// Y  

00060 {0x3F,0x03,0x06,0x0C,0x18,0x30,0x3F}  

// Z  

00061 };  

00062 static const uint8_t font5x7_rowminor[] [7] = {  

// Alfabeto minúsculas 5 x 7  

00063 {0x00,0x00,0x3C,0x06,0x3E,0x66,0x3E},  

// a  

00064 {0x60,0x60,0x7C,0x66,0x66,0x66,0x7C},  

// b  

00065 {0x00,0x00,0x3C,0x66,0x60,0x66,0x3C},  

// c  

00066 {0x06,0x06,0x3E,0x66,0x66,0x66,0x3E},  

// d  

00067 {0x00,0x00,0x3C,0x66,0x7E,0x60,0x3C},  

// e  

00068 {0x1C,0x30,0x30,0x7C,0x30,0x30,0x30},  

// f  

00069 {0x00,0x3E,0x66,0x66,0x3E,0x06,0x7C},  

// g  

00070 {0x60,0x60,0x7C,0x66,0x66,0x66,0x66},  

// h  

00071 {0x18,0x00,0x38,0x18,0x18,0x18,0x3C},  

// i  

00072 {0x06,0x00,0x06,0x06,0x66,0x66,0x3C},  

// j  

00073 {0x60,0x60,0x66,0x6C,0x78,0x6C,0x66},  

// k  

00074 {0x38,0x18,0x18,0x18,0x18,0x18,0x3C},  

// l  

00075 {0x00,0x00,0x6C,0x7E,0x7E,0x6C,0x6C},  

// m  

00076 {0x00,0x00,0x7C,0x66,0x66,0x66,0x66},  

// n  

00077 {0x00,0x00,0x3C,0x66,0x66,0x66,0x3C},  

// o  

00078 {0x00,0x7C,0x66,0x66,0x7C,0x60,0x60},  

// p  

00079 {0x00,0x3E,0x66,0x66,0x3E,0x06,0x06},  

// q  

00080 {0x00,0x00,0x6C,0x76,0x60,0x60,0x60},  

// r  

00081 {0x00,0x00,0x3E,0x60,0x3C,0x06,0x7C},  

// s  

00082 {0x30,0x30,0x7C,0x30,0x30,0x30,0x1C},  

// t  

00083 {0x00,0x00,0x66,0x66,0x66,0x66,0x3E},  

// u  

00084 {0x00,0x00,0x66,0x66,0x66,0x3C,0x18},  

// v  

00085 {0x00,0x00,0x66,0x66,0x7E,0x7E,0x6C},  

// w  

00086 {0x00,0x00,0x66,0x3C,0x18,0x3C,0x66},  

// x  

00087 {0x00,0x66,0x66,0x66,0x3E,0x06,0x7C},  

// y  

00088 {0x00,0x7E,0x0C,0x18,0x30,0x7E,0x00}  

// z  

00089 };  

00090 static const uint8_t font5x7_rownumber[] [7] = {  

// Números 5 x 7  

00091 {0x1E,0x33,0x37,0x3B,0x33,0x33,0x1E},  


```

```

// 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, 0x33, 0x33, 0x1E, 0x1E},
    // C
00106 {0x3C, 0x3C, 0x36, 0x36, 0x33, 0x33, 0x33, 0x33, 0x36, 0x36, 0x3C, 0x3C},
    // D
00107 {0x3F, 0x3F, 0x30, 0x30, 0x30, 0x3E, 0x3E, 0x30, 0x30, 0x30, 0x3F, 0x3F},
    // E
00108 {0x3F, 0x3F, 0x30, 0x30, 0x30, 0x3E, 0x3E, 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},
    // 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, 0x3F, 0x3F},
    // L
00115 {0x33, 0x33, 0x3F, 0x3F, 0x3F, 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, 0x37, 0x37, 0x36, 0x36, 0x1D, 0x1D},
    // Q
00120 {0x3E, 0x3E, 0x33, 0x33, 0x33, 0x3E, 0x3C, 0x3C, 0x36, 0x36, 0x33, 0x33},
    // R
00121 {0x1E, 0x1E, 0x33, 0x33, 0x30, 0x1E, 0x1E, 0x03, 0x03, 0x33, 0x1E, 0x1E},
    // S
00122 {0x3F, 0x3F, 0x0C, 0x0C, 0x0C, 0x0C, 0x0C, 0x0C, 0x0C, 0x0C, 0x0C, 0x0C},
    // T
00123 {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},
    // 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},
    // Y
00128 {0x3F, 0x3F, 0x03, 0x03, 0x06, 0x06, 0x0C, 0x0C, 0x18, 0x18, 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,0xC3,0x03,0x03,0x06,0x3C,0x06,0x03,0x03,0xC3,0x66,0x3C,0x00},
00167     // 3
00168     {0x06,0x0E,0x1E,0x36,0x66,0xC6,0x86,0xFF,0xFF,0x06,0x06,0x06,0x06,0x00},
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     {0x0F,0xC3,0x03,0x06,0x0C,0x18,0x30,0x60,0x60,0x60,0x60,0x00},
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;

```

```
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     }
00297 }
```

```

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 }

```

7.11. Referencia del archivo

C:/Users/User/LVFV_ESP32/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)

7.11.1. 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](#).

7.11.2. Documentación de «typedef»

7.11.2.1. sh1106_t

```
typedef struct sh1106_t sh1106_t
```

7.11.3. Documentación de funciones

7.11.3.1. sh1106_draw_arrow()

```
void sh1106_draw_arrow (
    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 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](#).

7.11.3.2. 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](#).

7.11.3.3. 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](#).

7.11.3.4. 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](#).

7.11.3.5. 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](#).

7.12. 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 pixeles, 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
```

7.13. Referencia del archivo

[C:/Users/User/LVFV_ESP32/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.

7.13.1. 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](#).

7.13.2. Documentación de «define»

7.13.2.1. CMD_CONTROL

```
#define CMD_CONTROL 0x00
```

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

7.13.2.2. DATA_CONTROL

```
#define DATA_CONTROL 0x40
```

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

7.13.2.3. I2C_DISPLAY

```
#define I2C_DISPLAY I2C_NUM_0
```

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

7.13.2.4. SH1106_ADDR

```
#define SH1106_ADDR 0x3C
```

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

7.13.3. Documentación de funciones

7.13.3.1. 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](#).

7.14. 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 }

```

7.15. Referencia del archivo

C:/Users/User/LVFV_ESP32/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.*

7.15.1. 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](#).

7.15.2. Documentación de enumeraciones

7.15.2.1. 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](#).

7.15.3. Documentación de funciones

7.15.3.1. 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](#).

7.16. 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
```

7.17. Referencia del archivo

C:/Users/User/LVFV_ESP32/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/FreeRTOS.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)
- 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.
- 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 mili segundos.

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

7.17.1. 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, protegiendo 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 antirruído de las entradas, evitando que se filtrean pulsos de ruido que hayan sido detectados 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](#).

7.17.2. Documentación de «define»

7.17.2.1. 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](#).

7.17.2.2. 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](#).

7.17.2.3. 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](#).

7.17.2.4. 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](#).

7.17.2.5. 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](#).

7.17.2.6. 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](#).

7.17.2.7. 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](#).

7.17.2.8. 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](#).

7.17.2.9. 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](#).

7.17.2.10. 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](#).

7.17.2.11. 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](#).

7.17.2.12. 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](#).

7.17.2.13. 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](#).

7.17.2.14. 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](#).

7.17.2.15. 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](#).

7.17.2.16. 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](#).

7.17.2.17. 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](#).

7.17.2.18. 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](#).

7.17.2.19. 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](#).

7.17.2.20. 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](#).

7.17.2.21. 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](#).

7.17.2.22. 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](#).

7.17.2.23. PWM_STEP_MS

```
#define PWM_STEP_MS 250
```

Cambio cada 500 ms

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

7.17.2.24. 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](#).

7.17.2.25. 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](#).

7.17.2.26. 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](#).

7.17.2.27. 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](#).

7.17.2.28. 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](#).

7.17.2.29. 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](#).

7.17.2.30. 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](#).

7.17.2.31. 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](#).

7.17.2.32. 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](#).

7.17.2.33. 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](#).

7.17.2.34. 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](#).

7.17.2.35. 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](#).

7.17.2.36. 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](#).

7.17.2.37. 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](#).

7.17.2.38. 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](#).

7.17.2.39. 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](#).

7.17.3. Documentación de funciones

7.17.3.1. 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	<i>buzzer_time_on</i>	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 604 del archivo [io_control.c](#).

7.17.3.2. 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](#).

7.17.3.3. 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](#).

7.17.3.4. **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

Tareas pendientes Revisar correctamente el funcionamiento del filtro antirebote.

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

7.17.3.5. **gpio_isr_handler()**

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

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

7.17.3.6. **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](#).

7.17.3.7. **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](#).

7.17.3.8. 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](#).

7.17.3.9. 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 ralay, 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 595 del archivo [io_control.c](#).

7.17.3.10. `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](#).

7.17.4. Documentación de variables

7.17.4.1. `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](#).

7.17.4.2. `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](#).

7.17.4.3. mcp_porta

```
uint8_t mcp_porta
```

Estado de entradas del puerto A del MCP23017

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

7.17.4.4. mcp_portb

```
uint8_t mcp_portb
```

Estado de entradas del puerto B del MCP23017

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

7.17.4.5. 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](#).

7.17.4.6. 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](#).

7.18. 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, "RelayEventPost: %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 }
```

```
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 (last_interrupt);
00534             RelayEventPost ( 1 );
00535         }
00536         break;
00537     case TERMO_SW_RELEASED:
00538         if ( mcp_porta & TERMO_SW_MASK ) {
00539             SystemEventPost (last_interrupt);
00540         }
00541         break;
00542     case STOP_PRESSED:
00543         if ( !(mcp_porta & STOP_SW_MASK) ) {
00544             SystemEventPost (last_interrupt);
00545         }
00546         break;
00547     case STOP_RELEASED:
00548         if ( mcp_porta & STOP_SW_MASK ) {
00549             SystemEventPost (last_interrupt);
00550         }
00551         break;
00552     case SPEED_SELECTOR_0:
00553     case SPEED_SELECTOR_1:
00554     case SPEED_SELECTOR_2:
00555     case SPEED_SELECTOR_3:
00556     case SPEED_SELECTOR_4:
00557     case SPEED_SELECTOR_5:
00558     case SPEED_SELECTOR_6:
00559     case SPEED_SELECTOR_7:
00560     case SPEED_SELECTOR_8:
00561     case SPEED_SELECTOR_9:
```

```

00562             mcp_porta = ~mcp_porta;
00563             speed_selector = mcp_porta & FREQ_SEL_MASK;
00564         // 00ss ss00
00565         // 0000 ssss
00566         if ( speed_selector + SPEED_SELECTOR_0 == last_interrupt ) {
00567             SystemEventPost(last_interrupt);
00568         }
00569         break;
00570     case EMERGENCI_STOP_PRESSED:
00571         if ( gpio_get_level(STOP_BUTTON_PIN) ) {
00572             SystemEventPost(last_interrupt);
00573             RelayEventPost( 1 );
00574         }
00575         break;
00576     case EMERGENCI_STOP_RELEASED:
00577         if ( !gpio_get_level(STOP_BUTTON_PIN) ) {
00578             SystemEventPost(last_interrupt);
00579         }
00580         break;
00581     case START_PRESSED:
00582         if ( !gpio_get_level(START_BUTTON_PIN) ) {
00583             SystemEventPost(last_interrupt);
00584         }
00585         break;
00586     case START_RELEASED:
00587         if ( gpio_get_level(START_BUTTON_PIN) ) {
00588             SystemEventPost(last_interrupt);
00589         }
00590         break;
00591     }
00592     last_interrupt = 0;
00593 }
00594
00595 esp_err_t RelayEventPost( uint8_t relay_event ) {
00596     uint8_t event = relay_event;
00597     ESP_LOGI(TAG, "Relay event: %d", event);
00598     if ( event == 1 || event == 0 ) {
00599         return xQueueSend(relay_evt_queue, &event, 0);
00600     }
00601     return ESP_FAIL;
00602 }
00603
00604 esp_err_t BuzzerEventPost( uint32_t buzzer_time_on ) {
00605     if ( buzzer_time_on > 3000 ) {
00606         buzzer_time_on = 3000;
00607     }
00608     return xQueueSend(buzzer_evt_queue, &buzzer_time_on, 0);
00609 }
```

7.19. Referencia del archivo

C:/Users/User/LVFV_ESP32/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.

7.19.1. 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](#).

7.19.2. Documentación de funciones

7.19.2.1. 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 [604](#) del archivo [io_control.c](#).

7.19.2.2. 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

Tareas pendientes Revisar correctamente el funcionamiento del filtro antirebote.

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

7.19.2.3. 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 595 del archivo [io_control.c](#).

7.19.2.4. 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](#).

7.20. io_control.h

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

```
00001
00009 #ifndef __IO_CONTROL_H__
00010
00011 #define __IO_CONTROL_H__
00012
00023 void GPIO_interrupt_attendance_task(void* arg);
00024
00039 esp_err_t set_freq_output(uint16_t freq);
00040
00055 esp_err_t RelayEventPost( uint8_t relay_event );
00056
00069 esp_err_t BuzzerEventPost( uint32_t buzzer_time_on );
00070
00071 #endif
```

7.21. Referencia del archivo

C:/Users/User/LVFV_ESP32/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/sempchr.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

7.21.1. 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.
- Sequential 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](#).

7.21.2. Documentación de «define»

7.21.2.1. 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](#).

7.21.2.2. 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](#).

7.21.2.3. 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](#).

7.21.2.4. 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](#).

7.21.2.5. 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](#).

7.21.2.6. 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](#).

7.21.2.7. 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](#).

7.21.3. Documentación de funciones

7.21.3.1. 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](#).

7.21.3.2. 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](#).

7.21.3.3. 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](#).

7.21.3.4. 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](#).

7.21.3.5. 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

<i>in</i>	<i>port</i>	Puede tomar los valores PORTA o PORTB, dependiendo del puerto que se quiera acceder.
<i>out</i>	<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](#).

7.21.3.6. 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](#).

7.21.3.7. 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](#).

7.21.3.8. 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 <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 146 del archivo [MCP23017.c](#).

7.21.3.9. 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](#).

7.21.3.10. 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](#).

7.21.3.11. 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 port

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](#).

7.21.4. Documentación de variables**7.21.4.1. TAG**

```
const char* TAG = "MCP23017" [static]
```

Variable que se imprime en los ESP_LOG

Definición en la línea 100 del archivo [MCP23017.c](#).

7.21.4.2. 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](#).

7.22. 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 }
```

```

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_OPPCODE__, 0);
00151     i2c_master_write_byte(cmd, register_address, true);
00152     i2c_master_start(cmd);
00153     i2c_master_write_byte(cmd, __MCP23017_READ_OPPCODE__, 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_OPPCODE__, 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_GPINTEA_t gpintena, gpintenb;
00187     MCP23017_DEFVAL_t defvala, defvalb;
00188     MCP23017_INTCON_t intcona, intconb;
00189
00190     iocon.all = 0x00; // Los pines de interrupción se
00191     iocon.bits.INTPOL = __MCP23017_INTPOL_POLARITY_HIGH__; ponen en 1 cuando hay una interrupción
00192     iocon.bits.ODR = __MCP23017_ODR_ACTIVE_DRIVER_OUTPUT__; // Los pines de interrupción
00193     configuran como salidas activas // Activo el Slew Rate en el SDA
00194     iocon.bits.DISSLW = __MCP23017_DISSLW_ENABLED__; para una mejor comunicación
00195     iocon.bits.SEQOP = __MCP23017_SEQOP_DISABLED__; // No activo el incremento del
00196     puntero de direcciones para poder leer constantemente el mismo registro si quiero
00197     iocon.bits.MIRROR = __MCP23017_MIRROR_UNCONNECTED__; // Los pines de interrupción A y B
00198     se controlan por separado // Uso el modo Byte para tratar
00199     iocon.bits.BANK = __MCP23017_FUNC_MODE__;
00200     los puertos como si guesen de 8 bits y no de 16
00201
00202     gppua.all = __MCP23017_GPPU_PULL_UP_DISABLE__;
00203
00204     gppub.all = __MCP23017_GPPU_PULL_UP_DISABLE__;
00205     gppub.bits.PU0 = __MCP23017_GPPU_PULL_UP_ENABLE__;
00206     gppub.bits.PU1 = __MCP23017_GPPU_PULL_UP_ENABLE__;
00207     gppub.bits.PU2 = __MCP23017_GPPU_PULL_UP_ENABLE__;
00208     gppub.bits.PU3 = __MCP23017_GPPU_PULL_UP_ENABLE__;
00209
00210     iodira.all = 0x00;
00211     iodira.bits.IO0 = __MCP23017_IODIR_OUTPUT__;
00212     iodira.bits.IO1 = __MCP23017_IODIR_OUTPUT__;
00213     iodira.bits.IO2 = __MCP23017_IODIR_INPUT__;
00214     iodira.bits.IO3 = __MCP23017_IODIR_INPUT__;
00215     iodira.bits.IO4 = __MCP23017_IODIR_INPUT__;
00216     iodira.bits.IO5 = __MCP23017_IODIR_INPUT__;
00217     iodira.bits.IO6 = __MCP23017_IODIR_INPUT__;
00218     iodira.bits.IO7 = __MCP23017_IODIR_OUTPUT__;
00219
00220     iodirb.all = 0x00;
00221     iodirb.bits.IO0 = __MCP23017_IODIR_INPUT__;
00222     iodirb.bits.IO1 = __MCP23017_IODIR_INPUT__;
00223     iodirb.bits.IO2 = __MCP23017_IODIR_INPUT__;
00224     iodirb.bits.IO3 = __MCP23017_IODIR_INPUT__;
00225     iodirb.bits.IO4 = __MCP23017_IODIR_OUTPUT__;
00226     iodirb.bits.IO5 = __MCP23017_IODIR_OUTPUT__;
00227     iodirb.bits.IO6 = __MCP23017_IODIR_OUTPUT__;
00228     iodirb.bits.IO7 = __MCP23017_IODIR_OUTPUT__;
00229
00230     gpintena.all = 0x00;

```

```

00227     gpintena.bits.GPIO2 = __MCP23017_GPIO_ENABLE__;
00228     gpintena.bits.GPIO3 = __MCP23017_GPIO_ENABLE__;
00229     gpintena.bits.GPIO4 = __MCP23017_GPIO_ENABLE__;
00230     gpintena.bits.GPIO5 = __MCP23017_GPIO_ENABLE__;
00231     gpintena.bits.GPIO6 = __MCP23017_GPIO_ENABLE__;
00232
00233     gpintenb.all = 0x00;
00234     gpintenb.bits.GPIO0 = __MCP23017_GPIO_ENABLE__;
00235     gpintenb.bits.GPIO1 = __MCP23017_GPIO_ENABLE__;
00236     gpintenb.bits.GPIO2 = __MCP23017_GPIO_ENABLE__;
00237     gpintenb.bits.GPIO3 = __MCP23017_GPIO_ENABLE__;
00238
00239     defvala.all = 0x00;
00240     defvala.bits.GPIO2 = 1;
00241     defvala.bits.GPIO3 = 1;
00242     defvala.bits.GPIO4 = 1;
00243     defvala.bits.GPIO5 = 1;
00244     defvala.bits.GPIO6 = 0;
00245
00246     defvalb.all = 0x0F;
00247
00248     intcona.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
00268     reset_val
00269     ret = mcp_register_write( (uint8_t) MCP23017_GPINTEA_REGISTER , gpintena.all ); //
00270     if ( ret != ESP_OK ) {
00271         ESP_LOGE(TAG, "Error al escribir en el registro GPINTEA");
00272         return ret;
00273     }
00274     ret = mcp_register_write( (uint8_t) MCP23017_GPINENB_REGISTER , gpintenb.all ); //
00275     if ( ret != ESP_OK ) {
00276         ESP_LOGE(TAG, "Error al escribir en el registro GPINENB");
00277         return ret;
00278     }
00279     ret = mcp_register_write( (uint8_t) MCP23017_DEFVALA_REGISTER , defvala.all ); //
00280     if ( ret != ESP_OK ) {
00281         ESP_LOGE(TAG, "Error al escribir en el registro DEFVALA");
00282         return ret;
00283     }
00284     ret = mcp_register_write( (uint8_t) MCP23017_DEFVALB_REGISTER , defvalb.all ); //
00285     if ( ret != ESP_OK ) {
00286         ESP_LOGE(TAG, "Error al escribir en el registro DEFVALB");
00287         return ret;
00288     }
00289     ret = mcp_register_write( (uint8_t) MCP23017_INTCONA_REGISTER , intcona.all ); //
00290     if ( ret != ESP_OK ) {
00291         ESP_LOGE(TAG, "Error al escribir en el registro DEFVALB");
00292         return ret;
00293     }
00294     ret = mcp_register_write( (uint8_t) MCP23017_INTCONB_REGISTER , intconb.all ); //
00295     if ( ret != ESP_OK ) {
00296         ESP_LOGE(TAG, "Error al escribir en el registro DEFVALB");
00297         return ret;
00298     }
00299     ret = mcp_register_write( (uint8_t) MCP23017_GPPUA_REGISTER , gppua.all ); //
00300     if ( ret != ESP_OK ) {
00301         ESP_LOGE(TAG, "Error al escribir en el registro DEFVALB");
00302         return ret;
00303     }
00304     ret = mcp_register_write( (uint8_t) MCP23017_GPPUB_REGISTER , gppub.all ); //
00305     if ( ret != ESP_OK ) {
00306         ESP_LOGE(TAG, "Error al escribir en el registro DEFVALB");
00307         return ret;
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 }
```

7.23. Referencia del archivo

C:/Users/User/LVFV_ESP32/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 escribir el registro OLAT del port del MCP23017.

7.23.1. 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](#).

7.23.2. Documentación de funciones

7.23.2.1. 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](#).

7.23.2.2. 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

<i>in</i>	<i>port</i>	Puede tomar los valores PORTA o PORTB, dependiendo del puerto que se quiera acceder.
<i>out</i>	<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](#).

7.23.2.3. 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](#).

7.23.2.4. 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](#).

7.23.2.5. 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	<code>port</code>	Puede tomar los valores PORTA o PORTB, dependiendo del puerto que se quiera escribir.
in	<code>pin</code>	Pin físico que quiere ser modificado su estado
in	<code>state</code>	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](#).

7.23.2.6. 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	<code>port</code>	Puede tomar los valores PORTA o PORTB, dependiendo del puerto que se quiera escribir.
in	<code>data</code>	Estado True o False que puede tomar los pines del <code>port</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 353 del archivo [MCP23017.c](#).

7.24. 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
```

7.25. Referencia del archivo

C:/Users/User/LVFV_ESP32/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_INTCAP_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_DISSLLW_DISABLED__ 0
- #define __MCP23017_DISSLLW_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 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.

7.25.1. Documentación de «define»

7.25.1.1. __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](#).

7.25.1.2. __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](#).

7.25.1.3. __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](#).

7.25.1.4. __MCP23017_DISSLW_DISABLED__

```
#define __MCP23017_DISSLW_DISABLED__ 0
```

Slew rate deshabilitado

Definición en la línea 20 del archivo [mcp23017_defs.h](#).

7.25.1.5. __MCP23017_DISSLW_ENABLED__

```
#define __MCP23017_DISSLW_ENABLED__ 1
```

Slew rate habilitado

Definición en la línea 21 del archivo [mcp23017_defs.h](#).

7.25.1.6. __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](#).

7.25.1.7. __MCP23017_GPINTEN_DISABLE__

```
#define __MCP23017_GPINTEN_DISABLE__ 0
```

Interrupción de GPIO deshabilitada

Definición en la línea 36 del archivo [mcp23017_defs.h](#).

7.25.1.8. __MCP23017_GPINTEN_ENABLE__

```
#define __MCP23017_GPINTEN_ENABLE__ 1
```

Interrupción de GPIO habilitada

Definición en la línea 35 del archivo [mcp23017_defs.h](#).

7.25.1.9. __MCP23017_GPPU_PULL_UP_DISABLE__

```
#define __MCP23017_GPPU_PULL_UP_DISABLE__ 0
```

DESactivación de pull up de GPIO

Definición en la línea 30 del archivo [mcp23017_defs.h](#).

7.25.1.10. __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](#).

7.25.1.11. __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](#).

7.25.1.12. __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](#).

7.25.1.13. __MCP23017_INTERRUPT_DISABLE__

```
#define __MCP23017_INTERRUPT_DISABLE__ 1
```

Habilitación de la interrupción del GPIO

Definición en la línea 48 del archivo [mcp23017_defs.h](#).

7.25.1.14. __MCP23017_INTERRUPT_ENABLE__

```
#define __MCP23017_INTERRUPT_ENABLE__ 0
```

Deshabilitación de la interrupción del GPIO

Definición en la línea 47 del archivo [mcp23017_defs.h](#).

7.25.1.15. __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](#).

7.25.1.16. __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](#).

7.25.1.17. __MCP23017_IODIR_INPUT__

```
#define __MCP23017_IODIR_INPUT__ 1
```

GPIO configurada como entrada

Definición en la línea 32 del archivo [mcp23017_defs.h](#).

7.25.1.18. __MCP23017_IODIR_OUTPUT__

```
#define __MCP23017_IODIR_OUTPUT__ 0
```

GPIO configurada como salida

Definición en la línea 33 del archivo [mcp23017_defs.h](#).

7.25.1.19. __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](#).

7.25.1.20. __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](#).

7.25.1.21. __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](#).

7.25.1.22. __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](#).

7.25.1.23. __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](#).

7.25.1.24. __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](#).

7.25.1.25. __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](#).

7.25.1.26. __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](#).

7.25.1.27. __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](#).

7.25.1.28. __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](#).

7.25.1.29. __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](#).

7.25.1.30. **MCP23017_WRITE_OPPCODE**

```
#define __MCP23017_WRITE_OPPCODE__ 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](#).

7.25.1.31. **MCP23017_DEFVALA_REGISTER**

```
#define MCP23017_DEFVALA_REGISTER 0x06
```

Dirección de memoria del registro DEFVALA

Definición en la línea 63 del archivo [mcp23017_defs.h](#).

7.25.1.32. **MCP23017_DEFVALB_REGISTER**

```
#define MCP23017_DEFVALB_REGISTER 0x07
```

Dirección de memoria del registro DEFVALB

Definición en la línea 64 del archivo [mcp23017_defs.h](#).

7.25.1.33. **MCP23017_GPINTENA_REGISTER**

```
#define MCP23017_GPINTENA_REGISTER 0x04
```

Dirección de memoria del registro GPINTENA

Definición en la línea 61 del archivo [mcp23017_defs.h](#).

7.25.1.34. **MCP23017_GPINTENB_REGISTER**

```
#define MCP23017_GPINTENB_REGISTER 0x05
```

Dirección de memoria del registro GPINTENB

Definición en la línea 62 del archivo [mcp23017_defs.h](#).

7.25.1.35. **MCP23017_GPIOA_REGISTER**

```
#define MCP23017_GPIOA_REGISTER 0x12
```

Dirección de memoria del registro GPIOA

Definición en la línea 75 del archivo [mcp23017_defs.h](#).

7.25.1.36. MCP23017_GPIOB_REGISTER

```
#define MCP23017_GPIOB_REGISTER 0x13
```

Dirección de memoria del registro GPIOB

Definición en la línea 76 del archivo [mcp23017_defs.h](#).

7.25.1.37. MCP23017_GPPUA_REGISTER

```
#define MCP23017_GPPUA_REGISTER 0x0C
```

Dirección de memoria del registro GPPUA

Definición en la línea 69 del archivo [mcp23017_defs.h](#).

7.25.1.38. MCP23017_GPPUB_REGISTER

```
#define MCP23017_GPPUB_REGISTER 0x0D
```

Dirección de memoria del registro GPPUB

Definición en la línea 70 del archivo [mcp23017_defs.h](#).

7.25.1.39. MCP23017_INTCAPA_REGISTER

```
#define MCP23017_INTCAPA_REGISTER 0x10
```

Dirección de memoria del registro INTCAPA

Definición en la línea 73 del archivo [mcp23017_defs.h](#).

7.25.1.40. MCP23017_INTCAPB_REGISTER

```
#define MCP23017_INTCAPB_REGISTER 0x11
```

Dirección de memoria del registro INTCAPB

Definición en la línea 74 del archivo [mcp23017_defs.h](#).

7.25.1.41. MCP23017_INTCONA_REGISTER

```
#define MCP23017_INTCONA_REGISTER 0x08
```

Dirección de memoria del registro INTCONA

Definición en la línea 65 del archivo [mcp23017_defs.h](#).

7.25.1.42. MCP23017_INTCONB_REGISTER

```
#define MCP23017_INTCONB_REGISTER 0x09
```

Dirección de memoria del registro INTCONB

Definición en la línea 66 del archivo [mcp23017_defs.h](#).

7.25.1.43. MCP23017_INTFA_REGISTER

```
#define MCP23017_INTFA_REGISTER 0x0E
```

Dirección de memoria del registro INTFA

Definición en la línea 71 del archivo [mcp23017_defs.h](#).

7.25.1.44. MCP23017_INTFB_REGISTER

```
#define MCP23017_INTFB_REGISTER 0x0F
```

Dirección de memoria del registro INTFB

Definición en la línea 72 del archivo [mcp23017_defs.h](#).

7.25.1.45. MCP23017_IOCON_REGISTER

```
#define MCP23017_IOCON_REGISTER 0x0A
```

Dirección de memoria del registro IOCON

Definición en la línea 67 del archivo [mcp23017_defs.h](#).

7.25.1.46. MCP23017_IODIRA_REGISTER

```
#define MCP23017_IODIRA_REGISTER 0x00
```

Dirección de memoria del registro IODIRA

Definición en la línea 57 del archivo [mcp23017_defs.h](#).

7.25.1.47. MCP23017_IODIRB_REGISTER

```
#define MCP23017_IODIRB_REGISTER 0x01
```

Dirección de memoria del registro IODIRB

Definición en la línea 58 del archivo [mcp23017_defs.h](#).

7.25.1.48. MCP23017_IPOLA_REGISTER

```
#define MCP23017_IPOLA_REGISTER 0x02
```

Dirección de memoria del registro IPOLA

Definición en la línea 59 del archivo [mcp23017_defs.h](#).

7.25.1.49. MCP23017_IPOLB_REGISTER

```
#define MCP23017_IPOLB_REGISTER 0x03
```

Dirección de memoria del registro IPOLB

Definición en la línea 60 del archivo [mcp23017_defs.h](#).

7.25.1.50. MCP23017_OLATA_REGISTER

```
#define MCP23017_OLATA_REGISTER 0x14
```

Dirección de memoria del registro OLATA

Definición en la línea 77 del archivo [mcp23017_defs.h](#).

7.25.1.51. MCP23017_OLATB_REGISTER

```
#define MCP23017_OLATB_REGISTER 0x15
```

Dirección de memoria del registro OLATB

Definición en la línea 78 del archivo [mcp23017_defs.h](#).

7.25.2. Documentación de enumeraciones

7.25.2.1. 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 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.

Valores de enumeraciones

PORTE	Puerto A del MCP23017
PORTB	Puerto B del MCP23017

Definición en la línea 340 del archivo [mcp23017_defs.h](#).

7.26. 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_DISSLW_DISABLED__ 0
00021 #define __MCP23017_DISSLW_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 /*!< Dirección de memoria del registro IOCON */
00069     #define MCP23017_GPPUA_REGISTER 0x0C
00070     #define MCP23017_GPPUB_REGISTER 0x0D
00071     #define MCP23017_INTFIA_REGISTER 0x0E
00072     #define MCP23017_INTFIB_REGISTER 0x0F
00073     #define MCP23017_INTCAPA_REGISTER 0x10
00074     #define MCP23017_INTCAPB_REGISTER 0x11
00075     #define MCP23017_GPIOA_REGISTER 0x12
00076     #define MCP23017_GPIOB_REGISTER 0x13
00077     #define MCP23017_OLATA_REGISTER 0x14
00078     #define MCP23017_OLATB_REGISTER 0x15

```

```

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     /*!< Dirección de memoria del registro IOCON */
00093     #define MCP23017_GPPUA_REGISTER          0x06
00094     #define MCP23017_GPPUB_REGISTER          0x16
00095     #define MCP23017_INTFIA_REGISTER         0x07
00096     #define MCP23017_INTFB_REGISTER          0x17
00097     #define MCP23017_INTCAPA_REGISTER        0x08
00098     #define MCP23017_INTCAPB_REGISTER        0x18
00099     #define MCP23017_GPIOA_REGISTER          0x09
00100    #define MCP23017_GPIOB_REGISTER          0x19
00101    #define MCP23017_OLATA_REGISTER         0x0A
00102    #define MCP23017_OLATB_REGISTER         0x1A
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 GPO : 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

```

7.27. Referencia del archivo

C:/Users/User/LVFV_ESP32/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 `security_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 `security_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

,

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 }

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.
- enum `system_status_e` {

SYSTEM_IDLE , SYSTEM_ACCEL_DESACCEL , SYSTEM_REGIME , SYSTEM_BREAKING ,

SYSTEM_EMERGENCY , SYSTEM_EMERGENCY_SENT , 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 }

7.27.1. 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](#).

7.27.2. Documentación de «define»

7.27.2.1. LINE_INCREMENT

```
#define LINE_INCREMENT 9
```

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

7.27.2.2. SH1106_SIZE_1

```
#define SH1106_SIZE_1 0
```

Definición en la línea 124 del archivo [LVFV_system.h](#).

7.27.2.3. SH1106_SIZE_2

```
#define SH1106_SIZE_2 1
```

Definición en la línea 125 del archivo [LVFV_system.h](#).

7.27.2.4. VARIABLE_EIGHTH

```
#define VARIABLE_EIGHTH 83
```

Definición en la línea 24 del archivo [LVFV_system.h](#).

7.27.2.5. VARIABLE_FIFTH

```
#define VARIABLE_FIFTH 56
```

Definición en la línea 21 del archivo [LVFV_system.h](#).

7.27.2.6. VARIABLE_FIRST

```
#define VARIABLE_FIRST 20
```

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

7.27.2.7. VARIABLE_FOURTH

```
#define VARIABLE_FOURTH 47
```

Definición en la línea 20 del archivo [LVFV_system.h](#).

7.27.2.8. VARIABLE_SECOND

```
#define VARIABLE_SECOND 29
```

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

7.27.2.9. VARIABLE_SEVENTH

```
#define VARIABLE_SEVENTH 74
```

Definición en la línea 23 del archivo [LVFV_system.h](#).

7.27.2.10. VARIABLE_SIXTH

```
#define VARIABLE_SIXTH 65
```

Definición en la línea 22 del archivo [LVFV_system.h](#).

7.27.2.11. VARIABLE_THIRD

```
#define VARIABLE_THIRD 38
```

Definición en la línea 19 del archivo [LVFV_system.h](#).

7.27.3. Documentación de «typedef»

7.27.3.1. frequency_settings_SH1106_t

```
typedef struct frequency_settings_SH1106_t frequency_settings_SH1106_t
```

7.27.3.2. frequency_settings_t

```
typedef struct frequency_settings_t frequency_settings_t
```

7.27.3.3. seccurity_settings_SH1106_t

```
typedef struct seccurity_settings_SH1106_t seccurity_settings_SH1106_t
```

7.27.3.4. seccurity_settings_t

```
typedef struct seccurity_settings_t seccurity_settings_t
```

7.27.3.5. sh1106_variable_lines_e

```
typedef enum sh1106_variable_lines_e sh1106_variable_lines_e
```

7.27.3.6. system_status_e

```
typedef enum system_status_e system_status_e
```

7.27.3.7. system_status_t

```
typedef struct system_status_t system_status_t
```

7.27.3.8. time_settings_SH1106_t

```
typedef struct time_settings_SH1106_t time_settings_SH1106_t
```

7.27.3.9. time_settings_t

```
typedef struct time_settings_t time_settings_t
```

7.27.4. Documentación de enumeraciones

7.27.4.1. sh1106_variable_lines_e

```
enum sh1106_variable_lines_e
```

Valores de enumeraciones

first	
second	
third	
fourth	
fifth	
sixth	
seventh	

Definición en la línea 77 del archivo [LVFV_system.h](#).

7.27.4.2. 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_SENT	
SYSTEM_EMERGENCY_OK	

Definición en la línea 67 del archivo [LVFV_system.h](#).

7.27.4.3. 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](#).

7.28. 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_SENT,
00074     SYSTEM_EMERGENCY_OK
00075 } system_status_e;
00076
00077 typedef enum sh1106_variable_lines_e{
00078     first = VARIABLE_FIRST,
00079     second = VARIABLE_SECOND,
00080     third = VARIABLE_THIRD,
00081     fourth = VARIABLE_FOURTH,
00082     fifth = VARIABLE_FIFTH,
00083     sixth = VARIABLE_SIXTH,
00084     seventh = VARIABLE_SEVENTH
00085 } sh1106_variable_lines_e;
00086
00087 typedef struct frequency_settings_t {
00088     uint16_t freq_regime;
00089     uint16_t acceleration;
00090     uint16_t desacceleration;
00091     uint16_t input_variable;
00092 } frequency_settings_t;
00093
00094 typedef struct time_settings_t {
00095     struct tm *time_system;
00096     struct tm *time_start;
00097     struct tm *time_stop;
00098 } time_settings_t;
00099
00100 typedef struct security_settings_t {
00101     uint16_t vbus_min;
00102     uint16_t ibus_max;
00103 } security_settings_t;
00104
```

```

00110 typedef struct system_status_t {
00111     uint16_t frequency;
00112     uint16_t frequency_destiny;
00113     uint16_t vbus_min;
00114     uint16_t ibus_max;
00115     uint16_t acceleration;
00116     uint16_t desacceleration;
00117     uint8_t inputs_status;
00118     system_status_e status;
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 seccurity_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

```

7.29. Referencia del archivo C:/Users/User/LVFV_ESP32/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"

7.29.1. 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](#).

7.29.2. Documentación de funciones

7.29.2.1. 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](#).

7.29.3. Documentación de variables

7.29.3.1. TAG

```
const char* TAG = "UTN-CA-PF2025" [static]
```

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

7.30. 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",        4096,
00053                             NULL, 10, NULL, 1);
00054     xTaskCreatePinnedToCore(adc_task,                                     "adc_task",           4096,
00055                             NULL, 9, NULL, 0);
00055     xTaskCreatePinnedToCore(SPI_communication,                         "SPI_communication", 2048,
00056                             NULL, 9, NULL, 0);
00056     xTaskCreatePinnedToCore(GPIO_interrupt_attendance_task, "GPIO_interrupt_attendance_task", 2048,
00057                             NULL, 10, NULL, 0);
00057
00058     wifi_init_softap();
00059     // start_mdns();
00060     start_webserver();
00061
00062     while (1) {
00063         vTaskDelay(pdMS_TO_TICKS(1000));
00064     }
00065 }
```

7.31. Referencia del archivo C:/Users/User/LVFV_ESP32/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** *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.

Variables

- static const char * **TAG** = "NVS"

7.31.1. 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](#).

7.31.2. Documentación de «define»

7.31.2.1. load_acceleration

```
#define load_acceleration(  
    freq_acceleration)
```

Valor:

```
load_16("freq_acce", (freq_acceleration), 5 )
```

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

7.31.2.2. load_desacceleration

```
#define load_desacceleration(  
    freq_desacceleration)
```

Valor:

```
load_16("freq_desa", (freq_desacceleration), 3 )
```

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

7.31.2.3. load_frequency

```
#define load_frequency(  
    freq_op)
```

Valor:

```
load_16("freq_freq", (freq_op), 50 )
```

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

7.31.2.4. `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](#).

7.31.2.5. `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](#).

7.31.2.6. `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](#).

7.31.2.7. `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](#).

7.31.2.8. `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](#).

7.31.2.9. 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](#).

7.31.2.10. 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](#).

7.31.2.11. save_acceleration

```
#define save_acceleration(  
    freq_acceleration)
```

Valor:

```
save_16("freq_acce", (freq_acceleration) )
```

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

7.31.2.12. save_desacceleration

```
#define save_desacceleration(  
    freq_desacceleration)
```

Valor:

```
save_16("freq_desa", (freq_desacceleration) )
```

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

7.31.2.13. save_frequency

```
#define save_frequency(  
    freq_op)
```

Valor:

```
save_16("freq_freq", (freq_op) )
```

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

7.31.2.14. `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](#).

7.31.2.15. `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](#).

7.31.2.16. `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](#).

7.31.2.17. `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](#).

7.31.2.18. `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](#).

7.31.2.19. 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](#).

7.31.2.20. 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](#).

7.31.3. Documentación de funciones

7.31.3.1. 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	<code>var_tag</code>	Nombre del namespace donde se almacena la variable que se está intentando leer. Es un string.
out	<code>value</code>	Puntero donde se guardará el dato que se desea leer de la memoria.
in	<code>defval</code>	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 oartición corrupa.
 - 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 espacia para une nueva entrad 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](#).

7.31.3.2. `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

<code>out</code>	<code>frequency_settings</code>	Estructura de parámetros de frecuencia a cargar.
<code>out</code>	<code>time_settings</code>	Estructura de parámetros de tiempo (ventanas start/stop) a cargar.
<code>out</code>	<code>seccurity_settings</code>	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](#).

7.31.3.3. 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](#).

7.31.3.4. 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](#).

7.31.3.5. 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](#).

7.31.4. Documentación de variables

7.31.4.1. TAG

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

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

7.32. 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 }
```

7.33. Referencia del archivo C:/Users/User/LVFV_ESP32/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.

```
#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.

7.33.1. Descripción detallada

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.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo [nvs.h](#).

7.33.2. Documentación de funciones

7.33.2.1. 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](#).

7.33.2.2. 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](#).

7.33.2.3. 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](#).

7.34. 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

```

7.35. Referencia del archivo C:/Users/User/LVFV_ESP32/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

7.35.1. 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](#).

7.35.2. Documentación de funciones**7.35.2.1. 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](#).

7.35.2.2. 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](#).

7.35.2.3. `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](#).

7.35.2.4. `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](#).

7.35.2.5. `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](#).

7.35.2.6. `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 <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](#).

7.35.2.7. `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>	Estructura 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](#).

7.35.3. Documentación de variables

7.35.3.1. alarm_settings

```
time_settings_t alarm_settings [static]
```

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

7.35.3.2. rtc_alarms

```
rtc_alarms_t rtc_alarms [static]
```

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

7.35.3.3. TAG

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

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

7.35.3.4. time_start

```
struct tm time_start [static]
```

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

7.35.3.5. time_stop

```
struct tm time_stop [static]
```

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

7.36. 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 rtc_alarms_t rtc_alarms; // Variable dedicada a la impresión de
00031 static time_settings_t alarm_settings; // Declaración de contenedor de handlers
00032 static time_start;
00033 static time_stop;
00034
00035
00036 static esp_err_t program_alarm(esp_timer_handle_t *timer_handle, void (*callback)(void *), int hour,
00037 int minute, const char *name);
00038
00039 static void alarm_start_cb(void *arg);
00040
00041 static void alarm_stop_cb(void *arg);
00042
00043 esp_err_t initRTC() {
00044     struct timeval tv = {
00045         .tv_sec = 0,
00046         .tv_usec = 0
00047     };
00048
00049     settimeofday(&tv, NULL); // Cargar en RTC interno
00050
00051     alarm_settings.time_start = &time_start;
00052     alarm_settings.time_stop = &time_stop;
00053
00054     return ESP_OK;
00055 }
00056
00057 esp_err_t setTime(struct tm *setting_time) {
00058
00059     if ( setting_time == NULL ) {
00060         return ESP_ERR_INVALID_ARG;
00061     }
00062
00063     time_t now = mktime(setting_time); // Convierte a timestamp UNIX
00064
00065     struct timeval tv = {
00066         .tv_sec = now,
00067         .tv_usec = 0
00068     };
00069
00070     settimeofday(&tv, NULL); // Cargar en RTC interno
00071     return ESP_OK;
00072 }
00073
00074 esp_err_t getTime(struct tm *current_timeinfo) {
00075     time_t now;
00076
00077     if ( current_timeinfo == NULL ) {
00078         return ESP_ERR_INVALID_ARG;
00079     }
00080
00081     time(&now);
00082     localtime_r(&now, current_timeinfo);
00083
00084     return ESP_OK;
00085 }
00086
00087 esp_err_t rtc_schedule_alarms(time_settings_t *time_settings) {
00088
00089     if ( time_settings != NULL ) {
00090         alarm_settings.time_start->tm_hour = time_settings->time_start->tm_hour;

```

```

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 SystemEventPost(START_PRESSED);
00229 }
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 }

```

7.37. Referencia del archivo C:/Users/User/LVFV_ESP32/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.

7.37.1. 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](#).

7.37.2. Documentación de funciones

7.37.2.1. 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](#).

7.37.2.2. 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](#).

7.37.2.3. 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](#).

7.37.2.4. `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](#).

7.38. 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
```

7.39. Referencia del archivo

C:/Users/User/LVFV_ESP32/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/FreRTOS.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)

Tareas dedicada a incrementar la frecuencia que es impresa en el display de acuerdo a la aceleración y hasta la frecuencia de regimen.
- static void **desaccelerating** (void *pvParameters)

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.
- 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.
- void **engine_emergency_stop_release** ()

Pasa a estado SYSTEM_EMERGENCY_OK para que el usuario pueda enviar la señal de stop al STM32 necesaria para poder arrancar nuevamente el motor.
- void **engine_emergency_stop** ()

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.
- 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

7.39.1. 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](#).

7.39.2. Documentación de funciones

7.39.2.1. 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_ACCLER_DESACCEL a SYSTEM_REGIME

Parámetros

in	<i>pvParameters</i>	Sin uso
----	---------------------	---------

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

7.39.2.2. 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 142 del archivo [sysAdmin.c](#).

7.39.2.3. 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](#).

7.39.2.4. engine_emergency_stop()

```
void engine_emergency_stop ()
```

Pasa la frecuencia de regimen a 0Hz y pasa a estado SYSTEM_EMERGENCY.

Termina las tareas accelerating y desaccelerating

Definición en la línea 128 del archivo [sysAdmin.c](#).

7.39.2.5. engine_emergency_stop_release()

```
void engine_emergency_stop_release ()
```

Pasa a estado SYSTEM_EMERGENCY_OK para que el usuario pueda enviar la señal de stop al STM32 necesaria para poder arrancar nuevamente el motor.

Definición en la línea 124 del archivo [sysAdmin.c](#).

7.39.2.6. 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](#).

7.39.2.7. 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](#).

7.39.2.8. get_status()

```
esp_err_t get_status (
    system_status_t * s_e)
```

Obtiene una réplica del status del sistema.

Parámetros

out	s_e	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 176 del archivo [sysAdmin.c](#).

7.39.2.9. `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

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 222 del archivo [sysAdmin.c](#).

7.39.2.10. `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

in	<i>f_s</i>	Puntero a la estructura con las variables de frecuencia a actualizar
in	<i>s_s</i>	Puntero a la estructura con las variables de seguridad a actualizar

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

7.39.2.11. `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
- SYSTEM_EMERGENCY_SENT: El sistema continúa en estado de emergencia

Definición en la línea 188 del archivo [sysAdmin.c](#).

7.39.3. Documentación de variables

7.39.3.1. accelerating_handle

```
TaskHandle_t accelerating_handle = NULL [static]
```

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

7.39.3.2. desaccelerating_handle

```
TaskHandle_t desaccelerating_handle = NULL [static]
```

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

7.39.3.3. frequency_table

```
uint16_t frequency_table[8] [static]
```

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

7.39.3.4. system_seccurity_settings

```
seccurity_settings_t system_seccurity_settings [static]
```

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

7.39.3.5. system_status

```
system_status_t system_status [static]
```

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

7.39.3.6. TAG

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

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

7.40. 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_security_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     system_status.status = SYSTEM_REGIME;
00065     accelerating_handle = NULL;
00066     vTaskDelete(NULL);
00067 }
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         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 || system_status.status == SYSTEM_EMERGENCY_SENT ) {
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 void engine_emergency_stop_release() {
00125     system_status.status = SYSTEM_EMERGENCY_OK;
00126 }
00127
00128 void engine_emergency_stop() {
00129     if ( accelerating_handle != NULL ) {
00130         vTaskDelete( accelerating_handle );
00131         accelerating_handle = NULL;
00132     }
00133     if ( desaccelerating_handle != NULL ) {
00134         vTaskDelete( desaccelerating_handle );
00135         desaccelerating_handle = NULL;
00136     }
00137     system_status.frequency_destiny = 0;
00138     system_status.frequency = 0;
00139     system_status.status = SYSTEM_EMERGENCY;
00140 }
00141
00142 uint16_t change_frequency(uint8_t speed_slector) {
00143     if (speed_slector > 8) {
00144         ESP_LOGE(TAG, "El numero del indice del selector de velocidad es muy grande %d",
00145             speed_slector);
00146         return 0;
00147     }
00148     system_status.inputs_status = speed_slector;
00149     if ( system_status.frequency > frequency_table[system_status.inputs_status] ) {
00150         if ( accelerating_handle != NULL ) {
00151             vTaskDelete( accelerating_handle );
00152             accelerating_handle = NULL;
00153         }
00154         uint16_t desacceleration = get_system_acceleration();
00155         system_status.frequency_destiny = frequency_table[system_status.inputs_status];
00156         if ( desaccelerating_handle == NULL ) {
00157             xTaskCreatePinnedToCore(desaccelerating, "desaccelerating", 1024, (void*)
00158                 &desacceleration, 9, &desaccelerating_handle, 1);
00159             ESP_LOGI(TAG, "Cambiando la frecuencia de regimen a %d. Desacelerando",
00160                 system_status.frequency_destiny);
00161         } else if ( system_status.frequency < frequency_table[system_status.inputs_status] ) {
00162             if ( desaccelerating_handle != NULL ) {
00163                 vTaskDelete( desaccelerating_handle );
00164                 desaccelerating_handle = NULL;
00165             }
00166             uint16_t acceleration = get_system_acceleration();
00167             system_status.frequency_destiny = frequency_table[system_status.inputs_status];
00168             if ( accelerating_handle == NULL ) {
00169                 xTaskCreatePinnedToCore(accelerating, "accelerating", 1024, (void*) &(acceleration), 9,
00170                     &accelerating_handle, 1);
00171             }
00172             ESP_LOGI(TAG, "Cambiando la frecuencia de regimen a %d. Acelerando",
00173                 system_status.frequency_destiny);
00174     }
00175     system_status.status = SYSTEM_ACCEL_DESACCEL;
00176     return system_status.frequency_destiny;
00177 }
00178
00179 esp_err_t get_status(system_status_t *s_e) {
00180     if (s_e == NULL)
00181         return ESP_ERR_NOT_ALLOWED;
00182     s_e->frequency = system_status.frequency;
00183     s_e->frequency_destiny = system_status.frequency_destiny;
00184     s_e->vbus_min = system_status.vbus_min;
00185     s_e->ibus_max = system_status.ibus_max;
00186     s_e->inputs_status = system_status.inputs_status;
00187     s_e->status = system_status.status;
00188     return ESP_OK;
00189 }
00190

```

```

00188 system_status_e update_meas(uint16_t vbus_meas, uint16_t ibus_meas) {
00189     bool in_emergency = false;
00190     system_status.vbus_min = vbus_meas;
00191     system_status.ibus_max = ibus_meas;
00192
00193     if ( system_status.ibus_max > system_security_settings.ibus_max ) {
00194         in_emergency = true;
00195         if ( system_status.status != SYSTEM_EMERGENCY && system_status.status != SYSTEM_EMERGENCY_SENT )
00196             ESP_LOGE( TAG, "Disparo de emergencia por sobre corriente.");
00197     }
00198
00199     if ( system_status.vbus_min < system_security_settings.vbus_min ) {
00200         in_emergency = true;
00201         if ( system_status.status != SYSTEM_EMERGENCY && system_status.status != SYSTEM_EMERGENCY_SENT )
00202             ESP_LOGE( TAG, "Disparo de emergencia por baja tensión.");
00203     }
00204
00205 }
00206
00207 if ( in_emergency == true ) {
00208     if ( system_status.status != SYSTEM_EMERGENCY_SENT ) {
00209         engine_emergency_stop();
00210         SystemEventPost(SECURITY_EXCEEDED);
00211         ESP_LOGI( TAG, "Mando señal.");
00212         system_status.status = SYSTEM_EMERGENCY_SENT;
00213     }
00214 } else if ( system_status.status == SYSTEM_EMERGENCY_SENT ) {
00215     SystemEventPost(SECURITY_OK);
00216     ESP_LOGI( TAG, "Salgo de emergencia por seguridad.");
00217 }
00218
00219 return system_status.status;
00220 }
00221
00222 void set_frequency_table( uint16_t input_variable, uint16_t freq_regime ) {
00223     if ( input_variable == 1 ) { //lineal
00224         uint16_t frequency_gap = freq_regime / 8;
00225         frequency_table[7] = frequency_gap;
00226         for ( uint8_t i = 6; i > 0; i-- ) {
00227             frequency_table[i] = frequency_table[i + 1] + frequency_gap;
00228             ESP_LOGI(TAG, "frequency_table[%d] =%d", i, frequency_table[i]);
00229         }
00230         frequency_table[0] = freq_regime;
00231         ESP_LOGI(TAG, "frequency_table[0] =%d", frequency_table[0]);
00232     } else if ( input_variable == 2 ) { //Cuadratica
00233         for ( uint8_t i = 0; i < 8; i++ ) {
00234             uint32_t num = (uint32_t)freq_regime * (uint32_t)(i + 1) * (uint32_t)(i + 1);
00235             uint16_t fi = (uint16_t)((num + 32) / 64);
00236             if (fi == 0)
00237                 fi = 1;
00238             frequency_table[7 - i] = fi;
00239             ESP_LOGI(TAG, "frequency_table[%d] =%d", 7 - i, frequency_table[7 - i]);
00240         }
00241     }
00242 }
00243
00244 void set_system_settings( frequency_settings_t *f_s, security_settings_t *s_s ) {
00245     system_status.acceleration = f_s->acceleration;
00246     system_status.desacceleration = f_s->desacceleration;
00247     system_seccurity_settings.vbus_min = s_s->vbus_min;
00248     system_seccurity_settings.ibus_max = s_s->ibus_max;
00249     set_frequency_table( f_s->input_variable, f_s->freq_regime );
00250 }

```

7.41. Referencia del archivo

C:/Users/User/LVFV_ESP32/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.
- `void engine_emergency_stop_release ()`
Pasa a estado SYSTEM_EMERGENCY_OK para que el usuario pueda enviar la señal de stop al STM32 necesaria para poder arrancar nuevamente el motor.
- `void engine_emergency_stop ()`
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.
- `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.

7.41.1. 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](#).

7.41.2. Documentación de funciones

7.41.2.1. `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 destín 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 142 del archivo [sysAdmin.c](#).

7.41.2.2. `engine_emergency_stop()`

```
void engine_emergency_stop ()
```

Pasa la frecuencia de regimen a 0Hz y pasa a estado SYSTEM_EMERGENCY.

Termina las tareas accelerating y desaccelerating

Definición en la línea 128 del archivo [sysAdmin.c](#).

7.41.2.3. `engine_emergency_stop_release()`

```
void engine_emergency_stop_release ()
```

Pasa a estado SYSTEM_EMERGENCY_OK para que el usuario pueda enviar la señal de stop al STM32 necesaria para poder arrancar nuevamente el motor.

Definición en la línea 124 del archivo [sysAdmin.c](#).

7.41.2.4. `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](#).

7.41.2.5. 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](#).

7.41.2.6. get_status()

```
esp_err_t get_status (
    system_status_t * s_e)
```

Obtiene una réplica del status del sistema.

Parámetros

out	s← _e	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 176 del archivo [sysAdmin.c](#).

7.41.2.7. 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 222 del archivo [sysAdmin.c](#).

7.41.2.8. `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	<i>f_s</i>	Puntero a la estructura con las variables de frecuencia a actualizar
in	<i>s_s</i>	Puntero a la estructura con las variables de seguridad a actualizar

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

7.41.2.9. `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
- SYSTEM_EMERGENCY_SENT: El sistema continúa en estado de emergencia

Definición en la línea 188 del archivo [sysAdmin.c](#).

7.42. 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
00056 void engine_emergency_stop_release();
00057
00065 void engine_emergency_stop();
00066
00079 uint16_t change_frequency(uint8_t speed_slector);
00080
00102 system_status_e update_meas(uint16_t vbus_meas, uint16_t ibus_meas);
00103
00117 void set_frequency_table( uint16_t input_variable, uint16_t freq_regime );
00118
00130 void set_system_settings( frequency_settings_t *f_s, security_settings_t *s_s );
00131
00132 #endif

```

7.43. Referencia del archivo

C:/Users/User/LVFV_ESP32/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/FreRTOS.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->set->Value 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

7.43.1. 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](#).

7.43.2. Documentación de «define»

7.43.2.1. 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](#).

7.43.2.2. 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](#).

7.43.2.3. 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](#).

7.43.2.4. 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](#).

7.43.2.5. 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](#).

7.43.2.6. 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](#).

7.43.2.7. 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](#).

7.43.3. Documentación de funciones

7.43.3.1. 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](#).

7.43.3.2. 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](#).

7.43.3.3. 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->set↔Value 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	<i>spi_cdm_item</i>	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](#).

7.43.3.4. 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 [363](#) del archivo [sysControl.c](#).

7.43.4. Documentación de variables**7.43.4.1. spi_handle**

```
spi_device_handle_t spi_handle = NULL [static]
```

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

7.43.4.2. system_event_queue

```
QueueHandle_t system_event_queue = NULL
```

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

7.43.4.3. TAG

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

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

7.44. 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
00045           // comunicación con el STM32. Inicializado en esp_err_t SPI_Init(void)
00046
00047 static SPI_Response SPI_SendRequest(spi_cmd_item_t *spi_cmd_item);
00048
00049 static SPI_Response SPI_SendRequest(spi_cmd_item_t *spi_cmd_item) {
00050
00051     uint8_t tx_buffer[4];
00052     uint8_t rx_buffer[4];
00053     esp_err_t ret;
00054     int flagDevolverValor; // Se necesita devolver el valor por parametro
00055
00056     ESP_LOGI(TAG, "[SPI Module] Request %d", spi_cmd_item->request);
00057
00058     // Chequeo de errores fuera de rango y comando desconocido
00059     if(spi_cmd_item->setValue > 512 || spi_cmd_item->setValue < 0) {
00060         return SPI_RESPONSE_ERR_DATA_INVALID;
00061     }
00062
00063     if(spi_cmd_item->request < SPI_REQUEST_START || spi_cmd_item->request > SPI_REQUEST_EMERGENCY) {
00064         ESP_LOGI(TAG, "[SPI Module] Comando desconocido\n");
00065     }
00066 }
```

```

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     flagDevolverValor = 1;
00089 }else {
00090     flagDevolverValor = 0;
00091 }
00092
00093 // Se arma la cadena a enviar
00094 tx_buffer[0] = spi_cmd_item->request;
00095
00096 if(spi_cmd_item->setValue > 255) {
00097     tx_buffer[1] = 255;
00098     tx_buffer[2] = spi_cmd_item->setValue - 255;
00099     tx_buffer[3] = ';';
00100 }else {
00101     tx_buffer[1] = spi_cmd_item->setValue;
00102     tx_buffer[2] = 0;
00103     tx_buffer[3] = ';';
00104 }
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 ESP_LOGI( TAG, "rx_buffer[0]:%d", rx_buffer[0]);
00142 if(flagDevolverValor && rx_buffer[0] == SPI_RESPONSE_OK) {
00143
00144     spi_cmd_item->getValue = rx_buffer[1] + rx_buffer[2];
00145     ESP_LOGI( TAG, "RxValores:%d - %d", rx_buffer[1], rx_buffer[2]);
00146
00147 }else {
00148     spi_cmd_item->getValue = 0;
00149 }
00150
00151
00152 return rx_buffer[0];
00153 }
00154
00155 esp_err_t SPI_Init(void) {
00156     spi_bus_config_t buscfg = {
00157         .mis0_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,
00170     .flags = SPI_DEVICE_NO_DUMMY           // Sin dummy bits
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 EMERGENCIA_STOP_PRESSED:
00233                 ESP_LOGI(TAG, "Botón de EMERGENCIA presionado");
00234                 item.request = SPI_REQUEST_EMERGENCY;
00235                 item.setValue = 0;
00236                 item.getValue = 0;
00237                 SPI_SendRequest(&item);
00238                 engine_emergency_stop();
00239                 break;
00240             case STOP_PRESSED:
00241                 if ( s_e.status != SYSTEM_EMERGENCY && s_e.status != SYSTEM_EMERGENCY_SENT ) {
00242                     engine_stop();
00243                     ESP_LOGI(TAG, "Botón de Parada presionado");
00244                     item.request = SPI_REQUEST_STOP;
00245                     item.setValue = 0;
00246                     item.getValue = 0;
00247                     SPI_SendRequest(&item);
00248                 } else {
00249                     ESP_LOGI(TAG,"Primero salir de estado de emergencia");
00250                 }
00251                 break;
00252             case SECURITY_OK:
00253                 ESP_LOGI(TAG, "Tensión y corriente normalizadas");
00254                 engine_emergency_stop_release();
00255                 break;

```

```

00256     case TERMO_SW_RELEASED:
00257         ESP_LOGI(TAG, "Termoswitch desactivado");
00258         engine_emergency_stop_release();
00259         break;
00260     case EMERGENCI_STOP_RELEASED:
00261         ESP_LOGI(TAG, "Botón de EMERGENCIA liberado");
00262         engine_emergency_stop_release();
00263         break;
00264     case START_PRESSED:
00265
00266         if ( s_e.status == SYSTEM_IDLE ) {
00267             ESP_LOGI(TAG, "Botón de Inicio presionado");
00268             SPI_Response SPI_Commando_response;
00269
00270             item.request = SPI_REQUEST_SET_FREC;
00271             item.setValue = get_system_frequency();
00272             item.getValue = 0;
00273             SPI_Commando_response = SPI_SendRequest(&item);
00274             if ( SPI_Commando_response != SPI_RESPONSE_OK ) {
00275                 ESP_LOGE(TAG, "Error cargando la frecuencia");
00276                 break;
00277             }
00278
00279             item.request = SPI_REQUEST_SET_ACCEL;
00280             item.setValue = get_system_acceleration();
00281             item.getValue = 0;
00282             SPI_Commando_response = SPI_SendRequest(&item);
00283             if ( SPI_Commando_response != SPI_RESPONSE_OK ) {
00284                 ESP_LOGE(TAG, "Error cargando la aceleracion");
00285                 break;
00286             }
00287
00288             item.request = SPI_REQUEST_SET_DESACEL;
00289             item.setValue = get_system_desacceleration();
00290             item.getValue = 0;
00291             SPI_Commando_response = SPI_SendRequest(&item);
00292             if ( SPI_Commando_response != SPI_RESPONSE_OK ) {
00293                 ESP_LOGE(TAG, "Error cargando la desaceleracion");
00294                 break;
00295             }
00296
00297             item.request = SPI_REQUEST_START;
00298             item.setValue = 0;
00299             item.getValue = 0;
00300             SPI_Commando_response = SPI_SendRequest(&item);
00301             if ( SPI_Commando_response != SPI_RESPONSE_OK ) {
00302                 ESP_LOGE(TAG, "Error arrancando el motor");
00303                 break;
00304             }
00305             uint16_t dest = engine_start();
00306             ESP_LOGI(TAG, "Motor arrancado. Frecuencia destino:%d", dest);
00307         } else {
00308             ESP_LOGE(TAG, "El motor debe estar detenido para poder arrancarlo");
00309         }
00310         break;
00311     case START_RELEASED:
00312         ESP_LOGI(TAG, "Botón de Inicio liberado");
00313         break;
00314     case SECURITY_EXCEEDED:
00315         ESP_LOGI(TAG, "Corriente elevada o tensión reducida");
00316         item.request = SPI_REQUEST_EMERGENCY;
00317         item.setValue = 0;
00318         item.getValue = 0;
00319         SPI_SendRequest(&item);
00320         RelayEventPost( 1 );
00321         break;
00322     case TERMO_SW_PRESSED:
00323         ESP_LOGI(TAG, "Termoswitch activo");
00324         item.request = SPI_REQUEST_EMERGENCY;
00325         item.setValue = 0;
00326         item.getValue = 0;
00327         SPI_SendRequest(&item);
00328         engine_emergency_stop();
00329         break;
00330     case SPEED_SELECTOR_0:
00331     case SPEED_SELECTOR_1:
00332     case SPEED_SELECTOR_2:
00333     case SPEED_SELECTOR_3:
00334     case SPEED_SELECTOR_4:
00335     case SPEED_SELECTOR_5:
00336     case SPEED_SELECTOR_6:
00337     case SPEED_SELECTOR_7:
00338     case SPEED_SELECTOR_8:
00339     case SPEED_SELECTOR_9:
00340
00341         if ( s_e.status == SYSTEM_REGIME || s_e.status == SYSTEM_ACCEL_DESACCEL ) {
00342             ESP_LOGI(TAG, "Cambio de velocidad:%d", new_button - SPEED_SELECTOR_0);

```

```

00343         uint8_t old_input_status = s_e.inputs_status;
00344         item.request = SPI_REQUEST_SET_FREC;
00345         item.setValue = change_frequency( new_button - SPEED_SELECTOR_0 );
00346         item.getValue = 0;
00347         if ( SPI_SendRequest(&item) != SPI_RESPONSE_OK ) {
00348             change_frequency( old_input_status );
00349             ESP_LOGE(TAG, "El STM32 no respondio correctamente");
00350             xQueueSend(system_event_queue, &new_button, pdMS_TO_TICKS(1000));
00351         }
00352     } else {
00353         ESP_LOGI(TAG, "No puede cambiar la velocidad, status =%d", s_e.status);
00354     }
00355     break;
00356 default:
00357     break;
00358 }
00359 }
00360 }
00361 }
00362 esp_err_t SystemEventPost(systemSignal_e event) {
00363     if (system_event_queue == NULL) {
00364         return ESP_FAIL;
00365     }
00366     return xQueueSend(system_event_queue, &event, 0);
00367 }
00368 }
```

7.45. Referencia del archivo

C:/Users/User/LVFV_ESP32/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 que pueda recibirse como respuesta en consecuencia.

Enumeraciones

- enum `SPI_Request` {
 `SPI_REQUEST_START` = 10, `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` = 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.

7.45.1. 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](#).

7.45.2. Documentación de enumeraciones

7.45.2.1. SPI_Request

`enum SPI_Request`

Posibles comandos que puede enviar el STM32.

Valores de enumeraciones

<code>SPI_REQUEST_START</code>	
<code>SPI_REQUEST_STOP</code>	
<code>SPI_REQUEST_SET_FREC</code>	
<code>SPI_REQUEST_SET_ACCEL</code>	
<code>SPI_REQUEST_SET_DESACCEL</code>	
<code>SPI_REQUEST_SET_DIR</code>	
<code>SPI_REQUEST_GET_FREC</code>	
<code>SPI_REQUEST_GET_ACCEL</code>	

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](#).

7.45.2.2. 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](#).

7.45.3. Documentación de funciones

7.45.3.1. 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	arg	Sin uso
---------	-----	---------

Definición en la línea 186 del archivo [sysControl.c](#).

7.45.3.2. 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](#).

7.45.3.3. 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 363 del archivo [sysControl.c](#).

7.46. 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
0019 typedef enum {
0020     SPI_REQUEST_START = 10,                                // 10 - Comando de arranque de motor
0021     SPI_REQUEST_STOP,                                     // 11 - Comando de parada de motor
0022     SPI_REQUEST_SET_FREC,                                 // 12 - Comando para configurar la frecuencia de
        régimen
0023     SPI_REQUEST_SET_ACCEL,                               // 13 - Comando para setear la aceleración hasta la
        frecuencia de régimen
0024     SPI_REQUEST_SET_DESACEL,                            // 14 - Comando para setear la desaceleración a 0Hz o
        frecuencia de régimen
0025     SPI_REQUEST_SET_DIR,                               // 15 - Comando para setear la dirección - TODO
0026     SPI_REQUEST_GET_FREC,                             // 16 - Comando de consulta de frecuencia de régimen
0027     SPI_REQUEST_GET_ACCEL,                            // 17 - Comando de consulta de aceleración
0028     SPI_REQUEST_GET_DESACEL,                          // 18 - Comando de consulta de desaceleración
0029     SPI_REQUEST_GET_DIR,                            // 19 - Comando de consulta de dirección de giro
0030     SPI_REQUEST_IS_STOP,                           // 20 - Comando de consulta para saber si el motor
        está parado o en movimiento
0031     SPI_REQUEST_EMERGENCY,                         // 21 - Comando para entrar en estado de emergencia -
        deja de conmutar la salida
0032     SPI_REQUEST_RESPONSE = 0x50                  // 80 - Comando para pedirle al STM32 la respuesta al
        comando enviado
0033 } SPI_Request;
0034
0040 typedef enum {
0041     SPI_RESPONSE_ERR = 0xA0,                         // 160 - Error en la transmisión del comando por
        problemas de inicialización del handler. El comando no sale del ESP32
0042     SPI_RESPONSE_ERR_CMD_UNKNOWN,                   // 161 - El comando enviado no está dentro los
        posibles
0043     SPI_RESPONSE_ERR_NO_COMMAND,                   // 162 - Llegó mensaje pero sin comando
0044     SPI_RESPONSE_ERR_MOVING,                      // 163 - Se está intentando enviar un comando de start
        con el motor en movimiento
0045     SPI_RESPONSE_ERR_NOT_MOVING,                 // 164 - Se está intentando enviar un comando de stop
        con el motor detenido
0046     SPI_RESPONSE_ERR_DATA_MISSING,                // 165 - Faltó enviar un dato dentro de la trama
0047     SPI_RESPONSE_ERR_DATA_INVALID,              // 166 - Los datos enviados como argumento dentro del
        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
        exitosa
0050 } SPI_Response;
0051
0059 typedef struct {
0060     SPI_Request request;                           // Comando a enviar al STM32
0061     int getValue;                                // Variable devuelta por el STM32 en caso de ser un
        comando get
0062     int setValue;                                // Dato enviado con los comandos set
0063 } spi_cmd_item_t;
0064
0079 esp_err_t SPI_Init(void);
0080
0091 void SPI_communication(void *arg);
0092
0095 esp_err_t SystemEventPost(systemSignal_e event);
0096
0097 #endif // SPI_MODULE_H

```

7.47. Referencia del archivo C:/Users/User/LVFV_ESP32/main/wifi/form.c

Variables

- const char * [HTML_FORM](#)

7.47.1. Documentación de variables

7.47.1.1. HTML_FORM

```
const char* HTML_FORM
```

Definición en la línea 1 del archivo `form.c`.

7.48. 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=\"lineal\">Lineal</option>
00059 "                <option value=\"cuadratica\">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>";
```

7.49. Referencia del archivo C:/Users/User/LVFV_ESP32/main/wifi/form.h

Variables

- const char * [HTML_FORM](#)

7.49.1. Documentación de variables

7.49.1.1. HTML_FORM

```
const char* HTML_FORM [extern]
```

Definición en la línea 1 del archivo [form.c](#).

7.50. 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
```

7.51. Referencia del archivo C:/Users/User/LVFV_ESP32/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"

7.51.1. Documentación de funciones

7.51.1.1. [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

<i>body</i>	cuerpo completo del POST (NUL-terminated).
<i>key</i>	nombre del parámetro (sin '=').
<i>dest</i>	buffer destino para el valor decodificado.
<i>dest_len</i>	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](#).

7.51.1.2. hex2int()

```
int hex2int (
    char c) [static]
```

Convierte un dígito hex a entero (0..15).

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

7.51.1.3. 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](#).

7.51.1.4. 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](#).

7.51.1.5. 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 288 del archivo [wifi.c](#).

7.51.1.6. 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](#).

7.51.1.7. 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 313 del archivo [wifi.c](#).

7.51.2. Documentación de variables

7.51.2.1. TAG

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

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

7.52. 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 variation_str[16] = {0};
00189 char sys_time_str[16] = {0};
00190 char start_time_str[16] = {0};
00191 char stop_time_str[16] = {0};
00192
00193 if (get_param_value(body, "frequency", buf, sizeof(buf))) {
00194     frequency = atoi(buf);
00195     ESP_LOGI(TAG, "Frecuencia: %d", frequency);
00196 }
00197 if (get_param_value(body, "accel", buf, sizeof(buf))) {
00198     accel = atoi(buf);
00199     ESP_LOGI(TAG, "Aceleración: %d", accel);
00200 }
00201 if (get_param_value(body, "decel", buf, sizeof(buf))) {
00202     decel = atoi(buf);
00203     ESP_LOGI(TAG, "Desaceleración: %d", decel);
00204 }
00205 if (get_param_value(body, "imax", buf, sizeof(buf))) {
00206     imax = atoi(buf);
00207     ESP_LOGI(TAG, "Corriente maxima: %d", imax);
00208 }
00209 if (get_param_value(body, "vmin", buf, sizeof(buf))) {
00210     vmin = atoi(buf);
00211     ESP_LOGI(TAG, "Tension minima: %d", vmin);
00212 }
00213 if (get_param_value(body, "variation_str", variation_str, sizeof(variation_str))) {
00214     ESP_LOGI(TAG, "Variacion de entradas %s", variation_str);
00215     if (strcmp(variation_str, "cuadratica") == 0) {
00216         variation = 1;
00217     } else {
00218         variation = 2;
00219     }
00220 // variation_str = "linear" o "quadratic"
00221 }
00222 get_param_value(body, "sys_time", sys_time_str, sizeof(sys_time_str));
00223 get_param_value(body, "start_time", start_time_str, sizeof(start_time_str));
00224 get_param_value(body, "stop_time", stop_time_str, sizeof(stop_time_str));
00225
00226 /* ----- Parsear horas ----- */
00227
00228 int sys_h=0, sys_m=0, sys_s=0;
00229 int start_h=0, start_m=0;
00230 int stop_h=0, stop_m=0;
00231
00232 if (sscanf(sys_time_str, "%2d:%2d:%2d", &sys_h, &sys_m, &sys_s) != 3) {
00233     ESP_LOGW(TAG, "Hora de sistema inválida: %s", sys_time_str);
00234 }
00235 if (sscanf(start_time_str, "%2d:%2d", &start_h, &start_m) != 2) {
00236     ESP_LOGW(TAG, "Hora de arranque inválida: %s", start_time_str);
00237 }
00238 if (sscanf(stop_time_str, "%2d:%2d", &stop_h, &stop_m) != 2) {
00239     ESP_LOGW(TAG, "Hora de parada inválida: %s", stop_time_str);
00240 }
00241 ESP_LOGI(TAG, "Hora de sistema: %02d: %02d: %02d", sys_h, sys_m, sys_s);
00242 ESP_LOGI(TAG, "Hora de arranque: %02d: %02d", start_h, start_m);
00243 ESP_LOGI(TAG, "Hora de parada: %02d: %02d", stop_h, stop_m);
00244
00245 struct tm time_system;
00246 time_system.tm_hour = sys_h;
00247 time_system.tm_min = sys_m;
00248 time_system.tm_sec = sys_s;
00249 struct tm time_start;
00250 time_start.tm_hour = start_h;
00251 time_start.tm_min = start_m;
00252 struct tm time_stop;
00253 time_stop.tm_hour = stop_h;
00254 time_stop.tm_min = stop_m;
00255
00256 frequency_edit.frequency_settings.freq_regime = frequency;
00257 frequency_edit.frequency_settings.acceleration = accel;
00258 frequency_edit.frequency_settings.desacceleration = decel;
00259 frequency_edit.frequency_settings.input_variable = variation;
00260
00261 seccurity_edit.security_settings.ibus_max = imax;

```

```

00263     security_edit.security_settings.vbus_min = vmin;
00264
00265     time_edit.time_settings.time_system = &time_system;
00266     time_edit.time_settings.time_start = &time_start;
00267     time_edit.time_settings.time_stop = &time_stop;
00268
00269     if (get_param_value(body, "cmd", cmd, sizeof(cmd))) {
00270         ESP_LOGI(TAG, "Comando: %s", cmd);
00271         if (strcmp(cmd, "save") == 0) {
00272             system_variables_save(&frequency_edit, &time_edit, &security_edit);
00273         } else if (strcmp(cmd, "start") == 0) {
00274             SystemEventPost(START_PRESSED);
00275         } else if (strcmp(cmd, "stop") == 0) {
00276             SystemEventPost(STOP_PRESSED);
00277         }
00278     }
00279
00280     /* ----- Respuesta al navegador ----- */
00281
00282     httpd_resp_set_type(req, "text/html");
00283     httpd_resp_send(req, HTML_FORM, HTTPD_RESP_USE_STRLEN);
00284
00285     return ESP_OK;
00286 }
00287
00288 httpd_handle_t start_webserver(void) {
00289     httpd_config_t config = HTTPD_DEFAULT_CONFIG();
00290     config.server_port = 80;
00291
00292     httpd_handle_t server = NULL;
00293     if (httpd_start(&server, &config) == ESP_OK) {
00294         httpd_uri_t root_uri = {
00295             .uri      = "/",
00296             .method   = HTTP_GET,
00297             .handler  = root_get_handler,
00298             .user_ctx = NULL
00299         };
00300         httpd_register_uri_handler(server, &root_uri);
00301
00302         httpd_uri_t save_uri = {
00303             .uri      = "/save",
00304             .method   = HTTP_POST,
00305             .handler  = save_post_handler,
00306             .user_ctx = NULL
00307         };
00308         httpd_register_uri_handler(server, &save_uri);
00309     }
00310     return server;
00311 }
00312
00313 void wifi_init_softap(void) {
00314     ESP_ERROR_CHECK(esp_netif_init());
00315     ESP_ERROR_CHECK(esp_event_loop_create_default());
00316     esp_netif_create_default_wifi_ap();
00317
00318     wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
00319     ESP_ERROR_CHECK(esp_wifi_init(&cfg));
00320
00321     wifi_config_t wifi_config = {
00322         .ap = {
00323             .ssid = "LVFV_2025",
00324             .ssid_len = 0,
00325             .channel = 1,
00326             .password = "LVFV_2025",
00327             .max_connection = 1,
00328             .authmode = WIFI_AUTH_WPA_WPA2_PSK
00329         },
00330     };
00331     if (strlen((char *)wifi_config.ap.password) == 0) {
00332         wifi_config.ap.authmode = WIFI_AUTH_OPEN;
00333     }
00334
00335     ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_AP));
00336     ESP_ERROR_CHECK(esp_wifi_set_config(WIFI_IF_AP, &wifi_config));
00337     ESP_ERROR_CHECK(esp_wifi_start());
00338
00339     ESP_LOGI(TAG, "AP iniciado. SSID: %s, pass: %s", wifi_config.ap.ssid, wifi_config.ap.password);
00340 }

```

7.53. Referencia del archivo C:/Users/User/LVFV_ESP32/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).

7.53.1. Documentación de funciones

7.53.1.1. `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 288 del archivo [wifi.c](#).

7.53.1.2. `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 313 del archivo [wifi.c](#).

7.54. 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, 155
__MCP23017_BANK_SEQUENTIAL__
 mcp23017_defs.h, 155
__MCP23017_BANK_SPLIT__
 mcp23017_defs.h, 155
__MCP23017_DISSLW_DISABLED__
 mcp23017_defs.h, 155
__MCP23017_DISSLW_ENABLED__
 mcp23017_defs.h, 155
__MCP23017_FUNC_MODE__
 mcp23017_defs.h, 155
__MCP23017_GPINTEN_DISABLE__
 mcp23017_defs.h, 156
__MCP23017_GPINTEN_ENABLE__
 mcp23017_defs.h, 156
__MCP23017_GPPU_PULL_UP_DISABLE__
 mcp23017_defs.h, 156
__MCP23017_GPPU_PULL_UP_ENABLE__
 mcp23017_defs.h, 156
__MCP23017_INTCON_CHANGE__
 mcp23017_defs.h, 156
__MCP23017_INTCON_DEFVAL__
 mcp23017_defs.h, 156
__MCP23017_INTERRUPT_DISABLE__
 mcp23017_defs.h, 157
__MCP23017_INTERRUPT_ENABLE__
 mcp23017_defs.h, 157
__MCP23017_INTPOL_POLARITY_HIGH__
 mcp23017_defs.h, 157
__MCP23017_INTPOL_POLARITY_LOW__
 mcp23017_defs.h, 157
__MCP23017_IODIR_INPUT__
 mcp23017_defs.h, 157
__MCP23017_IODIR_OUTPUT__
 mcp23017_defs.h, 157
__MCP23017_MIRROR_CONNECTED__
 mcp23017_defs.h, 158
__MCP23017_MIRROR_UNCONNECTED__
 mcp23017_defs.h, 158
__MCP23017_ODR_ACTIVE_DRIVER_OUTPUT__
 mcp23017_defs.h, 158
__MCP23017_ODR_OPEN_DRAIN_OUTPUT__
 mcp23017_defs.h, 158
__MCP23017_OPPOSITE_LOGIC__
 mcp23017_defs.h, 158
__MCP23017_POSITIVE_LOGIC__
 mcp23017_defs.h, 158
__MCP23017_READ_OPCODE__
 mcp23017_defs.h, 159
__MCP23017_READ__
 mcp23017_defs.h, 159
__MCP23017_SEQOP_DISABLED__
 mcp23017_defs.h, 159
__MCP23017_SEQOP_ENABLED__
 mcp23017_defs.h, 159
__MCP23017_WRITE_OPCODE__
 mcp23017_defs.h, 159
__MCP23017_WRITE__
 mcp23017_defs.h, 159

accelerating
 sysAdmin.c, 205
accelerating_handle
 sysAdmin.c, 210
acceleration
 frequency_settings_t, 16
 system_status_t, 44
adc.c
 ADC_ATTEN_USED, 50
 adc_cali_try_init, 52
 ADC_CH_GPIO34, 50
 ADC_CH_GPIO35, 50
 ADC_CH_GPIO36, 51
 ADC_CH_GPIO39, 51
 adc_init, 53
 ADC_PERIOD_MS, 51
 adc_task, 53
 ADC_UNIT_USED, 51
 bus_meas_evt_queue, 55
 calibration_3V3_source, 55
 calibration_3V3_source_ok, 55
 calibration_5V_source, 55
 calibration_5V_source_ok, 55
 calibration_bus_voltage, 55
 calibration_bus_voltage_ok, 55
 calibration_current, 56
 calibration_current_ok, 56
 readADC, 53
 s_adc, 56
 TAG, 56
adc.h
 adc_init, 60
 adc_task, 60
 readADC, 60
 ADC_ATTEN_USED
 adc.c, 50
 adc_cali_try_init

adc.c, 52
 ADC_CH_GPIO34
 adc.c, 50
 ADC_CH_GPIO35
 adc.c, 50
 ADC_CH_GPIO36
 adc.c, 51
 ADC_CH_GPIO39
 adc.c, 51
 adc_init
 adc.c, 53
 adc.h, 60
 ADC_PERIOD_MS
 adc.c, 51
 adc_task
 adc.c, 53
 adc.h, 60
 ADC_UNIT_USED
 adc.c, 51
 alarm_settings
 rtc.c, 198
 alarm_start_cb
 rtc.c, 195
 alarm_stop_cb
 rtc.c, 195
 all
 MCP23017_DEFVAL_t, 17
 MCP23017_GPINTEN_t, 19
 MCP23017_GPIO_t, 21
 MCP23017_GPPU_t, 23
 MCP23017_INTCAP_t, 26
 MCP23017_INTCON_t, 28
 MCP23017_INTF_t, 30
 MCP23017_IOCON_t, 32
 MCP23017_IODIR_t, 34
 MCP23017_IPOL_t, 36
 MCP23017_OLAT_t, 38
 app_main
 main.c, 176
 arrow_bmp
 sh1106_graphics.c, 92
 BANK
 MCP23017_IOCON_t, 32
 bitmap
 bitmap_t, 13
 bitmap_t, 13
 bitmap, 13
 h, 13
 sh1106_graphics.c, 90
 w, 14
 x, 14
 y, 14
 bits
 MCP23017_DEFVAL_t, 17
 MCP23017_GPINTEN_t, 19
 MCP23017_GPIO_t, 21
 MCP23017_GPPU_t, 23
 MCP23017_INTCAP_t, 26
 MCP23017_INTCON_t, 28
 MCP23017_INTF_t, 30
 MCP23017_IOCON_t, 32
 MCP23017_IODIR_t, 34
 MCP23017_IPOL_t, 36
 MCP23017_OLAT_t, 38
 sh1106_t, 42
 bus_meas_evt_queue
 adc.c, 55
 BUTTON_BACK
 LVFV_system.h, 173
 BUTTON_DOWN
 LVFV_system.h, 173
 button_evt_queue
 display.c, 72
 BUTTON_LEFT
 LVFV_system.h, 173
 BUTTON_MENU
 LVFV_system.h, 173
 BUTTON_OK
 LVFV_system.h, 173
 BUTTON_RIGHT
 LVFV_system.h, 173
 BUTTON_SAVE
 LVFV_system.h, 173
 BUTTON_UP
 LVFV_system.h, 173
 buzzer_evt_queue
 io_control.c, 124
 BUZZER_PIN
 io_control.c, 114
 BuzzerEventPost
 io_control.c, 120
 io_control.h, 133
 C:/Users/User/LVFV_ESP32/main/adc/adc.c, 49, 56
 C:/Users/User/LVFV_ESP32/main/adc/adc.h, 59, 61
 C:/Users/User/LVFV_ESP32/main/display/display.c, 61,
 73
 C:/Users/User/LVFV_ESP32/main/display/display.h, 84,
 88
 C:/Users/User/LVFV_ESP32/main/display/sh1106_graphics.c,
 88, 99
 C:/Users/User/LVFV_ESP32/main/display/sh1106_graphics.h,
 104, 107
 C:/Users/User/LVFV_ESP32/main/display/sh1106_i2c.c,
 107, 109
 C:/Users/User/LVFV_ESP32/main/display/sh1106_i2c.h,
 110, 111
 C:/Users/User/LVFV_ESP32/main/io_control/io_control.c,
 112, 125
 C:/Users/User/LVFV_ESP32/main/io_control/io_control.h,
 132, 135
 C:/Users/User/LVFV_ESP32/main/io_control/MCP23017.c,
 136, 145

C:/Users/User/LVFV_ESP32/main/io_control/MCP23017.hDEF6
 149, 153
C:/Users/User/LVFV_ESP32/main/io_control/mcp23017_DEF7
 153, 164
C:/Users/User/LVFV_ESP32/main/LVFV_system.h, 167, 173
C:/Users/User/LVFV_ESP32/main/main.c, 175, 177
C:/Users/User/LVFV_ESP32/main/nvs/nvs.c, 178, 187
C:/Users/User/LVFV_ESP32/main/nvs/nvs.h, 191, 194
C:/Users/User/LVFV_ESP32/main/rtc/rtc.c, 194, 199
C:/Users/User/LVFV_ESP32/main/rtc/rtc.h, 201, 203
C:/Users/User/LVFV_ESP32/main/system/sysAdmin.c,
 204, 211
C:/Users/User/LVFV_ESP32/main/system/sysAdmin.h,
 213, 218
C:/Users/User/LVFV_ESP32/main/system/sysControl.c,
 218, 223
C:/Users/User/LVFV_ESP32/main/system/sysControl.h,
 227, 231
C:/Users/User/LVFV_ESP32/main/wifi/form.c, 231, 232
C:/Users/User/LVFV_ESP32/main/wifi/form.h, 234
C:/Users/User/LVFV_ESP32/main/wifi/wifi.c, 234, 237
C:/Users/User/LVFV_ESP32/main/wifi/wifi.h, 241
calibration_3V3_source
 adc.c, 55
calibration_3V3_source_ok
 adc.c, 55
calibration_5V_source
 adc.c, 55
calibration_5V_source_ok
 adc.c, 55
calibration_bus_voltage
 adc.c, 55
calibration_bus_voltage_ok
 adc.c, 55
calibration_current
 adc.c, 56
calibration_current_ok
 adc.c, 56
change_frequency
 sysAdmin.c, 205
 sysAdmin.h, 214
CMD_CONTROL
 sh1106_i2c.c, 108
DATA_CONTROL
 sh1106_i2c.c, 108
DEF0
 MCP23017_DEFVAL_t, 17
DEF1
 MCP23017_DEFVAL_t, 17
DEF2
 MCP23017_DEFVAL_t, 17
DEF3
 MCP23017_DEFVAL_t, 18
DEF4
 MCP23017_DEFVAL_t, 18
DEF5
 MCP23017_DEFVAL_t, 18
 MCP23017_DEFVAL_t, 18
desaccelerating
 sysAdmin.c, 206
desaccelerating_handle
 sysAdmin.c, 210
desacceleration
 frequency_settings_t, 16
 system_status_t, 44
display.c
 blink, 72
 button_evt_queue, 72
 DisplayEventPost, 67
 EDIT_ACCELERATION_INDX, 63
 EDIT_DESACCELERATION_INDX, 63
 EDIT_FREQUENCY_INDX, 64
 EDIT_HFIN_LINE_INDX, 64
 EDIT_HINI_LINE_INDX, 64
 EDIT_IBUS_LINE_INDX, 64
 EDIT_INPUT_VARIATION_INDX, 64
 EDIT_TIME_LINE_INDX, 64
 EDIT_VBUS_LINE_INDX, 64
 FREC_LINE_INDX, 64
 FREQUENCY CUADRATIC VARIABLE, 65
 FREQUENCY LINEAR VARIABLE, 65
 get_system_acceleration, 67
 get_system_desacceleration, 67
 get_system_frequency, 67
 HFIN_LINE_INDX, 65
 HINI_LINE_INDX, 65
 I2C_DISPLAY_MASTER_NUM, 65
 I2C_MASTER_FREQ_HZ, 65
 i2c_master_init, 68
 I2C_MASTER_RX_BUF_DISABLE, 65
 I2C_MASTER_SCL_IO, 65
 I2C_MASTER_SDA_IO, 66
 I2C_MASTER_TX_BUF_DISABLE, 66
 IBUS_LINE_INDX, 66
 init_seq, 72
 oled, 72
 screen_displayed, 72
 SCREEN_FREQUENCY_EDIT, 67
 SCREEN_MAIN, 66
 SCREEN_SECURITY_EDIT, 66
 SCREEN_SELECT_VARIABLE, 66
 screen_selected_e, 66
 SCREEN_TIME_EDIT, 66
 sh1106_clear_buffer, 68
 sh1106_frequency_edit_variables, 68
 sh1106_init, 68
 sh1106_main_screen, 68
 sh1106_print_emergency, 69
 sh1106_print_frame, 69
 sh1106_refresh, 69
 sh1106_security_edit_variables, 69
 sh1106_select_edit_variables, 70

sh1106_splash_screen, 70
 sh1106_time_edit_variables, 70
 system_frequency_settings, 72
 system_security_settings, 73
 system_time_settings, 73
 system_variables_save, 70
 TAG, 73
 task_display, 71
 VBUS_LINE_INDX, 66
display.h
 DisplayEventPost, 85
 get_system_acceleration, 86
 get_system_desacceleration, 86
 get_system_frequency, 86
 sh1106_init, 86
 system_variables_save, 87
 task_display, 87
DisplayEventPost
 display.c, 67
 display.h, 85
DISSLW
 MCP23017_IOCON_t, 32
DUTY_MAX_PCT
 io_control.c, 114
DUTY_MIN_PCT
 io_control.c, 114
edit
 frequency_settings_SH1106_t, 15
 seccurity_settings_SH1106_t, 40
 time_settings_SH1106_t, 46
EDIT_ACCELERATION_INDX
 display.c, 63
EDIT_DESACCELERATION_INDX
 display.c, 63
edit_flag
 frequency_settings_SH1106_t, 15
 seccurity_settings_SH1106_t, 40
 time_settings_SH1106_t, 46
EDIT_FREQUENCY_INDX
 display.c, 64
EDIT_HFIN_LINE_INDX
 display.c, 64
EDIT_HINI_LINE_INDX
 display.c, 64
EDIT_IBUS_LINE_INDX
 display.c, 64
EDIT_INPUT_VARIATION_INDX
 display.c, 64
EDIT_TIME_LINE_INDX
 display.c, 64
edit_variable
 frequency_settings_SH1106_t, 15
 seccurity_settings_SH1106_t, 41
 time_settings_SH1106_t, 46
EDIT_VBUS_LINE_INDX
 display.c, 64
EMERGENCI_STOP_PRESSED
 LVFV_system.h, 173
EMERGENCI_STOP_RELEASED
 LVFV_system.h, 173
engine_emergency_stop
 sysAdmin.c, 206
 sysAdmin.h, 215
engine_emergency_stop_release
 sysAdmin.c, 206
 sysAdmin.h, 215
engine_start
 sysAdmin.c, 206
 sysAdmin.h, 215
engine_stop
 sysAdmin.c, 207
 sysAdmin.h, 215
fail_bmp
 sh1106_graphics.c, 92
fifth
 LVFV_system.h, 171
first
 LVFV_system.h, 171
font5x7_close_excl
 sh1106_graphics.c, 92
font5x7_close_quest
 sh1106_graphics.c, 92
font5x7_comma
 sh1106_graphics.c, 92
font5x7_dot
 sh1106_graphics.c, 92
font5x7_double_dot
 sh1106_graphics.c, 92
font5x7_minus
 sh1106_graphics.c, 93
font5x7_rowmajor
 sh1106_graphics.c, 93
font5x7_rowminor
 sh1106_graphics.c, 93
font5x7_rownumber
 sh1106_graphics.c, 94
font5x7_space
 sh1106_graphics.c, 95
font8x14_close_excl
 sh1106_graphics.c, 95
font8x14_close_quest
 sh1106_graphics.c, 95
font8x14_comma
 sh1106_graphics.c, 95
font8x14_dot
 sh1106_graphics.c, 95
font8x14_double_dot
 sh1106_graphics.c, 95
font8x14_minus
 sh1106_graphics.c, 96
font8x14_rowmajor
 sh1106_graphics.c, 96
font8x14_rowminor
 sh1106_graphics.c, 96
font8x14_rownumber
 sh1106_graphics.c, 97

font8x14_space
 sh1106_graphics.c, 98

form.c
 HTML_FORM, 232

form.h
 HTML_FORM, 234

fourth
 LVFV_system.h, 171

FREC_LINE_INDX
 display.c, 64

freq_0_10V_output
 io_control.c, 121

freq_regime
 frequency_settings_t, 16

FREQ_SEL_1_PIN
 io_control.c, 114

FREQ_SEL_2_PIN
 io_control.c, 115

FREQ_SEL_3_PIN
 io_control.c, 115

FREQ_SEL_MASK
 io_control.c, 115

frequency
 system_status_t, 45

FREQUENCY CUADRATIC VARIABLE
 display.c, 65

frequency_destiny
 system_status_t, 45

FREQUENCY LINEAR VARIABLE
 display.c, 65

frequency_settings
 frequency_settings_SH1106_t, 15

frequency_settings_SH1106_t, 14
 edit, 15
 edit_flag, 15
 edit_variable, 15
 frequency_settings, 15
 LVFV_system.h, 170
 multiplier, 15

frequency_settings_t, 15
 acceleration, 16
 desacceleration, 16
 freq_regime, 16
 input_variable, 16
 LVFV_system.h, 170

frequency_table
 sysAdmin.c, 210

get_param_value
 wifi.c, 235

get_status
 sysAdmin.c, 207
 sysAdmin.h, 216

get_system_acceleration
 display.c, 67
 display.h, 86

get_system_desacceleration
 display.c, 67
 display.h, 86

get_system_frequency
 display.c, 67
 display.h, 86

getTime
 rtc.c, 196
 rtc.h, 202

getValue
 spi_cmd_item_t, 43

GP0
 MCP23017_GPIO_t, 21

GP1
 MCP23017_GPIO_t, 22

GP2
 MCP23017_GPIO_t, 22

GP3
 MCP23017_GPIO_t, 22

GP4
 MCP23017_GPIO_t, 22

GP5
 MCP23017_GPIO_t, 22

GP6
 MCP23017_GPIO_t, 22

GP7
 MCP23017_GPIO_t, 23

GPINT0
 MCP23017_GPINTEN_t, 19

GPINT1
 MCP23017_GPINTEN_t, 19

GPINT2
 MCP23017_GPINTEN_t, 20

GPINT3
 MCP23017_GPINTEN_t, 20

GPINT4
 MCP23017_GPINTEN_t, 20

GPINT5
 MCP23017_GPINTEN_t, 20

GPINT6
 MCP23017_GPINTEN_t, 20

GPINT7
 MCP23017_GPINTEN_t, 20

GPIO_evt_queue
 io_control.c, 124

gpio_init_interrupts
 io_control.c, 121

GPIO_interrupt_attendance_task
 io_control.c, 121
 io_control.h, 133

gpio_isr_handler
 io_control.c, 122

h
 bitmap_t, 13

height
 sh1106_t, 42

hex2int
 wifi.c, 235

HFIN_LINE_INDX
 display.c, 65

HINI_LINE_INDX

display.c, 65
HTML_FORM
 form.c, 232
 form.h, 234

I2C_DISPLAY
 sh1106_i2c.c, 108
I2C_DISPLAY_MASTER_NUM
 display.c, 65
I2C_IO_MASTER_NUM
 MCP23017.c, 137
i2c_is_hardware_free
 MCP23017.c, 139
I2C_MASTER_FREQ_HZ
 display.c, 65
 MCP23017.c, 137
i2c_master_init
 display.c, 68
 MCP23017.c, 139
I2C_MASTER_RX_BUF_DISABLE
 display.c, 65
 MCP23017.c, 138
I2C_MASTER_SCL_IO
 display.c, 65
 MCP23017.c, 138
I2C_MASTER_SDA_IO
 display.c, 66
 MCP23017.c, 138
I2C_MASTER_TIMEOUT_MS
 MCP23017.c, 138
I2C_MASTER_TX_BUF_DISABLE
 display.c, 66
 MCP23017.c, 138
i2c_release_hardware
 MCP23017.c, 139
IBUS_LINE_INDX
 display.c, 66
ibus_max
 security_settings_t, 42
 system_status_t, 45
ICP0
 MCP23017_INTCAP_t, 26
ICP1
 MCP23017_INTCAP_t, 26
ICP2
 MCP23017_INTCAP_t, 26
ICP3
 MCP23017_INTCAP_t, 26
ICP4
 MCP23017_INTCAP_t, 26
ICP5
 MCP23017_INTCAP_t, 27
ICP6
 MCP23017_INTCAP_t, 27
ICP7
 MCP23017_INTCAP_t, 27
init_seq
 display.c, 72
initRTC

rtc.c, 196
 rtc.h, 202
input_variable
 frequency_settings_t, 16
inputs_status
 system_status_t, 45
INT0
 MCP23017_INTF_t, 30
INT1
 MCP23017_INTF_t, 30
INT2
 MCP23017_INTF_t, 30
INT3
 MCP23017_INTF_t, 30
INT4
 MCP23017_INTF_t, 30
INT5
 MCP23017_INTF_t, 31
INT6
 MCP23017_INTF_t, 31
INT7
 MCP23017_INTF_t, 31
INT_A_PIN
 io_control.c, 115
INT_B_PIN
 io_control.c, 115
INTPOL
 MCP23017_IOCON_t, 32
IO0
 MCP23017_IODIR_t, 34
IO1
 MCP23017_IODIR_t, 34
IO2
 MCP23017_IODIR_t, 34
IO3
 MCP23017_IODIR_t, 34
IO4
 MCP23017_IODIR_t, 34
IO5
 MCP23017_IODIR_t, 35
IO6
 MCP23017_IODIR_t, 35
IO7
 MCP23017_IODIR_t, 35
io_control.c
 buzzer_evt_queue, 124
 BUZZER_PIN, 114
 BuzzerEventPost, 120
 DUTY_MAX_PCT, 114
 DUTY_MIN_PCT, 114
 freq_0_10V_output, 121
 FREQ_SEL_1_PIN, 114
 FREQ_SEL_2_PIN, 115
 FREQ_SEL_3_PIN, 115
 FREQ_SEL_MASK, 115
 GPIO_evt_queue, 124
 gpio_init_interrupts, 121
 GPIO_interrupt_attendance_task, 121

gpio_isr_handler, 122
INT_A_PIN, 115
INT_B_PIN, 115
MATRIX_SW1, 115
MATRIX_SW2, 116
MATRIX_SW3, 116
MATRIX_SW4, 116
MATRIX_SW5, 116
MATRIX_SW6, 116
MATRIX_SW7, 116
MATRIX_SW8, 117
MCP23017_buzzer_control, 122
MCP23017_keyboard_control, 122
MCP23017_relay_control, 122
mcp_porta, 124
mcp_portb, 125
PWM_CHANNEL, 117
PWM_FREQ_HZ, 117
PWM_GPIO, 117
PWM_MODE, 117
PWM_RES, 117
PWM_STEP_MS, 118
PWM_TIMER, 118
relay_evt_queue, 125
RELAY_PIN, 118
RelayEventPost, 123
set_freq_output, 124
START_BUTTON_PIN, 118
STOP_BUTTON_PIN, 118
STOP_PIN, 118
STOP_SW_MASK, 119
SWITCH_BUTTON_BACK, 119
SWITCH_BUTTON_DOWN, 119
SWITCH_BUTTON_LEFT, 119
SWITCH_BUTTON_MENU, 119
SWITCH_BUTTON_OK, 119
SWITCH_BUTTON_RIGHT, 120
SWITCH_BUTTON_SAVE, 120
SWITCH_BUTTON_UP, 120
TAG, 125
TERMO_SW_MASK, 120
TERMO_SW_PIN, 120
io_control.h
 BuzzerEventPost, 133
 GPIO_interrupt_attendance_task, 133
 RelayEventPost, 133
 set_freq_output, 135
IOC0
 MCP23017_INTCON_t, 28
IOC1
 MCP23017_INTCON_t, 28
IOC2
 MCP23017_INTCON_t, 28
IOC3
 MCP23017_INTCON_t, 28
IOC4
 MCP23017_INTCON_t, 28
IOC5
 MCP23017_INTCON_t, 29
 MCP23017_INTCON_t, 29
IOC6
 MCP23017_INTCON_t, 29
IOC7
 MCP23017_INTCON_t, 29
IP0
 MCP23017_IPOL_t, 36
IP1
 MCP23017_IPOL_t, 36
IP2
 MCP23017_IPOL_t, 36
IP3
 MCP23017_IPOL_t, 36
IP4
 MCP23017_IPOL_t, 36
IP5
 MCP23017_IPOL_t, 37
IP6
 MCP23017_IPOL_t, 37
IP7
 MCP23017_IPOL_t, 37
line_bmp
 sh1106_graphics.c, 98
LINE_INCREMENT
 LVFV_system.h, 168
Lista de tareas pendientes, 1
load_16
 nvs.c, 183
load_acceleration
 nvs.c, 179
load_desacceleration
 nvs.c, 179
load_frequency
 nvs.c, 179
load_hour_fin
 nvs.c, 179
load_hour_ini
 nvs.c, 180
load_ibus_max
 nvs.c, 180
load_input_variable
 nvs.c, 180
load_min_fin
 nvs.c, 180
load_min_ini
 nvs.c, 180
load_variables
 nvs.c, 184
 nvs.h, 192
load_vbus_min
 nvs.c, 181
logo16_utn_bmp
 sh1106_graphics.c, 98
LVFV_system.h
 BUTTON_BACK, 173
 BUTTON_DOWN, 173
 BUTTON_LEFT, 173
 BUTTON_MENU, 173

BUTTON_OK, 173
 BUTTON_RIGHT, 173
 BUTTON_SAVE, 173
 BUTTON_UP, 173
 EMERGENCI_STOP_PRESSED, 173
 EMERGENCI_STOP_RELEASED, 173
 fifth, 171
 first, 171
 fourth, 171
 frequency_settings_SH1106_t, 170
 frequency_settings_t, 170
 LINE_INCREMENT, 168
 security_settings_SH1106_t, 170
 security_settings_t, 170
 second, 171
 SECURITY_EXCEEDED, 173
 SECURITY_OK, 173
 seventh, 171
 SH1106_SIZE_1, 168
 SH1106_SIZE_2, 169
 sh1106_variable_lines_e, 170, 171
 sixth, 171
 SPEED_SELECTOR_0, 173
 SPEED_SELECTOR_1, 173
 SPEED_SELECTOR_2, 173
 SPEED_SELECTOR_3, 173
 SPEED_SELECTOR_4, 173
 SPEED_SELECTOR_5, 173
 SPEED_SELECTOR_6, 173
 SPEED_SELECTOR_7, 173
 SPEED_SELECTOR_8, 173
 SPEED_SELECTOR_9, 173
 START_PRESSED, 173
 START_RELEASED, 173
 STOP_PRESSED, 173
 STOP_RELEASED, 173
 SYSTEM_ACCLE_DESACCEL, 171
 SYSTEM_BREAKING, 171
 SYSTEM_EMERGENCY, 171
 SYSTEM_EMERGENCY_OK, 171
 SYSTEM_EMERGENCY_SENT, 171
 SYSTEM_IDLE, 171
 SYSTEM_REGIME, 171
 system_status_e, 170, 171
 system_status_t, 170
 systemSignal_e, 171
 TERMO_SW_PRESSED, 173
 TERMO_SW_RELEASED, 173
 third, 171
 time_settings_SH1106_t, 170
 time_settings_t, 171
 VARIABLE_EIGTH, 169
 VARIABLE_FIFTH, 169
 VARIABLE_FIRST, 169
 VARIABLE_FOURTH, 169
 VARIABLE_SECOND, 169
 VARIABLE_SEVENTH, 169
 VARIABLE_SIXTH, 169
 VARIABLE_THIRD, 170

main.c
 app_main, 176
 TAG, 177

MATRIX_SW1
 io_control.c, 115

MATRIX_SW2
 io_control.c, 116

MATRIX_SW3
 io_control.c, 116

MATRIX_SW4
 io_control.c, 116

MATRIX_SW5
 io_control.c, 116

MATRIX_SW6
 io_control.c, 116

MATRIX_SW7
 io_control.c, 116

MATRIX_SW8
 io_control.c, 117

MCP23017.c
 I2C_IO_MASTER_NUM, 137
 i2c_is_hardware_free, 139
 I2C_MASTER_FREQ_HZ, 137
 i2c_master_init, 139
 I2C_MASTER_RX_BUF_DISABLE, 138
 I2C_MASTER_SCL_IO, 138
 I2C_MASTER_SDA_IO, 138
 I2C_MASTER_TIMEOUT_MS, 138
 I2C_MASTER_TX_BUF_DISABLE, 138
 i2c_release_hardware, 139
 MCP23017_INIT, 139
 mcp_get_on_interrupt_input, 140
 mcp_interrupt_flag, 140
 mcp_read_port, 142
 mcp_register_read, 142
 mcp_register_write, 143
 mcp_write_output_pin, 143
 mcp_write_output_port, 144
 TAG, 144
 xl2C_Mutex, 144

MCP23017.h
 MCP23017_INIT, 150
 mcp_get_on_interrupt_input, 150
 mcp_interrupt_flag, 150
 mcp_read_port, 151
 mcp_write_output_pin, 151
 mcp_write_output_port, 152

MCP23017_buzzer_control
 io_control.c, 122

mcp23017_defs.h
 __MCP23017_ADDRESS__, 155
 __MCP23017_BANK_SEQUENTIAL__, 155
 __MCP23017_BANK_SPLIT__, 155
 __MCP23017_DISSLW_DISABLED__, 155
 __MCP23017_DISSLW_ENABLED__, 155
 __MCP23017_FUNC_MODE__, 155
 __MCP23017_GPINTEN_DISABLE__, 156

__MCP23017_GPINTEN_ENABLE__, 156
__MCP23017_GPPU_PULL_UP_DISABLE__, 156
__MCP23017_GPPU_PULL_UP_ENABLE__, 156
__MCP23017_INTCON_CHANGE__, 156
__MCP23017_INTCON_DEFVAL__, 156
__MCP23017_INTERRUPT_DISABLE__, 157
__MCP23017_INTERRUPT_ENABLE__, 157
__MCP23017_INTPOL_POLARITY_HIGH__, 157
__MCP23017_INTPOL_POLARITY_LOW__, 157
__MCP23017_IODIR_INPUT__, 157
__MCP23017_IODIR_OUTPUT__, 157
__MCP23017_MIRROR_CONNECTED__, 158
__MCP23017_MIRROR_UNCONNECTED__, 158
__MCP23017_ODR_ACTIVE_DRIVER_OUTPUT__, 158
__MCP23017_ODR_OPEN_DRAIN_OUTPUT__, 158
__MCP23017_OPPOSITE_LOGIC__, 158
__MCP23017_POSITIVE_LOGIC__, 158
__MCP23017_READ_OPCODE__, 159
__MCP23017_READ__, 159
__MCP23017_SEQOP_DISABLED__, 159
__MCP23017_SEQOP_ENABLED__, 159
__MCP23017_WRITE_OPCODE__, 159
__MCP23017_WRITE__, 159
MCP23017_DEFVALA_REGISTER, 160
MCP23017_DEFVALB_REGISTER, 160
MCP23017_GPINTENA_REGISTER, 160
MCP23017_GPINTENB_REGISTER, 160
MCP23017_GPIOA_REGISTER, 160
MCP23017_GPIOB_REGISTER, 160
MCP23017_GPPUA_REGISTER, 161
MCP23017_GPPUB_REGISTER, 161
MCP23017_INTCAPA_REGISTER, 161
MCP23017_INTCAPB_REGISTER, 161
MCP23017_INTCONA_REGISTER, 161
MCP23017_INTCONB_REGISTER, 161
MCP23017_INTFA_REGISTER, 162
MCP23017_INTFB_REGISTER, 162
MCP23017_IOCON_REGISTER, 162
MCP23017_IODIRA_REGISTER, 162
MCP23017_IODIRB_REGISTER, 162
MCP23017_IPOLA_REGISTER, 162
MCP23017_IPOLB_REGISTER, 163
MCP23017_OLATA_REGISTER, 163
MCP23017_OLATB_REGISTER, 163
mcp_port_e, 163
PORTA, 163
PORTB, 163
MCP23017_DEFVAL_t, 16
all, 17
bits, 17
DEF0, 17
DEF1, 17
DEF2, 17
DEF3, 18
DEF4, 18
DEF5, 18
DEF6, 18
DEF7, 18
MCP23017_DEFVALA_REGISTER
mcp23017_defs.h, 160
MCP23017_DEFVALB_REGISTER
mcp23017_defs.h, 160
MCP23017_GPINTEN_t, 19
all, 19
bits, 19
GPINT0, 19
GPINT1, 19
GPINT2, 20
GPINT3, 20
GPINT4, 20
GPINT5, 20
GPINT6, 20
GPINT7, 20
MCP23017_GPINTENA_REGISTER
mcp23017_defs.h, 160
MCP23017_GPINTENB_REGISTER
mcp23017_defs.h, 160
MCP23017_GPIO_t, 21
all, 21
bits, 21
GP0, 21
GP1, 22
GP2, 22
GP3, 22
GP4, 22
GP5, 22
GP6, 22
GP7, 23
MCP23017_GPIOA_REGISTER
mcp23017_defs.h, 160
MCP23017_GPIOB_REGISTER
mcp23017_defs.h, 160
MCP23017_GPPU_t, 23
all, 23
bits, 23
PU0, 24
PU1, 24
PU2, 24
PU3, 24
PU4, 24
PU5, 24
PU6, 25
PU7, 25
MCP23017_GPPUA_REGISTER
mcp23017_defs.h, 161
MCP23017_GPPUB_REGISTER
mcp23017_defs.h, 161
MCP23017_INIT
MCP23017.c, 139
MCP23017.h, 150
MCP23017_INTCAP_t, 25
all, 26
bits, 26

ICP0, 26
 ICP1, 26
 ICP2, 26
 ICP3, 26
 ICP4, 26
 ICP5, 27
 ICP6, 27
 ICP7, 27
 MCP23017_INTCAPA_REGISTER
 mcp23017_defs.h, 161
 MCP23017_INTCAPB_REGISTER
 mcp23017_defs.h, 161
 MCP23017_INTCON_t, 27
 all, 28
 bits, 28
 IOC0, 28
 IOC1, 28
 IOC2, 28
 IOC3, 28
 IOC4, 28
 IOC5, 29
 IOC6, 29
 IOC7, 29
 MCP23017_INTCONA_REGISTER
 mcp23017_defs.h, 161
 MCP23017_INTCONB_REGISTER
 mcp23017_defs.h, 161
 MCP23017_INTF_t, 29
 all, 30
 bits, 30
 INT0, 30
 INT1, 30
 INT2, 30
 INT3, 30
 INT4, 30
 INT5, 31
 INT6, 31
 INT7, 31
 MCP23017_INTFA_REGISTER
 mcp23017_defs.h, 162
 MCP23017_INTFB_REGISTER
 mcp23017_defs.h, 162
 MCP23017_IOCON_REGISTER
 mcp23017_defs.h, 162
 MCP23017_IOCON_t, 31
 all, 32
 BANK, 32
 bits, 32
 DISSLW, 32
 INTPOL, 32
 MIRROR, 32
 ODR, 32
 reserved, 33
 reserved_2, 33
 SEQOP, 33
 MCP23017_IODIR_t, 33
 all, 34
 bits, 34
 IO0, 34
 IO1, 34
 IO2, 34
 IO3, 34
 IO4, 34
 IO5, 35
 IO6, 35
 IO7, 35
 MCP23017_IODIRA_REGISTER
 mcp23017_defs.h, 162
 MCP23017_IODIRB_REGISTER
 mcp23017_defs.h, 162
 MCP23017_IPOL_t, 35
 all, 36
 bits, 36
 IP0, 36
 IP1, 36
 IP2, 36
 IP3, 36
 IP4, 36
 IP5, 37
 IP6, 37
 IP7, 37
 MCP23017_IPOLA_REGISTER
 mcp23017_defs.h, 162
 MCP23017_IPOLB_REGISTER
 mcp23017_defs.h, 163
 MCP23017_keyboard_control
 io_control.c, 122
 MCP23017_OLAT_t, 37
 all, 38
 bits, 38
 OL0, 38
 OL1, 38
 OL2, 38
 OL3, 38
 OL4, 38
 OL5, 39
 OL6, 39
 OL7, 39
 MCP23017_OLATA_REGISTER
 mcp23017_defs.h, 163
 MCP23017_OLATB_REGISTER
 mcp23017_defs.h, 163
 MCP23017_relay_control
 io_control.c, 122
 mcp_get_on_interrupt_input
 MCP23017.c, 140
 MCP23017.h, 150
 mcp_interrupt_flag
 MCP23017.c, 140
 MCP23017.h, 150
 mcp_port_e
 mcp23017_defs.h, 163
 mcp_porta
 io_control.c, 124
 mcp_portb
 io_control.c, 125

mcp_read_port
 MCP23017.c, 142
 MCP23017.h, 151

mcp_register_read
 MCP23017.c, 142

mcp_register_write
 MCP23017.c, 143

mcp_write_output_pin
 MCP23017.c, 143
 MCP23017.h, 151

mcp_write_output_port
 MCP23017.c, 144
 MCP23017.h, 152

MIRROR
 MCP23017_IOCON_t, 32

multiplier
 frequency_settings_SH1106_t, 15
 seccurity_settings_SH1106_t, 41
 time_settings_SH1106_t, 46

nvs.c
 load_16, 183
 load_acceleration, 179
 load_desacceleration, 179
 load_frequency, 179
 load_hour_fin, 179
 load_hour_ini, 180
 load_ibus_max, 180
 load_input_variable, 180
 load_min_fin, 180
 load_min_ini, 180
 load_variables, 184
 load_vbus_min, 181
 nvs_init_once, 185
 save_16, 185
 save_acceleration, 181
 save_desacceleration, 181
 save_frequency, 181
 save_hour_fin, 181
 save_hour_ini, 182
 save_ibus_max, 182
 save_input_variable, 182
 save_min_fin, 182
 save_min_ini, 182
 save_variables, 186
 save_vbus_min, 183
 TAG, 187

nvs.h
 load_variables, 192
 nvs_init_once, 192
 save_variables, 193

nvs_init_once
 nvs.c, 185
 nvs.h, 192

ODR
 MCP23017_IOCON_t, 32

OL0
 MCP23017_OLAT_t, 38

OL1
 MCP23017_OLAT_t, 38

OL2
 MCP23017_OLAT_t, 38

OL3
 MCP23017_OLAT_t, 38

OL4
 MCP23017_OLAT_t, 38

OL5
 MCP23017_OLAT_t, 39

OL6
 MCP23017_OLAT_t, 39

OL7
 MCP23017_OLAT_t, 39

oled
 display.c, 72

PIN_NUM_CLK
 sysControl.c, 220

PIN_NUM_CS
 sysControl.c, 220

PIN_NUM_MISO
 sysControl.c, 220

PIN_NUM_MOSI
 sysControl.c, 220

PORTA
 mcp23017_defs.h, 163

PORTB
 mcp23017_defs.h, 163

program_alarm
 rtc.c, 196

PU0
 MCP23017_GPPU_t, 24

PU1
 MCP23017_GPPU_t, 24

PU2
 MCP23017_GPPU_t, 24

PU3
 MCP23017_GPPU_t, 24

PU4
 MCP23017_GPPU_t, 24

PU5
 MCP23017_GPPU_t, 24

PU6
 MCP23017_GPPU_t, 25

PU7
 MCP23017_GPPU_t, 25

PWM_CHANNEL
 io_control.c, 117

PWM_FREQ_HZ
 io_control.c, 117

PWM_GPIO
 io_control.c, 117

PWM_MODE
 io_control.c, 117

PWM_RES
 io_control.c, 117

PWM_STEP_MS
 io_control.c, 118

PWM_TIMER
 io_control.c, 118

readADC
 adc.c, 53
 adc.h, 60

Referencia del directorio C:/Users/User/LVFV_ESP32/main,
 10
 s_adc

Referencia del directorio C:/Users/User/LVFV_ESP32/main/adc,adc.c, 56
 9
 save_16

Referencia del directorio C:/Users/User/LVFV_ESP32/main/display,display.c, 185
 9
 save_acceleration

Referencia del directorio C:/Users/User/LVFV_ESP32/main/io_control,181
 10
 save_desacceleration

Referencia del directorio C:/Users/User/LVFV_ESP32/main/nvs,nvs.c, 181
 10
 save_frequency

Referencia del directorio C:/Users/User/LVFV_ESP32/main/rtc, nvs.c, 181
 11
 save_hour_fin

Referencia del directorio C:/Users/User/LVFV_ESP32/main/systems,c, 181
 11
 save_hour_ini

Referencia del directorio C:/Users/User/LVFV_ESP32/main/wifi,nvs.c, 182
 11
 save_ibus_max

relay_evt_queue
 io_control.c, 125

RELAY_PIN
 io_control.c, 118

RelayEventPost
 io_control.c, 123
 io_control.h, 133

request
 spi_cmd_item_t, 43

reserved
 MCP23017_IOCON_t, 33

reserved_2
 MCP23017_IOCON_t, 33

root_get_handler
 wifi.c, 236

rotation
 sh1106_t, 42

rtc.c
 alarm_settings, 198
 alarm_start_cb, 195
 alarm_stop_cb, 195
 getTime, 196
 initRTC, 196
 program_alarm, 196
 rtc_alarms, 198
 rtc_schedule_alarms, 197
 setTime, 197
 TAG, 198
 time_start, 198
 time_stop, 198

rtc.h
 getTime, 202
 initRTC, 202
 rtc_schedule_alarms, 202
 setTime, 203

rtc_alarms
 rtc.c, 198

rtc_alarms_t, 39
 start_timer, 40
 stop_timer, 40

rtc_schedule_alarms
 rtc.c, 197
 rtc.h, 202

s_adc

save_16

save_acceleration

save_desacceleration

save_frequency

save_hour_fin

save_hour_ini

save_ibus_max

save_input_variable
 nvs.c, 182

save_min_fin
 nvs.c, 182

save_min_ini
 nvs.c, 182

save_post_handler
 wifi.c, 236

save_variables
 nvs.c, 186
 nvs.h, 193

save_vbus_min
 nvs.c, 183

screen_displayed
 display.c, 72

SCREEN_FREQUENCY_EDIT
 display.c, 67

SCREEN_MAIN
 display.c, 66

SCREEN_SECURITY_EDIT
 display.c, 66

SCREEN_SELECT_VARIABLE
 display.c, 66

screen_selected_e
 display.c, 66

SCREEN_TIME_EDIT
 display.c, 66

seccurity_settings
 seccurity_settings_SH1106_t, 41

seccurity_settings_SH1106_t, 40
 edit, 40
 edit_flag, 40
 edit_variable, 41
 LVFV_system.h, 170
 multiplier, 41
 seccurity_settings, 41

security_settings_t, 41
ibus_max, 42
LVFV_system.h, 170
vbus_min, 42
second
 LVFV_system.h, 171
SECURITY_EXCEEDED
 LVFV_system.h, 173
SECURITY_OK
 LVFV_system.h, 173
SEQOP
 MCP23017_IOCON_t, 33
set_freq_output
 io_control.c, 124
 io_control.h, 135
set_frequency_table
 sysAdmin.c, 207
 sysAdmin.h, 216
set_system_settings
 sysAdmin.c, 209
 sysAdmin.h, 217
setTime
 rtc.c, 197
 rtc.h, 203
setValue
 spi_cmd_item_t, 44
seventh
 LVFV_system.h, 171
SH1106_ADDR
 sh1106_i2c.c, 109
sh1106_clear_buffer
 display.c, 68
SH1106_COMM_CMD
 sh1106_i2c.h, 111
SH1106_COMM_DATA
 sh1106_i2c.h, 111
sh1106_comm_type_t
 sh1106_i2c.h, 110
sh1106_draw_arrow
 sh1106_graphics.c, 90
 sh1106_graphics.h, 105
sh1106_draw_bitmap
 sh1106_graphics.c, 90
sh1106_draw_fail
 sh1106_graphics.c, 90
 sh1106_graphics.h, 106
sh1106_draw_line
 sh1106_graphics.c, 91
 sh1106_graphics.h, 106
sh1106_draw_text
 sh1106_graphics.c, 91
 sh1106_graphics.h, 106
sh1106_draw_utn_logo
 sh1106_graphics.c, 91
 sh1106_graphics.h, 107
sh1106_frequency_edit_variables
 display.c, 68
sh1106_graphics.c
arrow_bmp, 92
bitmap_t, 90
fail_bmp, 92
font5x7_close_excl, 92
font5x7_close_quest, 92
font5x7_comma, 92
font5x7_dot, 92
font5x7_double_dot, 92
font5x7_minus, 93
font5x7_rowmajor, 93
font5x7_rowminor, 93
font5x7_rownumber, 94
font5x7_space, 95
font8x14_close_excl, 95
font8x14_close_quest, 95
font8x14_comma, 95
font8x14_dot, 95
font8x14_double_dot, 95
font8x14_minus, 96
font8x14_rowmajor, 96
font8x14_rowminor, 96
font8x14_rownumber, 97
font8x14_space, 98
line_bmp, 98
logo16_utn.bmp, 98
sh1106_draw_arrow, 90
sh1106_draw_bitmap, 90
sh1106_draw_fail, 90
sh1106_draw_line, 91
sh1106_draw_text, 91
sh1106_draw_utn_logo, 91
slash, 98
sh1106_graphics.h
 sh1106_draw_arrow, 105
 sh1106_draw_fail, 106
 sh1106_draw_line, 106
 sh1106_draw_text, 106
 sh1106_draw_utn_logo, 107
 sh1106_t, 105
sh1106_i2c.c
 CMD_CONTROL, 108
 DATA_CONTROL, 108
 I2C_DISPLAY, 108
 SH1106_ADDR, 109
 sh1106_write, 109
sh1106_i2c.h
 SH1106_COMM_CMD, 111
 SH1106_COMM_DATA, 111
 sh1106_comm_type_t, 110
 sh1106_write, 111
sh1106_init
 display.c, 68
 display.h, 86
sh1106_main_screen
 display.c, 68
sh1106_print_emergency
 display.c, 69
sh1106_print_frame

display.c, 69
sh1106_refresh
 display.c, 69
sh1106_security_edit_variables
 display.c, 69
sh1106_select_edit_variables
 display.c, 70
SH1106_SIZE_1
 LVFV_system.h, 168
SH1106_SIZE_2
 LVFV_system.h, 169
sh1106_splash_screen
 display.c, 70
sh1106_t, 42
 buffer, 42
 height, 42
 rotation, 42
 sh1106_graphics.h, 105
 width, 43
sh1106_time_edit_variables
 display.c, 70
sh1106_variable_lines_e
 LVFV_system.h, 170, 171
sh1106_write
 sh1106_i2c.c, 109
 sh1106_i2c.h, 111
sixth
 LVFV_system.h, 171
slash
 sh1106_graphics.c, 98
SPEED_SELECTOR_0
 LVFV_system.h, 173
SPEED_SELECTOR_1
 LVFV_system.h, 173
SPEED_SELECTOR_2
 LVFV_system.h, 173
SPEED_SELECTOR_3
 LVFV_system.h, 173
SPEED_SELECTOR_4
 LVFV_system.h, 173
SPEED_SELECTOR_5
 LVFV_system.h, 173
SPEED_SELECTOR_6
 LVFV_system.h, 173
SPEED_SELECTOR_7
 LVFV_system.h, 173
SPEED_SELECTOR_8
 LVFV_system.h, 173
SPEED_SELECTOR_9
 LVFV_system.h, 173
SPI_CLOCK_HZ
 sysControl.c, 220
spi_cmd_item_t, 43
 getValue, 43
 request, 43
 setValue, 44
SPI_communication
 sysControl.c, 221
 sysControl.h, 229
spi_handle
 sysControl.c, 222
SPI_HOST_USED
 sysControl.c, 220
SPI_Init
 sysControl.c, 221
 sysControl.h, 229
SPI_QUEUE_TX_DEPTH
 sysControl.c, 220
SPI_Request
 sysControl.h, 228
SPI_REQUEST_EMERGENCY
 sysControl.h, 229
SPI_REQUEST_GET_ACCEL
 sysControl.h, 228
SPI_REQUEST_GET_DESACCEL
 sysControl.h, 229
SPI_REQUEST_GET_DIR
 sysControl.h, 229
SPI_REQUEST_GET_FREC
 sysControl.h, 228
SPI_REQUEST_IS_STOP
 sysControl.h, 229
SPI_REQUEST_RESPONSE
 sysControl.h, 229
SPI_REQUEST_SET_ACCEL
 sysControl.h, 228
SPI_REQUEST_SET_DESACCEL
 sysControl.h, 228
SPI_REQUEST_SET_DIR
 sysControl.h, 228
SPI_REQUEST_SET_FREC
 sysControl.h, 228
SPI_REQUEST_START
 sysControl.h, 228
SPI_REQUEST_STOP
 sysControl.h, 228
SPI_Response
 sysControl.h, 229
SPI_RESPONSE_ERR
 sysControl.h, 229
SPI_RESPONSE_ERR_CMD_UNKNOWN
 sysControl.h, 229
SPI_RESPONSE_ERR_DATA_INVALID
 sysControl.h, 229
SPI_RESPONSE_ERR_DATA_MISSING
 sysControl.h, 229
SPI_RESPONSE_ERR_DATA_OUT_RANGE
 sysControl.h, 229
SPI_RESPONSE_ERR_MOVING
 sysControl.h, 229
SPI_RESPONSE_ERR_NO_COMMAND
 sysControl.h, 229
SPI_RESPONSE_ERR_NOT_MOVING
 sysControl.h, 229
SPI_RESPONSE_OK
 sysControl.h, 229

SPI_SendRequest
 sysControl.c, 221

START_BUTTON_PIN
 io_control.c, 118

START_PRESSED
 LVFV_system.h, 173

START_RELEASED
 LVFV_system.h, 173

start_timer
 rtc_alarms_t, 40

start_webserver
 wifi.c, 236
 wifi.h, 241

status
 system_status_t, 45

STOP_BUTTON_PIN
 io_control.c, 118

STOP_PIN
 io_control.c, 118

STOP_PRESSED
 LVFV_system.h, 173

STOP_RELEASED
 LVFV_system.h, 173

STOP_SW_MASK
 io_control.c, 119

stop_timer
 rtc_alarms_t, 40

SWITCH_BUTTON_BACK
 io_control.c, 119

SWITCH_BUTTON_DOWN
 io_control.c, 119

SWITCH_BUTTON_LEFT
 io_control.c, 119

SWITCH_BUTTON_MENU
 io_control.c, 119

SWITCH_BUTTON_OK
 io_control.c, 119

SWITCH_BUTTON_RIGHT
 io_control.c, 120

SWITCH_BUTTON_SAVE
 io_control.c, 120

SWITCH_BUTTON_UP
 io_control.c, 120

sysAdmin.c
 accelerating, 205
 accelerating_handle, 210
 change_frequency, 205
 desaccelerating, 206
 desaccelerating_handle, 210
 engine_emergency_stop, 206
 engine_emergency_stop_release, 206
 engine_start, 206
 engine_stop, 207
 frequency_table, 210
 get_status, 207
 set_frequency_table, 207
 set_system_settings, 209
 system_security_settings, 210

system_status, 210

TAG, 210

update_meas, 209

sysAdmin.h
 change_frequency, 214
 engine_emergency_stop, 215
 engine_emergency_stop_release, 215
 engine_start, 215
 engine_stop, 215
 get_status, 216
 set_frequency_table, 216
 set_system_settings, 217
 update_meas, 217

sysControl.c
 PIN_NUM_CLK, 220
 PIN_NUM_CS, 220
 PIN_NUM_MISO, 220
 PIN_NUM_MOSI, 220
 SPI_CLOCK_HZ, 220
 SPI_communication, 221
 spi_handle, 222
 SPI_HOST_USED, 220
 SPI_Init, 221
 SPI_QUEUE_TX_DEPTH, 220
 SPI_SendRequest, 221
 system_event_queue, 222
 SystemEventPost, 222
 TAG, 223

sysControl.h
 SPI_communication, 229
 SPI_Init, 229
 SPI_Request, 228
 SPI_REQUEST_EMERGENCY, 229
 SPI_REQUEST_GET_ACCEL, 228
 SPI_REQUEST_GET_DESACCEL, 229
 SPI_REQUEST_GET_DIR, 229
 SPI_REQUEST_GET_FREQ, 228
 SPI_REQUEST_IS_STOP, 229
 SPI_REQUEST_RESPONSE, 229
 SPI_REQUEST_SET_ACCEL, 228
 SPI_REQUEST_SET_DESACCEL, 228
 SPI_REQUEST_SET_DIR, 228
 SPI_REQUEST_SET_FREQ, 228
 SPI_REQUEST_START, 228
 SPI_REQUEST_STOP, 228
 SPI_Response, 229
 SPI_RESPONSE_ERR, 229
 SPI_RESPONSE_ERR_CMD_UNKNOWN, 229
 SPI_RESPONSE_ERR_DATA_INVALID, 229
 SPI_RESPONSE_ERR_DATA_MISSING, 229
 SPI_RESPONSE_ERR_DATA_OUT_RANGE, 229
 SPI_RESPONSE_ERR_MOVING, 229
 SPI_RESPONSE_ERR_NO_COMMAND, 229
 SPI_RESPONSE_ERR_NOT_MOVING, 229
 SPI_RESPONSE_OK, 229
 SystemEventPost, 230

SYSTEM_ACCEL_DESACCEL
 LVFV_system.h, 171

SYSTEM_BREAKING
 LVFV_system.h, 171

SYSTEM_EMERGENCY
 LVFV_system.h, 171

SYSTEM_EMERGENCY_OK
 LVFV_system.h, 171

SYSTEM_EMERGENCY_SENT
 LVFV_system.h, 171

system_event_queue
 sysControl.c, 222

system_frequency_settings
 display.c, 72

SYSTEM_IDLE
 LVFV_system.h, 171

SYSTEM_REGIME
 LVFV_system.h, 171

system_security_settings
 display.c, 73

 sysAdmin.c, 210

system_status
 sysAdmin.c, 210

system_status_e
 LVFV_system.h, 170, 171

system_status_t, 44
 acceleration, 44

 desacceleration, 44

 frequency, 45

 frequency_destiny, 45

 ibus_max, 45

 inputs_status, 45

 LVFV_system.h, 170

 status, 45

 vbus_min, 45

system_time_settings
 display.c, 73

system_variables_save
 display.c, 70

 display.h, 87

SystemEventPost
 sysControl.c, 222

 sysControl.h, 230

systemSignal_e
 LVFV_system.h, 171

TAG
 adc.c, 56

 display.c, 73

 io_control.c, 125

 main.c, 177

 MCP23017.c, 144

 nvs.c, 187

 rtc.c, 198

 sysAdmin.c, 210

 sysControl.c, 223

 wifi.c, 237

task_display
 display.c, 71

 display.h, 87

TERMO_SW_MASK
 io_control.c, 120

TERMO_SW_PIN
 io_control.c, 120

TERMO_SW_PRESSED
 LVFV_system.h, 173

TERMO_SW_RELEASED
 LVFV_system.h, 173

third
 LVFV_system.h, 171

time_settings
 time_settings_SH1106_t, 46

time_settings_SH1106_t, 46
 edit, 46

 edit_flag, 46

 edit_variable, 46

 LVFV_system.h, 170

 multiplier, 46

 time_settings, 46

time_settings_t, 47
 LVFV_system.h, 171

 time_start, 47

 time_stop, 47

 time_system, 47

time_start
 rtc.c, 198

 time_settings_t, 47

time_stop
 rtc.c, 198

 time_settings_t, 47

time_system
 time_settings_t, 47

update_meas
 sysAdmin.c, 209

 sysAdmin.h, 217

url_decode
 wifi.c, 236

VARIABLE_EIGTH
 LVFV_system.h, 169

VARIABLE_FIFTH
 LVFV_system.h, 169

VARIABLE_FIRST
 LVFV_system.h, 169

VARIABLE_FOURTH
 LVFV_system.h, 169

VARIABLE_SECOND
 LVFV_system.h, 169

VARIABLE_SEVENTH
 LVFV_system.h, 169

VARIABLE_SIXTH
 LVFV_system.h, 169

VARIABLE_THIRD
 LVFV_system.h, 170

VBUS_LINE_INDX
 display.c, 66

vbus_min
 security_settings_t, 42

 system_status_t, 45

w

 bitmap_t, [14](#)

width

 sh1106_t, [43](#)

wifi.c

 get_param_value, [235](#)

 hex2int, [235](#)

 root_get_handler, [236](#)

 save_post_handler, [236](#)

 start_webserver, [236](#)

 TAG, [237](#)

 url_decode, [236](#)

 wifi_init_softap, [237](#)

wifi.h

 start_webserver, [241](#)

 wifi_init_softap, [241](#)

wifi_init_softap

 wifi.c, [237](#)

 wifi.h, [241](#)

x

 bitmap_t, [14](#)

xI2C_Mutex

 MCP23017.c, [144](#)

y

 bitmap_t, [14](#)