

# Low Voltage Variable Frequency Drive

## 2.1

Generado por Doxygen 1.15.0



# Capítulo 1

## Topic Index

### 1.1. Topics

Here is a list of all topics with brief descriptions:

Pines del inversor (puerto A) . . . . .	??
Pines de señalización (puerto B) . . . . .	??
CMSIS . . . . .	??
Stm32f1xx_system . . . . .	??
STM32F1xx_System_Private_Includes . . . . .	??
STM32F1xx_System_Private_TypesDefinitions . . . . .	??
STM32F1xx_System_Private_Defines . . . . .	??
STM32F1xx_System_Private_Macros . . . . .	??
STM32F1xx_System_Private_Variables . . . . .	??
STM32F1xx_System_Private_FunctionPrototypes . . . . .	??
STM32F1xx_System_Private_Functions . . . . .	??



## Capítulo 2

# Jerarquía de directorios

### 2.1. Directorios

adc . . . . .	??
adc.c . . . . .	??
adc.h . . . . .	??
display . . . . .	??
display.c . . . . .	??
display.h . . . . .	??
sh1106_graphics.c . . . . .	??
sh1106_graphics.h . . . . .	??
sh1106_i2c.c . . . . .	??
sh1106_i2c.h . . . . .	??
Gestor_Estados . . . . .	??
GestorEstados.c . . . . .	??
GestorEstados.h . . . . .	??
Gestor_SVM . . . . .	??
GestorSVM.c . . . . .	??
GestorSVM.h . . . . .	??
Gestor_Timers . . . . .	??
GestorTimers.c . . . . .	??
GestorTimers.h . . . . .	??
Inc . . . . .	??
main.h . . . . .	??
stm32f1xx_hal_conf.h . . . . .	??
stm32f1xx_it.h . . . . .	??
io_control . . . . .	??
io_control.c . . . . .	??
io_control.h . . . . .	??
MCP23017.c . . . . .	??
MCP23017.h . . . . .	??
mcp23017_defs.h . . . . .	??
LVFV_ESP32 . . . . .	??
main . . . . .	??
adc . . . . .	??
adc.c . . . . .	??
adc.h . . . . .	??
display . . . . .	??
display.c . . . . .	??

display.h . . . . .	??
sh1106_graphics.c . . . . .	??
sh1106_graphics.h . . . . .	??
sh1106_i2c.c . . . . .	??
sh1106_i2c.h . . . . .	??
io_control . . . . .	??
io_control.c . . . . .	??
io_control.h . . . . .	??
MCP23017.c . . . . .	??
MCP23017.h . . . . .	??
mcp23017_defs.h . . . . .	??
nvs . . . . .	??
nvs.c . . . . .	??
nvs.h . . . . .	??
rtc . . . . .	??
rtc.c . . . . .	??
rtc.h . . . . .	??
system . . . . .	??
sysAdmin.c . . . . .	??
sysAdmin.h . . . . .	??
sysControl.c . . . . .	??
sysControl.h . . . . .	??
wifi . . . . .	??
form.c . . . . .	??
form.h . . . . .	??
wifi.c . . . . .	??
wifi.h . . . . .	??
LVFV_system.h . . . . .	??
main.c . . . . .	??
pytest_hello_world.py . . . . .	??
main . . . . .	??
adc . . . . .	??
adc.c . . . . .	??
adc.h . . . . .	??
display . . . . .	??
display.c . . . . .	??
display.h . . . . .	??
sh1106_graphics.c . . . . .	??
sh1106_graphics.h . . . . .	??
sh1106_i2c.c . . . . .	??
sh1106_i2c.h . . . . .	??
io_control . . . . .	??
io_control.c . . . . .	??
io_control.h . . . . .	??
MCP23017.c . . . . .	??
MCP23017.h . . . . .	??
mcp23017_defs.h . . . . .	??
nvs . . . . .	??
nvs.c . . . . .	??
nvs.h . . . . .	??
rtc . . . . .	??
rtc.c . . . . .	??
rtc.h . . . . .	??
system . . . . .	??
sysAdmin.c . . . . .	??
sysAdmin.h . . . . .	??
sysControl.c . . . . .	??

sysControl.h	??
wifi	??
form.c	??
form.h	??
wifi.c	??
wifi.h	??
LVFV_system.h	??
main.c	??
Modules	??
Gestor_Estados	??
GestorEstados.c	??
GestorEstados.h	??
Gestor_SVM	??
GestorSVM.c	??
GestorSVM.h	??
Gestor_Timers	??
GestorTimers.c	??
GestorTimers.h	??
SPI_Interfase	??
SPIModule.c	??
SPIModule.h	??
nvs	??
nvs.c	??
nvs.h	??
rtc	??
rtc.c	??
rtc.h	??
Software ESP32	??
SPI_ESP32_TestModule	??
main.c	??
SPI_Module.c	??
SPI_Module.h	??
UART_Module.c	??
UART_Module.h	??
SPI_ESP32_TestModule	??
main.c	??
SPI_Module.c	??
SPI_Module.h	??
UART_Module.c	??
UART_Module.h	??
SPI_Interfase	??
SPIModule.c	??
SPIModule.h	??
Src	??
main.c	??
stm32f1xx_hal_msp.c	??
stm32f1xx_it.c	??
syscalls.c	??
sysmem.c	??
system_stm32f1xx.c	??
SVM Space Vector Modulation	??
SVM_Calculos.py	??
SVM_DiagramaPolar.py	??
svm_DiagramaSalidaTemporal.py	??
svm_interactivo.py	??

SVM_SwitchAnimacion.py	??
SVM_SwitchTime.py	??
svm_v2.py	??
system	??
sysAdmin.c	??
sysAdmin.h	??
sysControl.c	??
sysControl.h	??
wifi	??
form.c	??
form.h	??
wifi.c	??
wifi.h	??

# Capítulo 3

# Índice de espacios de nombres

### **3.1. Lista de espacios de nombres**

Lista de los espacios de nombres documentados, con breves descripciones:

```
pytest_hello_world . . . . . ??  
SVM_Calculos . . . . . ??  
SVM_DiagramaPolar . . . . . ??  
svm_DiagramaSalidaTemporal . . . . . ??  
svm_interactivo . . . . . ??  
SVM_SwitchAnimacion . . . . . ??  
SVM_SwitchTime . . . . . ??  
svm_v2 . . . . . ??
```



## Capítulo 4

# Índice de estructuras de datos

### 4.1. Estructuras de datos

Lista de estructuras con breves descripciones:

<a href="#">bitmap_t</a>	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 . . . . .	??
<a href="#">ConfiguracionSVM</a>	Parámetros de configuración del módulo SVM . . . . .	??
<a href="#">DatoCalculado</a>		??
<a href="#">frequency_settings_SH1106_t</a>		??
<a href="#">frequency_settings_t</a>		??
<a href="#">MCP23017_DEFVAL_t</a>		??
<a href="#">MCP23017_GPINTEN_t</a>		??
<a href="#">MCP23017_GPIO_t</a>		??
<a href="#">MCP23017_GPPU_t</a>		??
<a href="#">MCP23017_INTCAP_t</a>		??
<a href="#">MCP23017_INTCON_t</a>		??
<a href="#">MCP23017_INTF_t</a>		??
<a href="#">MCP23017_IOCON_t</a>		??
<a href="#">MCP23017_IODIR_t</a>		??
<a href="#">MCP23017_IPOL_t</a>		??
<a href="#">MCP23017_OLAT_t</a>		??
<a href="#">Parametros</a>	Parámetros “sombra” para sincronización atómica con el lazo de cálculo . . . . .	??
<a href="#">rtc_alarms_t</a>		??
<a href="#">seccurity_settings_SH1106_t</a>		??
<a href="#">seccurity_settings_t</a>		??
<a href="#">sh1106_t</a>		??
<a href="#">spi_cmd_item_t</a>	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 . . . . .	??
<a href="#">system_status_t</a>	Estructura con las variables necesarias para establecer las condiciones de trabajo del motor . . . . .	??
<a href="#">time_settings_SH1106_t</a>		??
<a href="#">time_settings_t</a>		??
<a href="#">ValoresSwitching</a>	Estructura de trabajo del ISR del timer de switching . . . . .	??



# Capítulo 5

## Índice de archivos

### 5.1. Lista de archivos

Lista de todos los archivos con breves descripciones:

Inc/main.h	Header de la aplicación principal . . . . .	??
Inc/stm32f1xx_hal_conf.h	HAL configuration file . . . . .	??
Inc/stm32f1xx_it.h	This file contains the headers of the interrupt handlers . . . . .	??
Modules/Gestor_Estados/GestorEstados.c	Implementa la máquina de estados del LVFV . . . . .	??
Modules/Gestor_Estados/GestorEstados.h		??
Modules/Gestor_SVM/GestorSVM.c		??
Modules/Gestor_SVM/GestorSVM.h		??
Modules/Gestor_Timers/GestorTimers.c		??
Modules/Gestor_Timers/GestorTimers.h		??
Modules/SPI_Interfase/SPIModule.c		??
Modules/SPI_Interfase/SPIModule.h	Interfaz del módulo SPI (esclavo) con DMA . . . . .	??
Src/main.c	Punto de entrada del firmware y configuración de periféricos . . . . .	??
Src/stm32f1xx_hal_msp.c	This file provides code for the MSP Initialization and de-Initialization codes . . . . .	??
Src/stm32f1xx_it.c	Interrupt Service Routines . . . . .	??
Src/syscalls.c	STM32CubeIDE Minimal System calls file . . . . .	??
Src/sysmem.c	STM32CubeIDE System Memory calls file . . . . .	??
Src/system_stm32f1xx.c	CMSIS Cortex-M3 Device Peripheral Access Layer System Source File . . . . .	??
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV_ESP32/pytest_hello_world.py	??	
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV_ESP32/main/LVFV_system.h	Declaraciones de estructuras, variables, definiciones y estados generales del sistema utilizados en varios de los archivos . . . . .	??
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/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 . . . . .	??

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV_ESP32/main/adc/ <a href="#">adc.c</a>	??
Funcion de inicialización y tarea lectura del ADC . . . . .	??
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV_ESP32/main/adc/ <a href="#">adc.h</a>	??
Declaración de funcion de inicialización y tarea lectura del ADC . . . . .	??
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV_ESP32/main/display/ <a href="#">display.c</a>	??
Funciones de consulta de los seteos hechos por el usuario, tarea que controla el display y posteо de eventos para controlar el display . . . . .	??
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV_ESP32/main/display/ <a href="#">display.h</a>	??
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 . . . . .	??
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV_ESP32/main/display/ <a href="#">sh1106_graphics.c</a>	??
Funciones que permiten operar sobre el display . . . . .	??
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV_ESP32/main/display/ <a href="#">sh1106_graphics.h</a>	??
Declaración de funciones que permiten operar sobre el display . . . . .	??
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV_ESP32/main/display/ <a href="#">sh1106_i2c.c</a>	??
Funciones de hardware para el puerto I2C . . . . .	??
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV_ESP32/main/display/ <a href="#">sh1106_i2c.h</a>	??
Declaración de funciones de hardware para el puerto I2C . . . . .	??
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV_ESP32/main/io_↔control/ <a href="#">io_control.c</a>	??
Funciones de control del relay, buzzer y tarea del uso del expansor de entradas y salidas y entradas y salidas aisladas directas al ESP32 . . . . .	??
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV_ESP32/main/io_↔control/ <a href="#">io_control.h</a>	??
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 . . . . .	??
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV_ESP32/main/io_↔control/ <a href="#">MCP23017.c</a>	??
Declaración de funciones de inicialización, lectura y escritura a través del puerto I2C el dispositivo MCP23017 . . . . .	??
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV_ESP32/main/io_↔control/ <a href="#">MCP23017.h</a>	??
Declaración de funciones de inicialización, lectura y escritura a través del puerto I2C el dispositivo MCP23017 . . . . .	??
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV_ESP32/main/io_↔control/ <a href="#">mcp23017_defs.h</a>	??
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV_ESP32/main/nvs/ <a href="#">nvs.c</a>	??
Funciones de lectura y escritura de variables no volátils. Carga en sistema además las variables de seguridad al momento de ser guardadas en la memoria . . . . .	??
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV_ESP32/main/nvs/ <a href="#">nvs.h</a>	??
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 . . . . .	??
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV_ESP32/main/rtc/ <a href="#">rtc.c</a>	??
Funciones de lectura y escritura del RTC y alarmas . . . . .	??
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV_ESP32/main/rtc/ <a href="#">rtc.h</a>	??
Header con prototipos de funciones básicas para el lectura y escritura del RTC y alarmas . . . . .	??
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV_ESP32/main/system/ <a href="#">sysAdmin.c</a>	??
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 . . . . .	??
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV_ESP32/main/system/ <a href="#">sysAdmin.h</a>	??
Header con las funciones de funcionamiento del sistema . . . . .	??
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV_ESP32/main/system/ <a href="#">sysControl.c</a>	??
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 . . . . .	??

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/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 . . . . . ??  
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/wifi/[form.c](#)  
??  
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/wifi/[form.h](#)  
??  
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/wifi/[wifi.c](#)  
??  
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/wifi/[wifi.h](#)  
??  
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/Software ESP32/SPI\_↔  
ESP32\_TestModule/[main.c](#) . . . . . ??  
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/Software ESP32/SPI\_↔  
ESP32\_TestModule/[SPI\\_Module.c](#) . . . . . ??  
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/Software ESP32/SPI\_↔  
ESP32\_TestModule/[SPI\\_Module.h](#) . . . . . ??  
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/Software ESP32/SPI\_↔  
ESP32\_TestModule/[UART\\_Module.c](#) . . . . . ??  
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/Software ESP32/SPI\_↔  
ESP32\_TestModule/[UART\\_Module.h](#) . . . . . ??  
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/SVM Space Vector  
Modulation/[SVM\\_Calculos.py](#) . . . . . ??  
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/SVM Space Vector  
Modulation/[SVM\\_DiagramaPolar.py](#) . . . . . ??  
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/SVM Space Vector  
Modulation/[svm\\_DiagramaSalidaTemporal.py](#) . . . . . ??  
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/SVM Space Vector  
Modulation/[svm\\_interactivo.py](#) . . . . . ??  
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/SVM Space Vector  
Modulation/[SVM\\_SwitchAnimacion.py](#) . . . . . ??  
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/SVM Space Vector  
Modulation/[SVM\\_SwitchTime.py](#) . . . . . ??  
C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/SVM Space Vector  
Modulation/[svm\\_v2.py](#) . . . . . ??



# Capítulo 6

## Topic Documentation

### 6.1. Pines del inversor (puerto A)

Pines de control del módulo SVM en GPIOA.

#### defines

- `#define GPIO_U_IN GPIO_PIN_1`  
*Entrada lógica de la pierna U (GPIOA, PIN\_1).*
- `#define GPIO_U_SD GPIO_PIN_2`  
*Shutdown/Habilitación driver pierna U (GPIOA, PIN\_2).*
- `#define GPIO_V_IN GPIO_PIN_3`  
*Entrada lógica de la pierna V (GPIOA, PIN\_3).*
- `#define GPIO_V_SD GPIO_PIN_4`  
*Shutdown/Habilitación driver pierna V (GPIOA, PIN\_4).*
- `#define GPIO_W_IN GPIO_PIN_5`  
*Entrada lógica de la pierna W (GPIOA, PIN\_5).*
- `#define GPIO_W_SD GPIO_PIN_6`  
*Shutdown/Habilitación driver pierna W (GPIOA, PIN\_6).*
- `#define GPIO_TERMO_SWITCH GPIO_PIN_11`  
*Entrada digital: Termo-switch (GPIOA, PIN\_11).*
- `#define GPIO_STOP_BUTTON GPIO_PIN_12`  
*Entrada digital: Botón de parada (GPIOA, PIN\_12).*

#### 6.1.1. Descripción detallada

Pines de control del módulo SVM en GPIOA.

#### 6.1.2. Documentación de «define»

##### 6.1.2.1. GPIO\_STOP\_BUTTON

```
#define GPIO_STOP_BUTTON GPIO_PIN_12
```

Entrada digital: Botón de parada (GPIOA, PIN\_12).

Definición en la línea 34 del archivo `main.h`.

### 6.1.2.2. GPIO\_TERMO\_SWITCH

```
#define GPIO_TERMO_SWITCH GPIO_PIN_11
```

Entrada digital: Termo–switch (GPIOA, PIN\_11).

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

### 6.1.2.3. GPIO\_U\_IN

```
#define GPIO_U_IN GPIO_PIN_1
```

Entrada lógica de la pierna U (GPIOA, PIN\_1).

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

### 6.1.2.4. GPIO\_U\_SD

```
#define GPIO_U_SD GPIO_PIN_2
```

Shutdown/Habilitación driver pierna U (GPIOA, PIN\_2).

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

### 6.1.2.5. GPIO\_V\_IN

```
#define GPIO_V_IN GPIO_PIN_3
```

Entrada lógica de la pierna V (GPIOA, PIN\_3).

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

### 6.1.2.6. GPIO\_V\_SD

```
#define GPIO_V_SD GPIO_PIN_4
```

Shutdown/Habilitación driver pierna V (GPIOA, PIN\_4).

Definición en la línea [25](#) del archivo [main.h](#).

### 6.1.2.7. GPIO\_W\_IN

```
#define GPIO_W_IN GPIO_PIN_5
```

Entrada lógica de la pierna W (GPIOA, PIN\_5).

Definición en la línea [27](#) del archivo [main.h](#).

### 6.1.2.8. GPIO\_W\_SD

```
#define GPIO_W_SD GPIO_PIN_6
```

Shutdown/Habilitación driver pierna W (GPIOA, PIN\_6).

Definición en la línea 29 del archivo [main.h](#).

## 6.2. Pines de señalización (puerto B)

LEDs de estado en GPIOB.

### defines

- `#define GPIO_LED_STATE GPIO_PIN_0`  
*LED de estado (GPIOB, PIN\_0).*
- `#define GPIO_LED_ERROR GPIO_PIN_2`  
*LED de error (GPIOB, PIN\_2).*

### 6.2.1. Descripción detallada

LEDs de estado en GPIOB.

### 6.2.2. Documentación de «define»

#### 6.2.2.1. GPIO\_LED\_ERROR

```
#define GPIO_LED_ERROR GPIO_PIN_2
```

LED de error (GPIOB, PIN\_2).

Definición en la línea 45 del archivo [main.h](#).

#### 6.2.2.2. GPIO\_LED\_STATE

```
#define GPIO_LED_STATE GPIO_PIN_0
```

LED de estado (GPIOB, PIN\_0).

Definición en la línea 43 del archivo [main.h](#).

## 6.3. CMSIS

### Topics

- `Stm32f1xx_system . . . . . ??`

### 6.3.1. Descripción detallada

### 6.3.2. Stm32f1xx\_system

#### Topics

- STM32F1xx\_System\_Private\_Includes . . . . . ??
- STM32F1xx\_System\_Private\_TypesDefinitions . . . . . ??
- STM32F1xx\_System\_Private\_Defines . . . . . ??
- STM32F1xx\_System\_Private\_Macros . . . . . ??
- STM32F1xx\_System\_Private\_Variables . . . . . ??
- STM32F1xx\_System\_Private\_FunctionPrototypes . . . . . ??
- STM32F1xx\_System\_Private\_Functions . . . . . ??

#### 6.3.2.1. Descripción detallada

#### 6.3.2.2. STM32F1xx\_System\_Private\_Includes

#### 6.3.2.3. STM32F1xx\_System\_Private\_TypesDefinitions

#### 6.3.2.4. STM32F1xx\_System\_Private\_Defines

##### defines

- #define HSE\_VALUE 8000000U
- #define HSI\_VALUE 8000000U

##### 6.3.2.4.1. Descripción detallada

##### 6.3.2.4.2. Documentación de «define»

###### 6.3.2.4.2.1. HSE\_VALUE

```
#define HSE_VALUE 8000000U
```

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

Definición en la línea 77 del archivo [system\\_stm32f1xx.c](#).

### 6.3.2.4.2.2. HSI\_VALUE

```
#define HSI_VALUE 8000000U
```

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

Definición en la línea 82 del archivo [system\\_stm32f1xx.c](#).

### 6.3.2.5. STM32F1xx\_System\_Private\_Macros

### 6.3.2.6. STM32F1xx\_System\_Private\_Variables

#### Variables

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

#### 6.3.2.6.1. Descripción detallada

#### 6.3.2.6.2. Documentación de variables

##### 6.3.2.6.2.1. AHBPrescTable

```
const uint8_t AHBPrescTable[16U] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
```

Definición en la línea 142 del archivo [system\\_stm32f1xx.c](#).

##### 6.3.2.6.2.2. APBPrescTable

```
const uint8_t APBPrescTable[8U] = {0, 0, 0, 0, 1, 2, 3, 4}
```

Definición en la línea 143 del archivo [system\\_stm32f1xx.c](#).

##### 6.3.2.6.2.3. SystemCoreClock

```
uint32_t SystemCoreClock = 8000000
```

Definición en la línea 141 del archivo [system\\_stm32f1xx.c](#).

### 6.3.2.7. STM32F1xx\_System\_Private\_FunctionPrototypes

### 6.3.2.8. STM32F1xx\_System\_Private\_Functions

#### Funciones

- `void SystemInit (void)`  
*Setup the microcontroller system Initialize the Embedded Flash Interface, the PLL and update the SystemCoreClock variable.*
- `void SystemCoreClockUpdate (void)`  
*Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.*

### 6.3.2.8.1. Descripción detallada

### 6.3.2.8.2. Documentación de funciones

#### 6.3.2.8.2.1. SystemCoreClockUpdate()

```
void SystemCoreClockUpdate (
    void )
```

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

##### Nota

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

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

- If SYSCLK source is HSI, SystemCoreClock will contain the [HSI\\_VALUE\(\\*\)](#)
- If SYSCLK source is HSE, SystemCoreClock will contain the [HSE\\_VALUE\(\\*\\*\)](#)
- If SYSCLK source is PLL, SystemCoreClock will contain the [HSE\\_VALUE\(\\*\\*\)](#) or [HSI\\_VALUE\(\\*\)](#) multiplied by the PLL factors.

(\*) HSI\_VALUE is a constant defined in stm32f1xx.h file (default value 8 MHz) but the real value may vary depending on the variations in voltage and temperature.

(\*\*) HSE\_VALUE is a constant defined in stm32f1xx.h file (default value 8 MHz or 25 MHz, depending on the product used), user has to ensure that HSE\_VALUE is same as the real frequency of the crystal used. Otherwise, this function may have wrong result.

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

##### Parámetros

<a href="#">None</a>	<input type="button" value=""/>
----------------------	---------------------------------

##### Valores devueltos

<a href="#">None</a>	<input type="button" value=""/>
----------------------	---------------------------------

Definición en la línea [224](#) del archivo [system\\_stm32f1xx.c](#).

#### 6.3.2.8.2.2. SystemInit()

```
void SystemInit (
    void )
```

Setup the microcontroller system Initialize the Embedded Flash Interface, the PLL and update the SystemCoreClock variable.

##### Nota

This function should be used only after reset.

##### Parámetros

<i>None</i>	
-------------	--

**Valores devueltos**

<i>None</i>	
-------------	--

Definición en la línea 175 del archivo [system\\_stm32f1xx.c](#).



## Capítulo 7

# Documentación de directorios

### 7.1. Referencia del directorio C:/Users/User/Desktop/ProyectoFinal- Grupo5-VariadorDeFrecuencia/Software/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.*

### 7.2. Referencia del directorio C:/Users/User/Desktop/ProyectoFinal- Grupo5-VariadorDeFrecuencia/Software/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.*

## 7.3. Referencia del directorio Modules/Gestor\_Estados

### Archivos

- archivo [GestorEstados.c](#)  
*Implementa la máquina de estados del LVFV.*
- archivo [GestorEstados.h](#)

## 7.4. Referencia del directorio Modules/Gestor\_SVM

### Archivos

- archivo [GestorSVM.c](#)
- archivo [GestorSVM.h](#)

## 7.5. Referencia del directorio Modules/Gestor\_Timers

### Archivos

- archivo [GestorTimers.c](#)
- archivo [GestorTimers.h](#)

## 7.6. Referencia del directorio Inc

### Archivos

- archivo [main.h](#)  
*Header de la aplicación principal.*
- archivo [stm32f1xx\\_hal\\_conf.h](#)  
*HAL configuration file.*
- archivo [stm32f1xx\\_it.h](#)  
*This file contains the headers of the interrupt handlers.*

## 7.7. Referencia del directorio

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDe←  
Frecuencia/Software/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](#)

7.8. Referencia del directorio C:/Users/User/Desktop/ProyectoFinal-  
Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32

## Directorios

- #### ■ directorio main

## Archivos

- archivo `pytest_hello_world.py`

### **7.8.1. Descripción detallada**

## 7.8.2. Hello World Example

Starts a FreeRTOS task to print "Hello World".

(See the [README.md](#) file in the upper level 'examples' directory for more information about examples.)

#### **7.8.2.1. How to use example**

Follow detailed instructions provided specifically for this example.

Select the instructions depending on Espressif chip installed on your development board:

- [ESP32 Getting Started Guide](#)
  - [ESP32-S2 Getting Started Guide](#)

### 7.8.2.2. Example folder contents

The project **hello\_world** contains one source file in C language **hello\_world\_main.c**. The file is located in folder [main](main).

ESP-IDF projects are built using CMake. The project build configuration is contained in `CMakeLists.txt` files that provide set of directives and instructions describing the project's source files and targets (executable, library, or both).

Below is short explanation of remaining files in the project folder.

```
CMakeLists.txt          Python script used for automated testing
pytest_hello_world.py
main
    CMakeLists.txt
    hello_world_main.c
README.md              This is the file you are currently reading
```

For more information on structure and contents of ESP-IDF projects, please refer to Section [Build System](#) of the ESP-IDF Programming Guide.

### 7.8.2.3. Troubleshooting

- Program upload failure
  - Hardware connection is not correct: run `idf.py -p PORT monitor`, and reboot your board to see if there are any output logs.
  - The baud rate for downloading is too high: lower your baud rate in the `menuconfig` menu, and try again.

### 7.8.2.4. Technical support and feedback

Please use the following feedback channels:

- For technical queries, go to the [esp32.com](#) forum
- For a feature request or bug report, create a [GitHub issue](#)

We will get back to you as soon as possible.

## 7.9. Referencia del directorio C:/Users/User/Desktop/ProyectoFinal- Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main

### Directories

- 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.*

## 7.10. Referencia del directorio Modules

### Directories

- directorio [Gestor\\_Estados](#)
- directorio [Gestor\\_SVM](#)
- directorio [Gestor\\_Timers](#)
- directorio [SPI\\_Interfase](#)

## 7.11. Referencia del directorio C:/Users/User/Desktop/ProyectoFinal- Grupo5-VariadorDeFrecuencia/Software/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 volatil. Útiles para los reset y no tener que reconfigurar siempre las variables básicas del sistema.*

## 7.12. Referencia del directorio C:/Users/User/Desktop/ProyectoFinal- Grupo5-VariadorDeFrecuencia/Software/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.*

## 7.13. Referencia del directorio C:/Users/User/Desktop/ProyectoFinal- Grupo5-VariadorDeFrecuencia/Software/Software ESP32

### Directorios

- directorio [SPI\\_ESP32\\_TestModule](#)

## 7.14. Referencia del directorio C:/Users/User/Desktop/ProyectoFinal- Grupo5-VariadorDeFrecuencia/Software/Software ESP32/SPI\_ESP32\_TestModule

### Archivos

- archivo [main.c](#)

- archivo [SPI\\_Module.c](#)

- archivo [SPI\\_Module.h](#)

- archivo [UART\\_Module.c](#)

- archivo [UART\\_Module.h](#)

## 7.15. Referencia del directorio Modules/SPI\_Interfase

### Archivos

- archivo SPIModule.c
- archivo SPIModule.h

*Interfaz del módulo SPI (esclavo) con DMA.*

## 7.16. Referencia del directorio Src

### Archivos

- archivo main.c

*Punto de entrada del firmware y configuración de periféricos.*

- archivo stm32f1xx\_hal\_msp.c

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

- archivo stm32f1xx\_it.c

*Interrupt Service Routines.*

- archivo syscalls.c

*STM32CubeIDE Minimal System calls file.*

- archivo sysmem.c

*STM32CubeIDE System Memory calls file.*

- archivo system\_stm32f1xx.c

*CMSIS Cortex-M3 Device Peripheral Access Layer System Source File.*

## 7.17. Referencia del directorio C:/Users/User/Desktop/ProyectoFinal- Grupo5-VariadorDeFrecuencia/Software/SVM Space Vector Modulation

### Archivos

- archivo SVM\_Calculos.py
- archivo SVM\_DiagramaPolar.py
- archivo svm\_DiagramaSalidaTemporal.py
- archivo svm\_interactivo.py
- archivo SVM\_SwitchAnimacion.py
- archivo SVM\_SwitchTime.py
- archivo svm\_v2.py

## 7.18 Referencia del directorio

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/system29

## 7.18. Referencia del directorio C:/Users/User/Desktop/ProyectoFinal- Grupo5-VariadorDeFrecuencia/Software/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.*

## 7.19. Referencia del directorio C:/Users/User/Desktop/ProyectoFinal- Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/wifi

### Archivos

- archivo [form.c](#)
- archivo [form.h](#)
- archivo [wifi.c](#)
- archivo [wifi.h](#)



## Capítulo 8

# Documentación de espacios de nombres

## 8.1. Referencia del espacio de nombres `pytest_hello_world`

### Funciones

- None `test_hello_world` (IdfDut dut, Callable[..., None] log\_minimum\_free\_heap\_size)
- None `test_hello_world_linux` (IdfDut dut)
- None `test_hello_world_macos` (IdfDut dut)
- None `verify_elf_sha256_embedding` (QemuApp app, str sha256\_reported)
- None `test_hello_world_host` (QemuApp app, QemuDut dut)

### 8.1.1. Documentación de funciones

#### 8.1.1.1. `test_hello_world()`

```
None test_hello_world (
    IdfDut dut,
    Callable[..., None] log_minimum_free_heap_size)
```

Definición en la línea 16 del archivo [pytest\\_hello\\_world.py](#).

#### 8.1.1.2. `test_hello_world_host()`

```
None test_hello_world_host (
    QemuApp app,
    QemuDut dut)
```

Definición en la línea 51 del archivo [pytest\\_hello\\_world.py](#).

#### 8.1.1.3. `test_hello_world_linux()`

```
None test_hello_world_linux (
    IdfDut dut)
```

Definición en la línea 23 del archivo [pytest\\_hello\\_world.py](#).

#### 8.1.1.4. test\_hello\_world\_macos()

```
None test_hello_world_macos (
    IdfDut dut)
```

Definición en la línea 30 del archivo [pytest\\_hello\\_world.py](#).

#### 8.1.1.5. verify\_elf\_sha256\_embedding()

```
None verify_elf_sha256_embedding (
    QemuApp app,
    str sha256_reported)
```

Definición en la línea 34 del archivo [pytest\\_hello\\_world.py](#).

## 8.2. Referencia del espacio de nombres SVM\_Calculos

### Funciones

- [get\\_time\\_vector \(Ts, M, angle\)](#)
- [getSector \(angle\)](#)
- [getSecuencia \(sector\)](#)
- [getSecuenciaLabel \(sector\)](#)

### Variables

- int [SECTOR1](#) = 1
- int [SECTOR2](#) = 2
- int [SECTOR3](#) = 3
- int [SECTOR4](#) = 4
- int [SECTOR5](#) = 5
- int [SECTOR6](#) = 6
- int [Ts](#) = 1
- int [M](#) = 1
- list [vectors](#) = []
- int [current\\_angle](#) = 0

## 8.2.1. Documentación de funciones

### 8.2.1.1. get\_time\_vector()

```
get_time_vector (
    Ts,
    M,
    angle)
```

Definición en la línea 24 del archivo [SVM\\_Calculos.py](#).

### 8.2.1.2. `getSector()`

```
getSector (
    angle)
```

Definición en la línea 58 del archivo [SVM\\_Calculos.py](#).

### 8.2.1.3. `getSecuencia()`

```
getSecuencia (
    sector)
```

Definición en la línea 78 del archivo [SVM\\_Calculos.py](#).

### 8.2.1.4. `getSecuenciaLabel()`

```
getSecuenciaLabel (
    sector)
```

Definición en la línea 92 del archivo [SVM\\_Calculos.py](#).

## 8.2.2. Documentación de variables

### 8.2.2.1. `current_angle`

```
int current_angle = 0
```

Definición en la línea 20 del archivo [SVM\\_Calculos.py](#).

### 8.2.2.2. `M`

```
int M = 1
```

Definición en la línea 16 del archivo [SVM\\_Calculos.py](#).

### 8.2.2.3. `SECTOR1`

```
int SECTOR1 = 1
```

Definición en la línea 6 del archivo [SVM\\_Calculos.py](#).

### 8.2.2.4. `SECTOR2`

```
int SECTOR2 = 2
```

Definición en la línea 7 del archivo [SVM\\_Calculos.py](#).

### 8.2.2.5. SECTOR3

```
int SECTOR3 = 3
```

Definición en la línea 8 del archivo [SVM\\_Calculos.py](#).

### 8.2.2.6. SECTOR4

```
int SECTOR4 = 4
```

Definición en la línea 9 del archivo [SVM\\_Calculos.py](#).

### 8.2.2.7. SECTOR5

```
int SECTOR5 = 5
```

Definición en la línea 10 del archivo [SVM\\_Calculos.py](#).

### 8.2.2.8. SECTOR6

```
int SECTOR6 = 6
```

Definición en la línea 11 del archivo [SVM\\_Calculos.py](#).

### 8.2.2.9. Ts

```
int Ts = 1
```

Definición en la línea 14 del archivo [SVM\\_Calculos.py](#).

### 8.2.2.10. vectors

```
list vectors = [ ]
```

Definición en la línea 19 del archivo [SVM\\_Calculos.py](#).

## 8.3. Referencia del espacio de nombres SVM\_DiagramaPolar

### Funciones

- [update\\_plot \(angle\)](#)
- [slider\\_update \(val\)](#)
- [animate \(\)](#)

## Variables

- list `vectors` = []
- int `current_angle` = 0
- `root` = tk.Tk()
- `polar_ax1`
- `polar_ax2`
- `subplot_kw`
- `figsize`
- `va`
- `canvas` = FigureCanvasTkAgg(fig, master=`root`)
- `canvas_widget` = `canvas.get_tk_widget()`

## 8.3.1. Documentación de funciones

### 8.3.1.1. `animate()`

```
animate ()
```

Definición en la línea 116 del archivo [SVM\\_DiagramaPolar.py](#).

### 8.3.1.2. `slider_update()`

```
slider_update (
    val)
```

Definición en la línea 111 del archivo [SVM\\_DiagramaPolar.py](#).

### 8.3.1.3. `update_plot()`

```
update_plot (
    angle)
```

Definición en la línea 11 del archivo [SVM\\_DiagramaPolar.py](#).

## 8.3.2. Documentación de variables

### 8.3.2.1. `canvas`

```
canvas = FigureCanvasTkAgg(fig, master=root)
```

Definición en la línea 147 del archivo [SVM\\_DiagramaPolar.py](#).

### 8.3.2.2. `canvas_widget`

```
canvas_widget = canvas.get_tk_widget()
```

Definición en la línea 148 del archivo [SVM\\_DiagramaPolar.py](#).

### 8.3.2.3. current\_angle

```
int current_angle = 0
```

Definición en la línea 8 del archivo [SVM\\_DiagramaPolar.py](#).

### 8.3.2.4. figsize

```
figsize
```

Definición en la línea 128 del archivo [SVM\\_DiagramaPolar.py](#).

### 8.3.2.5. polar\_ax1

```
polar_ax1
```

Definición en la línea 128 del archivo [SVM\\_DiagramaPolar.py](#).

### 8.3.2.6. polar\_ax2

```
polar_ax2
```

Definición en la línea 128 del archivo [SVM\\_DiagramaPolar.py](#).

### 8.3.2.7. root

```
root = tk.Tk()
```

Definición en la línea 124 del archivo [SVM\\_DiagramaPolar.py](#).

### 8.3.2.8. subplot\_kw

```
subplot_kw
```

Definición en la línea 128 del archivo [SVM\\_DiagramaPolar.py](#).

### 8.3.2.9. va

```
va
```

Definición en la línea 131 del archivo [SVM\\_DiagramaPolar.py](#).

### 8.3.2.10. vectors

```
list vectors = []
```

Definición en la línea 7 del archivo [SVM\\_DiagramaPolar.py](#).

## 8.4. Referencia del espacio de nombres svm\_DiagramaSalidaTemporal

### Funciones

- `generar_vector_salida_temporal (Ts, M, angSwitch, tRise, tf)`
- `calcular_fase (t, out1, out2, out3, VBus)`
- `calcular_rms (signal)`

### Variables

- `int SECTOR1 = 1`
- `int SECTOR2 = 2`
- `int SECTOR3 = 3`
- `int SECTOR4 = 4`
- `int SECTOR5 = 5`
- `int SECTOR6 = 6`
- `int Q1 = 0`
- `int Q2 = 1`
- `int Q3 = 2`
- `int Ts = 1`
- `int M = 1`
- `dict SECTOR_VECTORS`
- `int angle_deg = 45`
- `t`
- `out1`
- `out2`
- `out3`
- `angSwitch`
- `tRise`
- `tf`
- `t2`
- `faseRS`
- `faseST`
- `rms_RS = calcular_rms(faseRS)`
- `rms_ST = calcular_rms(faseST)`
- `figsize`
- `label`

### 8.4.1. Documentación de funciones

#### 8.4.1.1. calcular\_fase()

```
calcular_fase (
    t,
    out1,
    out2,
    out3,
    VBus)
```

Definición en la línea 154 del archivo `svm_DiagramaSalidaTemporal.py`.

#### 8.4.1.2. calcular\_rms()

```
calcular_rms (
    signal)
```

Definición en la línea 164 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.1.3. generar\_vector\_salida\_temporal()

```
generar_vector_salida_temporal (
    Ts,
    M,
    angSwitch,
    tRise,
    tf)
```

Definición en la línea 36 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

### 8.4.2. Documentación de variables

#### 8.4.2.1. angle\_deg

```
int angle_deg = 45
```

Definición en la línea 170 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.2. angSwitch

```
angSwitch
```

Definición en la línea 172 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.3. faseRS

```
faseRS
```

Definición en la línea 174 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.4. faseST

```
faseST
```

Definición en la línea 174 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.5. `figsize`

```
figsize
```

Definición en la línea 185 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.6. `label`

```
label
```

Definición en la línea 189 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.7. `M`

```
float M = 1
```

Definición en la línea 21 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.8. `out1`

```
out1
```

Definición en la línea 172 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.9. `out2`

```
out2
```

Definición en la línea 172 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.10. `out3`

```
out3
```

Definición en la línea 172 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.11. `Q1`

```
int Q1 = 0
```

Definición en la línea 14 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.12. `Q2`

```
int Q2 = 1
```

Definición en la línea 15 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.13. Q3

```
int Q3 = 2
```

Definición en la línea 16 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.14. rms\_RS

```
rms_RS = calcular_rms(faseRS)
```

Definición en la línea 176 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.15. rms\_ST

```
rms_ST = calcular_rms(faseST)
```

Definición en la línea 177 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.16. SECTOR1

```
int SECTOR1 = 1
```

Definición en la línea 6 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.17. SECTOR2

```
int SECTOR2 = 2
```

Definición en la línea 7 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.18. SECTOR3

```
int SECTOR3 = 3
```

Definición en la línea 8 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.19. SECTOR4

```
int SECTOR4 = 4
```

Definición en la línea 9 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.20. SECTOR5

```
int SECTOR5 = 5
```

Definición en la línea 10 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.21. SECTOR6

```
int SECTOR6 = 6
```

Definición en la línea 11 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.22. SECTOR\_VECTORS

```
dict SECTOR_VECTORS
```

##### Valor inicial:

```
00001 = {  
00002     SECTOR1: [2, 1, 0],  
00003     SECTOR2: [1, 2, 0],  
00004     SECTOR3: [1, 0, 2],  
00005     SECTOR4: [0, 1, 2],  
00006     SECTOR5: [0, 2, 1],  
00007     SECTOR6: [2, 0, 1],  
00008 }
```

Definición en la línea 25 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.23. t

```
t
```

Definición en la línea 172 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.24. t2

```
t2
```

Definición en la línea 174 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.25. tf

```
tf
```

Definición en la línea 172 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.26. tRise

```
tRise
```

Definición en la línea 172 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

#### 8.4.2.27. Ts

```
int Ts = 1
```

Definición en la línea 19 del archivo [svm\\_DiagramaSalidaTemporal.py](#).

## 8.5. Referencia del espacio de nombres svm\_interactivo

### Funciones

- [calculate\\_modulation\\_index \(angle\)](#)
- [update\\_plot \(angle\\_slider\)](#)

### Variables

- [root = tk.Tk\(\)](#)
- [fig](#)
- [ax](#)
- [figsize](#)
- [angles = np.linspace\(0, 360, 100\)](#)
- [list modulation\\_indices = \[calculate\\_modulation\\_index\(a\) for a in angles\]](#)
- [label](#)
- [canvas = FigureCanvasTkAgg\(fig, master=root\)](#)
- [canvas\\_widget = canvas.get\\_tk\\_widget\(\)](#)
- [angle\\_slider = tk.Scale\(root, from\\_=0, to=360, orient="horizontal", label="Ángulo", command=lambda val: update\\_plot\(angle\\_slider\)\)](#)

### 8.5.1. Documentación de funciones

#### 8.5.1.1. calculate\_modulation\_index()

```
calculate_modulation_index (
    angle)
```

Definición en la línea 7 del archivo [svm\\_interactivo.py](#).

#### 8.5.1.2. update\_plot()

```
update_plot (
    angle_slider)
```

Definición en la línea 12 del archivo [svm\\_interactivo.py](#).

### 8.5.2. Documentación de variables

#### 8.5.2.1. angle\_slider

```
angle_slider = tk.Scale(root, from_=0, to=360, orient="horizontal", label="Ángulo", command=lambda val: update_plot(angle_slider))
```

Definición en la línea 45 del archivo [svm\\_interactivo.py](#).

### 8.5.2.2. angles

```
angles = np.linspace(0, 360, 100)
```

Definición en la línea 32 del archivo [svm\\_interactivo.py](#).

### 8.5.2.3. ax

```
ax
```

Definición en la línea 31 del archivo [svm\\_interactivo.py](#).

### 8.5.2.4. canvas

```
canvas = FigureCanvasTkAgg(fig, master=root)
```

Definición en la línea 40 del archivo [svm\\_interactivo.py](#).

### 8.5.2.5. canvas\_widget

```
canvas_widget = canvas.get_tk_widget()
```

Definición en la línea 41 del archivo [svm\\_interactivo.py](#).

### 8.5.2.6. fig

```
fig
```

Definición en la línea 31 del archivo [svm\\_interactivo.py](#).

### 8.5.2.7. figsize

```
figsize
```

Definición en la línea 31 del archivo [svm\\_interactivo.py](#).

### 8.5.2.8. label

```
label
```

Definición en la línea 34 del archivo [svm\\_interactivo.py](#).

### 8.5.2.9. modulation\_indices

```
modulation_indices = [calculate_modulation_index(a) for a in angles]
```

Definición en la línea 33 del archivo [svm\\_interactivo.py](#).

### 8.5.2.10. root

```
root = tk.Tk()
```

Definición en la línea 27 del archivo [svm\\_interactivo.py](#).

## 8.6. Referencia del espacio de nombres SVM\_SwitchAnimacion

### Variables

- `screen_width`
- `screen_height`
- `screen = pygame.display.set_mode((screen_width, screen_height))`
- tuple `rect_color` = (255, 0, 0)
- tuple `background_color` = (128, 128, 128)
- tuple `text_color` = (255, 255, 255)
- tuple `circle_color` = (0, 0, 0)
- tuple `vector_color` = (100, 100, 100)
- tuple `black_color` = (0,0,0)
- tuple `green_color` = (0,255,0)
- tuple `red_color` = (255,0,0)
- tuple `blue_color` = (0,0,255)
- `rect_width`
- `rect_height`
- `rect_x`
- `rect_y`
- int `rect_speed` = 5
- `font_tiempo` = pygame.font.SysFont(None, 36)
- `font_caracteristicas` = pygame.font.SysFont(None, 20)
- int `line_spacing` = 30
- str `image_folder` = "image"
- list `image_files` = [f'E{i}.png' for i in range(8)]
- list `imageSwitch` = []
- `image_path` = os.path.join(image\_folder, file\_name)
- `image` = pygame.image.load(image\_path)
- `clock` = pygame.time.Clock()
- `start_time` = pygame.time.get\_ticks()
- tuple `circle_center` = (screen\_width // 2, screen\_height // 2)
- int `center_x` = screen\_width // 2
- int `center_y` = screen\_height // 2
- int `circle_radius` = 250
- int `vector_length` = 250
- int `num_vectors` = 6
- int `timeScale` = 100000000
- int `rotFreq` = 50
- int `swFreq` = 10000
- int `forward` = 1
- int `angSw` = rotFreq \* 360 / swFreq
- list `text_lines`
- list `comunicacion_lineas` = []
- int `proxTiempoTimer` = 0
- int `ultTiempoTimer` = 0

- int `ang` = 0
- int `sector` = 0
- int `t0` = 0
- int `t1` = 0
- int `t2` = 0
- str `secuenciaLabel` = ""
- int `numImageSwitch` = 0
- tuple `elapsed_time` = (pygame.time.get\_ticks() - `start_time`)
- `microsTime` = int(`elapsed_time/timeScale` \* 1000000)
- str `time_text` = f"Tiempo: {`microsTime`}us"
- `text_surface` = font\_tiempos.render(`time_text`, True, `text_color`)
- tuple `angle` = (360 / `num_vectors`) \* i
- `angle_rad` = math.radians(`angle`)
- tuple `end_x` = `circle_center`[0] + `vector_length` \* math.cos(`angle_rad`)
- tuple `end_y` = `circle_center`[1] + `vector_length` \* math.sin(`angle_rad`)
- int `angMag` = 250
- int `duty0` = 80
- int `duty1` = 60
- int `duty2` = 40
- tuple `tiempoEnCiclo` = (`microsTime` - `ultTiempoTimer`) / 1000000
- `rendered_text` = font\_caracteristicas.render(f"Sector {`sector`}, Secuencia:", True, `text_color`)
- `width`

## 8.6.1. Documentación de variables

### 8.6.1.1. `ang`

```
tuple ang = 0
```

Definición en la línea 97 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.2. `angle`

```
tuple angle = (360 / num_vectors) * i
```

Definición en la línea 127 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.3. `angle_rad`

```
angle_rad = math.radians(angle)
```

Definición en la línea 128 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.4. `angMag`

```
int angMag = 250
```

Definición en la línea 137 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.5. **angSw**

```
int angSw = rotFreq * 360 / swFreq
```

Definición en la línea 80 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.6. **background\_color**

```
tuple background_color = (128, 128, 128)
```

Definición en la línea 20 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.7. **black\_color**

```
tuple black_color = (0, 0, 0)
```

Definición en la línea 25 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.8. **blue\_color**

```
tuple blue_color = (0, 0, 255)
```

Definición en la línea 28 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.9. **center\_x**

```
int center_x = screen_width // 2
```

Definición en la línea 61 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.10. **center\_y**

```
int center_y = screen_height // 2
```

Definición en la línea 62 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.11. **circle\_center**

```
tuple circle_center = (screen_width // 2, screen_height // 2)
```

Definición en la línea 60 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.12. **circle\_color**

```
tuple circle_color = (0, 0, 0)
```

Definición en la línea 22 del archivo [SVM\\_SwitchAnimacion.py](#).

**8.6.1.13. circle\_radius**

```
int circle_radius = 250
```

Definición en la línea 63 del archivo [SVM\\_SwitchAnimacion.py](#).

**8.6.1.14. clock**

```
clock = pygame.time.Clock()
```

Definición en la línea 56 del archivo [SVM\\_SwitchAnimacion.py](#).

**8.6.1.15. conmutacion\_lineas**

```
list conmutacion_lineas = []
```

Definición en la línea 92 del archivo [SVM\\_SwitchAnimacion.py](#).

**8.6.1.16. duty0**

```
int duty0 = 80
```

Definición en la línea 139 del archivo [SVM\\_SwitchAnimacion.py](#).

**8.6.1.17. duty1**

```
int duty1 = 60
```

Definición en la línea 140 del archivo [SVM\\_SwitchAnimacion.py](#).

**8.6.1.18. duty2**

```
int duty2 = 40
```

Definición en la línea 141 del archivo [SVM\\_SwitchAnimacion.py](#).

**8.6.1.19. elapsed\_time**

```
tuple elapsed_time = (pygame.time.get_ticks() - start_time)
```

Definición en la línea 114 del archivo [SVM\\_SwitchAnimacion.py](#).

**8.6.1.20. end\_x**

```
tuple end_x = circle_center[0] + vector_length * math.cos(angle_rad)
```

Definición en la línea 129 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.21. `end_y`

```
tuple end_y = circle_center[1] + vector_length * math.sin(angle_rad)
```

Definición en la línea 130 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.22. `font_caracteristicas`

```
font_caracteristicas = pygame.font.SysFont(None, 20)
```

Definición en la línea 37 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.23. `font_tiempo`

```
font_tiempo = pygame.font.SysFont(None, 36)
```

Definición en la línea 36 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.24. `forward`

```
int forward = 1
```

Definición en la línea 76 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.25. `green_color`

```
tuple green_color = (0, 255, 0)
```

Definición en la línea 26 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.26. `image`

```
image = pygame.image.load(image_path)
```

Definición en la línea 48 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.27. `image_files`

```
list image_files = [f"E{i}.png" for i in range(8)]
```

Definición en la línea 42 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.28. `image_folder`

```
str image_folder = "image"
```

Definición en la línea 41 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.29. **image\_path**

```
image_path = os.path.join(image_folder, file_name)
```

Definición en la línea 46 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.30. **imageSwitch**

```
list imageSwitch = []
```

Definición en la línea 44 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.31. **line\_spacing**

```
int line_spacing = 30
```

Definición en la línea 38 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.32. **microsTime**

```
microsTime = int(elapsed_time/timeScale * 1000000)
```

Definición en la línea 115 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.33. **num\_vectors**

```
int num_vectors = 6
```

Definición en la línea 65 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.34. **numImageSwitch**

```
numImageSwitch = 0
```

Definición en la línea 103 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.35. **proxTiempoTimer**

```
int proxTiempoTimer = 0
```

Definición en la línea 95 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.36. **rect\_color**

```
tuple rect_color = (255, 0, 0)
```

Definición en la línea 19 del archivo [SVM\\_SwitchAnimacion.py](#).

#### 8.6.1.37. rect\_height

```
rect_height
```

Definición en la línea 31 del archivo [SVM\\_SwitchAnimacion.py](#).

#### 8.6.1.38. rect\_speed

```
int rect_speed = 5
```

Definición en la línea 33 del archivo [SVM\\_SwitchAnimacion.py](#).

#### 8.6.1.39. rect\_width

```
rect_width
```

Definición en la línea 31 del archivo [SVM\\_SwitchAnimacion.py](#).

#### 8.6.1.40. rect\_x

```
rect_x
```

Definición en la línea 32 del archivo [SVM\\_SwitchAnimacion.py](#).

#### 8.6.1.41. rect\_y

```
rect_y
```

Definición en la línea 32 del archivo [SVM\\_SwitchAnimacion.py](#).

#### 8.6.1.42. red\_color

```
tuple red_color = (255, 0, 0)
```

Definición en la línea 27 del archivo [SVM\\_SwitchAnimacion.py](#).

#### 8.6.1.43. rendered\_text

```
rendered_text = font_caracteristicas.render(f"Sector {sector}, Secuencia:", True, text_color)
```

Definición en la línea 189 del archivo [SVM\\_SwitchAnimacion.py](#).

#### 8.6.1.44. rotFreq

```
int rotFreq = 50
```

Definición en la línea 74 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.45. screen

```
screen = pygame.display.set_mode((screen_width, screen_height))
```

Definición en la línea 15 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.46. screen\_height

```
screen_height
```

Definición en la línea 14 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.47. screen\_width

```
screen_width
```

Definición en la línea 14 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.48. sector

```
sector = 0
```

Definición en la línea 98 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.49. secuenciaLabel

```
secuenciaLabel = ""
```

Definición en la línea 102 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.50. start\_time

```
start_time = pygame.time.get_ticks()
```

Definición en la línea 57 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.51. swFreq

```
int swFreq = 10000
```

Definición en la línea 75 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.52. t0

```
t0 = 0
```

Definición en la línea 99 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.53. t1

```
t1 = 0
```

Definición en la línea 100 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.54. t2

```
t2 = 0
```

Definición en la línea 101 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.55. text\_color

```
tuple text_color = (255, 255, 255)
```

Definición en la línea 21 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.56. text\_lines

```
list text_lines
```

#### Valor inicial:

```
00001 = [
00002     f"Ang conmutacion: {angSw}",
00003     f"Frecuencia rotor: {rotFreq}",
00004     f"Frecuencia switch: {swFreq}"
00005 ]
```

Definición en la línea 85 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.57. text\_surface

```
text_surface = font_tiempo.render(time_text, True, text_color)
```

Definición en la línea 117 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.58. tiempoEnCiclo

```
tuple tiempoEnCiclo = (microsTime - ultTiempoTimer) / 1000000
```

Definición en la línea 160 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.59. time\_text

```
str time_text = f"Tiempo: {microsTime}us"
```

Definición en la línea 116 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.60. timeScale

```
int timeScale = 100000000
```

Definición en la línea 73 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.61. ultTiempoTimer

```
ultTiempoTimer = 0
```

Definición en la línea 96 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.62. vector\_color

```
tuple vector_color = (100, 100, 100)
```

Definición en la línea 23 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.63. vector\_length

```
int vector_length = 250
```

Definición en la línea 64 del archivo [SVM\\_SwitchAnimacion.py](#).

### 8.6.1.64. width

```
width
```

Definición en la línea 200 del archivo [SVM\\_SwitchAnimacion.py](#).

## 8.7. Referencia del espacio de nombres SVM\_SwitchTime

### Funciones

- [update\\_rectangle \(new\\_x, new\\_y\)](#)

### Variables

- [image](#) = np.random.random((100, 100))
- [fig](#)
- [ax](#)
- [im](#) = ax.imshow([image](#), cmap="gray")
- [rect](#) = patches.Rectangle((30, 30), 20, 20, linewidth=2, edgecolor='red', facecolor='none')
- [random\\_x](#) = random.randint(0, 80)
- [random\\_y](#) = random.randint(0, 80)

## 8.7.1. Documentación de funciones

### 8.7.1.1. update\_rectangle()

```
update_rectangle (
    new_x,
    new_y)
```

Definición en la línea 19 del archivo [SVM\\_SwitchTime.py](#).

## 8.7.2. Documentación de variables

### 8.7.2.1. ax

```
ax
```

Definición en la línea 11 del archivo [SVM\\_SwitchTime.py](#).

### 8.7.2.2. fig

```
fig
```

Definición en la línea 11 del archivo [SVM\\_SwitchTime.py](#).

### 8.7.2.3. im

```
im = ax.imshow(image, cmap="gray")
```

Definición en la línea 12 del archivo [SVM\\_SwitchTime.py](#).

### 8.7.2.4. image

```
image = np.random.random((100, 100))
```

Definición en la línea 8 del archivo [SVM\\_SwitchTime.py](#).

### 8.7.2.5. random\_x

```
random_x = random.randint(0, 80)
```

Definición en la línea 31 del archivo [SVM\\_SwitchTime.py](#).

### 8.7.2.6. random\_y

```
random_y = random.randint(0, 80)
```

Definición en la línea 32 del archivo [SVM\\_SwitchTime.py](#).

### 8.7.2.7. rect

```
rect = patches.Rectangle((30, 30), 20, 20, linewidth=2, edgecolor='red', facecolor='none')
```

Definición en la línea 15 del archivo [SVM\\_SwitchTime.py](#).

## 8.8. Referencia del espacio de nombres svm\_v2

### Funciones

- [calculate\\_modulation\\_index \(angle\)](#)
- [update\\_plot1 \(angle\\_slider\)](#)
- [update\\_plot2 \(angle\\_slider\)](#)
- [update\\_plot \(angle\\_slider\)](#)
- [animate \(\)](#)

### Variables

- [root = tk.Tk\(\)](#)
- [fig](#)
- [polar\\_ax](#)
- [subplot\\_kw](#)
- [figsize](#)
- [va](#)
- [list vectors = \[\]](#)
- [int current\\_angle = 0](#)
- [canvas = FigureCanvasTkAgg\(fig, master=root\)](#)
- [canvas\\_widget = canvas.get\\_tk\\_widget\(\)](#)
- [angle\\_slider = tk.Scale\(root, from\\_=0, to=360, orient="horizontal", label="Ángulo", command=lambda val: update\\_plot\(angle\\_slider\)\)](#)

### 8.8.1. Documentación de funciones

#### 8.8.1.1. animate()

```
animate ()
```

Definición en la línea 79 del archivo [svm\\_v2.py](#).

#### 8.8.1.2. calculate\_modulation\_index()

```
calculate_modulation_index (
    angle)
```

Definición en la línea 7 del archivo [svm\\_v2.py](#).

### 8.8.1.3. update\_plot()

```
update_plot (
    angle_slider)
```

Definición en la línea 52 del archivo [svm\\_v2.py](#).

### 8.8.1.4. update\_plot1()

```
update_plot1 (
    angle_slider)
```

Definición en la línea 12 del archivo [svm\\_v2.py](#).

### 8.8.1.5. update\_plot2()

```
update_plot2 (
    angle_slider)
```

Definición en la línea 34 del archivo [svm\\_v2.py](#).

## 8.8.2. Documentación de variables

### 8.8.2.1. angle\_slider

```
angle_slider = tk.Scale(root, from_=0, to=360, orient="horizontal", label="Ángulo", command=lambda
val: update_plot(angle_slider))
```

Definición en la línea 111 del archivo [svm\\_v2.py](#).

### 8.8.2.2. canvas

```
canvas = FigureCanvasTkAgg(fig, master=root)
```

Definición en la línea 106 del archivo [svm\\_v2.py](#).

### 8.8.2.3. canvas\_widget

```
canvas_widget = canvas.get_tk_widget()
```

Definición en la línea 107 del archivo [svm\\_v2.py](#).

### 8.8.2.4. current\_angle

```
int current_angle = 0
```

Definición en la línea 104 del archivo [svm\\_v2.py](#).

### 8.8.2.5. `fig`

`fig`

Definición en la línea 94 del archivo [svm\\_v2.py](#).

### 8.8.2.6. `figsize`

`figsize`

Definición en la línea 94 del archivo [svm\\_v2.py](#).

### 8.8.2.7. `polar_ax`

`polar_ax`

Definición en la línea 94 del archivo [svm\\_v2.py](#).

### 8.8.2.8. `root`

`root = tk.Tk()`

Definición en la línea 90 del archivo [svm\\_v2.py](#).

### 8.8.2.9. `subplot_kw`

`subplot_kw`

Definición en la línea 94 del archivo [svm\\_v2.py](#).

### 8.8.2.10. `va`

`va`

Definición en la línea 99 del archivo [svm\\_v2.py](#).

### 8.8.2.11. `vectors`

`list vectors = [ ]`

Definición en la línea 103 del archivo [svm\\_v2.py](#).



## Capítulo 9

# Documentación de estructuras de datos

### 9.1. 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`

#### 9.1.1. 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](#).

#### 9.1.2. Documentación de campos

##### 9.1.2.1. `bitmap`

```
const uint8_t* bitmap
```

Puntero al carácter

Definición en la línea 177 del archivo [sh1106\\_graphics.c](#).

### 9.1.2.2. h

```
uint8_t h
```

Alto del carácter

Definición en la línea 179 del archivo [sh1106\\_graphics.c](#).

### 9.1.2.3. w

```
uint8_t w
```

Ancho del carácter

Definición en la línea 178 del archivo [sh1106\\_graphics.c](#).

### 9.1.2.4. x

```
int x
```

Posición en X del carácter

Definición en la línea 180 del archivo [sh1106\\_graphics.c](#).

### 9.1.2.5. 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/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/display/[sh1106\\_graphics.c](#)

## 9.2. Referencia de la estructura ConfiguracionSVM

Parámetros de configuración del módulo SVM.

```
#include <GestorSVM.h>
```

### Campos de datos

- int `frec_switch`  
*Frecuencia de switching del PWM SVM [Hz].*
- int `frecReferencia`  
*Frecuencia de referencia (target) en régimen [Hz].*
- int `direccionRotacion`  
*1 = sentido horario, -1 = antihorario.*
- int `acel`  
*Aceleración dinámica [Hz/seg].*
- int `desacel`

#### 9.2.1. Descripción detallada

Parámetros de configuración del módulo SVM.

- `frec_switch`: frecuencia del timer de switching (Hz).
- `frecReferencia`: frecuencia objetivo de marcha estable (Hz).
- `direccionRotacion`: 1 horario, -1 antihorario.
- `acel` / `desacel`: rampas dinámicas [Hz/seg].

Definición en la línea 31 del archivo [GestorSVM.h](#).

#### 9.2.2. Documentación de campos

##### 9.2.2.1. `acel`

```
int acel
```

1 = sentido horario, -1 = antihorario.

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

##### 9.2.2.2. `desacel`

```
int desacel
```

Aceleración dinámica [Hz/seg].

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

##### 9.2.2.3. `direccionRotacion`

```
int direccionRotacion
```

Frecuencia de referencia (target) en régimen [Hz].

Definición en la línea 34 del archivo [GestorSVM.h](#).

### 9.2.2.4. `frec_switch`

```
int frec_switch
```

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

### 9.2.2.5. `frecReferencia`

```
int frecReferencia
```

Frecuencia de switching del PWM SVM [Hz].

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

La documentación de esta estructura está generada del siguiente archivo:

- Modules/Gestor\_SVM/[GestorSVM.h](#)

## 9.3. Referencia de la estructura `DatoCalculado`

### Campos de datos

- int `ticksChannel1` [`BUFFER_CALCULO_SIZE`]
- int `ticksChannel2` [`BUFFER_CALCULO_SIZE`]
- int `ticksChannel3` [`BUFFER_CALCULO_SIZE`]
- `CasoInterferenciaTimer` `casoInterferencia` [`BUFFER_CALCULO_SIZE`]
- int `cuadranteActual` [`BUFFER_CALCULO_SIZE`]
- int `indiceEscritura`
- int `indiceLectura`  
*Índice de escritura del productor.*
- int `contadorDeDatos`  
*Índice de lectura del consumidor.*

### 9.3.1. Descripción detallada

Definición en la línea 134 del archivo [GestorSVM.c](#).

### 9.3.2. Documentación de campos

#### 9.3.2.1. `casoInterferencia`

```
CasoInterferenciaTimer casoInterferencia[BUFFER_CALCULO_SIZE]
```

Definición en la línea 138 del archivo [GestorSVM.c](#).

### 9.3.2.2. contadorDeDatos

```
int contadorDeDatos
```

Índice de lectura del consumidor.

Definición en la línea 142 del archivo [GestorSVM.c](#).

### 9.3.2.3. cuadranteActual

```
int cuadranteActual[BUFFER_CALCULO_SIZE]
```

Definición en la línea 139 del archivo [GestorSVM.c](#).

### 9.3.2.4. indiceEscritura

```
int indiceEscritura
```

Definición en la línea 140 del archivo [GestorSVM.c](#).

### 9.3.2.5. indiceLectura

```
int indiceLectura
```

Índice de escritura del productor.

Definición en la línea 141 del archivo [GestorSVM.c](#).

### 9.3.2.6. ticksChannel1

```
int ticksChannel1[BUFFER_CALCULO_SIZE]
```

Definición en la línea 135 del archivo [GestorSVM.c](#).

### 9.3.2.7. ticksChannel2

```
int ticksChannel2[BUFFER_CALCULO_SIZE]
```

Definición en la línea 136 del archivo [GestorSVM.c](#).

### 9.3.2.8. ticksChannel3

```
int ticksChannel3[BUFFER_CALCULO_SIZE]
```

Definición en la línea 137 del archivo [GestorSVM.c](#).

La documentación de esta estructura está generada del siguiente archivo:

- Modules/Gestor\_SVM/[GestorSVM.c](#)

## 9.4. Referencia de la estructura frequency\_settings\_SH1106\_t

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

### Campos de datos

- `frequency_settings_t frequency_settings`
- `sh1106_variable_lines_e * edit`
- `uint8_t edit_variable`
- `uint8_t * edit_flag`
- `uint8_t * multiplier`

### 9.4.1. Descripción detallada

Definición en la línea 127 del archivo [LVFV\\_system.h](#).

### 9.4.2. Documentación de campos

#### 9.4.2.1. edit

```
sh1106_variable_lines_e* edit
```

Definición en la línea 129 del archivo [LVFV\\_system.h](#).

#### 9.4.2.2. edit\_flag

```
uint8_t* edit_flag
```

Definición en la línea 131 del archivo [LVFV\\_system.h](#).

#### 9.4.2.3. edit\_variable

```
uint8_t edit_variable
```

Definición en la línea 130 del archivo [LVFV\\_system.h](#).

#### 9.4.2.4. frequency\_settings

```
frequency_settings_t frequency_settings
```

Definición en la línea 128 del archivo [LVFV\\_system.h](#).

#### 9.4.2.5. 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/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/[LVFV\\_system.h](#)

## 9.5. 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

### 9.5.1. Descripción detallada

Definición en la línea 86 del archivo [LVFV\\_system.h](#).

### 9.5.2. Documentación de campos

#### 9.5.2.1. acceleration

```
uint16_t acceleration
```

Definición en la línea 88 del archivo [LVFV\\_system.h](#).

#### 9.5.2.2. desacceleration

```
uint16_t desacceleration
```

Definición en la línea 89 del archivo [LVFV\\_system.h](#).

#### 9.5.2.3. freq\_regime

```
uint16_t freq_regime
```

Definición en la línea 87 del archivo [LVFV\\_system.h](#).

### 9.5.2.4. input\_variable

```
uint16_t input_variable
```

Definición en la línea 90 del archivo [LVFV\\_system.h](#).

La documentación de esta estructura está generada del siguiente archivo:

- C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/[LVFV\\_system.h](#)

## 9.6. 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

### 9.6.1. Descripción detallada

Definición en la línea 237 del archivo [mcp23017\\_defs.h](#).

### 9.6.2. Documentación de campos

#### 9.6.2.1. all

```
uint8_t all
```

Unificación de la variable

Definición en la línea 238 del archivo [mcp23017\\_defs.h](#).

#### 9.6.2.2. [struct]

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

### 9.6.2.3. DEF0

```
uint8_t DEF0
```

Bit 0 de DEFVAL

Definición en la línea 240 del archivo [mcp23017\\_defs.h](#).

### 9.6.2.4. DEF1

```
uint8_t DEF1
```

Bit 1 de DEFVAL

Definición en la línea 241 del archivo [mcp23017\\_defs.h](#).

### 9.6.2.5. DEF2

```
uint8_t DEF2
```

Bit 2 de DEFVAL

Definición en la línea 242 del archivo [mcp23017\\_defs.h](#).

### 9.6.2.6. DEF3

```
uint8_t DEF3
```

Bit 3 de DEFVAL

Definición en la línea 243 del archivo [mcp23017\\_defs.h](#).

### 9.6.2.7. DEF4

```
uint8_t DEF4
```

Bit 4 de DEFVAL

Definición en la línea 244 del archivo [mcp23017\\_defs.h](#).

### 9.6.2.8. DEF5

```
uint8_t DEF5
```

Bit 5 de DEFVAL

Definición en la línea 245 del archivo [mcp23017\\_defs.h](#).

### 9.6.2.9. DEF6

```
uint8_t DEF6
```

Bit 6 de DEFVAL

Definición en la línea 246 del archivo [mcp23017\\_defs.h](#).

### 9.6.2.10. 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/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/io\_←control/[mcp23017\\_defs.h](#)

## 9.7. 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

### 9.7.1. Descripción detallada

Definición en la línea 153 del archivo [mcp23017\\_defs.h](#).

## 9.7.2. Documentación de campos

### 9.7.2.1. all

```
uint8_t all
```

Unificación de la variable

Definición en la línea 154 del archivo [mcp23017\\_defs.h](#).

### 9.7.2.2. [struct]

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

### 9.7.2.3. GPINT0

```
uint8_t GPINT0
```

Bit 0 de GPINEN

Definición en la línea 156 del archivo [mcp23017\\_defs.h](#).

### 9.7.2.4. GPINT1

```
uint8_t GPINT1
```

Bit 1 de GPINEN

Definición en la línea 157 del archivo [mcp23017\\_defs.h](#).

### 9.7.2.5. GPINT2

```
uint8_t GPINT2
```

Bit 2 de GPINEN

Definición en la línea 158 del archivo [mcp23017\\_defs.h](#).

### 9.7.2.6. GPINT3

```
uint8_t GPINT3
```

Bit 3 de GPINEN

Definición en la línea 159 del archivo [mcp23017\\_defs.h](#).

### 9.7.2.7. GPINT4

```
uint8_t GPINT4
```

Bit 4 de GPINTEN

Definición en la línea 160 del archivo [mcp23017\\_defs.h](#).

### 9.7.2.8. GPINT5

```
uint8_t GPINT5
```

Bit 5 de GPINTEN

Definición en la línea 161 del archivo [mcp23017\\_defs.h](#).

### 9.7.2.9. GPINT6

```
uint8_t GPINT6
```

Bit 6 de GPINTEN

Definición en la línea 162 del archivo [mcp23017\\_defs.h](#).

### 9.7.2.10. 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/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/io\_←  
control/mcp23017\_defs.h

## 9.8. 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
```

### 9.8.1. Descripción detallada

Definición en la línea 195 del archivo [mcp23017\\_defs.h](#).

### 9.8.2. Documentación de campos

#### 9.8.2.1. all

```
uint8_t all
```

Unificación de la variable

Definición en la línea 196 del archivo [mcp23017\\_defs.h](#).

#### 9.8.2.2. [struct]

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

#### 9.8.2.3. GP0

```
uint8_t GP0
```

Bit 0 de GPIO

Definición en la línea 198 del archivo [mcp23017\\_defs.h](#).

#### 9.8.2.4. GP1

```
uint8_t GP1
```

Bit 1 de GPIO

Definición en la línea 199 del archivo [mcp23017\\_defs.h](#).

### 9.8.2.5. GP2

```
uint8_t GP2
```

Bit 2 de GPIO

Definición en la línea 200 del archivo [mcp23017\\_defs.h](#).

### 9.8.2.6. GP3

```
uint8_t GP3
```

Bit 3 de GPIO

Definición en la línea 201 del archivo [mcp23017\\_defs.h](#).

### 9.8.2.7. GP4

```
uint8_t GP4
```

Bit 4 de GPIO

Definición en la línea 202 del archivo [mcp23017\\_defs.h](#).

### 9.8.2.8. GP5

```
uint8_t GP5
```

Bit 5 de GPIO

Definición en la línea 203 del archivo [mcp23017\\_defs.h](#).

### 9.8.2.9. GP6

```
uint8_t GP6
```

Bit 6 de GPIO

Definición en la línea 204 del archivo [mcp23017\\_defs.h](#).

### 9.8.2.10. 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/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/io\_→control/mcp23017\_defs.h

## 9.9. 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`

### 9.9.1. Descripción detallada

Definición en la línea 174 del archivo [mcp23017\\_defs.h](#).

### 9.9.2. Documentación de campos

#### 9.9.2.1. all

```
uint8_t all
```

Unificación de la variable

Definición en la línea 175 del archivo [mcp23017\\_defs.h](#).

#### 9.9.2.2. [struct]

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

#### 9.9.2.3. PU0

```
uint8_t PU0
```

Bit 0 de GPPU

Definición en la línea 177 del archivo [mcp23017\\_defs.h](#).

#### 9.9.2.4. PU1

```
uint8_t PU1
```

Bit 1 de GPPU

Definición en la línea 178 del archivo [mcp23017\\_defs.h](#).

#### 9.9.2.5. PU2

```
uint8_t PU2
```

Bit 2 de GPPU

Definición en la línea 179 del archivo [mcp23017\\_defs.h](#).

#### 9.9.2.6. PU3

```
uint8_t PU3
```

Bit 3 de GPPU

Definición en la línea 180 del archivo [mcp23017\\_defs.h](#).

#### 9.9.2.7. PU4

```
uint8_t PU4
```

Bit 4 de GPPU

Definición en la línea 181 del archivo [mcp23017\\_defs.h](#).

#### 9.9.2.8. PU5

```
uint8_t PU5
```

Bit 5 de GPPU

Definición en la línea 182 del archivo [mcp23017\\_defs.h](#).

#### 9.9.2.9. PU6

```
uint8_t PU6
```

Bit 6 de GPPU

Definición en la línea 183 del archivo [mcp23017\\_defs.h](#).

### 9.9.2.10. 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/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/io\_control/mcp23017\_defs.h

## 9.10. 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

### 9.10.1. Descripción detallada

Definición en la línea 321 del archivo [mcp23017\\_defs.h](#).

### 9.10.2. Documentación de campos

#### 9.10.2.1. all

uint8\_t all

Unificación de la variable

Definición en la línea 322 del archivo [mcp23017\\_defs.h](#).

### 9.10.2.2. [struct]

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

### 9.10.2.3. ICP0

uint8\_t ICP0

Bit 0 de INTCAP

Definición en la línea [324](#) del archivo [mcp23017\\_defs.h](#).

### 9.10.2.4. ICP1

uint8\_t ICP1

Bit 1 de INTCAP

Definición en la línea [325](#) del archivo [mcp23017\\_defs.h](#).

### 9.10.2.5. ICP2

uint8\_t ICP2

Bit 2 de INTCAP

Definición en la línea [326](#) del archivo [mcp23017\\_defs.h](#).

### 9.10.2.6. ICP3

uint8\_t ICP3

Bit 3 de INTCAP

Definición en la línea [327](#) del archivo [mcp23017\\_defs.h](#).

### 9.10.2.7. ICP4

uint8\_t ICP4

Bit 4 de INTCAP

Definición en la línea [328](#) del archivo [mcp23017\\_defs.h](#).

### 9.10.2.8. ICP5

```
uint8_t ICP5
```

Bit 5 de INTCAP

Definición en la línea 329 del archivo [mcp23017\\_defs.h](#).

### 9.10.2.9. ICP6

```
uint8_t ICP6
```

Bit 6 de INTCAP

Definición en la línea 330 del archivo [mcp23017\\_defs.h](#).

### 9.10.2.10. 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/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/io\_control/mcp23017\_defs.h

## 9.11. 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

### 9.11.1. Descripción detallada

Definición en la línea [258](#) del archivo [mcp23017\\_defs.h](#).

### 9.11.2. Documentación de campos

#### 9.11.2.1. all

```
uint8_t all
```

Unificación de la variable

Definición en la línea [259](#) del archivo [mcp23017\\_defs.h](#).

#### 9.11.2.2. [struct]

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

#### 9.11.2.3. IOC0

```
uint8_t IOC0
```

Bit 0 de INTCON

Definición en la línea [261](#) del archivo [mcp23017\\_defs.h](#).

#### 9.11.2.4. IOC1

```
uint8_t IOC1
```

Bit 1 de INTCON

Definición en la línea [262](#) del archivo [mcp23017\\_defs.h](#).

#### 9.11.2.5. IOC2

```
uint8_t IOC2
```

Bit 2 de INTCON

Definición en la línea [263](#) del archivo [mcp23017\\_defs.h](#).

#### 9.11.2.6. IOC3

```
uint8_t IOC3
```

Bit 3 de INTCON

Definición en la línea [264](#) del archivo [mcp23017\\_defs.h](#).

### 9.11.2.7. IOC4

```
uint8_t IOC4
```

Bit 4 de INTCON

Definición en la línea 265 del archivo [mcp23017\\_defs.h](#).

### 9.11.2.8. IOC5

```
uint8_t IOC5
```

Bit 5 de INTCON

Definición en la línea 266 del archivo [mcp23017\\_defs.h](#).

### 9.11.2.9. IOC6

```
uint8_t IOC6
```

Bit 6 de INTCON

Definición en la línea 267 del archivo [mcp23017\\_defs.h](#).

### 9.11.2.10. 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/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/io\_←  
control/mcp23017\_defs.h

## 9.12. 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
```

### 9.12.1. Descripción detallada

Definición en la línea 300 del archivo [mcp23017\\_defs.h](#).

### 9.12.2. Documentación de campos

#### 9.12.2.1. all

```
uint8_t all
```

Unificación de la variable

Definición en la línea 301 del archivo [mcp23017\\_defs.h](#).

#### 9.12.2.2. [struct]

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

#### 9.12.2.3. INT0

```
uint8_t INT0
```

Bit 0 de INTF

Definición en la línea 303 del archivo [mcp23017\\_defs.h](#).

#### 9.12.2.4. INT1

```
uint8_t INT1
```

Bit 1 de INTF

Definición en la línea 304 del archivo [mcp23017\\_defs.h](#).

### 9.12.2.5. INT2

```
uint8_t INT2
```

Bit 2 de INTF

Definición en la línea 305 del archivo [mcp23017\\_defs.h](#).

### 9.12.2.6. INT3

```
uint8_t INT3
```

Bit 3 de INTF

Definición en la línea 306 del archivo [mcp23017\\_defs.h](#).

### 9.12.2.7. INT4

```
uint8_t INT4
```

Bit 4 de INTF

Definición en la línea 307 del archivo [mcp23017\\_defs.h](#).

### 9.12.2.8. INT5

```
uint8_t INT5
```

Bit 5 de INTF

Definición en la línea 308 del archivo [mcp23017\\_defs.h](#).

### 9.12.2.9. INT6

```
uint8_t INT6
```

Bit 6 de INTF

Definición en la línea 309 del archivo [mcp23017\\_defs.h](#).

### 9.12.2.10. 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/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/io\_→  
control/mcp23017\_defs.h

## 9.13. 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

### 9.13.1. Descripción detallada

Definición en la línea 279 del archivo [mcp23017\\_defs.h](#).

### 9.13.2. Documentación de campos

#### 9.13.2.1. all

```
uint8_t all
```

Unificación de la variable

Definición en la línea 280 del archivo [mcp23017\\_defs.h](#).

#### 9.13.2.2. 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](#).

#### 9.13.2.3. [struct]

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

#### 9.13.2.4. DISSLW

```
uint8_t DISSLW
```

Bit 4 de IOCON - Configuración de velocidad del MCP23017

Definición en la línea 286 del archivo [mcp23017\\_defs.h](#).

#### 9.13.2.5. INTPOL

```
uint8_t INTPOL
```

Bit 1 de IOCON - Configuración de polaridad de las entradas

Definición en la línea 283 del archivo [mcp23017\\_defs.h](#).

#### 9.13.2.6. 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](#).

#### 9.13.2.7. 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](#).

#### 9.13.2.8. reserved

```
uint8_t reserved
```

Bit 0 de IOCON - Sin uso

Definición en la línea 282 del archivo [mcp23017\\_defs.h](#).

#### 9.13.2.9. reserved\_2

```
uint8_t reserved_2
```

Bit 3 de IOCON - Sin uso

Definición en la línea 285 del archivo [mcp23017\\_defs.h](#).

### 9.13.2.10. 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/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/io\_control/mcp23017\_defs.h

## 9.14. 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

### 9.14.1. Descripción detallada

Definición en la línea 111 del archivo [mcp23017\\_defs.h](#).

### 9.14.2. Documentación de campos

#### 9.14.2.1. all

```
uint8_t all
```

Unificación de la variable

Definición en la línea 112 del archivo [mcp23017\\_defs.h](#).

**9.14.2.2. [struct]**

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

**9.14.2.3. IO0**

```
uint8_t IO0
```

Bit 0 de IODIR

Definición en la línea 114 del archivo [mcp23017\\_defs.h](#).

**9.14.2.4. IO1**

```
uint8_t IO1
```

Bit 1 de IODIR

Definición en la línea 115 del archivo [mcp23017\\_defs.h](#).

**9.14.2.5. IO2**

```
uint8_t IO2
```

Bit 2 de IODIR

Definición en la línea 116 del archivo [mcp23017\\_defs.h](#).

**9.14.2.6. IO3**

```
uint8_t IO3
```

Bit 3 de IODIR

Definición en la línea 117 del archivo [mcp23017\\_defs.h](#).

**9.14.2.7. IO4**

```
uint8_t IO4
```

Bit 4 de IODIR

Definición en la línea 118 del archivo [mcp23017\\_defs.h](#).

### 9.14.2.8. IO5

```
uint8_t IO5
```

Bit 5 de IODIR

Definición en la línea 119 del archivo [mcp23017\\_defs.h](#).

### 9.14.2.9. IO6

```
uint8_t IO6
```

Bit 6 de IODIR

Definición en la línea 120 del archivo [mcp23017\\_defs.h](#).

### 9.14.2.10. 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/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/io\_control/mcp23017\_defs.h

## 9.15. 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}

### 9.15.1. Descripción detallada

Definición en la línea 132 del archivo [mcp23017\\_defs.h](#).

### 9.15.2. Documentación de campos

#### 9.15.2.1. all

```
uint8_t all
```

Unificación de la variable

Definición en la línea 133 del archivo [mcp23017\\_defs.h](#).

#### 9.15.2.2. [struct]

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

#### 9.15.2.3. IPO

```
uint8_t IPO
```

Bit 0 de IPOL

Definición en la línea 135 del archivo [mcp23017\\_defs.h](#).

#### 9.15.2.4. IP1

```
uint8_t IP1
```

Bit 1 de IPOL

Definición en la línea 136 del archivo [mcp23017\\_defs.h](#).

#### 9.15.2.5. IP2

```
uint8_t IP2
```

Bit 2 de IPOL

Definición en la línea 137 del archivo [mcp23017\\_defs.h](#).

#### 9.15.2.6. IP3

```
uint8_t IP3
```

Bit 3 de IPOL

Definición en la línea 138 del archivo [mcp23017\\_defs.h](#).

### 9.15.2.7. IP4

```
uint8_t IP4
```

Bit 4 de IPOL

Definición en la línea 139 del archivo [mcp23017\\_defs.h](#).

### 9.15.2.8. IP5

```
uint8_t IP5
```

Bit 5 de IPOL

Definición en la línea 140 del archivo [mcp23017\\_defs.h](#).

### 9.15.2.9. IP6

```
uint8_t IP6
```

Bit 6 de IPOL

Definición en la línea 141 del archivo [mcp23017\\_defs.h](#).

### 9.15.2.10. 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/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/io\_←  
control/mcp23017\_defs.h

## 9.16. 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
```

### 9.16.1. Descripción detallada

Definición en la línea 216 del archivo [mcp23017\\_defs.h](#).

### 9.16.2. Documentación de campos

#### 9.16.2.1. all

```
uint8_t all
```

Unificación de la variable

Definición en la línea 217 del archivo [mcp23017\\_defs.h](#).

#### 9.16.2.2. [struct]

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

#### 9.16.2.3. OL0

```
uint8_t OL0
```

Bit 0 de OLAT

Definición en la línea 219 del archivo [mcp23017\\_defs.h](#).

#### 9.16.2.4. OL1

```
uint8_t OL1
```

Bit 1 de OLAT

Definición en la línea 220 del archivo [mcp23017\\_defs.h](#).

### 9.16.2.5. OL2

uint8\_t OL2

Bit 2 de OLAT

Definición en la línea 221 del archivo [mcp23017\\_defs.h](#).

### 9.16.2.6. OL3

uint8\_t OL3

Bit 3 de OLAT

Definición en la línea 222 del archivo [mcp23017\\_defs.h](#).

### 9.16.2.7. OL4

uint8\_t OL4

Bit 4 de OLAT

Definición en la línea 223 del archivo [mcp23017\\_defs.h](#).

### 9.16.2.8. OL5

uint8\_t OL5

Bit 5 de OLAT

Definición en la línea 224 del archivo [mcp23017\\_defs.h](#).

### 9.16.2.9. OL6

uint8\_t OL6

Bit 6 de OLAT

Definición en la línea 225 del archivo [mcp23017\\_defs.h](#).

### 9.16.2.10. 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/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/io\_→control/mcp23017\_defs.h

## 9.17. Referencia de la estructura Parametros

Parámetros “sombra” para sincronización atómica con el lazo de cálculo.

### Campos de datos

- int32\_t **frecObjetivo**
- uint32\_t **cambioFrecuenciaPorCiclo**
- int **direccionRotacion**
- int **flagMotorRunning**
- int **flagEsAcelerado**
- int **flagChangingFrecuencia**

### 9.17.1. Descripción detallada

Parámetros “sombra” para sincronización atómica con el lazo de cálculo.

Definición en la línea 150 del archivo [GestorSVM.c](#).

### 9.17.2. Documentación de campos

#### 9.17.2.1. cambioFrecuenciaPorCiclo

```
uint32_t cambioFrecuenciaPorCiclo
```

Definición en la línea 152 del archivo [GestorSVM.c](#).

#### 9.17.2.2. direccionRotacion

```
int direccionRotacion
```

Definición en la línea 153 del archivo [GestorSVM.c](#).

#### 9.17.2.3. flagChangingFrecuencia

```
int flagChangingFrecuencia
```

Definición en la línea 156 del archivo [GestorSVM.c](#).

#### 9.17.2.4. flagEsAcelerado

```
int flagEsAcelerado
```

Definición en la línea 155 del archivo [GestorSVM.c](#).

### 9.17.2.5. flagMotorRunning

```
int flagMotorRunning
```

Definición en la línea 154 del archivo [GestorSVM.c](#).

### 9.17.2.6. freqObjetivo

```
int32_t freqObjetivo
```

Definición en la línea 151 del archivo [GestorSVM.c](#).

La documentación de esta estructura está generada del siguiente archivo:

- Modules/Gestor\_SVM/[GestorSVM.c](#)

## 9.18. Referencia de la estructura rtc\_alarms\_t

### Campos de datos

- esp\_timer\_handle\_t [start\\_timer](#)
- esp\_timer\_handle\_t [stop\\_timer](#)

### 9.18.1. Descripción detallada

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

### 9.18.2. Documentación de campos

#### 9.18.2.1. start\_timer

```
esp_timer_handle_t start_timer
```

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

#### 9.18.2.2. 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/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/rtc/rtc.c

## 9.19. Referencia de la estructura seecurity\_settings\_SH1106\_t

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

### Campos de datos

- `seecurity_settings_t` `seecurity_settings`
- `sh1106_variable_lines_e` \* `edit`
- `uint8_t` `edit_variable`
- `uint8_t` \* `edit_flag`
- `uint8_t` \* `multiplier`

### 9.19.1. Descripción detallada

Definición en la línea 143 del archivo [LVFV\\_system.h](#).

### 9.19.2. Documentación de campos

#### 9.19.2.1. edit

```
sh1106_variable_lines_e* edit
```

Definición en la línea 145 del archivo [LVFV\\_system.h](#).

#### 9.19.2.2. edit\_flag

```
uint8_t* edit_flag
```

Definición en la línea 147 del archivo [LVFV\\_system.h](#).

#### 9.19.2.3. edit\_variable

```
uint8_t edit_variable
```

Definición en la línea 146 del archivo [LVFV\\_system.h](#).

#### 9.19.2.4. multiplier

```
uint8_t* multiplier
```

Definición en la línea 148 del archivo [LVFV\\_system.h](#).

### 9.19.2.5. security\_settings

```
security_settings_t security_settings
```

Definición en la línea 144 del archivo [LVFV\\_system.h](#).

La documentación de esta estructura está generada del siguiente archivo:

- C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/[LVFV\\_system.h](#)

## 9.20. Referencia de la estructura security\_settings\_t

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

### Campos de datos

- uint16\_t [vbus\\_min](#)
- uint16\_t [ibus\\_max](#)

### 9.20.1. Descripción detallada

Definición en la línea 99 del archivo [LVFV\\_system.h](#).

### 9.20.2. Documentación de campos

#### 9.20.2.1. ibus\_max

```
uint16_t ibus_max
```

Definición en la línea 101 del archivo [LVFV\\_system.h](#).

#### 9.20.2.2. vbus\_min

```
uint16_t vbus_min
```

Definición en la línea 100 del archivo [LVFV\\_system.h](#).

La documentación de esta estructura está generada del siguiente archivo:

- C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/[LVFV\\_system.h](#)

## 9.21. 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]`

#### 9.21.1. Descripción detallada

Definición en la línea 13 del archivo [sh1106\\_graphics.h](#).

#### 9.21.2. Documentación de campos

##### 9.21.2.1. buffer

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

Definición en la línea 17 del archivo [sh1106\\_graphics.h](#).

##### 9.21.2.2. height

```
uint8_t height
```

Definición en la línea 15 del archivo [sh1106\\_graphics.h](#).

##### 9.21.2.3. rotation

```
uint8_t rotation
```

Definición en la línea 16 del archivo [sh1106\\_graphics.h](#).

##### 9.21.2.4. 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/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/display/[sh1106\\_graphics.h](#)

## 9.22. 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 puede recibirse como respuesta en consecuencia.

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

## Campos de datos

- `SPI_Request request`
- `int getValue`
- `int setValue`

### 9.22.1. 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 puede 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](#).

### 9.22.2. Documentación de campos

#### 9.22.2.1. `getValue`

```
int getValue
```

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

#### 9.22.2.2. `request`

```
SPI_Request request
```

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

#### 9.22.2.3. `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/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVVF\_ESP32/main/system/[sysControl.h](#)

## 9.23. Referencia de la estructura `system_status_t`

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

```
#include <LVVF_system.h>
```

### Campos de datos

- `uint16_t frequency`
- `uint16_t frequency_destiny`
- `uint16_t vbus_min`
- `uint16_t ibus_max`
- `uint16_t acceleration`
- `uint16_t desacceleration`
- `uint8_t inputs_status`
- `system_status_e status`
- `uint8_t emergency_signals`

#### 9.23.1. Descripción detallada

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

Definición en la línea 109 del archivo [LVFV\\_system.h](#).

#### 9.23.2. Documentación de campos

##### 9.23.2.1. acceleration

```
uint16_t acceleration
```

Definición en la línea 114 del archivo [LVFV\\_system.h](#).

##### 9.23.2.2. desacceleration

```
uint16_t desacceleration
```

Definición en la línea 115 del archivo [LVFV\\_system.h](#).

##### 9.23.2.3. emergency\_signals

```
uint8_t emergency_signals
```

Definición en la línea 118 del archivo [LVFV\\_system.h](#).

##### 9.23.2.4. frequency

```
uint16_t frequency
```

Definición en la línea 110 del archivo [LVFV\\_system.h](#).

### 9.23.2.5. frequency\_destiny

```
uint16_t frequency_destiny
```

Definición en la línea 111 del archivo [LVFV\\_system.h](#).

### 9.23.2.6. ibus\_max

```
uint16_t ibus_max
```

Definición en la línea 113 del archivo [LVFV\\_system.h](#).

### 9.23.2.7. inputs\_status

```
uint8_t inputs_status
```

Definición en la línea 116 del archivo [LVFV\\_system.h](#).

### 9.23.2.8. status

```
system_status_e status
```

Definición en la línea 117 del archivo [LVFV\\_system.h](#).

### 9.23.2.9. vbus\_min

```
uint16_t vbus_min
```

Definición en la línea 112 del archivo [LVFV\\_system.h](#).

La documentación de esta estructura está generada del siguiente archivo:

- C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/[LVFV\\_system.h](#)

## 9.24. 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`

### 9.24.1. Descripción detallada

Definición en la línea 135 del archivo [LVFV\\_system.h](#).

### 9.24.2. Documentación de campos

#### 9.24.2.1. edit

```
sh1106_variable_lines_e* edit
```

Definición en la línea 137 del archivo [LVFV\\_system.h](#).

#### 9.24.2.2. edit\_flag

```
uint8_t* edit_flag
```

Definición en la línea 139 del archivo [LVFV\\_system.h](#).

#### 9.24.2.3. edit\_variable

```
uint8_t edit_variable
```

Definición en la línea 138 del archivo [LVFV\\_system.h](#).

#### 9.24.2.4. multiplier

```
uint8_t* multiplier
```

Definición en la línea 140 del archivo [LVFV\\_system.h](#).

#### 9.24.2.5. 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/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/[LVFV\\_system.h](#)

## 9.25. 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](#)

#### 9.25.1. Descripción detallada

Definición en la línea [93](#) del archivo [LVFV\\_system.h](#).

#### 9.25.2. Documentación de campos

##### 9.25.2.1. time\_start

```
struct tm* time_start
```

Definición en la línea [95](#) del archivo [LVFV\\_system.h](#).

##### 9.25.2.2. time\_stop

```
struct tm* time_stop
```

Definición en la línea [96](#) del archivo [LVFV\\_system.h](#).

##### 9.25.2.3. time\_system

```
struct tm* time_system
```

Definición en la línea [94](#) del archivo [LVFV\\_system.h](#).

La documentación de esta estructura está generada del siguiente archivo:

- C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/[LVFV\\_system.h](#)

### 9.26. Referencia de la estructura ValoresSwitching

Estructura de trabajo del ISR del timer de switching.

```
#include <GestorSVM.h>
```

## Campos de datos

- char **cuadrante**
- char **flagSwitch** [3]
  - Sector SVM actual (0..5). Lo usa el ISR para decidir qué pierna conmutar.*
- int **ticks1** [2]
  - Flags de acción por pierna {U,V,W}: 1=encender, 0=apagar (par de valores por flanco).*
- int **ticks2** [2]
  - Ticks CCR del evento 1 (subida/bajada).*
- int **ticks3** [2]
  - Ticks CCR del evento 2 (subida/bajada).*
- int **contador**
  - Ticks CCR del evento 3 (subida/bajada).*

### 9.26.1. Descripción detallada

Estructura de trabajo del ISR del timer de switching.

Contiene los parámetros que determinan en qué puntos (ticks) se disparan las comparaciones CCR para conmutar las fases y el estado de conmutación (encendido/apagado) de cada pierna. El campo **cuadrante** indica el sector SVM actual (0..5).

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

### 9.26.2. Documentación de campos

#### 9.26.2.1. contador

```
int contador
```

Ticks CCR del evento 3 (subida/bajada).

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

#### 9.26.2.2. cuadrante

```
char cuadrante
```

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

#### 9.26.2.3. flagSwitch

```
char flagSwitch[3]
```

Sector SVM actual (0..5). Lo usa el ISR para decidir qué pierna conmutar.

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

#### 9.26.2.4. ticks1

```
int ticks1[2]
```

Flags de acción por pierna {U,V,W}: 1=encender, 0=apagar (par de valores por flanco).

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

#### 9.26.2.5. ticks2

```
int ticks2[2]
```

Ticks CCR del evento 1 (subida/bajada).

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

#### 9.26.2.6. ticks3

```
int ticks3[2]
```

Ticks CCR del evento 2 (subida/bajada).

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

La documentación de esta estructura está generada del siguiente archivo:

- Modules/Gestor\_SVM/[GestorSVM.h](#)

## Capítulo 10

# Documentación de archivos

### 10.1. Referencia del archivo Inc/main.h

Header de la aplicación principal.

```
#include "stm32f1xx_hal.h"
```

#### defines

- #define GPIO\_U\_IN GPIO\_PIN\_1  
*Entrada lógica de la pierna U (GPIOA, PIN\_1).*
- #define GPIO\_U\_SD GPIO\_PIN\_2  
*Shutdown/Habilitación driver pierna U (GPIOA, PIN\_2).*
- #define GPIO\_V\_IN GPIO\_PIN\_3  
*Entrada lógica de la pierna V (GPIOA, PIN\_3).*
- #define GPIO\_V\_SD GPIO\_PIN\_4  
*Shutdown/Habilitación driver pierna V (GPIOA, PIN\_4).*
- #define GPIO\_W\_IN GPIO\_PIN\_5  
*Entrada lógica de la pierna W (GPIOA, PIN\_5).*
- #define GPIO\_W\_SD GPIO\_PIN\_6  
*Shutdown/Habilitación driver pierna W (GPIOA, PIN\_6).*
- #define GPIO\_TERMO\_SWITCH GPIO\_PIN\_11  
*Entrada digital: Termo-switch (GPIOA, PIN\_11).*
- #define GPIO\_STOP\_BUTTON GPIO\_PIN\_12  
*Entrada digital: Botón de parada (GPIOA, PIN\_12).*
- #define GPIO\_LED\_STATE GPIO\_PIN\_0  
*LED de estado (GPIOB, PIN\_0).*
- #define GPIO\_LED\_ERROR GPIO\_PIN\_2  
*LED de error (GPIOB, PIN\_2).*

#### Funciones

- void Error\_Handler (void)  
*Manejador global de errores fatales.*

### 10.1.1. Descripción detallada

Header de la aplicación principal.

Contiene definiciones comunes, asignación de pines y prototipos globales compartidos por todo el proyecto.

Definición en el archivo [main.h](#).

### 10.1.2. Documentación de funciones

#### 10.1.2.1. Error\_Handler()

```
void Error_Handler (
    void )
```

Manejador global de errores fatales.

Manejador genérico de errores.

Detiene la ejecución en un bucle y puede ser utilizado para reportar/diagnosticar condiciones críticas. Implementado en [main.c](#).

Deshabilita IRQs y entra en bucle infinito para provocar reset por IWDG si está activo.

Definición en la línea [375](#) del archivo [main.c](#).

## 10.2. main.h

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

```
00001
00007
00008 #ifndef __MAIN_H
00009 #define __MAIN_H
00010
00011 #include "stm32f1xx_hal.h"
00012
00017
00019 #define GPIO_U_IN          GPIO_PIN_1
00021 #define GPIO_U_SD          GPIO_PIN_2
00023 #define GPIO_V_IN          GPIO_PIN_3
00025 #define GPIO_V_SD          GPIO_PIN_4
00027 #define GPIO_W_IN          GPIO_PIN_5
00029 #define GPIO_W_SD          GPIO_PIN_6
00030
00032 #define GPIO_TERMO_SWITCH   GPIO_PIN_11
00034 #define GPIO_STOP_BUTTON    GPIO_PIN_12
00036
00041
00043 #define GPIO_LED_STATE     GPIO_PIN_0
00045 #define GPIO_LED_ERROR      GPIO_PIN_2
00047
00054 void Error_Handler(void);
00055
00056 #endif /* __MAIN_H */
```

## 10.3. Referencia del archivo Inc/stm32f1xx\_hal\_conf.h

HAL configuration file.

```
#include "stm32f1xx_hal_rcc.h"
#include "stm32f1xx_hal_gpio.h"
#include "stm32f1xx_hal_exti.h"
#include "stm32f1xx_hal_dma.h"
#include "stm32f1xx_hal_cortex.h"
#include "stm32f1xx_hal_flash.h"
#include "stm32f1xx_hal_pwr.h"
#include "stm32f1xx_hal_spi.h"
#include "stm32f1xx_hal_tim.h"
#include "stm32f1xx_hal_uart.h"
```

### defines

- `#define HAL_MODULE_ENABLED`

*This is the list of modules to be used in the HAL driver.*

- `#define HAL_DMA_MODULE_ENABLED`
- `#define HAL_GPIO_MODULE_ENABLED`
- `#define HAL_SPI_MODULE_ENABLED`
- `#define HAL_TIM_MODULE_ENABLED`
- `#define HAL_UART_MODULE_ENABLED`
- `#define HAL_CORTEX_MODULE_ENABLED`
- `#define HAL_DMA_MODULE_ENABLED`
- `#define HAL_FLASH_MODULE_ENABLED`
- `#define HAL_EXTI_MODULE_ENABLED`
- `#define HAL_GPIO_MODULE_ENABLED`
- `#define HAL_PWR_MODULE_ENABLED`
- `#define HAL_RCC_MODULE_ENABLED`
- `#define HSE_VALUE 8000000U`

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

- `#define HSE_STARTUP_TIMEOUT 100U`
- `#define HSI_VALUE 8000000U`

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

- `#define LSI_VALUE 40000U`
- Internal Low Speed oscillator (LSI) value.*
- `#define LSE_VALUE 32768U`

*External Low Speed oscillator (LSE) value. This value is used by the UART, RTC HAL module to compute the system frequency.*

- `#define LSE_STARTUP_TIMEOUT 5000U`
- `#define VDD_VALUE 3300U`

*This is the HAL system configuration section.*

- `#define TICK_INT_PRIORITY 15U`
- `#define USERTOS 0U`
- `#define PREFETCH_ENABLE 1U`
- `#define USE_HAL_ADC_REGISTER_CALLBACKS 0U /* ADC register callback disabled */`
- `#define USE_HAL_CAN_REGISTER_CALLBACKS 0U /* CAN register callback disabled */`
- `#define USE_HAL_CEC_REGISTER_CALLBACKS 0U /* CEC register callback disabled */`

- #define USE\_HAL\_DAC\_REGISTER\_CALLBACKS 0U /\* DAC register callback disabled \*/
- #define USE\_HAL\_ETH\_REGISTER\_CALLBACKS 0U /\* ETH register callback disabled \*/
- #define USE\_HAL\_HCD\_REGISTER\_CALLBACKS 0U /\* HCD register callback disabled \*/
- #define USE\_HAL\_I2C\_REGISTER\_CALLBACKS 0U /\* I2C register callback disabled \*/
- #define USE\_HAL\_I2S\_REGISTER\_CALLBACKS 0U /\* I2S register callback disabled \*/
- #define USE\_HAL\_MMC\_REGISTER\_CALLBACKS 0U /\* MMC register callback disabled \*/
- #define USE\_HAL\_NAND\_REGISTER\_CALLBACKS 0U /\* NAND register callback disabled \*/
- #define USE\_HAL\_NOR\_REGISTER\_CALLBACKS 0U /\* NOR register callback disabled \*/
- #define USE\_HAL\_PCCARD\_REGISTER\_CALLBACKS 0U /\* PCCARD register callback disabled \*/
- #define USE\_HAL\_PCD\_REGISTER\_CALLBACKS 0U /\* PCD register callback disabled \*/
- #define USE\_HAL\_RTC\_REGISTER\_CALLBACKS 0U /\* RTC register callback disabled \*/
- #define USE\_HAL\_SD\_REGISTER\_CALLBACKS 0U /\* SD register callback disabled \*/
- #define USE\_HAL\_SMARTCARD\_REGISTER\_CALLBACKS 0U /\* SMARTCARD register callback disabled \*/
- #define MAC\_ADDR0 2U

*Uncomment the line below to expand the "assert\_param" macro in the HAL drivers code.*

- #define MAC\_ADDR1 0U
- #define MAC\_ADDR2 0U
- #define MAC\_ADDR3 0U
- #define MAC\_ADDR4 0U
- #define MAC\_ADDR5 0U
- #define ETH\_RX\_BUF\_SIZE ETH\_MAX\_PACKET\_SIZE /\* buffer size for receive \*/
- #define ETH\_TX\_BUF\_SIZE ETH\_MAX\_PACKET\_SIZE /\* buffer size for transmit \*/
- #define ETH\_RXBUFN 8U /\* 4 Rx buffers of size ETH\_RX\_BUF\_SIZE \*/
- #define ETH\_TXBUFN 4U /\* 4 Tx buffers of size ETH\_TX\_BUF\_SIZE \*/
- #define DP83848\_PHY\_ADDRESS 0x01U
- #define PHY\_RESET\_DELAY 0x000000FFU
- #define PHY\_CONFIG\_DELAY 0x00000FFFU
- #define PHY\_READ\_TO 0x0000FFFFU
- #define PHY\_WRITE\_TO 0x0000FFFFU
- #define PHY\_BCR ((uint16\_t)0x00)
- #define PHY\_BSR ((uint16\_t)0x01)
- #define PHY\_RESET ((uint16\_t)0x8000)
- #define PHY\_LOOPBACK ((uint16\_t)0x4000)
- #define PHY\_FULLDUPLEX\_100M ((uint16\_t)0x2100)
- #define PHY\_HALFDUPLEX\_100M ((uint16\_t)0x2000)
- #define PHY\_FULLDUPLEX\_10M ((uint16\_t)0x0100)
- #define PHY\_HALFDUPLEX\_10M ((uint16\_t)0x0000)
- #define PHY\_AUTONEGOTIATION ((uint16\_t)0x1000)
- #define PHY\_RESTART\_AUTONEGOTIATION ((uint16\_t)0x0200)
- #define PHY\_POWERDOWN ((uint16\_t)0x0800)
- #define PHY\_ISOLATE ((uint16\_t)0x0400)
- #define PHY\_AUTONEGO\_COMPLETE ((uint16\_t)0x0020)
- #define PHY\_LINKED\_STATUS ((uint16\_t)0x0004)
- #define PHY\_JABBER\_DETECTION ((uint16\_t)0x0002)
- #define PHY\_SR ((uint16\_t)0x10U)
- #define PHY\_SPEED\_STATUS ((uint16\_t)0x0002U)
- #define PHY\_DUPLEX\_STATUS ((uint16\_t)0x0004U)
- #define USE\_SPI\_CRC 0U
- #define assert\_param(expr)

*Include module's header file.*

### 10.3.1. Descripción detallada

HAL configuration file.

#### Atención

Copyright (c) 2017 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

Definición en el archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2. Documentación de «define»

#### 10.3.2.1. assert\_param

```
#define assert_param(  
    expr)
```

##### Valor:

```
(void) 0U
```

Include module's header file.

Definición en la línea 383 del archivo [stm32f1xx\\_hal\\_conf.h](#).

#### 10.3.2.2. DP83848\_PHY\_ADDRESS

```
#define DP83848_PHY_ADDRESS 0x01U
```

Definición en la línea 188 del archivo [stm32f1xx\\_hal\\_conf.h](#).

#### 10.3.2.3. ETH\_RX\_BUF\_SIZE

```
#define ETH_RX_BUF_SIZE ETH_MAX_PACKET_SIZE /* buffer size for receive */
```

Definición en la línea 180 del archivo [stm32f1xx\\_hal\\_conf.h](#).

#### 10.3.2.4. ETH\_RXBUFN

```
#define ETH_RXBUFN 8U /* 4 Rx buffers of size ETH_RX_BUF_SIZE */
```

Definición en la línea 182 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.5. ETH\_TX\_BUF\_SIZE

```
#define ETH_TX_BUF_SIZE ETH_MAX_PACKET_SIZE /* buffer size for transmit */
```

Definición en la línea 181 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.6. ETH\_TXBUFN

```
#define ETH_TXBUFN 4U /* 4 Tx buffers of size ETH_TX_BUF_SIZE */
```

Definición en la línea 183 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.7. HAL\_CORTEX\_MODULE\_ENABLED

```
#define HAL_CORTEX_MODULE_ENABLED
```

Definición en la línea 72 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.8. HAL\_DMA\_MODULE\_ENABLED [1/2]

```
#define HAL_DMA_MODULE_ENABLED
```

Definición en la línea 45 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.9. HAL\_DMA\_MODULE\_ENABLED [2/2]

```
#define HAL_DMA_MODULE_ENABLED
```

Definición en la línea 45 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.10. HAL\_EXTI\_MODULE\_ENABLED

```
#define HAL_EXTI_MODULE_ENABLED
```

Definición en la línea 75 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.11. HAL\_FLASH\_MODULE\_ENABLED

```
#define HAL_FLASH_MODULE_ENABLED
```

Definición en la línea 74 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.12. HAL\_GPIO\_MODULE\_ENABLED [1/2]

```
#define HAL_GPIO_MODULE_ENABLED
```

Definición en la línea 48 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.13. HAL\_GPIO\_MODULE\_ENABLED [2/2]

```
#define HAL_GPIO_MODULE_ENABLED
```

Definición en la línea 48 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.14. HAL\_MODULE\_ENABLED

```
#define HAL_MODULE_ENABLED
```

This is the list of modules to be used in the HAL driver.

Definición en la línea 36 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.15. HAL\_PWR\_MODULE\_ENABLED

```
#define HAL_PWR_MODULE_ENABLED
```

Definición en la línea 77 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.16. HAL\_RCC\_MODULE\_ENABLED

```
#define HAL_RCC_MODULE_ENABLED
```

Definición en la línea 78 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.17. HAL\_SPI\_MODULE\_ENABLED

```
#define HAL_SPI_MODULE_ENABLED
```

Definición en la línea 65 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.18. HAL\_TIM\_MODULE\_ENABLED

```
#define HAL_TIM_MODULE_ENABLED
```

Definición en la línea 67 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.19. HAL\_UART\_MODULE\_ENABLED

```
#define HAL_UART_MODULE_ENABLED
```

Definición en la línea 68 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.20. HSE\_STARTUP\_TIMEOUT

```
#define HSE_STARTUP_TIMEOUT 100U
```

Time out for HSE start up, in ms

Definición en la línea 91 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.21. HSE\_VALUE

```
#define HSE_VALUE 8000000U
```

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

Value of the External oscillator in Hz

Definición en la línea 87 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.22. HSI\_VALUE

```
#define HSI_VALUE 8000000U
```

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

Value of the Internal oscillator in Hz

Definición en la línea 100 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.23. LSE\_STARTUP\_TIMEOUT

```
#define LSE_STARTUP_TIMEOUT 5000U
```

Time out for LSE start up, in ms

Definición en la línea 121 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.24. LSE\_VALUE

```
#define LSE_VALUE 32768U
```

External Low Speed oscillator (LSE) value. This value is used by the UART, RTC HAL module to compute the system frequency.

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

Definición en la línea 117 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.25. LSI\_VALUE

```
#define LSI_VALUE 40000U
```

Internal Low Speed oscillator (LSI) value.

LSI Typical Value in Hz

Definición en la línea 107 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.26. MAC\_ADDR0

```
#define MAC_ADDR0 2U
```

Uncomment the line below to expand the "assert\_param" macro in the HAL drivers code.

Definición en la línea 172 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.27. MAC\_ADDR1

```
#define MAC_ADDR1 0U
```

Definición en la línea 173 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.28. MAC\_ADDR2

```
#define MAC_ADDR2 0U
```

Definición en la línea 174 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.29. MAC\_ADDR3

```
#define MAC_ADDR3 0U
```

Definición en la línea 175 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.30. MAC\_ADDR4

```
#define MAC_ADDR4 0U
```

Definición en la línea 176 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.31. MAC\_ADDR5

```
#define MAC_ADDR5 0U
```

Definición en la línea 177 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.32. PHY\_AUTONEGO\_COMPLETE

```
#define PHY_AUTONEGO_COMPLETE ((uint16_t)0x0020)
```

Auto-Negotiation process completed

Definición en la línea 213 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.33. PHY\_AUTONEGOTIATION

```
#define PHY_AUTONEGOTIATION ((uint16_t)0x1000)
```

Enable auto-negotiation function

Definición en la línea 208 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.34. PHY\_BCR

```
#define PHY_BCR ((uint16_t)0x00)
```

Transceiver Basic Control Register

Definición en la línea 199 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.35. PHY\_BSR

```
#define PHY_BSR ((uint16_t)0x01)
```

Transceiver Basic Status Register

Definición en la línea 200 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.36. PHY\_CONFIG\_DELAY

```
#define PHY_CONFIG_DELAY 0x00000FFFU
```

Definición en la línea 192 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.37. PHY\_DUPLEX\_STATUS

```
#define PHY_DUPLEX_STATUS ((uint16_t)0x0004U)
```

PHY Duplex mask

Definición en la línea 221 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.38. **PHY\_FULLDUPLEX\_100M**

```
#define PHY_FULLDUPLEX_100M ((uint16_t)0x2100)
```

Set the full-duplex mode at 100 Mb/s

Definición en la línea 204 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.39. **PHY\_FULLDUPLEX\_10M**

```
#define PHY_FULLDUPLEX_10M ((uint16_t)0x0100)
```

Set the full-duplex mode at 10 Mb/s

Definición en la línea 206 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.40. **PHY\_HALFDUPLEX\_100M**

```
#define PHY_HALFDUPLEX_100M ((uint16_t)0x2000)
```

Set the half-duplex mode at 100 Mb/s

Definición en la línea 205 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.41. **PHY\_HALFDUPLEX\_10M**

```
#define PHY_HALFDUPLEX_10M ((uint16_t)0x0000)
```

Set the half-duplex mode at 10 Mb/s

Definición en la línea 207 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.42. **PHY\_ISOLATE**

```
#define PHY_ISOLATE ((uint16_t)0x0400)
```

Isolate PHY from MII

Definición en la línea 211 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.43. **PHY\_JABBER\_DETECTION**

```
#define PHY_JABBER_DETECTION ((uint16_t)0x0002)
```

Jabber condition detected

Definición en la línea 215 del archivo [stm32f1xx\\_hal\\_conf.h](#).

#### 10.3.2.44. **PHY\_LINKED\_STATUS**

```
#define PHY_LINKED_STATUS ((uint16_t)0x0004)
```

Valid link established

Definición en la línea 214 del archivo [stm32f1xx\\_hal\\_conf.h](#).

#### 10.3.2.45. **PHY\_LOOPBACK**

```
#define PHY_LOOPBACK ((uint16_t)0x4000)
```

Select loop-back mode

Definición en la línea 203 del archivo [stm32f1xx\\_hal\\_conf.h](#).

#### 10.3.2.46. **PHY\_POWERDOWN**

```
#define PHY_POWERDOWN ((uint16_t)0x0800)
```

Select the power down mode

Definición en la línea 210 del archivo [stm32f1xx\\_hal\\_conf.h](#).

#### 10.3.2.47. **PHY\_READ\_TO**

```
#define PHY_READ_TO 0x0000FFFFU
```

Definición en la línea 194 del archivo [stm32f1xx\\_hal\\_conf.h](#).

#### 10.3.2.48. **PHY\_RESET**

```
#define PHY_RESET ((uint16_t)0x8000)
```

PHY Reset

Definición en la línea 202 del archivo [stm32f1xx\\_hal\\_conf.h](#).

#### 10.3.2.49. **PHY\_RESET\_DELAY**

```
#define PHY_RESET_DELAY 0x0000000FFU
```

Definición en la línea 190 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.50. PHY\_RESTART\_AUTONEGOTIATION

```
#define PHY_RESTART_AUTONEGOTIATION ((uint16_t)0x0200)
```

Restart auto-negotiation function

Definición en la línea 209 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.51. PHY\_SPEED\_STATUS

```
#define PHY_SPEED_STATUS ((uint16_t)0x0002U)
```

PHY Speed mask

Definición en la línea 220 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.52. PHY\_SR

```
#define PHY_SR ((uint16_t)0x10U)
```

PHY status register Offset

Definición en la línea 218 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.53. PHY\_WRITE\_TO

```
#define PHY_WRITE_TO 0x0000FFFFU
```

Definición en la línea 195 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.54. PREFETCH\_ENABLE

```
#define PREFETCH_ENABLE 1U
```

Definición en la línea 134 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.55. TICK\_INT\_PRIORITY

```
#define TICK_INT_PRIORITY 15U
```

tick interrupt priority (lowest by default)

Definición en la línea 132 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.56. USE\_HAL\_ADC\_REGISTER\_CALLBACKS

```
#define USE_HAL_ADC_REGISTER_CALLBACKS 0U /* ADC register callback disabled */
```

Definición en la línea 136 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.57. USE\_HAL\_CAN\_REGISTER\_CALLBACKS

```
#define USE_HAL_CAN_REGISTER_CALLBACKS 0U /* CAN register callback disabled */
```

Definición en la línea 137 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.58. USE\_HAL\_CEC\_REGISTER\_CALLBACKS

```
#define USE_HAL_CEC_REGISTER_CALLBACKS 0U /* CEC register callback disabled */
```

Definición en la línea 138 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.59. USE\_HAL\_DAC\_REGISTER\_CALLBACKS

```
#define USE_HAL_DAC_REGISTER_CALLBACKS 0U /* DAC register callback disabled */
```

Definición en la línea 139 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.60. USE\_HAL\_ETH\_REGISTER\_CALLBACKS

```
#define USE_HAL_ETH_REGISTER_CALLBACKS 0U /* ETH register callback disabled */
```

Definición en la línea 140 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.61. USE\_HAL\_HCD\_REGISTER\_CALLBACKS

```
#define USE_HAL_HCD_REGISTER_CALLBACKS 0U /* HCD register callback disabled */
```

Definición en la línea 141 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.62. USE\_HAL\_I2C\_REGISTER\_CALLBACKS

```
#define USE_HAL_I2C_REGISTER_CALLBACKS 0U /* I2C register callback disabled */
```

Definición en la línea 142 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.63. USE\_HAL\_I2S\_REGISTER\_CALLBACKS

```
#define USE_HAL_I2S_REGISTER_CALLBACKS 0U /* I2S register callback disabled */
```

Definición en la línea 143 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.64. USE\_HAL\_IRDA\_REGISTER\_CALLBACKS

```
#define USE_HAL_IRDA_REGISTER_CALLBACKS 0U /* IRDA register callback disabled */
```

Definición en la línea 152 del archivo [stm32f1xx\\_hal\\_conf.h](#).

**10.3.2.65. USE\_HAL\_MMC\_REGISTER\_CALLBACKS**

```
#define USE_HAL_MMC_REGISTER_CALLBACKS 0U /* MMC register callback disabled */
```

Definición en la línea 144 del archivo [stm32f1xx\\_hal\\_conf.h](#).

**10.3.2.66. USE\_HAL\_NAND\_REGISTER\_CALLBACKS**

```
#define USE_HAL_NAND_REGISTER_CALLBACKS 0U /* NAND register callback disabled */
```

Definición en la línea 145 del archivo [stm32f1xx\\_hal\\_conf.h](#).

**10.3.2.67. USE\_HAL\_NOR\_REGISTER\_CALLBACKS**

```
#define USE_HAL_NOR_REGISTER_CALLBACKS 0U /* NOR register callback disabled */
```

Definición en la línea 146 del archivo [stm32f1xx\\_hal\\_conf.h](#).

**10.3.2.68. USE\_HAL\_PCCARD\_REGISTER\_CALLBACKS**

```
#define USE_HAL_PCCARD_REGISTER_CALLBACKS 0U /* PCCARD register callback disabled */
```

Definición en la línea 147 del archivo [stm32f1xx\\_hal\\_conf.h](#).

**10.3.2.69. USE\_HAL\_PCD\_REGISTER\_CALLBACKS**

```
#define USE_HAL_PCD_REGISTER_CALLBACKS 0U /* PCD register callback disabled */
```

Definición en la línea 148 del archivo [stm32f1xx\\_hal\\_conf.h](#).

**10.3.2.70. USE\_HAL\_RTC\_REGISTER\_CALLBACKS**

```
#define USE_HAL_RTC_REGISTER_CALLBACKS 0U /* RTC register callback disabled */
```

Definición en la línea 149 del archivo [stm32f1xx\\_hal\\_conf.h](#).

**10.3.2.71. USE\_HAL\_SD\_REGISTER\_CALLBACKS**

```
#define USE_HAL_SD_REGISTER_CALLBACKS 0U /* SD register callback disabled */
```

Definición en la línea 150 del archivo [stm32f1xx\\_hal\\_conf.h](#).

**10.3.2.72. USE\_HAL\_SMARTCARD\_REGISTER\_CALLBACKS**

```
#define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U /* SMARTCARD register callback disabled */
```

Definición en la línea 151 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.73. USE\_HAL\_SPI\_REGISTER\_CALLBACKS

```
#define USE_HAL_SPI_REGISTER_CALLBACKS 0U /* SPI register callback disabled */
```

Definición en la línea 154 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.74. USE\_HAL\_SRAM\_REGISTER\_CALLBACKS

```
#define USE_HAL_SRAM_REGISTER_CALLBACKS 0U /* SRAM register callback disabled */
```

Definición en la línea 153 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.75. USE\_HAL\_TIM\_REGISTER\_CALLBACKS

```
#define USE_HAL_TIM_REGISTER_CALLBACKS 0U /* TIM register callback disabled */
```

Definición en la línea 155 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.76. USE\_HAL\_UART\_REGISTER\_CALLBACKS

```
#define USE_HAL_UART_REGISTER_CALLBACKS 0U /* UART register callback disabled */
```

Definición en la línea 156 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.77. USE\_HAL\_USART\_REGISTER\_CALLBACKS

```
#define USE_HAL_USART_REGISTER_CALLBACKS 0U /* USART register callback disabled */
```

Definición en la línea 157 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.78. USE\_HAL\_WWDG\_REGISTER\_CALLBACKS

```
#define USE_HAL_WWDG_REGISTER_CALLBACKS 0U /* WWDG register callback disabled */
```

Definición en la línea 158 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.79. USE\_RTOS

```
#define USE_RTOS 0U
```

Definición en la línea 133 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.80. USE\_SPI\_CRC

```
#define USE_SPI_CRC 0U
```

Definición en la línea 230 del archivo [stm32f1xx\\_hal\\_conf.h](#).

### 10.3.2.81. VDD\_VALUE

```
#define VDD_VALUE 3300U
```

This is the HAL system configuration section.

Value of VDD in mv

Definición en la línea 131 del archivo [stm32f1xx\\_hal\\_conf.h](#).

## 10.4. stm32f1xx\_hal\_conf.h

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

```
00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00020 /* Define to prevent recursive inclusion -----*/
00021 #ifndef __STM32F1XX_HAL_CONF_H
00022 #define __STM32F1XX_HAL_CONF_H
00023
00024 #ifdef __cplusplus
00025     extern "C" {
00026 #endif
00027
00028 /* Exported types -----*/
00029 /* Exported constants -----*/
00030
00031 /* ##### Module Selection ##### */
00035
00036 #define HAL_MODULE_ENABLED
00037 /*#define HAL_ADC_MODULE_ENABLED */
00038 /*#define HAL_CRYP_MODULE_ENABLED */
00039 /*#define HAL_CAN_MODULE_ENABLED */
00040 /*#define HAL_CAN_LEGACY_MODULE_ENABLED */
00041 /*#define HAL_CEC_MODULE_ENABLED */
00042 /*#define HAL_CORTEX_MODULE_ENABLED */
00043 /*#define HAL_CRC_MODULE_ENABLED */
00044 /*#define HAL_DAC_MODULE_ENABLED */
00045 #define HAL_DMA_MODULE_ENABLED
00046 /*#define HAL_ETH_MODULE_ENABLED */
00047 /*#define HAL_FLASH_MODULE_ENABLED */
00048 #define HAL_GPIO_MODULE_ENABLED
00049 /*#define HAL_I2C_MODULE_ENABLED */
00050 /*#define HAL_I2S_MODULE_ENABLED */
00051 /*#define HAL_IRDA_MODULE_ENABLED */
00052 /*#define HAL_IWDG_MODULE_ENABLED */
00053 /*#define HAL_NOR_MODULE_ENABLED */
00054 /*#define HAL_NAND_MODULE_ENABLED */
00055 /*#define HAL_PCCARD_MODULE_ENABLED */
00056 /*#define HAL_PCD_MODULE_ENABLED */
00057 /*#define HAL_HCD_MODULE_ENABLED */
00058 /*#define HAL_PWR_MODULE_ENABLED */
00059 /*#define HAL_RCC_MODULE_ENABLED */
00060 /*#define HAL_RTC_MODULE_ENABLED */
00061 /*#define HAL_SD_MODULE_ENABLED */
00062 /*#define HAL_MMC_MODULE_ENABLED */
00063 /*#define HAL_SDRAM_MODULE_ENABLED */
00064 /*#define HAL_SMARTCARD_MODULE_ENABLED */
00065 #define HAL_SPI_MODULE_ENABLED
00066 /*#define HAL_SRAM_MODULE_ENABLED */
00067 #define HAL_TIM_MODULE_ENABLED
00068 #define HAL_UART_MODULE_ENABLED
00069 /*#define HAL_USART_MODULE_ENABLED */
00070 /*#define HAL_WWDG_MODULE_ENABLED */
00071
00072 #define HAL_CORTEX_MODULE_ENABLED
00073 #define HAL_DMA_MODULE_ENABLED
00074 #define HAL_FLASH_MODULE_ENABLED
00075 #define HAL_EXTI_MODULE_ENABLED
00076 #define HAL_GPIO_MODULE_ENABLED
00077 #define HAL_PWR_MODULE_ENABLED
00078 #define HAL_RCC_MODULE_ENABLED
00079
00080 /* ##### Oscillator Values adaptation ##### */
00086 #if !defined (HSE_VALUE)
00087     #define HSE_VALUE    8000000U
```

```

00088 #endif /* HSE_VALUE */
00089
00090 #if !defined (HSE_STARTUP_TIMEOUT)
00091   #define HSE_STARTUP_TIMEOUT    100U
00092 #endif /* HSE_STARTUP_TIMEOUT */
00093
00099 #if !defined (HSI_VALUE)
00100   #define HSI_VALUE     8000000U
00101 #endif /* HSI_VALUE */
00102
00106 #if !defined (LSI_VALUE)
00107   #define LSI_VALUE      40000U
00108 #endif /* LSI_VALUE */
00111
00116 #if !defined (LSE_VALUE)
00117   #define LSE_VALUE     32768U
00118 #endif /* LSE_VALUE */
00119
00120 #if !defined (LSE_STARTUP_TIMEOUT)
00121   #define LSE_STARTUP_TIMEOUT 5000U
00122 #endif /* LSE_STARTUP_TIMEOUT */
00123
00124 /* Tip: To avoid modifying this file each time you need to use different HSE,
00125 === you can define the HSE value in your toolchain compiler preprocessor. */
00126
00127 /* ##### System Configuration #####
00131 #define VDD_VALUE           3300U
00132 #define TICK_INT_PRIORITY   15U
00133 #define USERTOS              0U
00134 #define PREFETCH_ENABLE      1U
00135
00136 #define USE_HAL_ADC_REGISTER_CALLBACKS      0U /* ADC register callback disabled */ */
00137 #define USE_HAL_CAN_REGISTER_CALLBACKS      0U /* CAN register callback disabled */ */
00138 #define USE_HAL_CEC_REGISTER_CALLBACKS      0U /* CEC register callback disabled */ */
00139 #define USE_HAL_DAC_REGISTER_CALLBACKS      0U /* DAC register callback disabled */ */
00140 #define USE_HAL_ETH_REGISTER_CALLBACKS      0U /* ETH register callback disabled */ */
00141 #define USE_HAL_HCD_REGISTER_CALLBACKS      0U /* HCD register callback disabled */ */
00142 #define USE_HAL_I2C_REGISTER_CALLBACKS      0U /* I2C register callback disabled */ */
00143 #define USE_HAL_I2S_REGISTER_CALLBACKS      0U /* I2S register callback disabled */ */
00144 #define USE_HAL_MMIC_REGISTER_CALLBACKS     0U /* MMC register callback disabled */ */
00145 #define USE_HAL_NAND_REGISTER_CALLBACKS     0U /* NAND register callback disabled */ */
00146 #define USE_HAL_NOR_REGISTER_CALLBACKS     0U /* NOR register callback disabled */ */
00147 #define USE_HAL_PCCARD_REGISTER_CALLBACKS   0U /* PCCARD register callback disabled */ */
00148 #define USE_HAL_PCD_REGISTER_CALLBACKS     0U /* PCD register callback disabled */ */
00149 #define USE_HAL_RTC_REGISTER_CALLBACKS     0U /* RTC register callback disabled */ */
00150 #define USE_HAL_SD_REGISTER_CALLBACKS      0U /* SD register callback disabled */ */
00151 #define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U /* SMARTCARD register callback disabled */ */
00152 #define USE_HAL_IRDA_REGISTER_CALLBACKS    0U /* IRDA register callback disabled */ */
00153 #define USE_HAL_SRAM_REGISTER_CALLBACKS    0U /* SRAM register callback disabled */ */
00154 #define USE_HAL_SPI_REGISTER_CALLBACKS    0U /* SPI register callback disabled */ */
00155 #define USE_HAL_TIM_REGISTER_CALLBACKS    0U /* TIM register callback disabled */ */
00156 #define USE_HAL_UART_REGISTER_CALLBACKS   0U /* UART register callback disabled */ */
00157 #define USE_HAL_USART_REGISTER_CALLBACKS   0U /* USART register callback disabled */ */
00158 #define USE_HAL_WWDG_REGISTER_CALLBACKS   0U /* WWDG register callback disabled */ */
00159
00160 /* ##### Assert Selection #####
00165 /* #define USE_FULL_ASSERT    1U */
00166
00167 /* ##### Ethernet peripheral configuration #####
00169 /* Section 1 : Ethernet peripheral configuration */
00170
00171 /* MAC ADDRESS: MAC_ADDR0:MAC_ADDR1:MAC_ADDR2:MAC_ADDR3:MAC_ADDR4:MAC_ADDR5 */
00172 #define MAC_ADDR0    2U
00173 #define MAC_ADDR1    0U
00174 #define MAC_ADDR2    0U
00175 #define MAC_ADDR3    0U
00176 #define MAC_ADDR4    0U
00177 #define MAC_ADDR5    0U
00178
00179 /* Definition of the Ethernet driver buffers size and count */
00180 #define ETH_RX_BUF_SIZE          ETH_MAX_PACKET_SIZE /* buffer size for receive */ */
00181 #define ETH_TX_BUF_SIZE          ETH_MAX_PACKET_SIZE /* buffer size for transmit */ */
00182 #define ETH_RXBUFNFB             8U    /* 4 Rx buffers of size ETH_RX_BUF_SIZE */
00183 #define ETH_TXBUFNFB             4U    /* 4 Tx buffers of size ETH_TX_BUF_SIZE */
00184
00185 /* Section 2: PHY configuration section */
00186
00187 /* DP83848_PHY_ADDRESS Address*/
00188 #define DP83848_PHY_ADDRESS      0x01U
00189 /* PHY Reset delay these values are based on a 1 ms Systick interrupt*/
00190 #define PHY_RESET_DELAY         0x0000000FFU
00191 /* PHY Configuration delay */
00192 #define PHY_CONFIG_DELAY        0x000000FFFU
00193
00194 #define PHY_READ_TO             0x0000FFFFU
00195 #define PHY_WRITE_TO            0x0000FFFFU

```

```

00196 /* Section 3: Common PHY Registers */
00197
00198 #define PHY_BCR ((uint16_t)0x00)
00199 #define PHY_BSR ((uint16_t)0x01)
00200
00201 #define PHY_RESET ((uint16_t)0x8000)
00202 #define PHY_LOOPBACK ((uint16_t)0x4000)
00203 #define PHY_FULLDUPLEX_100M ((uint16_t)0x2100)
00204 #define PHY_HALFDUPLEX_100M ((uint16_t)0x2000)
00205 #define PHY_FULLDUPLEX_10M ((uint16_t)0x0100)
00206 #define PHY_HALFDUPLEX_10M ((uint16_t)0x0000)
00207 #define PHY_AUTONEGOTIATION ((uint16_t)0x1000)
00208 #define PHY_RESTART_AUTONEGOTIATION ((uint16_t)0x0200)
00209 #define PHY_POWERDOWN ((uint16_t)0x0800)
00210 #define PHY_ISOLATE ((uint16_t)0x0400)
00211
00212 #define PHY_AUTONEGO_COMPLETE ((uint16_t)0x0020)
00213 #define PHY_LINKED_STATUS ((uint16_t)0x0004)
00214 #define PHY_JABBER_DETECTION ((uint16_t)0x0002)
00215
00216
00217 /* Section 4: Extended PHY Registers */
00218 #define PHY_SR ((uint16_t)0x10U)
00219
00220 #define PHY_SPEED_STATUS ((uint16_t)0x0002U)
00221 #define PHY_DUPLEX_STATUS ((uint16_t)0x0004U)
00222
00223 /* ##### SPI peripheral configuration ##### */
00224
00225 /* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver
00226 * Activated: CRC code is present inside driver
00227 * Deactivated: CRC code cleaned from driver
00228 */
00229
00230 #define USE_SPI_CRC 0U
00231
00232 /* Includes -----*/
00233
00234 #ifdef HAL_RCC_MODULE_ENABLED
00235 #include "stm32f1xx_hal_rcc.h"
00236 #endif /* HAL_RCC_MODULE_ENABLED */
00237
00238 #ifdef HAL_GPIO_MODULE_ENABLED
00239 #include "stm32f1xx_hal_gpio.h"
00240 #endif /* HAL_GPIO_MODULE_ENABLED */
00241
00242 #ifdef HAL_EXTI_MODULE_ENABLED
00243 #include "stm32f1xx_hal_exti.h"
00244 #endif /* HAL_EXTI_MODULE_ENABLED */
00245
00246 #ifdef HAL_DMA_MODULE_ENABLED
00247 #include "stm32f1xx_hal_dma.h"
00248 #endif /* HAL_DMA_MODULE_ENABLED */
00249
00250 #ifdef HAL_ETH_MODULE_ENABLED
00251 #include "stm32f1xx_hal_eth.h"
00252 #endif /* HAL_ETH_MODULE_ENABLED */
00253
00254 #ifdef HAL_CAN_MODULE_ENABLED
00255 #include "stm32f1xx_hal_can.h"
00256 #endif /* HAL_CAN_MODULE_ENABLED */
00257
00258 #ifdef HAL_CAN_LEGACY_MODULE_ENABLED
00259 #include "Legacy/stm32f1xx_hal_can_legacy.h"
00260 #endif /* HAL_CAN_LEGACY_MODULE_ENABLED */
00261
00262 #ifdef HAL_CEC_MODULE_ENABLED
00263 #include "stm32f1xx_hal_cec.h"
00264 #endif /* HAL_CEC_MODULE_ENABLED */
00265
00266 #ifdef HAL_CORTEX_MODULE_ENABLED
00267 #include "stm32f1xx_hal_cortex.h"
00268 #endif /* HAL_CORTEX_MODULE_ENABLED */
00269
00270 #ifdef HAL_ADC_MODULE_ENABLED
00271 #include "stm32f1xx_hal_adc.h"
00272 #endif /* HAL_ADC_MODULE_ENABLED */
00273
00274 #ifdef HAL_CRC_MODULE_ENABLED
00275 #include "stm32f1xx_hal_crc.h"
00276 #endif /* HAL_CRC_MODULE_ENABLED */
00277
00278 #ifdef HAL_DAC_MODULE_ENABLED
00279 #include "stm32f1xx_hal_dac.h"
00280 #endif /* HAL_DAC_MODULE_ENABLED */
00281
00282 #ifdef HAL_FLASH_MODULE_ENABLED

```

```
00286 #include "stm32f1xx_hal_flash.h"
00287 #endif /* HAL_FLASH_MODULE_ENABLED */
00288
00289 #ifdef HAL_SRAM_MODULE_ENABLED
00290 #include "stm32f1xx_hal_sram.h"
00291 #endif /* HAL_SRAM_MODULE_ENABLED */
00292
00293 #ifdef HAL_NOR_MODULE_ENABLED
00294 #include "stm32f1xx_hal_nor.h"
00295 #endif /* HAL_NOR_MODULE_ENABLED */
00296
00297 #ifdef HAL_I2C_MODULE_ENABLED
00298 #include "stm32f1xx_hal_i2c.h"
00299 #endif /* HAL_I2C_MODULE_ENABLED */
00300
00301 #ifdef HAL_I2S_MODULE_ENABLED
00302 #include "stm32f1xx_hal_i2s.h"
00303 #endif /* HAL_I2S_MODULE_ENABLED */
00304
00305 #ifdef HAL_IWDG_MODULE_ENABLED
00306 #include "stm32f1xx_hal_iwdg.h"
00307 #endif /* HAL_IWDG_MODULE_ENABLED */
00308
00309 #ifdef HAL_PWR_MODULE_ENABLED
00310 #include "stm32f1xx_hal_pwr.h"
00311 #endif /* HAL_PWR_MODULE_ENABLED */
00312
00313 #ifdef HAL_RTC_MODULE_ENABLED
00314 #include "stm32f1xx_hal_rtc.h"
00315 #endif /* HAL_RTC_MODULE_ENABLED */
00316
00317 #ifdef HAL_PCCARD_MODULE_ENABLED
00318 #include "stm32f1xx_hal_pccard.h"
00319 #endif /* HAL_PCCARD_MODULE_ENABLED */
00320
00321 #ifdef HAL_SD_MODULE_ENABLED
00322 #include "stm32f1xx_hal_sd.h"
00323 #endif /* HAL_SD_MODULE_ENABLED */
00324
00325 #ifdef HAL_NAND_MODULE_ENABLED
00326 #include "stm32f1xx_hal_nand.h"
00327 #endif /* HAL_NAND_MODULE_ENABLED */
00328
00329 #ifdef HAL_SPI_MODULE_ENABLED
00330 #include "stm32f1xx_hal_spi.h"
00331 #endif /* HAL_SPI_MODULE_ENABLED */
00332
00333 #ifdef HAL_TIM_MODULE_ENABLED
00334 #include "stm32f1xx_hal_tim.h"
00335 #endif /* HAL_TIM_MODULE_ENABLED */
00336
00337 #ifdef HAL_UART_MODULE_ENABLED
00338 #include "stm32f1xx_hal_uart.h"
00339 #endif /* HAL_UART_MODULE_ENABLED */
00340
00341 #ifdef HAL_USART_MODULE_ENABLED
00342 #include "stm32f1xx_hal_usart.h"
00343 #endif /* HAL_USART_MODULE_ENABLED */
00344
00345 #ifdef HAL_IRDA_MODULE_ENABLED
00346 #include "stm32f1xx_hal_irda.h"
00347 #endif /* HAL_IRDA_MODULE_ENABLED */
00348
00349 #ifdef HAL_SMARTCARD_MODULE_ENABLED
00350 #include "stm32f1xx_hal_smartcard.h"
00351 #endif /* HAL_SMARTCARD_MODULE_ENABLED */
00352
00353 #ifdef HAL_WWDG_MODULE_ENABLED
00354 #include "stm32f1xx_hal_wwdg.h"
00355 #endif /* HAL_WWDG_MODULE_ENABLED */
00356
00357 #ifdef HAL_PCD_MODULE_ENABLED
00358 #include "stm32f1xx_hal_pcd.h"
00359 #endif /* HAL_PCD_MODULE_ENABLED */
00360
00361 #ifdef HAL_HCD_MODULE_ENABLED
00362 #include "stm32f1xx_hal_hcd.h"
00363 #endif /* HAL_HCD_MODULE_ENABLED */
00364
00365 #ifdef HAL_MMC_MODULE_ENABLED
00366 #include "stm32f1xx_hal_mmc.h"
00367 #endif /* HAL_MMC_MODULE_ENABLED */
00368
00369 /* Exported macro -----*/
00370 #ifdef USE_FULL_ASSERT
00371 #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t *)__FILE__, __LINE__))
00380 /* Exported functions ----- */
```

```
00381 void assert_failed(uint8_t* file, uint32_t line);  
00382 #else  
00383 #define assert_param(expr) ((void)0U)  
00384 #endif /* USE_FULL_ASSERT */  
00385  
00386 #ifdef __cplusplus  
00387 }  
00388 #endif  
00389  
00390 #endif /* __STM32F1xx_HAL_CONF_H */  
00391
```

## 10.5. Referencia del archivo Inc/stm32f1xx\_it.h

This file contains the headers of the interrupt handlers.

### Funciones

- void **NMI\_Handler** (void)  
*This function handles Non maskable interrupt.*
- void **HardFault\_Handler** (void)  
*This function handles Hard fault interrupt.*
- void **MemManage\_Handler** (void)  
*This function handles Memory management fault.*
- void **BusFault\_Handler** (void)  
*This function handles Prefetch fault, memory access fault.*
- void **UsageFault\_Handler** (void)  
*This function handles Undefined instruction or illegal state.*
- void **SVC\_Handler** (void)  
*This function handles System service call via SWI instruction.*
- void **DebugMon\_Handler** (void)  
*This function handles Debug monitor.*
- void **PendSV\_Handler** (void)  
*This function handles Pendable request for system service.*
- void **SysTick\_Handler** (void)  
*This function handles System tick timer.*
- void **DMA1\_Channel4\_IRQHandler** (void)  
*This function handles DMA1 channel4 global interrupt.*
- void **DMA1\_Channel5\_IRQHandler** (void)  
*This function handles DMA1 channel5 global interrupt.*
- void **TIM2\_IRQHandler** (void)  
*This function handles TIM2 global interrupt.*
- void **TIM3\_IRQHandler** (void)  
*This function handles TIM3 global interrupt.*
- void **SPI2\_IRQHandler** (void)  
*This function handles SPI2 global interrupt.*

### 10.5.1. Descripción detallada

This file contains the headers of the interrupt handlers.

#### Atención

Copyright (c) 2025 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

Definición en el archivo [stm32f1xx\\_it.h](#).

### 10.5.2. Documentación de funciones

#### 10.5.2.1. BusFault\_Handler()

```
void BusFault_Handler (
    void )
```

This function handles Prefetch fault, memory access fault.

Definición en la línea [62](#) del archivo [stm32f1xx\\_it.c](#).

#### 10.5.2.2. DebugMon\_Handler()

```
void DebugMon_Handler (
    void )
```

This function handles Debug monitor.

Definición en la línea [89](#) del archivo [stm32f1xx\\_it.c](#).

#### 10.5.2.3. DMA1\_Channel4\_IRQHandler()

```
void DMA1_Channel4_IRQHandler (
    void )
```

This function handles DMA1 channel4 global interrupt.

Definición en la línea [110](#) del archivo [stm32f1xx\\_it.c](#).

#### 10.5.2.4. DMA1\_Channel5\_IRQHandler()

```
void DMA1_Channel5_IRQHandler (
    void )
```

This function handles DMA1 channel5 global interrupt.

Definición en la línea [117](#) del archivo [stm32f1xx\\_it.c](#).

### 10.5.2.5. HardFault\_Handler()

```
void HardFault_Handler (
    void )
```

This function handles Hard fault interrupt.

Definición en la línea [42](#) del archivo [stm32f1xx\\_it.c](#).

### 10.5.2.6. MemManage\_Handler()

```
void MemManage_Handler (
    void )
```

This function handles Memory management fault.

Definición en la línea [52](#) del archivo [stm32f1xx\\_it.c](#).

### 10.5.2.7. NMI\_Handler()

```
void NMI_Handler (
    void )
```

This function handles Non maskable interrupt.

Definición en la línea [32](#) del archivo [stm32f1xx\\_it.c](#).

### 10.5.2.8. PendSV\_Handler()

```
void PendSV_Handler (
    void )
```

This function handles Pendable request for system service.

Definición en la línea [96](#) del archivo [stm32f1xx\\_it.c](#).

### 10.5.2.9. SPI2\_IRQHandler()

```
void SPI2_IRQHandler (
    void )
```

This function handles SPI2 global interrupt.

Definición en la línea [139](#) del archivo [stm32f1xx\\_it.c](#).

### 10.5.2.10. SVC\_Handler()

```
void SVC_Handler (
    void )
```

This function handles System service call via SWI instruction.

Definición en la línea [82](#) del archivo [stm32f1xx\\_it.c](#).

#### 10.5.2.11. SysTick\_Handler()

```
void SysTick_Handler (
    void )
```

This function handles System tick timer.

Definición en la línea 103 del archivo [stm32f1xx\\_it.c](#).

#### 10.5.2.12. TIM2\_IRQHandler()

```
void TIM2_IRQHandler (
    void )
```

This function handles TIM2 global interrupt.

Definición en la línea 124 del archivo [stm32f1xx\\_it.c](#).

#### 10.5.2.13. TIM3\_IRQHandler()

```
void TIM3_IRQHandler (
    void )
```

This function handles TIM3 global interrupt.

Definición en la línea 132 del archivo [stm32f1xx\\_it.c](#).

#### 10.5.2.14. UsageFault\_Handler()

```
void UsageFault_Handler (
    void )
```

This function handles Undefined instruction or illegal state.

Definición en la línea 72 del archivo [stm32f1xx\\_it.c](#).

## 10.6. stm32f1xx\_it.h

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

```

00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00020 /* Define to prevent recursive inclusion -----*/
00021 #ifndef __STM32F1xx_IT_H
00022 #define __STM32F1xx_IT_H
00023
00024 #ifdef __cplusplus
00025 extern "C" {
00026 #endif
00027
00028 /* Private includes -----*/
00029 /* USER CODE BEGIN Includes */
00030
00031 /* USER CODE END Includes */
00032
00033 /* Exported types -----*/
00034 /* USER CODE BEGIN ET */
00035
00036 /* USER CODE END ET */
00037
00038 /* Exported constants -----*/
00039 /* USER CODE BEGIN EC */
00040
00041 /* USER CODE END EC */
00042
00043 /* Exported macro -----*/
00044 /* USER CODE BEGIN EM */
00045
00046 /* USER CODE END EM */
00047
00048 /* Exported functions prototypes -----*/
00049 void NMI_Handler(void);
00050 void HardFault_Handler(void);
00051 void MemManage_Handler(void);
00052 void BusFault_Handler(void);
00053 void UsageFault_Handler(void);
00054 void SVC_Handler(void);
00055 void DebugMon_Handler(void);
00056 void PendSV_Handler(void);
00057 void SysTick_Handler(void);
00058 void DMA1_Channel4_IRQHandler(void);
00059 void DMA1_Channel5_IRQHandler(void);
00060 void TIM2_IRQHandler(void);
00061 void TIM3_IRQHandler(void);
00062 void SPI2_IRQHandler(void);
00063 /* USER CODE BEGIN EFP */
00064
00065 /* USER CODE END EFP */
00066
00067 #ifdef __cplusplus
00068 }
00069 #endif
00070
00071 #endif /* __STM32F1xx_IT_H */

```

## 10.7. Referencia del archivo Modules/Gestor\_Estados/GestorEstados.c

Implementa la máquina de estados del LVFV.

```
#include "GestorEstados.h"
#include "../Gestor_SVM/GestorSVM.h"
```

### Funciones

- **SystemActionResponse GestorEstados\_Action (SystemAction sysAct, int value)**  
*Ejecuta una acción de la máquina de estados y devuelve el resultado.*

## Variables

- static SystemState currentState = STATE\_INIT

### 10.7.1. Descripción detallada

Implementa la máquina de estados del LVFV.

Ver [SystemState](#), [SystemAction](#) y [SystemActionResponse](#).

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

### 10.7.2. Documentación de funciones

#### 10.7.2.1. GestorEstados\_Action()

```
SystemActionResponse GestorEstados_Action (
    SystemAction sysAct,
    int value)
```

Ejecuta una acción de la máquina de estados y devuelve el resultado.

Evalúa la acción `sysAct` contra el estado actual ([SystemState](#)) y, según corresponda, invoca rutinas del SVM (p.ej. `GestorSVM_MotorStart()`, `GestorSVM_MotorStop()`, `GestorSVM_Estop()`, `GestorSVM_SetFrec/SetAcel/SetDecel/SetDir`) y actualiza la variable interna `currentState`.

Las reglas de transición/retorno (según el código mostrado) incluyen, entre otras:

- `ACTION_INIT_DONE` : solo desde `STATE_INIT` → `STATE_IDLE`.
- `ACTION_START` : desde `STATE_IDLE` → `STATE_VEL_CHANGE`; si hay movimiento → `ACTION_RESP_MOVING`; en emergencia → `ACTION_RESP_EMERGENCY_ACTIVE`.
- `ACTION_STOP` : desde `RUNNING/VEL_CHANGE` → `STATE_BRAKING`; en `EMERGENCY` → `STATE_IDLE`; en `IDLE` → `ACTION_RESP_NOT_MOVING`.
- `ACTION_MOTOR_STOPPED` : en `BRAKING` → `STATE_IDLE`.
- `ACTION_EMERGENCY` : a `STATE_EMERGENCY` (si no lo estaba).
- `ACTION_TO_CONST_RUNNING` : en `VEL_CHANGE` → `STATE_RUNNING`.
- `ACTION_SET_FREQ/SET_ACCEL/SET_DEACCEL/SET_DIR` : validan rango/estado y pueden dejar el estado o forzar `STATE_VEL_CHANGE` (según retorno de `GestorSVM_*`).

## Parámetros

in	<code>sysAct</code>	Acción solicitada ( <a href="#">SystemAction</a> ).
in	<code>value</code>	Parámetro asociado a la acción (p.ej., frecuencia, acel./desacel., dirección).

Devuelve

#### SystemActionResponse

- **ACTION\_RESP\_OK** : La acción fue válida y se ejecutó (posible cambio de estado).
- **ACTION\_RESP\_ERR** : Secuencia inválida para el estado actual o fallo de configuración.
- **ACTION\_RESP\_MOVING** : Rechazo/indicación de que el motor está en movimiento.
- **ACTION\_RESP\_NOT\_MOVING** : Rechazo/indicación de que el motor está detenido.
- **ACTION\_RESP\_OUT\_RANGE** : Parámetro value fuera de límites admitidos.
- **ACTION\_RESP\_EMERGENCY\_ACTIVE**: Acción no permitida bajo **STATE\_EMERGENCY**.

Nota

Las acciones de lectura (**ACTION\_GET\_FREC**, **ACTION\_GET\_ACCEL**, **ACTION\_GET\_DESACEL**, **ACTION\_GET\_DIR**) no cambian de estado en este switch.

Atención

Verificar que cada case finalice con break (en el código provisto faltaba break tras **ACTION\_EMERGENCY** para evitar fall-through).

Definición en la línea 12 del archivo [GestorEstados.c](#).

### 10.7.3. Documentación de variables

#### 10.7.3.1. currentState

```
SystemState currentState = STATE_INIT [static]
```

Definición en la línea 10 del archivo [GestorEstados.c](#).

## 10.8. GestorEstados.c

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

```
00001
00006
00007 #include "GestorEstados.h"
00008 #include "../Gestor_SVM/GestorSVM.h"
00009
00010 static SystemState currentState = STATE_INIT;
00011
00012 SystemActionResponse GestorEstados_Action(SystemAction sysAct, int value) {
00013     SystemActionResponse retVal = ACTION_RESP_ERR; // default seguro
00014     switch(sysAct) {
00015         case ACTION_INIT_DONE:
00016             if(currentState == STATE_INIT) {
00017                 currentState = STATE_IDLE;
00018                 retVal = ACTION_RESP_OK;
00019             } else {
00020                 retVal = ACTION_RESP_ERR;
00021             }
00022             break;
00023         case ACTION_TO_IDLE:
00024             if(currentState == STATE_BRAKING) {
00025                 currentState = STATE_IDLE;
00026                 retVal = ACTION_RESP_OK;
00027             } else {
00028                 retVal = ACTION_RESP_ERR;
00029             }
00030     }
00031 }
```

```

00030         break;
00031     case ACTION_START:
00032         if(currentState == STATE_IDLE) {
00033             GestorSVM_MotorStart();
00034             currentState = STATE_VEL_CHANGE;
00035             retVal = ACTION_RESP_OK;
00036         } else if(currentState == STATE_RUNNING || currentState == STATE_VEL_CHANGE || currentState
00037 == STATE_BRAKING) {
00038             retVal = ACTION_RESP_MOVING;
00039         } else if(currentState == STATE_EMERGENCY) {
00040             retVal = ACTION_RESP_EMERGENCY_ACTIVE;
00041         } else {
00042             retVal = ACTION_RESP_ERR;
00043         }
00044         break;
00045     case ACTION_STOP:
00046         if(currentState == STATE_RUNNING || currentState == STATE_VEL_CHANGE) {
00047             GestorSVM_MotorStop();
00048             currentState = STATE_BRAKING;
00049             retVal = ACTION_RESP_OK;
00050         } else if(currentState == STATE_EMERGENCY) {
00051             GestorSVM_Estop();
00052             currentState = STATE_IDLE;
00053             retVal = ACTION_RESP_OK;
00054         } else if (currentState == STATE_IDLE) {
00055             retVal = ACTION_RESP_NOT_MOVING;
00056         } else {
00057             retVal = ACTION_RESP_ERR;
00058         }
00059         break;
00060     case ACTION_MOTOR_STOPPED:
00061         if(currentState == STATE_BRAKING) {
00062             currentState = STATE_IDLE;
00063             retVal = ACTION_RESP_OK;
00064         } else {
00065             retVal = ACTION_RESP_ERR;
00066         }
00067         break;
00068     case ACTION_EMERGENCY:
00069         if(currentState != STATE_EMERGENCY) {
00070             GestorSVM_Estop();
00071             currentState = STATE_EMERGENCY;
00072             retVal = ACTION_RESP_OK;
00073         } else {
00074             retVal = ACTION_RESP_ERR;
00075         }
00076         break;
00077     case ACTION_TO_CONST_RUNNING:
00078         if(currentState == STATE_VEL_CHANGE) {
00079             currentState = STATE_RUNNING;
00080             retVal = ACTION_RESP_OK;
00081         } else {
00082             retVal = ACTION_RESP_ERR;
00083         }
00084         break;
00085     case ACTION_SET_FREC:
00086         if(currentState == STATE_IDLE || currentState == STATE_RUNNING || currentState ==
00087 STATE_VEL_CHANGE) {
00088             int conf_retVal = GestorSVM_SetFrec(value);
00089             if(conf_retVal == 0 || conf_retVal == -2) {
00090                 retVal = ACTION_RESP_OK;
00091             } else if(conf_retVal == 1) {
00092                 currentState = STATE_VEL_CHANGE;
00093                 retVal = ACTION_RESP_OK;
00094             } else if(conf_retVal == -1) {
00095                 retVal = ACTION_RESP_OUT_RANGE;
00096             }
00097         } else if( currentState == STATE_BRAKING) {
00098             retVal = ACTION_RESP_MOVING;
00099         } else {
00100             retVal = ACTION_RESP_ERR;
00101         }
00102         break;
00103     case ACTION_SET_ACCEL:
00104         // Implementar la logica para cambiar la aceleracion
00105         if(currentState == STATE_IDLE || currentState == STATE_RUNNING) {
00106             switch( GestorSVM_SetAcel(value) ) {
00107                 case 0:
00108                     retVal = ACTION_RESP_OK;
00109                     break;
00110                 case -1:
00111                     retVal = ACTION_RESP_OUT_RANGE;
00112                     break;
00113                 case -2:
00114                     retVal = ACTION_RESP_ERR;
00115                     break;
00116             }
00117         }
00118     default:
00119         break;
00120     }
00121 }
```

```

00115         }
00116     } else {
00117         retVal = ACTION_RESP_ERR;
00118     }
00119     break;
00120 case ACTION_SET_DESACEL:
00121     if(currentState == STATE_IDLE || currentState == STATE_RUNNING) {
00122         switch( GestorSVM_SetDecel(value) ) {
00123             case 0:
00124                 retVal = ACTION_RESP_OK;
00125                 break;
00126             case -1:
00127                 retVal = ACTION_RESP_OUT_RANGE;
00128                 break;
00129             case -2:
00130             default:
00131                 retVal = ACTION_RESP_ERR;
00132                 break;
00133         }
00134     } else {
00135         retVal = ACTION_RESP_ERR;
00136     }
00137     break;
00138 case ACTION_SET_DIR:
00139     if(currentState == STATE_IDLE) {
00140         switch( GestorSVM_SetDir(value)) {
00141             case 0:
00142                 retVal = ACTION_RESP_OK;
00143                 break;
00144             case -1:
00145                 retVal = ACTION_RESP_OUT_RANGE;
00146                 break;
00147             case -2:
00148                 retVal = ACTION_RESP_MOVING;
00149                 break;
00150             default:
00151                 retVal = ACTION_RESP_ERR;
00152                 break;
00153         }
00154     } else {
00155         retVal = ACTION_RESP_MOVING;
00156     }
00157     break;
00158 // case ACTION_GET_FREC:
00159 // case ACTION_GET_ACCEL:
00160 // case ACTION_GET_DESACEL:
00161 // case ACTION_GET_DIR:
00162 //     retVal = ACTION_RESP_OK;      // lectura válida, sin transición
00163 //     break;
00164 case ACTION_IS_MOTOR_STOP:
00165     if(currentState == STATE_IDLE || currentState == STATE_EMERGENCY) {
00166         retVal = ACTION_RESP_NOT_MOVING;
00167     } else {
00168         retVal = ACTION_RESP_MOVING;
00169     }
00170     break;
00171 default:
00172     retVal = ACTION_RESP_ERR;
00173     break;
00174 }
00175 return retVal;
00176 }

```

## 10.9. Referencia del archivo Modules/Gestor\_Estados/GestorEstados.h

### Enumeraciones

- enum **SystemState** {
   
STATE\_INIT = 0 , STATE\_IDLE , STATE\_RUNNING , STATE\_VEL\_CHANGE ,
   
STATE\_BRAKING , STATE\_EMERGENCY }

*Estados de la máquina de estados del variador.*

- enum **SystemAction** {
   
ACTION\_NONE = 0 , ACTION\_INIT\_DONE = 1 , ACTION\_TO\_IDLE , ACTION\_START ,
   
ACTION\_STOP , ACTION\_MOTOR\_STOPPED , ACTION\_EMERGENCY , ACTION\_SET\_FREC ,
   
ACTION\_SET\_ACCEL , ACTION\_SET\_DESACEL , ACTION\_SET\_DIR , ACTION\_TO\_CONST\_RUNNING ,
   
ACTION\_GET\_FREC , ACTION\_GET\_ACCEL , ACTION\_GET\_DESACEL , ACTION\_GET\_DIR ,
   
ACTION\_IS\_MOTOR\_STOP }

*Acciones externas/internas que gobiernan la máquina de estados del VFD.*

- enum SystemActionResponse {
 ACTION\_RESP\_OK = 0, ACTION\_RESP\_ERR = 1, ACTION\_RESP\_MOVING = 2, ACTION\_RESP\_NOT\_MOVING = 3,
 ACTION\_RESP\_OUT\_RANGE = 4, ACTION\_RESP\_EMERGENCY\_ACTIVE = 5
 }

*Códigos de respuesta del gestor de estados ante una acción recibida.*

## Funciones

- [SystemActionResponse GestorEstados\\_Action \(SystemAction sysAct, int value\)](#)

*Ejecuta una acción de la máquina de estados y devuelve el resultado.*

### 10.9.1. Documentación de enumeraciones

#### 10.9.1.1. SystemAction

enum [SystemAction](#)

Acciones externas/internas que gobiernan la máquina de estados del VFD.

Cada acción puede provocar un cambio de estado ([SystemState](#)) y devuelve un [SystemActionResponse](#) según las reglas implementadas en [GestorEstados\\_Action\(\)](#). Las referencias a estados (STATE\_\*) y respuestas (ACTION↔\_RESP\_\*) indican el flujo esperado y los códigos de retorno típicos.

#### Valores de enumeraciones

ACTION_NONE	Sin uso. No debe enviarse.
ACTION_INIT_DONE	Fin de inicialización. Válida <b>solo</b> si currentState==@ref STATE_INIT: pasa a <a href="#">STATE_IDLE</a> y responde ACTION_RESP_OK; en cualquier otro estado responde ACTION_RESP_ERR.
ACTION_TO_IDLE	Forzar estado inactivo. Válida cuando currentState==@ref STATE_BRAKING: pasa a <a href="#">STATE_IDLE</a> y responde ACTION_RESP_OK; si no está frenando, responde ACTION_RESP_ERR.
ACTION_START	Solicitud de arranque. Si currentState==@ref STATE_IDLE llama a <a href="#">GestorSVM_MotorStart()</a> , pasa a <a href="#">STATE_VEL_CHANGE</a> y responde ACTION_RESP_OK. Si está en <a href="#">STATE_RUNNING</a> , <a href="#">STATE_VEL_CHANGE</a> o <a href="#">STATE_BRAKING</a> responde ACTION_RESP_MOVING. En <a href="#">STATE_EMERGENCY</a> responde ACTION_RESP_EMERGENCY_ACTIVE. En cualquier otro caso ACTION_RESP_ERR.
ACTION_STOP	Solicitud de parada. Si está en <a href="#">STATE_RUNNING</a> o <a href="#">STATE_VEL_CHANGE</a> , llama a <a href="#">GestorSVM_MotorStop()</a> , pasa a <a href="#">STATE_BRAKING</a> y responde ACTION_RESP_OK. Si está en <a href="#">STATE_EMERGENCY</a> , ejecuta <a href="#">GestorSVM_Estop()</a> , pasa a <a href="#">STATE_IDLE</a> y responde ACTION_RESP_OK. Si ya está en <a href="#">STATE_IDLE</a> responde ACTION_RESP_NOT_MOVING. Otro caso: ACTION_RESP_ERR.

ACTION_MOTOR_STOPPED	Confirmación de motor detenido. Válida durante STATE_BRAKING: pasa a STATE_IDLE y responde ACTION_RESP_OK; si no está frenando responde ACTION_RESP_ERR.
ACTION_EMERGENCY	Activación de emergencia. Si no está en STATE_EMERGENCY, ejecuta GestorSVM_Estop(), pasa a STATE_EMERGENCY y responde ACTION_RESP_OK; si ya estaba en emergencia responde ACTION_RESP_ERR.
ACTION_SET_FREC	Fijar frecuencia de régimen. Permitido en STATE_IDLE, STATE_RUNNING o STATE_VEL_CHANGE. Llama a GestorSVM_SetFrec(value): <ul style="list-style-type: none"> <li>• 0 o -2 → ACTION_RESP_OK (permanece en el estado actual)</li> <li>• 1 → pasa a STATE_VEL_CHANGE y ACTION_RESP_OK</li> <li>• -1 → ACTION_RESP_OUT_RANGE Si está en STATE_BRAKING → ACTION_RESP_MOVING; otro estado → ACTION_RESP_ERR.</li> </ul>
ACTION_SET_ACCEL	Configurar aceleración. Permitido en STATE_IDLE o STATE_RUNNING. Llama a GestorSVM_SetAcel(value): <ul style="list-style-type: none"> <li>• 0 → ACTION_RESP_OK</li> <li>• -1 → ACTION_RESP_OUT_RANGE</li> <li>• -2 u otro → ACTION_RESP_ERR En otros estados → ACTION_RESP_ERR.</li> </ul>
ACTION_SET_DESACCEL	Configurar desaceleración. Igual a ACTION_SET_ACCEL pero llamando GestorSVM_SetDecel(value). Respuestas: ACTION_RESP_OK, ACTION_RESP_OUT_RANGE o ACTION_RESP_ERR. En otros estados → ACTION_RESP_ERR.
ACTION_SET_DIR	Configurar dirección de giro. Permitido solo en STATE_IDLE. Llama a GestorSVM_SetDir(value): <ul style="list-style-type: none"> <li>• 0 → ACTION_RESP_OK</li> <li>• -1 → ACTION_RESP_OUT_RANGE</li> <li>• -2 → ACTION_RESP_MOVING</li> <li>• otro → ACTION_RESP_ERR Si no está en IDLE → ACTION_RESP_MOVING.</li> </ul>
ACTION_TO_CONST_RUNNING	Transición a régimen constante. Válida cuando currentState==@ref STATE_VEL_CHANGE: pasa a STATE_RUNNING y responde ACTION_RESP_OK; caso contrario ACTION_RESP_ERR.
ACTION_GET_FREC	Consulta de frecuencia actual. Acción de lectura (sin cambio de estado). La entrega del valor se realiza por la capa que invoca a GestorEstados_Action() (no hay respuesta numérica directa en este switch).
ACTION_GET_ACCEL	Consulta de aceleración actual. Acción de lectura (sin cambio de estado). La obtención del valor queda a cargo de la capa llamadora.
ACTION_GET_DESACCEL	Consulta de desaceleración actual. Acción de lectura (sin cambio de estado). La obtención del valor queda a cargo de la capa llamadora.
ACTION_GET_DIR	Consulta de dirección actual. Acción de lectura (sin cambio de estado). La obtención del valor queda a cargo de la capa llamadora.

ACTION_IS_MOTOR_STOP	¿El motor está detenido? Si <code>currentState==@ref STATE_IDLE</code> o <code>STATE_EMERGENCY</code> → ACTION_RESP_NOT_MOVING; en otro estado → ACTION_RESP_MOVING.
----------------------	--

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

#### 10.9.1.2. SystemActionResponse

enum `SystemActionResponse`

Códigos de respuesta del gestor de estados ante una acción recibida.

Los valores indican el resultado de `GestorEstados_Action()` según la acción solicitada (`SystemAction`) y el estado actual (`SystemState`). Se utilizan también como payload de respuesta en el enlace SPI con el ESP32.

##### Valores de enumeraciones

ACTION_RESP_OK	
ACTION_RESP_ERR	La acción fue válida y se ejecutó correctamente. Si correspondía, la transición de estado se realizó.
ACTION_RESP_MOVING	Error genérico: secuencia inválida para el estado actual o fallo al aplicar la configuración. No hay garantía de cambio de estado.
ACTION_RESP_NOT_MOVING	El motor está en movimiento (p. ej., en <code>STATE_RUNNING</code> , <code>STATE_VEL_CHANGE</code> o <code>STATE_BRAKING</code> ). Puede usarse como respuesta a una consulta o para rechazar acciones no permitidas mientras hay movimiento.
ACTION_RESP_OUT_RANGE	El motor está detenido (p. ej., en <code>STATE_IDLE</code> o <code>STATE_EMERGENCY</code> ). Puede ser respuesta a consulta o a un STOP redundante.
ACTION_RESP_EMERGENCY_ACTIVE	El parámetro de configuración recibido (frecuencia, aceleración, etc.) está fuera de los límites admitidos por la lógica (p. ej., <code>GestorSVM_*</code> ).

Definición en la línea 212 del archivo [GestorEstados.h](#).

#### 10.9.1.3. SystemState

enum `SystemState`

Estados de la máquina de estados del variador.

Representan el ciclo de vida del sistema y condicionan qué acciones (`SystemAction`) son válidas y qué respuestas (`SystemActionResponse`) devuelve `GestorEstados_Action()`. Las transiciones típicas son:

- `STATE_INIT` → ( `ACTION_INIT_DONE` ) → `STATE_IDLE`
- `STATE_IDLE` → ( `ACTION_START` ) → `STATE_VEL_CHANGE`
- `STATE_VEL_CHANGE` → ( `ACTION_TO_CONST_RUNNING` ) → `STATE_RUNNING`
- `STATE_RUNNING` → ( `ACTION_STOP` ) → `STATE_BRAKING`
- `STATE_BRAKING` → ( `ACTION_MOTOR_STOPPED` ) → `STATE_IDLE`
- Cualquier estado → ( `ACTION_EMERGENCY` ) → `STATE_EMERGENCY`

##### Valores de enumeraciones

STATE_INIT	
STATE_IDLE	Sistema en inicialización. Solo es válida ACTION_INIT_DONE para pasar a STATE_IDLE.
STATE_RUNNING	Reposo: motor detenido (o salida de emergencia). Admite configuración (ACTION_SET_FREC, ACTION_SET_ACCEL, ACTION_SET_DESACCEL, ACTION_SET_DIR) y arranque con ACTION_START.
STATE_VEL_CHANGE	Marcha a velocidad constante. Acepta cambio de parámetros permitidos y ACTION_STOP para transicionar a STATE_BRAKING.
STATE_BRAKING	Transitorio de cambio de velocidad (acel./desacel.). Al alcanzar régimen : ACTION_TO_CONST_RUNNING → STATE_RUNNING. ACTION_STOP lleva a STATE_BRAKING.
STATE_EMERGENCY	Frenado en curso. Al detenerse: ACTION_MOTOR_STOPPED → STATE_IDLE. ACTION_TO_IDLE fuerza el paso a inactivo. Emergencia activa (salidas deshabilitadas). Solo acciones de recuperación/acondicionamiento; p.ej. ACTION_STOP puede llevar a STATE_IDLE según la lógica de seguridad.

Definición en la línea 25 del archivo [GestorEstados.h](#).

## 10.9.2. Documentación de funciones

### 10.9.2.1. GestorEstados\_Action()

```
SystemActionResponse GestorEstados_Action (
    SystemAction sysAct,
    int value)
```

Ejecuta una acción de la máquina de estados y devuelve el resultado.

Evalúa la acción sysAct contra el estado actual ([SystemState](#)) y, según corresponda, invoca rutinas del SVM (p.ej. [GestorSVM\\_MotorStart\(\)](#), [GestorSVM\\_MotorStop\(\)](#), [GestorSVM\\_Estop\(\)](#), [GestorSVM\\_SetFrec](#)/[SetAcel](#)/[SetDecel](#)/[SetDir](#)) y actualiza la variable interna [currentState](#).

Las reglas de transición/retorno (según el código mostrado) incluyen, entre otras:

- ACTION\_INIT\_DONE : solo desde STATE\_INIT → STATE\_IDLE.
- ACTION\_START : desde STATE\_IDLE → STATE\_VEL\_CHANGE; si hay movimiento → ACTION\_RESP\_MOVING; en emergencia → ACTION\_RESP\_EMERGENCY\_ACTIVE.
- ACTION\_STOP : desde RUNNING/VEL\_CHANGE → STATE\_BRAKING; en EMERGENCY → STATE\_IDLE; en IDLE → ACTION\_RESP\_NOT\_MOVING.
- ACTION\_MOTOR\_STOPPED : en BRAKING → STATE\_IDLE.
- ACTION\_EMERGENCY : a STATE\_EMERGENCY (si no lo estaba).
- ACTION\_TO\_CONST\_RUNNING : en VEL\_CHANGE → STATE\_RUNNING.
- ACTION\_SET\_FREC/SET\_ACCEL/SET\_DESACCEL/SET\_DIR : validan rango/estado y pueden dejar el estado o forzar STATE\_VEL\_CHANGE (según retorno de [GestorSVM\\_\\*](#)).

### Parámetros

---

in	<code>sysAct</code>	Acción solicitada ( <a href="#">SystemAction</a> ).
in	<code>value</code>	Parámetro asociado a la acción (p.ej., frecuencia, acel./desacel., dirección).

Devuelve

[SystemActionResponse](#)

- [ACTION\\_RESP\\_OK](#) : La acción fue válida y se ejecutó (posible cambio de estado).
- [ACTION\\_RESP\\_ERR](#) : Secuencia inválida para el estado actual o fallo de configuración.
- [ACTION\\_RESP\\_MOVING](#) : Rechazo/indicación de que el motor está en movimiento.
- [ACTION\\_RESP\\_NOT\\_MOVING](#) : Rechazo/indicación de que el motor está detenido.
- [ACTION\\_RESP\\_OUT\\_RANGE](#) : Parámetro `value` fuera de límites admitidos.
- [ACTION\\_RESP\\_EMERGENCY\\_ACTIVE](#): Acción no permitida bajo [STATE\\_EMERGENCY](#).

Nota

Las acciones de lectura ([ACTION\\_GET\\_FREC](#), [ACTION\\_GET\\_ACCEL](#), [ACTION\\_GET\\_DESACEL](#), [ACTION\\_GET\\_DIR](#)) no cambian de estado en este switch.

Atención

Verificar que cada `case` finalice con `break` (en el código provisto faltaba `break` tras [ACTION\\_EMERGENCY](#) para evitar fall-through).

Definición en la línea 12 del archivo [GestorEstados.c](#).

## 10.10. GestorEstados.h

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

```

00001 /*
00002  * GestorEstados.h
00003 *
00004  * Created on: Aug 10, 2025
00005  * Author: elian
00006 */
00007
00008 #ifndef MODULES_GESTOR_ESTADOS_GESTORESTADOS_H_
00009 #define MODULES_GESTOR_ESTADOS_GESTORESTADOS_H_
00010
00025 typedef enum {
00026     STATE_INIT = 0,
00028
00029     STATE_IDLE,
00034
00035     STATE_RUNNING,
00038
00039     STATE_VEL_CHANGE,
00043
00044     STATE_BRAKING,
00047
00048     STATE_EMERGENCY
00052 } SystemState;
00053
00054
00064 typedef enum {
00065     ACTION_NONE = 0,
00066
00073     ACTION_INIT_DONE = 1,
00074
00081     ACTION_TO_IDLE,

```

```

00082
00091     ACTION_START,
00092
00101     ACTION_STOP,
00102
00108     ACTION_MOTOR_STOPPED,
00109
00116     ACTION_EMERGENCY,
00117
00127     ACTION_SET_FREC,
00128
00138     ACTION_SET_ACCEL,
00139
00146     ACTION_SET_DESACEL,
00147
00157     ACTION_SET_DIR,
00158
00165     ACTION_TO_CONST_RUNNING,
00166
00173     ACTION_GET_FREC,
00174
00180     ACTION_GET_ACCEL,
00181
00187     ACTION_GET_DESACEL,
00188
00194     ACTION_GET_DIR,
00195
00201     ACTION_IS_MOTOR_STOP,
00202 } SystemAction;
00203
00212 typedef enum {
00213     ACTION_RESP_OK = 0,
00215
00216     ACTION_RESP_ERR = 1,
00219
00220     ACTION_RESP_MOVING = 2,
00225
00226     ACTION_RESP_NOT_MOVING = 3,
00229
00230     ACTION_RESP_OUT_RANGE = 4,
00233
00234     ACTION_RESP_EMERGENCY_ACTIVE = 5
00236 } SystemActionResponse;
00237
00238
00273 SystemActionResponse GestorEstados_Action(SystemAction sysAct, int value);
00274
00275
00276 #endif /* MODULES_GESTOR_ESTADOS_GESTORESTADOS_H_ */

```

## 10.11. Referencia del archivo Modules/Gestor\_SVM/GestorSVM.c

```

#include <stdio.h>
#include "GestorSVM.h"
#include "stm32f103xb.h"
#include "../Inc/main.h"
#include "../Gestor_Estados/GestorEstados.h"
#include "../Gestor_Timers/GestorTimers.h"

```

### Estructuras de datos

- struct **DatoCalculado**
- struct **Parametros**

*Parámetros “sombra” para sincronización atómica con el lazo de cálculo.*

**defines**

- `#define MAX_TICKS 256`  
ARR efectivo del timer de switching (resolución 256 pasos).
- `#define TICKS_DESHABILITAR_CANAL 280`  
Valor mayor a ARR para forzar que un CCR nunca dispare (deshabilitar canal por "posición imposible").
- `#define MIN_TICKS_DIF 5`  
Mínima separación (en ticks) entre eventos para considerar que no hay interferencia entre t1/t2/t3.
- `#define BUFFER_CALCULO_SIZE 3`  
Buffer circular (productor: timer de cálculo; consumidor: timer de switching).

**Constantes de cálculo t1/t2 por regresión lineal**

*t1 y t2 se obtienen con una regresión lineal en función del ángulo parcial y el índice de modulación.*

- `#define CONST_CALC_T1_PROP (float)-3.59145E-6`
- `#define CONST_CALC_T1_ORD_ORG (float)224.2845426`
- `#define CONST_CALC_T2_PROP (float)+3.59145E-6`
- `#define CONST_CALC_T2_ORD_ORG (float)8.797345358`

**Enumeraciones**

- enum `CasoInterferenciaTimer {`
- `INTERF_NULA = 0 , INTERF_T1T2 , INTERF_T1T3 , INTERF_T2T3 ,`
- `INTERF_T1T2_T2T3 }`

*Casos de interferencia temporal entre eventos t1/t2/t3 (demasiado cercanos).*

**Funciones**

- static void `GestorSVM_CalcuLoaceleracioneracion (void)`  
*Aplica la rampa de velocidad (aceleracion/desaceleracion) sobre frecuenciaSalida y actualiza flags/estado.*
- static int `pinMap (int x)`  
*Convierte el XOR de estados (bit U/V/W) al índice interno {0,1,2}.*
- static void `GestorSVM_CalcularValoresSwitching (void)`  
*Calcula t1, t2 y t0 (y casos de interferencia) y los deja en ticksChannel[].*
- static void `GestorSVM_SwitchInterrupt (SwitchInterruptType intType)`  
*Handler interno llamado por el ISR de TIM3 según la fuente (CH1..CH4, RESET, CLEAN).*
- static void `GestorSVM_SwitchPuertos (OrdenSwitch orden, char estado, int numCuadrante)`  
*Escribe en BSRR los pines correspondientes a la orden y cuadrante.*
- void `GestorSVM_SetConfiguration (ConfiguracionSVM *configuracion)`  
*Carga la configuración base de SVM y prepara tablas de BSRR por cuadrante.*
- void `GestorSVM_CalcInterrupt ()`  
*Handler llamado por el timer de cálculo (productor). Encola t1/t2/t3 si hay espacio.*
- int `GestorSVM_SetFrec (int frec)`  
*Solicita nueva frecuencia objetivo (Hz). Inicia rampa si el motor está en marcha.*
- int `GestorSVM_SetDir (int dir)`  
*Cambia el sentido de giro si el motor está detenido.*
- int `GestorSVM_Setaceleracion (int nuevaaceleracion)`  
*Actualiza acelerada [Hz/s].*
- int `GestorSVM_SetDecel (int nuevaDecel)`  
*Actualiza desacelerada [Hz/s].*
- int `GestorSVM_MotorStart ()`

- Arranca el motor: habilita drivers, inicia timers y rampa hacia [frecuenciaReferencia](#).*
- int [GestorSVM\\_MotorStop](#) ()
 

*Ordena frenado con rampa de [desaceleracion](#) hasta 0 Hz.*
  - int [GestorSVM\\_Estop](#) ()
 

*Parada de emergencia inmediata: desactiva timers, apaga drivers y salidas.*
  - int [GestorSVM\\_GetFrec](#) ()
 

*Devuelve la frecuencia de referencia configurada (Hz).*
  - int [GestorSVM\\_Getaceleracion](#) ()
 

*Devuelve acelerada actual [Hz/s].*
  - int [GestorSVM\\_GetDesaceleracion](#) ()
 

*Devuelve desacelerada actual [Hz/s].*
  - int [GestorSVM\\_GetDir](#) ()
 

*Devuelve sentido de giro (1 horario, -1 antihorario).*
  - void [HAL\\_TIM\\_OC\\_DelayElapsedCallback](#) (TIM\_HandleTypeDef \*htim)
 

*Callback HAL de Output Compare: enruta eventos CCR de TIM3 a [GestorSVM\\_SwitchInterrupt](#).*

## Variables

- static int [frecuenciaSwitching](#)

*Frecuencia de switching [Hz] (TIM3).*
- static int [direccionRotacion](#) = 1
 

*Sentido de rotación: 1 horario, -1 antihorario.*
- static int32\_t [frecuenciaSalida](#)

*Frecuencia de salida INSTANTÁNEA (escalada  $\times 1e6$ ) usada por el lazo de rampa.*
- static uint32\_t [anguloSwitching](#)

*Incremento angular por switching (grados $\times 1e3$ ).*
- static volatile int [indiceModulacion](#)

*Índice de modulación (0..100). Controla la tensión de salida. La relación v/f debe ser constante.*
- static int [aceleracion](#)

*acelerada configurada [Hz/s].*
- static int [desaceleracion](#)

*Desacelerada configurada [Hz/s].*
- static uint32\_t [anguloActual](#)

*Ángulo absoluto (grados $\times 1e3$ ).*
- static int32\_t [anguloParcial](#)

*Ángulo parcial 0..60°(grados $\times 1e3$ ), usado para t1/t2. Puede ser negativo durante el diente.*
- static volatile int [cuadranteActual](#)

*Cuadrante SVM actual (0..5).*
- const int [vectorSecuenciaPorCuadrante](#) [6][7]
 

*Secuencia SVM por cuadrante ( $V0 \rightarrow V1 \rightarrow V2 \rightarrow V7 \rightarrow V2 \rightarrow V1 \rightarrow V0$ ).*
- int [pinTogglePorCuadranteYOrden](#) [6][3]
 

*Mapa de qué pin conmuta en cada orden y cuadrante.*
- uint32\_t [estadoGPIOPorCuadranteYOrden](#) [6][2]
 

*Palabras BSRR precalculadas por cuadrante/estado (0/1).*
- uint32\_t [estadoGPIOOff](#)

*BSRR para apagar U/V/W simultáneo.*
- uint32\_t [estadoGPIOOn](#)

*BSRR para encender U/V/W simultáneo.*
- static volatile char [flagAscensoAnguloParcial](#) = 1
 

*Bandera de rampa del ángulo parcial (subida/bajada en diente).*

- static volatile char **estadoLogicoSalida** [3]  
*Estado lógico de salidas U/V/W (para depuración).*
- static volatile int **ticksChannel** [3]  
*t1/t2/t3 en ticks para el ciclo actual.*
- static volatile **CasoInterferenciaTimer casolInterferencia**  
*Caso de interferencia detectado para el ciclo actual.*
- static volatile int32\_t **frecObjetivo**  
*Frecuencia objetivo (target) escalada ×1e6.*
- static volatile int **flagChangingFrecuencia**  
*Indica cambio de frecuencia en curso (rampa activa).*
- static volatile int **flagEsAcelerado**  
*1 si la rampa corresponde a una aceleración, 0 para desaceleración.*
- static volatile int **flagMotorRunning**  
*1 si el motor está en movimiento.*
- static volatile int **frecuenciaReferencia**  
*Frecuencia "de referencia" reportada (Hz).*
- static volatile uint32\_t **cambioFrecuenciaPorCiclo**  
*Delta por ciclo de switching (escalado ×1e6) para la rampa.*
- volatile **ValoresSwitching valoresSwitching**  
*Estructura auxiliar de switching usada por el ISR.*
- volatile **DatoCalculado bufferCalculo**
- volatile **Parametros paramSombra**
- volatile int **flagActualizarParamSombra**

### 10.11.1. Documentación de «define»

#### 10.11.1.1. BUFFER\_CALCULO\_SIZE

```
#define BUFFER_CALCULO_SIZE 3
```

Buffer circular (productor: timer de cálculo; consumidor: timer de switching).

Tamaño fijo 3: el productor se detiene si está lleno.

Definición en la línea 132 del archivo [GestorSVM.c](#).

#### 10.11.1.2. CONST\_CALC\_T1\_ORD\_ORG

```
#define CONST_CALC_T1_ORD_ORG (float)224.2845426
```

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

#### 10.11.1.3. CONST\_CALC\_T1\_PROP

```
#define CONST_CALC_T1_PROP (float)-3.59145E-6
```

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

#### 10.11.1.4. CONST\_CALC\_T2\_ORD\_ORG

```
#define CONST_CALC_T2_ORD_ORG (float) 8.797345358
```

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

#### 10.11.1.5. CONST\_CALC\_T2\_PROP

```
#define CONST_CALC_T2_PROP (float) +3.59145E-6
```

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

#### 10.11.1.6. MAX\_TICKS

```
#define MAX_TICKS 256
```

ARR efectivo del timer de switching (resolución 256 pasos).

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

#### 10.11.1.7. MIN\_TICKS\_DIF

```
#define MIN_TICKS_DIF 5
```

Mínima separación (en ticks) entre eventos para considerar que no hay interferencia entre t1/t2/t3.

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

#### 10.11.1.8. TICKS\_DESHABILITAR\_CANAL

```
#define TICKS_DESHABILITAR_CANAL 280
```

Valor mayor a ARR para forzar que un CCR nunca dispare (deshabilitar canal por “posición imposible”).

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

### 10.11.2. Documentación de enumeraciones

#### 10.11.2.1. CasoInterferenciaTimer

```
enum CasoInterferenciaTimer
```

Casos de interferencia temporal entre eventos t1/t2/t3 (demasiado cercanos).

#### Valores de enumeraciones

INTERF_NULA	
INTERF_T1T2	Sin interferencias.
INTERF_T1T3	t1 y t2 demasiado cercanos.
INTERF_T2T3	t1 y t3 demasiado cercanos.
INTERF_T1T2_T2T3	t2 y t3 demasiado cercanos. t1~t2 y t2~t3 simultáneamente.

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

### 10.11.3. Documentación de funciones

#### 10.11.3.1. GestorSVM\_CalcInterrupt()

```
void GestorSVM_CalcInterrupt (
    void )
```

Handler llamado por el timer de cálculo (productor). Encola t1/t2/t3 si hay espacio.

ISR (o handler llamado por ISR) del timer de cálculo.

Corre a 2x [FREC\\_SWITCH](#) (según tu diseño) para anticipar cómputos: si el buffer circular tiene espacio, calcula t1/t2/t3, cuadrante y posibles interferencias, y los encola para que el timer de switching (TIM3) los consuma. No bloquea si el buffer está lleno.

#### Nota

Tamaño de buffer: 3 entradas (pipeline productor/consumidor).

Definición en la línea 640 del archivo [GestorSVM.c](#).

#### 10.11.3.2. GestorSVM\_CalcularValoresSwitching()

```
void GestorSVM_CalcularValoresSwitching (
    void ) [static]
```

Calcula t1, t2 y t0 (y casos de interferencia) y los deja en [ticksChannel\[\]](#).

- Actualiza ángulos ([anguloActual](#) y [anguloParcial](#)) y [cuadranteActual](#).
- t1/t2 por regresión lineal (constantes CONST\_CALC\_T\* y [indiceModulacion](#)).
- t0 = (255 - t1 - t2)/2. Luego: ticksC1=t0, ticksC2=t0+t1, ticksC3=t0+t1+t2.
- Clasifica interferencias según [MIN\\_TICKS\\_DIF](#) y setea [casoInterferencia](#).

Definición en la línea 238 del archivo [GestorSVM.c](#).

### 10.11.3.3. GestorSVM\_Calculoaceleracioneracion()

```
void GestorSVM_Calculoaceleracioneracion (
    void ) [static]
```

Aplica la rampa de velocidad (aceleracion/desaceleracion) sobre `frecuenciaSalida` y actualiza flags/estado.

- Si `flagActualizarParamSombra` está activo, copia los parámetros de `paramSombra`.
- Ajusta `frecuenciaSalida` en  $\pm$  `cambioFrecuenciaPorCiclo` hasta llegar a `frecObjetivo`.
- Al llegar a 0 Hz: detiene timers, limpia buffer, apaga GPIO y notifica `ACTION_MOTOR_STOPPED`.

#### Nota

En esta funcion se puede llegar a dar una incoherencia debido a la sincronizacion de las variables sombra. Si se esta ejecutando un cambio de frecuencia en la funcion `GestorSVM_SetFrecOut`, es decir que se habia escrito `flagChangingFrec` y antes de llegar levantar el `flagActualizarParamSombra`, justo en ese momento llega una interrupcion de este timer de calculo y en esa iteracion se terminaba de llegar a la velocidad taget. En ese caso se va a pisar ese `flagChangingFrec` y se va a quedar como si nunca hubiese pasado. Debido a que es un caso muy extremo no se analiza

Definición en la línea 513 del archivo [GestorSVM.c](#).

### 10.11.3.4. GestorSVM\_Estop()

```
int GestorSVM_Estop (
    void )
```

Parada de emergencia inmediata: desactiva timers, apaga drivers y salidas.

Parada de emergencia inmediata.

#### Devuelve

Siempre 0.  
Siempre 0.

Deshabilita timers/interrupts, apaga drivers y salidas sin rampa.

#### Atención

Uso ante condiciones de falla. Requiere nueva orden de arranque.

Definición en la línea 857 del archivo [GestorSVM.c](#).

### 10.11.3.5. GestorSVM\_Getaceleracion()

```
int GestorSVM_Getaceleracion ()
```

Devuelve acelerada actual [Hz/s].

Definición en la línea 892 del archivo [GestorSVM.c](#).

### 10.11.3.6. GestorSVM\_GetDesaceleracion()

```
int GestorSVM_GetDesaceleracion ()
```

Devuelve desacelerada actual [Hz/s].

Definición en la línea 897 del archivo [GestorSVM.c](#).

### 10.11.3.7. GestorSVM\_GetDir()

```
int GestorSVM_GetDir ()
```

Devuelve sentido de giro (1 horario, -1 antihorario).

Obtiene el sentido de giro actual (1 horario, -1 antihorario).

Definición en la línea 902 del archivo [GestorSVM.c](#).

### 10.11.3.8. GestorSVM\_GetFrec()

```
int GestorSVM_GetFrec (
    void )
```

Devuelve la frecuencia de referencia configurada (Hz).

Lee la frecuencia objetivo de referencia (no escalada).

Devuelve

[frecuenciaReferencia](#).

Frecuencia de referencia [Hz].

Nota

Si querés la frecuencia *instantánea* en rampa, podrías exponer otra API.

Definición en la línea 887 del archivo [GestorSVM.c](#).

### 10.11.3.9. GestorSVM\_MotorStart()

```
int GestorSVM_MotorStart (
    void )
```

Arranca el motor: habilita drivers, inicia timers y rampa hacia [frecuenciaReferencia](#).

Inicia la marcha: arranca rampa de aceleración hacia [frecReferencia](#).

Devuelve

0 si arranca; 1 si ya estaba en marcha.

0 si inicia correctamente, 1 si ya estaba en marcha.

Habilita drivers, inicia timers (cálculo y switching), precarga ticks, y pone flags para rampa ascendente. La frecuencia objetivo proviene de [GestorSVM\\_SetFrec](#) o [GestorSVM\\_SetConfiguration](#).

Definición en la línea 794 del archivo [GestorSVM.c](#).

### 10.11.3.10. GestorSVM\_MotorStop()

```
int GestorSVM_MotorStop (
    void )
```

Ordena frenado con rampa de [desaceleracion](#) hasta 0 Hz.

Ordena frenado controlado (rampa de desacel hasta 0 Hz).

#### Devuelve

- 0 si acepta; 1 si ya estaba detenido.
- 0 si se acepta la orden, 1 si ya estaba detenido.

No corta drivers instantáneamente: mantiene el switching hasta llegar a 0 Hz, luego apaga salidas y notifica al gestor de estados.

Definición en la línea [831](#) del archivo [GestorSVM.c](#).

### 10.11.3.11. GestorSVM\_Setaceleracion()

```
int GestorSVM_Setaceleracion (
    int nuevaaceleracion)
```

Actualiza acelerada [Hz/s].

#### Devuelve

- 0 OK; -1 fuera de rango; -2 si hay rampa activa.

Definición en la línea [762](#) del archivo [GestorSVM.c](#).

### 10.11.3.12. GestorSVM\_SetConfiguration()

```
void GestorSVM_SetConfiguration (
    ConfiguracionSVM * configuracion)
```

Carga la configuración base de SVM y prepara tablas de BSRR por cuadrante.

Carga la configuración base del SVM.

#### Parámetros

<i>configuracion</i>	Puntero a <a href="#">ConfiguracionSVM</a> .
<i>configuracion</i>	Puntero a <a href="#">ConfiguracionSVM</a> con <i>frec_switch</i> , <i>frecReferencia</i> , <a href="#">direccionRotacion</a> , <i>acel</i> y <i>desacel</i> .

Aplica *frec\_switch*, rampas y sentido; y propaga *frecReferencia* al pipeline de cambio de velocidad. Internamente recalcula lookups/BSRR por cuadrante.

#### Atención

- Validar rangos antes de invocar si la config proviene de UI/externo.

Definición en la línea [585](#) del archivo [GestorSVM.c](#).

### 10.11.3.13. GestorSVM\_SetDecel()

```
int GestorSVM_SetDecel (
    int decel)
```

Actualiza desacelerada [Hz/s].

Actualiza la desaceleración dinámica [Hz/seg].

Devuelve

0 OK; -1 fuera de rango; -2 si hay rampa activa.

#### Parámetros

<i>decel</i>	Nueva desaceleración.
--------------	-----------------------

Devuelve

0 OK; -1 fuera de rango; -2 si hay cambio de velocidad en curso.

Definición en la línea 778 del archivo [GestorSVM.c](#).

### 10.11.3.14. GestorSVM\_SetDir()

```
int GestorSVM_SetDir (
    int dir)
```

Cambia el sentido de giro si el motor está detenido.

Cambia el sentido de giro (solo con motor detenido).

#### Parámetros

<i>dir</i>	0 o 1 (mapeo a antihorario/horario).
------------	--------------------------------------

Devuelve

0 OK; -1 fuera de rango; -2 si motor en marcha o rampa activa.

#### Parámetros

<i>dir</i>	0 o 1 (mapea a antihorario/horario según tu implementación).
------------	--

Devuelve

- 0: Modificado.
- -1: Fuerza de rango.
- -2: Rechazado (motor en marcha o cambio en curso).

Recalcula la tabla BSRR por cuadrante para invertir U/V.

Definición en la línea 717 del archivo [GestorSVM.c](#).

### 10.11.3.15. GestorSVM\_SetFreq()

```
int GestorSVM_SetFreq (
    int freq)
```

Solicita nueva frecuencia objetivo (Hz). Inicia rampa si el motor está en marcha.

Solicita una nueva frecuencia objetivo.

#### Parámetros

<i>freq</i>	Frecuencia objetivo [Hz].
-------------	---------------------------

#### Devuelve

0 si aceptada (motor detenido), 1 si aceptada con rampa en curso; -1 fuera de rango; -2 misma que la actual.

#### Parámetros

<i>freq</i>	Frecuencia objetivo [Hz].
-------------	---------------------------

#### Devuelve

- 0: Aceptada con motor detenido (queda como referencia).
- 1: Aceptada con motor en marcha (inicia rampa).
- -1: Fuera de rango ([FERC\\_OUT\\_MIN..FERC\\_OUT\\_MAX](#)).
- -2: Igual a la frecuencia actual (sin cambios).

Calcula delta por ciclo a partir de acel/ desacel y freq\_switch, y actualiza parámetros sombra para el lazo de cálculo.

Definición en la línea [668](#) del archivo [GestorSVM.c](#).

### 10.11.3.16. GestorSVM\_SwitchInterrupt()

```
void GestorSVM_SwitchInterrupt (
    SwitchInterruptType intType) [static]
```

Handler interno llamado por el ISR de TIM3 según la fuente (CH1..CH4, RESET, CLEAN).

#### Parámetros

<i>intType</i>	Fuente de interrupción <a href="#">SwitchInterruptType</a> .
----------------	--

- RESET: consume una entrada del buffer y precarga CCR2/CCR3/CCR4.
- CH1/CH2/CH3: conmuta GPIO según orden/fase y maneja interferencias habilitando/deshabilitando CCxIE.
- CLEAN: limpia flags y estado interno del switching.

Definición en la línea [300](#) del archivo [GestorSVM.c](#).

### 10.11.3.17. GestorSVM\_SwitchPuertos()

```
void GestorSVM_SwitchPuertos (
    OrdenSwitch orden,
    char estado,
    int numCuadrante) [static]
```

Escribe en BSRR los pines correspondientes a la orden y cuadrante.

#### Parámetros

---

<i>orden</i>	Orden de conmutación OrdenSwitch.
<i>estado</i>	0=primer estado del sector, 1=segundo estado (usa <a href="#">estadoGPIOPorCuadranteYOrden</a> ).
<i>numCuadrante</i>	Sector SVM (0..5).

Definición en la línea 496 del archivo [GestorSVM.c](#).

#### 10.11.3.18. HAL\_TIM\_OC\_DelayElapsedCallback()

```
void HAL_TIM_OC_DelayElapsedCallback (
    TIM_HandleTypeDef * htim)
```

Callback HAL de Output Compare: enruta eventos CCR de TIM3 a [GestorSVM\\_SwitchInterrupt](#).

##### Parámetros

<i>htim</i>	Puntero al handle del timer que disparó el evento.
-------------	--

- CH1 → [SWITCH\\_INT\\_RESET](#)
- CH2 → [SWITCH\\_INT\\_CH1](#)
- CH3 → [SWITCH\\_INT\\_CH2](#)
- CH4 → [SWITCH\\_INT\\_CH3](#)

Definición en la línea 918 del archivo [GestorSVM.c](#).

#### 10.11.3.19. pinMap()

```
int pinMap (
    int x)  [static]
```

Convierte el XOR de estados (bit U/V/W) al índice interno {0,1,2}.

##### Parámetros

<i>x</i>	Máscara de pin que cambia (1bit activo entre U/V/W).
----------	--

Devuelve

Índice 0→U, 1→V, 2→W.

Definición en la línea 223 del archivo [GestorSVM.c](#).

## 10.11.4. Documentación de variables

### 10.11.4.1. aceleracion

```
int aceleracion [static]
```

acelerada configurada [Hz/s].

Definición en la línea 72 del archivo [GestorSVM.c](#).

### 10.11.4.2. anguloActual

```
uint32_t anguloActual [static]
```

Ángulo absoluto (grados×1e3).

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

### 10.11.4.3. anguloParcial

```
int32_t anguloParcial [static]
```

Ángulo parcial 0..60° (grados×1e3), usado para t1/t2. Puede ser negativo durante el diente.

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

### 10.11.4.4. anguloSwitching

```
uint32_t anguloSwitching [static]
```

Incremento angular por switching (grados×1e3).

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

### 10.11.4.5. bufferCalculo

```
volatile DatoCalculado bufferCalculo
```

Definición en la línea 145 del archivo [GestorSVM.c](#).

### 10.11.4.6. cambioFrecuenciaPorCiclo

```
volatile uint32_t cambioFrecuenciaPorCiclo [static]
```

Delta por ciclo de switching (escalado ×1e6) para la rampa.

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

#### 10.11.4.7. casoInterferencia

```
volatile CasoInterferenciaTimer casoInterferencia [static]
```

Caso de interferencia detectado para el ciclo actual.

Definición en la línea 111 del archivo [GestorSVM.c](#).

#### 10.11.4.8. cuadranteActual

```
volatile int cuadranteActual [static]
```

Cuadrante SVM actual (0..5).

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

#### 10.11.4.9. desaceleracion

```
int desaceleracion [static]
```

Desacelerada configurada [Hz/s].

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

#### 10.11.4.10. direccionRotacion

```
int direccionRotacion = 1 [static]
```

Sentido de rotación: 1 horario, -1 antihorario.

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

#### 10.11.4.11. estadoGPIOOff

```
uint32_t estadoGPIOOff
```

BSRR para apagar U/V/W simultáneo.

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

#### 10.11.4.12. estadoGPIOOn

```
uint32_t estadoGPIOOn
```

BSRR para encender U/V/W simultáneo.

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

#### 10.11.4.13. **estadoGPIOPorCuadranteYOrden**

```
uint32_t estadoGPIOPorCuadranteYOrden[6][2]
```

Palabras BSRR precalculadas por cuadrante/estado (0/1).

Definición en la línea 98 del archivo [GestorSVM.c](#).

#### 10.11.4.14. **estadoLogicoSalida**

```
volatile char estadoLogicoSalida[3] [static]
```

Estado lógico de salidas U/V/W (para depuración).

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

#### 10.11.4.15. **flagActualizarParamSombra**

```
volatile int flagActualizarParamSombra
```

Definición en la línea 160 del archivo [GestorSVM.c](#).

#### 10.11.4.16. **flagAscensoAnguloParcial**

```
volatile char flagAscensoAnguloParcial = 1 [static]
```

Bandera de rampa del ángulo parcial (subida/bajada en diente).

Definición en la línea 105 del archivo [GestorSVM.c](#).

#### 10.11.4.17. **flagChangingFrecuencia**

```
volatile int flagChangingFrecuencia [static]
```

Indica cambio de frecuencia en curso (rampa activa).

Definición en la línea 116 del archivo [GestorSVM.c](#).

#### 10.11.4.18. **flagEsAcelerado**

```
volatile int flagEsAcelerado [static]
```

1 si la rampa corresponde a una aceleración, 0 para desaceleración.

Definición en la línea 118 del archivo [GestorSVM.c](#).

#### 10.11.4.19. flagMotorRunning

```
volatile int flagMotorRunning [static]
```

1 si el motor está en movimiento.

Definición en la línea 120 del archivo [GestorSVM.c](#).

#### 10.11.4.20. freqObjetivo

```
volatile int32_t freqObjetivo [static]
```

Frecuencia objetivo (target) escalada ×1e6.

Definición en la línea 114 del archivo [GestorSVM.c](#).

#### 10.11.4.21. frecuenciaReferenica

```
volatile int frecuenciaReferenica [static]
```

Frecuencia “de referencia” reportada (Hz).

Definición en la línea 122 del archivo [GestorSVM.c](#).

#### 10.11.4.22. frecuenciaSalida

```
int32_t frecuenciaSalida [static]
```

Frecuencia de salida INSTANTÁNEA (escalada ×1e6) usada por el lazo de rampa.

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

#### 10.11.4.23. frecuenciaSwitching

```
int frecuenciaSwitching [static]
```

Frecuencia de switching [Hz] (TIM3).

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

#### 10.11.4.24. indiceModulacion

```
volatile int indiceModulacion [static]
```

Índice de modulación (0..100). Controla la tensión de salida. La relación v/f debe ser constante.

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

#### 10.11.4.25. paramSombra

```
volatile Parametros paramSombra
```

Definición en la línea 159 del archivo [GestorSVM.c](#).

#### 10.11.4.26. pinTogglePorCuadranteYOrden

```
int pinTogglePorCuadranteYOrden[6][3]
```

Mapa de qué pin conmuta en cada orden y cuadrante.

Definición en la línea 96 del archivo [GestorSVM.c](#).

#### 10.11.4.27. ticksChannel

```
volatile int ticksChannel[3] [static]
```

t1/t2/t3 en ticks para el ciclo actual.

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

#### 10.11.4.28. valoresSwitching

```
volatile ValoresSwitching valoresSwitching
```

Estructura auxiliar de switching usada por el ISR.

Definición en la línea 126 del archivo [GestorSVM.c](#).

#### 10.11.4.29. vectorSecuenciaPorCuadrante

```
const int vectorSecuenciaPorCuadrante[6][7]
```

##### Valor inicial:

```
= {
    {0b000, 0b100, 0b110, 0b111, 0b110, 0b100, 0b000},
    {0b000, 0b010, 0b110, 0b111, 0b110, 0b010, 0b000},
    {0b000, 0b010, 0b011, 0b111, 0b011, 0b010, 0b000},
    {0b000, 0b001, 0b011, 0b111, 0b011, 0b001, 0b000},
    {0b000, 0b001, 0b101, 0b111, 0b101, 0b001, 0b000},
    {0b000, 0b100, 0b101, 0b111, 0b101, 0b100, 0b000}
}
```

Secuencia SVM por cuadrante (V0→V1→V2→V7→V2→V1→V0).

Cada fila representa el vector lógico {U,V,W} en 7 pasos por sector.

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

## 10.12. GestorSVM.c

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

```

00001
00014
00015 #include <stdio.h>
00016 #include "GestorSVM.h"
00017 #include "stm32f103xb.h"
00018 #include "../Inc/main.h"
00019 #include "../Gestor_Estados/GestorEstados.h"
00020 #include "../Gestor_Timers/GestorTimers.h"
00021
00026 #define MAX_TICKS 256
00027
00032 #define TICKS_DESHABILITAR_CANAL 280
00033
00038 #define MIN_TICKS_DIF 5
00039
00043 #define CONST_CALC_T1_PROP      (float)-3.59145E-6
00044 #define CONST_CALC_T1_ORD_ORG  (float)224.2845426
00045 #define CONST_CALC_T2_PROP      (float)+3.59145E-6
00046 #define CONST_CALC_T2_ORD_ORG  (float)8.797345358
00048
00053 typedef enum {
00054     INTERF_NULA = 0,
00055     INTERF_T1T2,
00056     INTERF_T1T3,
00057     INTERF_T2T3,
00058     INTERF_T1T2_T2T3
00059 } CasoInterferenciaTimer;
00060
00062 static int frecuenciaSwitching;
00064 static int direccionRotacion = 1;
00066 static int32_t frecuenciaSalida;
00068 static uint32_t anguloSwitching;
00070 static volatile int indiceModulacion;
00072 static int aceleracion;
00074 static int desaceleracion;
00076 static uint32_t anguloActual;
00078 static int32_t anguloParcial;
00080 static volatile int cuadranteActual;
00081
00086 const int vectorSecuenciaPorCuadrante[6][7] = {
00087     {0b000,0b100,0b110,0b111,0b100,0b100,0b000}, // Sector 1
00088     {0b000,0b010,0b110,0b111,0b010,0b000}, // Sector 2
00089     {0b000,0b010,0b011,0b111,0b011,0b010,0b000}, // Sector 3
00090     {0b000,0b001,0b011,0b111,0b011,0b001,0b000}, // Sector 4
00091     {0b000,0b001,0b101,0b111,0b101,0b001,0b000}, // Sector 5
00092     {0b000,0b100,0b101,0b111,0b101,0b100,0b000} // Sector 6
00093 };
00094
00096 int pinTogglePorCuadranteYOrden[6][3];
00098 uint32_t estadoGPIOPorCuadranteYOrden[6][2];
00100 uint32_t estadoGPIOOff;
00102 uint32_t estadoGPIOOn;
00103
00105 static volatile char flagAscensoAnguloParcial = 1;
00107 static volatile char estadoLogicoSalida[3];
00109 static volatile int ticksChannel[3];
00111 static volatile CasoInterferenciaTimer casoInterferencia;
00112
00114 static volatile int32_t freqObjetivo;
00116 static volatile int flagChangingFrecuencia;
00118 static volatile int flagEsAcelerado;
00120 static volatile int flagMotorRunning;
00122 static volatile int frecuenciaReferencia;
00124 static volatile uint32_t cambioFrecuenciaPorCiclo;
00126 volatile ValoresSwitching valoresSwitching;
00127
00132 #define BUFFER_CALCULO_SIZE 3
00133
00134 typedef struct {
00135     int ticksChannel1[BUFFER_CALCULO_SIZE];
00136     int ticksChannel2[BUFFER_CALCULO_SIZE];
00137     int ticksChannel3[BUFFER_CALCULO_SIZE];
00138     CasoInterferenciaTimer casoInterferencia[BUFFER_CALCULO_SIZE];
00139     int cuadranteActual[BUFFER_CALCULO_SIZE];
00140     int indiceEscritura;
00141     int indiceLectura;
00142     int contadorDeDatos;
00143 } DatoCalculado;
00144
00145 volatile DatoCalculado bufferCalculo;
00146
00150 typedef struct {

```

```

00151     int32_t frecObjetivo;
00152     uint32_t cambioFrecuenciaPorCiclo;
00153     int direccionRotacion;
00154     int flagMotorRunning;
00155     int flagEsAcelerado;
00156     int flagChangingFrecuencia;
00157 } Parametros;
00158
00159 volatile Parametros paramSombra;
00160 volatile int flagActualizarParamSombra;
00161
00162 /* ===== Prototipos privados ===== */
00163
00164 static void GestorSVM_CalcuLoaceleracioneracion(void);
00165
00166 static int pinMap(int x);
00167
00168 static void GestorSVM_CalcularValoresSwitching(void);
00169
00170 static void GestorSVM_SwitchInterrupt(SwitchInterruptType intType);
00171
00172 static void GestorSVM_SwitchPuertos(OrdenSwitch orden, char estado, int numCuadrante);
00173
00174 /* ===== Implementación privada ===== */
00175
00176 static int pinMap(int x) {
00177     int r = 0;
00178     x &= 0x07; // limitar a 3 bits (U,V,W)
00179     if (x == 1) {
00180         r = 2; // W
00181     }
00182     else if (x == 2) {
00183         r = 1; // V
00184     }
00185     else if (x == 4) {
00186         r = 0; // U
00187     }
00188     return r;
00189 }
00190
00191 static void GestorSVM_CalcularValoresSwitching() {
00192     int t1, t2, t0;
00193     int ticksC1, ticksC2, ticksC3;
00194
00195     /* Rampa de velocidad si corresponde */
00196     if (flagChangingFrecuencia) {
00197         GestorSVM_CalcuLoaceleracioneracion();
00198     }
00199
00200     /* Avance angular y gestión de diente 0..60° */
00201     anguloActual += anguloSwitching;
00202     if (anguloActual >= 360 * 1000 * 1000) {
00203         anguloActual -= 360 * 1000 * 1000;
00204         cuadranteActual = 0;
00205         flagAscensoAnguloParcial = 1;
00206         anguloParcial = anguloActual;
00207     } else {
00208         if (flagAscensoAnguloParcial == 1) {
00209             anguloParcial += anguloSwitching;
00210             if (anguloParcial >= 60 * 1000 * 1000) {
00211                 anguloParcial = 2 * 60 * 1000 * 1000 - anguloParcial; // espejo
00212                 flagAscensoAnguloParcial = 0;
00213                 cuadranteActual = (cuadranteActual + 1) % 6;
00214             }
00215         } else {
00216             anguloParcial -= anguloSwitching;
00217             if (anguloParcial <= 0) {
00218                 anguloParcial = -anguloParcial; // espejo
00219                 flagAscensoAnguloParcial = 1;
00220                 cuadranteActual = (cuadranteActual + 1) % 6;
00221             }
00222         }
00223     }
00224
00225     /* t1/t2 por regresión e índice de modulación */
00226     t1 = (int)((CONST_CALC_T1_PROP * (float)anguloParcial + CONST_CALC_T1_ORD_ORG) *
00227     indiceModulacion) / 100;
00228     t2 = (int)((CONST_CALC_T2_PROP * (float)anguloParcial + CONST_CALC_T2_ORD_ORG) *
00229     indiceModulacion) / 100;
00230     t0 = (int)((float)(255 - t1 - t2) * 0.5);
00231
00232     /* Ticks absolutos en el período */
00233     ticksC1 = t0;
00234     ticksC2 = t0 + t1;
00235     ticksC3 = t0 + t1 + t2;
00236
00237     /* Clasificación de interferencias */
00238 }
```

```

00283     if (ticksC3 - ticksC1 < MIN_TICKS_DIF) {
00284         casoInterferencia = INTERF_T1T3;
00285     } else if (ticksC2 - ticksC1 < MIN_TICKS_DIF && ticksC3 - ticksC2 < MIN_TICKS_DIF) {
00286         casoInterferencia = INTERF_T1T2_T2T3;
00287     } else if (ticksC2 - ticksC1 < MIN_TICKS_DIF) {
00288         casoInterferencia = INTERF_T1T2;
00289     } else if (ticksC3 - ticksC2 < MIN_TICKS_DIF) {
00290         casoInterferencia = INTERF_T2T3;
00291     } else {
00292         casoInterferencia = INTERF_NULA;
00293     }
00294
00295     ticksChannel[0] = ticksC1;
00296     ticksChannel[1] = ticksC2;
00297     ticksChannel[2] = ticksC3;
00298 }
00299
00300 static void GestorSVM_SwitchInterrupt(SwitchInterruptType intType) {
00301     volatile static int countUpTx[3];           // 1=subiendo, 0=bajando
00302     volatile static int flagTxActivo[3];        // espera de evento por canal
00303     volatile static CasoInterferenciaTimer interferencia;
00304     volatile static int cuadrante = 0;
00305     int ticks1, ticks2, ticks3;
00306
00307     /* Si no hay datos precargados, no hacer nada */
00308     if (bufferCalculo.contadorDeDatos <= 0) {
00309         return;
00310     }
00311
00312     switch (intType) {
00313         case SWITCH_INT_CH1:
00314             if (flagTxActivo[0]) {
00315                 flagTxActivo[0] = 0;
00316                 if (countUpTx[0]) {
00317                     GestorSVM_SwitchPuertos(ORDEN_SWITCH_1_UP, 1, cuadrante);
00318                     countUpTx[0] = 0;
00319                 } else {
00320                     countUpTx[0] = 1;
00321                     GestorSVM_SwitchPuertos(ORDEN_SWITCH_1_DOWN, 0, cuadrante);
00322                     /* Rehabilita interrupciones de los otros canales */
00323                     TIM3->DIER |= TIM_DIER_CC2IE;
00324                     TIM3->DIER |= TIM_DIER_CC3IE;
00325                     TIM3->DIER |= TIM_DIER_CC4IE;
00326                 }
00327             } else {
00328                 flagTxActivo[0] = 1;
00329             }
00330             break;
00331
00332         case SWITCH_INT_CH2:
00333             if (flagTxActivo[1]) {
00334                 flagTxActivo[1] = 0;
00335                 if (countUpTx[1]) {
00336                     GestorSVM_SwitchPuertos(ORDEN_SWITCH_2_UP, 1, cuadrante);
00337                     countUpTx[1] = 0;
00338                 } else {
00339                     GestorSVM_SwitchPuertos(ORDEN_SWITCH_2_DOWN, 0, cuadrante);
00340                     countUpTx[1] = 1;
00341                 }
00342             } else {
00343                 if (interferencia == INTERF_T1T2_T2T3) {
00344                     flagTxActivo[1] = 0;
00345                     TIM3->DIER &= ~TIM_DIER_CC3IE; // desactiva T2
00346                 } else {
00347                     flagTxActivo[1] = 1;
00348                 }
00349             }
00350             break;
00351
00352         case SWITCH_INT_CH3:
00353             if (flagTxActivo[2]) {
00354                 flagTxActivo[2] = 0;
00355                 if (countUpTx[2]) {
00356                     GestorSVM_SwitchPuertos(ORDEN_SWITCH_3_UP, 1, cuadrante);
00357
00358                     /* Manejo de interferencias */
00359                     switch (interferencia) {
00360                         case INTERF_NULA:
00361                             flagTxActivo[0] = 1;
00362                             flagTxActivo[1] = 1;
00363                             flagTxActivo[2] = 1;
00364                             break;
00365                         case INTERF_T1T2:
00366                             flagTxActivo[0] = 0; // T1 pendiente
00367                             flagTxActivo[1] = 0; // desactiva T2
00368                             flagTxActivo[2] = 1;
00369                             // Deshabilitacion T2

```

```

00370             TIM3->DIER &= ~TIM_DIER_CC3IE;
00371             // Activacion T1
00372             TIM3->DIER |= TIM_DIER_CC2IE;
00373             break;
00374         case INTERF_T2T3:
00375             flagTxActivo[0] = 1;
00376             flagTxActivo[1] = 0; // T2 pendiente
00377             flagTxActivo[2] = 0; // T3 off
00378             TIM3->DIER &= ~TIM_DIER_CC4IE; // CC4 off
00379             TIM3->DIER |= TIM_DIER_CC3IE; // CC3 on
00380             break;
00381     case INTERF_T1T3:
00382         /* No debe ocurrir aquí */
00383         break;
00384     case INTERF_T1T2_T2T3:
00385         flagTxActivo[0] = 0;
00386         flagTxActivo[1] = 0;
00387         flagTxActivo[2] = 1;
00388         TIM3->DIER &= ~TIM_DIER_CC3IE; // CC3 off
00389         TIM3->DIER |= TIM_DIER_CC2IE; // CC2 on
00390         break;
00391     }
00392     /* Próximas interrupciones: cuenta abajo */
00393     countUpTx[0] = 0;
00394     countUpTx[1] = 0;
00395     countUpTx[2] = 0;
00396 } else {
00397     GestorSVM_SwitchPuertos(ORDEN_SWITCH_3_DOWN, 0, cuadrante);
00398     countUpTx[2] = 1;
00399 }
00400 } else {
00401     if (interferencia == INTERF_T1T3) {
00402         flagTxActivo[0] = 0;
00403         flagTxActivo[1] = 0;
00404         flagTxActivo[2] = 0;
00405     } else {
00406         flagTxActivo[2] = 1;
00407     }
00408 }
00409 break;
00410
00411 case SWITCH_INT_RESET:
00412     /* Re-sync / precarga */
00413     countUpTx[0] = 1;
00414     countUpTx[1] = 1;
00415     countUpTx[2] = 1;
00416
00417     /* Consumo entrada del buffer */
00418     ticks1 = bufferCalculo.ticksChannel1[bufferCalculo.indiceLectura];
00419     ticks2 = bufferCalculo.ticksChannel2[bufferCalculo.indiceLectura];
00420     ticks3 = bufferCalculo.ticksChannel3[bufferCalculo.indiceLectura];
00421     interferencia = bufferCalculo.casoInterferencia[bufferCalculo.indiceLectura];
00422     cuadrante = bufferCalculo.cuadranteActual[bufferCalculo.indiceLectura];
00423     bufferCalculo.indiceLectura = (bufferCalculo.indiceLectura + 1) % BUFFER_CALCULO_SIZE;
00424     bufferCalculo.contadorDeDatos--;
00425
00426     /* Habilitación según interferencias */
00427     switch (interferencia) {
00428         case INTERF_NULA:
00429             flagTxActivo[0] = 1;
00430             flagTxActivo[1] = 1;
00431             flagTxActivo[2] = 1;
00432             TIM3->DIER |= TIM_DIER_CC2IE;
00433             TIM3->DIER |= TIM_DIER_CC3IE;
00434             TIM3->DIER |= TIM_DIER_CC4IE;
00435             break;
00436         case INTERF_T1T2:
00437             flagTxActivo[0] = 0;
00438             flagTxActivo[1] = 1;
00439             flagTxActivo[2] = 1;
00440             TIM3->DIER &= ~TIM_DIER_CC2IE;
00441             TIM3->DIER |= TIM_DIER_CC3IE;
00442             TIM3->DIER |= TIM_DIER_CC4IE;
00443             break;
00444         case INTERF_T2T3:
00445             flagTxActivo[0] = 1;
00446             flagTxActivo[1] = 0;
00447             flagTxActivo[2] = 1;
00448             TIM3->DIER |= TIM_DIER_CC2IE;
00449             TIM3->DIER &= ~TIM_DIER_CC3IE;
00450             TIM3->DIER |= TIM_DIER_CC4IE;
00451             break;
00452         case INTERF_T1T3:
00453             flagTxActivo[0] = 0;
00454             flagTxActivo[1] = 0;
00455             flagTxActivo[2] = 0;
00456             /* Carga posiciones distantes (nunca disparan simultáneo) */

```

```

00457             ticks1 = 20;
00458             ticks2 = 60;
00459             ticks3 = 100;
00460             TIM3->DIER |= TIM_DIER_CC2IE;
00461             TIM3->DIER |= TIM_DIER_CC3IE;
00462             TIM3->DIER |= TIM_DIER_CC4IE;
00463             break;
00464         case INTERF_T1T2_T2T3:
00465             // Desactivacion de T1 y Activacion de T2 y T3
00466             flagTxActivo[0] = 0;
00467             flagTxActivo[1] = 1;
00468             flagTxActivo[2] = 1;
00469             TIM3->DIER &= ~TIM_DIER_CC2IE;
00470             TIM3->DIER |= TIM_DIER_CC3IE;
00471             TIM3->DIER |= TIM_DIER_CC4IE;
00472             break;
00473     }
00474 
00475     /* Precarga CCRs (t1→CCR2, t2→CCR3, t3→CCR4) */
00476     TIM3->CCR2 = ticks1;
00477     TIM3->CCR3 = ticks2;
00478     TIM3->CCR4 = ticks3;
00479     break;
00480 
00481     case SWITCH_INT_CLEAN:
00482         /* Limpieza de estado */
00483         countUpTx[0] = 1;
00484         countUpTx[1] = 1;
00485         countUpTx[2] = 1;
00486         flagTxActivo[0] = 0;
00487         flagTxActivo[1] = 0;
00488         flagTxActivo[2] = 0;
00489         break;
00490 
00491     default:
00492         break;
00493 }
00494 }

00495 static void GestorSVM_SwitchPuertos(OrdenSwitch orden, char estado, int numCuadrante) {
00496     switch (orden) {
00497         case ORDEN_SWITCH_1_UP:
00498         case ORDEN_SWITCH_2_UP:
00499         case ORDEN_SWITCH_3_DOWN:
00500         case ORDEN_SWITCH_2_DOWN:
00501             GPIOA->BSRR = estadoGPIOPorCuadranteYOrden[numCuadrante][estado];
00502             break;
00503         case ORDEN_SWITCH_3_UP:
00504             GPIOA->BSRR = estadoGPIOON;
00505             break;
00506         case ORDEN_SWITCH_1_DOWN:
00507             GPIOA->BSRR = estadoGPIOOff;
00508             break;
00509     }
00510 }
00511 }

00512 static void GestorSVM_Calculoaceleracioneracion() {
00513     /* Sincronización con parámetros sombra, si corresponde */
00514     if (flagActualizarParamSombra) {
00515         frecObjetivo = paramSombra.frecObjetivo;
00516         cambioFrecuenciaPorCiclo = paramSombra.cambioFrecuenciaPorCiclo;
00517         direccionRotacion = paramSombra.direccionRotacion;
00518         flagMotorRunning = paramSombra.flagMotorRunning;
00519         flagEsAcelerado = paramSombra.flagEsAcelerado;
00520         flagChangingFrecuencia = paramSombra.flagChangingFrecuencia;
00521         flagActualizarParamSombra = 0;
00522     }
00523 }

00524 
00525     /* Aplicación de rampa */
00526     if (flagEsAcelerado) {
00527         frecuenciaSalida += cambioFrecuenciaPorCiclo;
00528         if (frecuenciaSalida >= frecObjetivo) {
00529             frecuenciaSalida = frecObjetivo;
00530             GestorEstados_Action(ACTION_TO_CONST_RUNNING, 0);
00531             flagChangingFrecuencia = 0;
00532         }
00533     } else {
00534         frecuenciaSalida -= cambioFrecuenciaPorCiclo;
00535         if (frecuenciaSalida <= frecObjetivo) {
00536             frecuenciaSalida = frecObjetivo;
00537 
00538             if (frecObjetivo == 0) {
00539                 flagMotorRunning = 0;
00540                 GestorTimers_DetenerTimerSVM();
00541                 GestorSVM_SwitchInterrupt(SWITCH_INT_CLEAN);
00542 
00543             /* Limpia buffer productor/consumidor */
00544         }
00545     }
00546 }
```

```

00544         bufferCalculo.indiceEscritura = 0;
00545         bufferCalculo.indiceLectura   = 0;
00546         bufferCalculo.contadorDeDatos = 0;
00547
00548         /* Apaga salidas y drivers */
00549         HAL_GPIO_WritePin(GPIOA, GPIO_U_IN, GPIO_PIN_RESET);
00550         HAL_GPIO_WritePin(GPIOA, GPIO_V_IN, GPIO_PIN_RESET);
00551         HAL_GPIO_WritePin(GPIOA, GPIO_W_IN, GPIO_PIN_RESET);
00552
00553         HAL_GPIO_WritePin(GPIOA, GPIO_U_SD, GPIO_PIN_RESET);
00554         HAL_GPIO_WritePin(GPIOA, GPIO_V_SD, GPIO_PIN_RESET);
00555         HAL_GPIO_WritePin(GPIOA, GPIO_W_SD, GPIO_PIN_RESET);
00556
00557         /* Notifica detención */
00558         GestorEstados_Action(ACTION_MOTOR_STOPPED, 0);
00559     } else {
00560         GestorEstados_Action(ACTION_TO_CONST_RUNNING, 0);
00561     }
00562     flagChangingFrecuencia = 0;
00563 }
00564 }
00565
00566 /* Índice de modulación (V/f) */
00567 if (frecuenciaSalida < 50 * 1000 * 1000) {
00568     indiceModulacion = frecuenciaSalida / (500 * 1000); // lineal hasta 50 Hz
00569 } else {
00570     indiceModulacion = 100;
00571 }
00572 if (indiceModulacion < 1) {
00573     indiceModulacion = 1;
00574 }
00575
00576 /* Incremento angular por switching (cuidar orden para evitar overflow) */
00577 anguloSwitching = (frecuenciaSalida / frecuenciaSwitching) * 360;
00578 }
00579
00580 void GestorSVM_SetConfiguration(ConfiguracionSVM* configuracion) {
00581     int i;
00582     /* Dinámicos */
00583     frecuenciaSwitching      = configuracion->frecuenciaSwitching;
00584     direccionRotacion        = configuracion->direccionRotacion;
00585     aceleracion              = configuracion->aceleracion;
00586     desaceleracion           = configuracion->desaceleracion;
00587
00588     /* Referencia (inicia como target) */
00589     GestorSVM_SetFrec(configuracion->frecuenciaReferencia);
00590
00591     printf("Configuracion Seteada \n");
00592     int estado0, estado1, estado2, estado3;
00593     int pinToggle1, pinToggle2, pinToggle3;
00594     uint32_t myBSRR;
00595
00596     estadoGPIOff = (GPIO_U_IN | GPIO_V_IN | GPIO_W_IN) << 16;
00597     estadoGPIOOn = (GPIO_U_IN | GPIO_V_IN | GPIO_W_IN);
00598
00599     for (i = 0; i < 6; i++) {
00600         estado0 = vectorSecuenciaPorCuadrante[i][0];
00601         estado1 = vectorSecuenciaPorCuadrante[i][1];
00602         estado2 = vectorSecuenciaPorCuadrante[i][2];
00603         estado3 = vectorSecuenciaPorCuadrante[i][3];
00604
00605         /* Estado 0→1 */
00606         myBSRR = 0;
00607         myBSRR |= ((estado1 & 0b100) > 0) ? puerto_senal_pierna[0] : (puerto_senal_pierna[0] << 16);
00608         myBSRR |= ((estado1 & 0b010) > 0) ? puerto_senal_pierna[1] : (puerto_senal_pierna[1] << 16);
00609         myBSRR |= ((estado1 & 0b001) > 0) ? puerto_senal_pierna[2] : (puerto_senal_pierna[2] << 16);
00610         estadoGPIOPorCuadranteYOrden[i][0] = myBSRR;
00611
00612         /* Estado 1→2 */
00613         myBSRR = 0;
00614         myBSRR |= ((estado2 & 0b100) > 0) ? puerto_senal_pierna[0] : (puerto_senal_pierna[0] << 16);
00615         myBSRR |= ((estado2 & 0b010) > 0) ? puerto_senal_pierna[1] : (puerto_senal_pierna[1] << 16);
00616         myBSRR |= ((estado2 & 0b001) > 0) ? puerto_senal_pierna[2] : (puerto_senal_pierna[2] << 16);
00617         estadoGPIOPorCuadranteYOrden[i][1] = myBSRR;
00618
00619         /* Pines que comutaron entre estados (XOR) */
00620         pinToggle1 = pinMap(estado0 ^ estado1);
00621         pinToggle2 = pinMap(estado1 ^ estado2);
00622         pinToggle3 = pinMap(estado2 ^ estado3);
00623
00624         /* Mapea a U/V/W con desplazamiento */
00625         pinTogglePorCuadranteYOrden[i][0] = GPIO_U_IN << (pinToggle1 * 2);
00626         pinTogglePorCuadranteYOrden[i][1] = GPIO_U_IN << (pinToggle2 * 2);
00627         pinTogglePorCuadranteYOrden[i][2] = GPIO_U_IN << (pinToggle3 * 2);
00628
00629     }
00630 }
```

```

00640 void GestorSVM_CalcInterrupt() {
00641     int indiceEscritura;
00642
00643     if (bufferCalculo.contadorDeDatos < BUFFER_CALCULO_SIZE) {
00644         indiceEscritura = bufferCalculo.indiceEscritura;
00645
00646         GestorSVM_CalcularValoresSwitching();
00647
00648         /* Encolar resultados */
00649         bufferCalculo.ticksChannel1[indiceEscritura] = ticksChannel[0];
00650         bufferCalculo.ticksChannel2[indiceEscritura] = ticksChannel[1];
00651         bufferCalculo.ticksChannel3[indiceEscritura] = ticksChannel[2];
00652         bufferCalculo.casoInterferencia[indiceEscritura] = casoInterferencia;
00653         bufferCalculo.cuadranteActual[indiceEscritura] = cuadranteActual;
00654
00655         bufferCalculo.indiceEscritura = (indiceEscritura + 1) % BUFFER_CALCULO_SIZE;
00656         bufferCalculo.contadorDeDatos++;
00657     }
00658 }
00659
00660 /* ----- API invocada por el Gestor de Estados (SPI) ----- */
00661
00662 int GestorSVM_SetFrec(int freq) {
00663     int flagEsAcelerado_local;
00664     int32_t cambioFrecuenciaPorCiclo_local;
00665     int32_t freqTarget_local;
00666     int32_t nuevaFrec;
00667
00668     if (freq < FERC_OUT_MIN || freq > FERC_OUT_MAX) {
00669         return -1;
00670     }
00671     nuevaFrec = (int32_t)freq * 1000 * 1000;
00672     if (frecuenciaSalida == nuevaFrec) {
00673         return -2;
00674     }
00675
00676     if (flagMotorRunning) {
00677         /* Determinar sentido de la rampa */
00678         if (frecuenciaSalida < nuevaFrec) {
00679             flagEsAcelerado_local = 1;
00680             cambioFrecuenciaPorCiclo_local = (aceleracion * 1000 * 1000) / (frecuenciaSwitching);
00681         } else {
00682             flagEsAcelerado_local = 0;
00683             cambioFrecuenciaPorCiclo_local = (desaceleracion * 1000 * 1000) / (frecuenciaSwitching);
00684         }
00685
00686         freqTarget_local = nuevaFrec;
00687         frecuenciaReferencia = freq;
00688
00689         /* Parámetros sombra */
00690         paramSombra.flagMotorRunning = 1;
00691         paramSombra.flagChangingFrecuencia = 1;
00692         paramSombra.flagEsAcelerado = flagEsAcelerado_local;
00693         paramSombra.cambioFrecuenciaPorCiclo = cambioFrecuenciaPorCiclo_local;
00694         paramSombra.freqObjetivo = freqTarget_local;
00695         flagActualizarParamSombra = 1;
00696
00697         flagChangingFrecuencia = 1;
00698         return 1;
00699     } else {
00700         frecuenciaReferencia = freq;
00701         return 0;
00702     }
00703 }
00704
00705 } else {
00706     frecuenciaReferencia = freq;
00707     return 0;
00708 }
00709 }
00710
00711 int GestorSVM_SetDir(int dir) {
00712     int estado1, estado2;
00713     int i;
00714     uint32_t myBSRR;
00715
00716     if (flagMotorRunning) {
00717         return -2;
00718     }
00719     if (flagChangingFrecuencia) {
00720         return -2;
00721     }
00722     if (dir != 0 && dir != 1) {
00723         return -1;
00724     }
00725     if (dir == direccionRotacion) {
00726         return 0;
00727     }
00728
00729     /* Recacular tablas BSRR intercambiando UV */
00730     for (i = 0; i < 6; i++) {
00731         estado1 = vectorSecuenciaPorCuadrante[i][1];
00732         estado2 = vectorSecuenciaPorCuadrante[i][2];
00733     }
00734
00735     for (i = 0; i < 6; i++) {
00736         vectorSecuenciaPorCuadrante[i][1] = estado2;
00737         vectorSecuenciaPorCuadrante[i][2] = estado1;
00738     }
00739 }
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01840
01841
01842
01843
01844
01845
01846
01847
01848
01849
01850
01851
01852
01853
01854
01855
01856
01857
01858
01859
01860
01861
01862
01863
01864
01865
01866
01867
01868
01869
01870
01871
01872
01873
01874
01875
01876
01877
01878
01879
01880
01881
01882
01883
01884
01885
01886
01887
01888
01889
01890
01891
01892
01893
01894
01895
01896
01897
01898
01899
01900
01901
01902
01903
01904
01905
01906
01907
01908
01909
01910
01911
01912
01913
01914
01915
01916
01917
01918
01919
01920
01921
01922
01923
01924
01925
01926
01927
01928
01929
01930
01931
01932
01933
01934
01935
01936
01937
01938
01939
01940
01941
01942
01943
01944
01945
01946
01947
01948
01949
01950
01951
01952
01953
01954
01955
01956
01957
01958
01959
01960
01961
01962
01963
01964
01965
01966
01967
01968
01969
01970
01971
01972
01973
01974
01975
01976
01977
01978
01979
01980
01981
01982
01983
01984
01985
01986
01987
01988
01989
01990
01991
01992
01993
01994
01995
01996
01997
01998
01999
02000
02001
02002
02003
02004
02005
02006
02007
02008
02009
02010
02011
02012
02013
02014
02015
02016
02017
02018
02019
02020
02021
02022
02023
02024
02025
02026
02027
02028
02029
02030
02031
02032
02033
02034
02035
02036
02037
02038
02039
02040
02041
02042
02043
02044
02045
02046
02047
02048
02049
02050
02051
02052
02053
02054
02055
02056
02057
02058
02059
02060
02061
02062
02063
02064
02065
02066
02067
02068
02069
02070
02071
02072
02073
02074
02075
02076
02077
02078
02079
02080
02081
02082
02083
02084
02085
02086
02087
02088
02089
02090
02091
02092
02093
02094
02095
02096
02097
02098
02099
02100
02101
02102
02103
02104
02105
02106
02107
02108
02109
02110
02111
02112
02113
02114
02115
02116
02117
02118
02119
02120
02121
02122
02123
02124
02125
02126
02127
02128
02129
02130
02131
02132
02133
02134
02135
02136
02137
02138
02139
02140
02141
02142
02143
02144
02145
02146
02147
02148
02149
02150
02151
02152
02153
02154
02155
02156
02157
02158
02159
02160
02161
02162
02163
02164
02165
02166
02167
02168
02169
02170
02171
02172
02173
02174
02175
02176
02177
02178
02179
02180
02181
02182
02183
02184
02185
02186
02187
02188
02189
02190
02191
02192
02193
02194
02195
02196
02197
02198
02199
02200
02201
02202
02203
02204
02205
02206
02207
02208
02209
02210
02211
02212
02213
02214
02215
02216
02217
02218
02219
02220
02221
02222
02223
02224
02225
02226
02227
02228
02229
02230
02231
02232
02233
02234
02235
02236
02237
02238
02239
02240
02241
02242
02243
02244
02245
02246
02247
02248
02249
02250
02251
02252
02253
02254
02255
02256
02257
02258
02259
02260
02261
02262
02263
02264
02265
02266
02267
02268
02269
02270
02271
02272
02273
02274
02275
02276
02277
02278
02279
02280
02281
02282
02283
02284
02285
02286
02287
02288
02289
02290
02291
02292
02293
02294
02295
02296
02297
02298
02299
02300
02301
02302
02303
02304
02305
02306
02307
02308
02309
02310
02311
02312
02313
02314
02315
02316
02317
02318
02319
02320
02321
02322
02323
02324
02325
02326
02327
02328
02329
02330
02331
02332
02333
02334
02335
02336
02337
02338
02339
02340
02341
02342
02343
02344
02345
02346
02347
02348
02349
02350
02351
02352

```

```

00739     myBSRR = 0;
00740     myBSRR |= ((estad01 & 0b100) ? GPIO_V_IN : (GPIO_V_IN << 16));
00741     myBSRR |= ((estad01 & 0b010) ? GPIO_U_IN : (GPIO_U_IN << 16));
00742     myBSRR |= ((estad01 & 0b001) ? GPIO_W_IN : (GPIO_W_IN << 16));
00743     estadoGPIOPorCuadranteYOrden[i][0] = myBSRR;
00744
00745     myBSRR = 0;
00746     myBSRR |= ((estad02 & 0b100) ? GPIO_V_IN : (GPIO_V_IN << 16));
00747     myBSRR |= ((estad02 & 0b010) ? GPIO_U_IN : (GPIO_U_IN << 16));
00748     myBSRR |= ((estad02 & 0b001) ? GPIO_W_IN : (GPIO_W_IN << 16));
00749     estadoGPIOPorCuadranteYOrden[i][1] = myBSRR;
00750 }
00751 }
00752
00753     direccionRotacion = dir;
00754     return 0;
00755 }
00756
00762 int GestorSVM_Setaceleracion(int nuevaaceleracion) {
00763     if (flagChangingFrecuencia) {
00764         return -2;
00765     }
00766     if (nuevaaceleracion < ACELERACION_MINIMA || nuevaaceleracion > ACCELERACION_MAXIMA) {
00767         return -1;
00768     }
00769     aceleracion = nuevaaceleracion;
00770     return 0;
00771 }
00772
00778 int GestorSVM_SetDecel(int nuevaDecel) {
00779     if (flagChangingFrecuencia) {
00780         return -2;
00781     }
00782     if (nuevaDecel < DESACELERACION_MINIMA || nuevaDecel > DESACELERACION_MAXIMA) {
00783         return -1;
00784     }
00785     desaceleracion = nuevaDecel;
00786     return 0;
00787 }
00788
00794 int GestorSVM_MotorStart() {
00795     if (!flagMotorRunning) {
00796         flagMotorRunning = 1;
00797         flagEsAcelerado = 1;
00798         flagChangingFrecuencia = 1;
00799
00800         cambioFrecuenciaPorCiclo = (aceleracion * 1000 * 1000) / (frecuenciaSwitching);
00801
00802         /* Parámetros sombra de arranque */
00803         paramSombra.frecObjetivo = (uint32_t)frecuenciaReferencia * 1000 * 1000;
00804         paramSombra.flagMotorRunning = 1;
00805         paramSombra.flagEsAcelerado = 1;
00806         paramSombra.flagChangingFrecuencia = 1;
00807         paramSombra.cambioFrecuenciaPorCiclo = cambioFrecuenciaPorCiclo;
00808         flagActualizarParamSombra = 1;
00809
00810         /* Habilitar drivers */
00811         HAL_GPIO_WritePin(GPIOA, GPIO_U_SD, GPIO_PIN_SET);
00812         HAL_GPIO_WritePin(GPIOA, GPIO_V_SD, GPIO_PIN_SET);
00813         HAL_GPIO_WritePin(GPIOA, GPIO_W_SD, GPIO_PIN_SET);
00814
00815         /* Precargar una muestra y sincronizar switching */
00816         GestorSVM_CalcInterrupt();
00817         GestorSVM_SwitchInterrupt(SWITCH_INT_RESET);
00818
00819         /* Iniciar timer de switching */
00820         GestorTimers_IniciarTimerSVM();
00821         return 0;
00822     }
00823     return 1;
00824 }
00825
00831 int GestorSVM_MotorStop() {
00832     if (flagMotorRunning) {
00833         flagEsAcelerado = 0;
00834         flagChangingFrecuencia = 1;
00835
00836         cambioFrecuenciaPorCiclo = (desaceleracion * 1000 * 1000) / (frecuenciaSwitching);
00837         frecObjetivo = 0;
00838
00839         /* Parámetros sombra */
00840         paramSombra.flagMotorRunning = 1;
00841         paramSombra.flagEsAcelerado = 0;
00842         paramSombra.flagChangingFrecuencia = 1;
00843         paramSombra.cambioFrecuenciaPorCiclo = cambioFrecuenciaPorCiclo;
00844         paramSombra.frecObjetivo = 0;
00845         flagActualizarParamSombra = 1;

```

```

00846         return 0;
00847     }
00848     return 1;
00850 }
00851
00852 int GestorSVM_Estop() {
00853     if (flagMotorRunning) {
00854         GestorTimers_DetenerTimerSVM();
00855
00856         HAL_GPIO_WritePin(GPIOA, GPIO_U_SD, GPIO_PIN_RESET);
00857         HAL_GPIO_WritePin(GPIOA, GPIO_V_SD, GPIO_PIN_RESET);
00858         HAL_GPIO_WritePin(GPIOA, GPIO_W_SD, GPIO_PIN_RESET);
00859
00860         GestorSVM_SwitchInterrupt(SWITCH_INT_CLEAN);
00861
00862         bufferCalculo.indiceEscritura = 0;
00863         bufferCalculo.indiceLectura = 0;
00864         bufferCalculo.contadorDeDatos = 0;
00865
00866         HAL_GPIO_WritePin(GPIOA, GPIO_U_IN, GPIO_PIN_RESET);
00867         HAL_GPIO_WritePin(GPIOA, GPIO_V_IN, GPIO_PIN_RESET);
00868         HAL_GPIO_WritePin(GPIOA, GPIO_W_IN, GPIO_PIN_RESET);
00869
00870         flagChangingFrecuencia = 0;
00871         flagMotorRunning = 0;
00872         frecuenciaSalida = 0;
00873     }
00874     return 0;
00875 }
00876
00877 int GestorSVM_GetFrec() {
00878     return frecuenciaReferencia;
00879 }
00880
00881 int GestorSVM_Getaceleracion() {
00882     return aceleracion;
00883 }
00884
00885 int GestorSVM_GetDesaceleracion() {
00886     return desaceleracion;
00887 }
00888
00889 int GestorSVM_GetDir() {
00890     return direccionRotacion;
00891 }
00892
00893 /* ===== ISR de TIM3 (HAL) ===== */
00894
00895 void HAL_TIM_OC_DelayElapsedCallback(TIM_HandleTypeDef *htim) {
00896     if (htim->Instance != TIM3) {
00897         return;
00898     }
00899
00900     switch (htim->Channel) {
00901         case HAL_TIM_ACTIVE_CHANNEL_1:
00902             GestorSVM_SwitchInterrupt(SWITCH_INT_RESET);
00903             break;
00904         case HAL_TIM_ACTIVE_CHANNEL_2:
00905             GestorSVM_SwitchInterrupt(SWITCH_INT_CH1);
00906             break;
00907         case HAL_TIM_ACTIVE_CHANNEL_3:
00908             GestorSVM_SwitchInterrupt(SWITCH_INT_CH2);
00909             break;
00910         case HAL_TIM_ACTIVE_CHANNEL_4:
00911             GestorSVM_SwitchInterrupt(SWITCH_INT_CH3);
00912             break;
00913         default:
00914             break;
00915     }
00916 }
00917
00918 }
```

## 10.13. Referencia del archivo Modules/Gestor\_SVM/GestorSVM.h

### Estructuras de datos

- struct [ValoresSwitching](#)  
*Estructura de trabajo del ISR del timer de switching.*
- struct [ConfiguracionSVM](#)  
*Parámetros de configuración del módulo SVM.*

**defines**

- #define FREC\_SWITCH 2511
- #define FREC\_REFERENCIA 50
- #define DIRECCION\_ROTACION 1
- #define RESOLUCION\_TIMER 255
- #define ACCELERACION\_DEFUAL 5
- #define FERC\_OUT\_MIN 1
- #define FERC\_OUT\_MAX 150
- #define DESACELERACION\_DEFAULT 3
- #define ACCLERACION\_MAXIMA 50
- #define ACELERACION\_MINIMA 1
- #define DESACELERACION\_MAXIMA 50
- #define DESACELERACION\_MINIMA 1

**typedefs**

- typedef struct ValoresSwitching **ValoresSwitching**
- typedef struct ConfiguracionSVM **ConfiguracionSVM**

**Enumeraciones**

- enum OrdenSwitch {
 ORDEN\_SWITCH\_1\_UP = 0 , ORDEN\_SWITCH\_2\_UP = 1 , ORDEN\_SWITCH\_3\_UP = 2 , ORDEN\_SWITCH\_3\_DOWN = 3 ,
 ORDEN\_SWITCH\_2\_DOWN = 4 , ORDEN\_SWITCH\_1\_DOWN = 5 }
   
*Ordenes simbólicas de commutación por canal/flujo (UP/DOWN).*
- enum SwitchInterruptType {
 SWITCH\_INT\_CH1 = 0 , SWITCH\_INT\_CH2 , SWITCH\_INT\_CH3 , SWITCH\_INT\_RESET ,
 SWITCH\_INT\_CLEAN }

*Fuentes de interrupción de switching usadas por el ISR.*

**Funciones**

- void **GestorSVM\_Init ()**

*Inicializa el gestor SVM y su estado interno.*
- void **GestorSVM\_SetConfiguration (ConfiguracionSVM \*configuracion)**

*Carga la configuración base de SVM y prepara tablas de BSRR por cuadrante.*
- int **GestorSVM\_MotorStart ()**

*Arranca el motor: habilita drivers, inicia timers y rampa hacia **frecuenciaReferencia**.*
- int **GestorSVM\_MotorStop ()**

*Ordena frenado con rampa de **desaceleracion** hasta 0 Hz.*
- int **GestorSVM\_Estop ()**

*Parada de emergencia inmediata: desactiva timers, apaga drivers y salidas.*
- int **GestorSVM\_SetFrec (int freq)**

*Solicita nueva frecuencia objetivo (Hz). Inicia rampa si el motor está en marcha.*
- int **GestorSVM\_SetDir (int dir)**

*Cambia el sentido de giro si el motor está detenido.*
- int **GestorSVM\_SetAcel (int acel)**

*Actualiza la aceleración dinámica [Hz/seg].*
- int **GestorSVM\_SetDecel (int decel)**

- int [GestorSVM\\_GetFrec \(\)](#)  
*Actualiza desacelerada [Hz/s]. Devuelve la frecuencia de referencia configurada (Hz).*
- int [GestorSVM\\_GetAcel \(\)](#)  
*Obtiene la aceleración actual [Hz/seg].*
- int [GestorSVM\\_GetDesacel \(\)](#)  
*Obtiene la desaceleración actual [Hz/seg].*
- int [GestorSVM\\_GetDir \(\)](#)  
*Devuelve sentido de giro (1 horario, -1 antihorario).*
- void [GestorSVM\\_CalcInterrupt \(\)](#)  
*Handler llamado por el timer de cálculo (productor). Encola t1/t2/t3 si hay espacio.*

### 10.13.1. Documentación de «define»

#### 10.13.1.1. ACCELERACION\_DEFUAL

```
#define ACCELERACION_DEFUAL 5
```

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

#### 10.13.1.2. ACCLERACION\_MAXIMA

```
#define ACCLERACION_MAXIMA 50
```

Definición en la línea 82 del archivo [GestorSVM.h](#).

#### 10.13.1.3. ACELERACION\_MINIMA

```
#define ACELERACION_MINIMA 1
```

Definición en la línea 83 del archivo [GestorSVM.h](#).

#### 10.13.1.4. DESACELERACION\_DEFAULT

```
#define DESACELERACION_DEFAULT 3
```

Definición en la línea 81 del archivo [GestorSVM.h](#).

#### 10.13.1.5. DESACELERACION\_MAXIMA

```
#define DESACELERACION_MAXIMA 50
```

Definición en la línea 84 del archivo [GestorSVM.h](#).

### 10.13.1.6. DESACELERACION\_MINIMA

```
#define DESACELERACION_MINIMA 1
```

Definición en la línea 85 del archivo [GestorSVM.h](#).

### 10.13.1.7. DIRECCION\_ROTACION

```
#define DIRECCION_ROTACION 1
```

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

### 10.13.1.8. FERC\_OUT\_MAX

```
#define FERC_OUT_MAX 150
```

Definición en la línea 80 del archivo [GestorSVM.h](#).

### 10.13.1.9. FERC\_OUT\_MIN

```
#define FERC_OUT_MIN 1
```

Definición en la línea 79 del archivo [GestorSVM.h](#).

### 10.13.1.10. FREC\_REFERENCIA

```
#define FREC_REFERENCIA 50
```

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

### 10.13.1.11. FREC\_SWITCH

```
#define FREC_SWITCH 2511
```

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

### 10.13.1.12. RESOLUCION\_TIMER

```
#define RESOLUCION_TIMER 255
```

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

## 10.13.2. Documentación de «typedef»

### 10.13.2.1. ConfiguracionSVM

```
typedef struct ConfiguracionSVM ConfiguracionSVM
```

### 10.13.2.2. ValoresSwitching

```
typedef struct ValoresSwitching ValoresSwitching
```

### 10.13.3. Documentación de enumeraciones

#### 10.13.3.1. OrdenSwitch

```
enum OrdenSwitch
```

Ordenes simbólicas de conmutación por canal/flujo (UP/DOWN).

Mapear a eventos CCR (t1/t2/t3) y flancos (subida/bajada) facilita el código del ISR. El orden debe coincidir con la secuencia SVM configurada.

#### Valores de enumeraciones

ORDEN_SWITCH_1_UP	
ORDEN_SWITCH_2_UP	Evento t1 en flanco de subida.
ORDEN_SWITCH_3_UP	Evento t2 en flanco de subida.
ORDEN_SWITCH_3_DOWN	Evento t3 en flanco de subida.
ORDEN_SWITCH_2_DOWN	Evento t3 en flanco de bajada.
ORDEN_SWITCH_1_DOWN	Evento t2 en flanco de bajada.

Definición en la línea 46 del archivo [GestorSVM.h](#).

#### 10.13.3.2. SwitchInterruptType

```
enum SwitchInterruptType
```

Fuentes de interrupción de switching usadas por el ISR.

- CH1..CH3: comparaciones CCR activas.
- RESET: sincroniza y precarga nuevos ticks (cambio de sector).
- CLEAN: limpia flags/estado (p.ej. al detener motor).

#### Valores de enumeraciones

SWITCH_INT_CH1	
SWITCH_INT_CH2	Interrupción por CCR de t1.
SWITCH_INT_CH3	Interrupción por CCR de t2.
SWITCH_INT_RESET	Interrupción por CCR de t3.
SWITCH_INT_CLEAN	Re-armado de ticks/sincronización por ciclo.

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

### 10.13.4. Documentación de funciones

#### 10.13.4.1. GestorSVM\_CalcInterrupt()

```
void GestorSVM_CalcInterrupt (
    void )
```

Handler llamado por el timer de cálculo (productor). Encola t1/t2/t3 si hay espacio.

ISR (o handler llamado por ISR) del timer de cálculo.

Corre a 2x **FREC\_SWITCH** (según tu diseño) para anticipar cómputos: si el buffer circular tiene espacio, calcula t1/t2/t3, cuadrante y posibles interferencias, y los encola para que el timer de switching (TIM3) los consuma. No bloquea si el buffer está lleno.

##### Nota

Tamaño de buffer: 3 entradas (pipeline productor/consumidor).

Definición en la línea [640](#) del archivo [GestorSVM.c](#).

#### 10.13.4.2. GestorSVM\_Estop()

```
int GestorSVM_Estop (
    void )
```

Parada de emergencia inmediata: desactiva timers, apaga drivers y salidas.

Parada de emergencia inmediata.

##### Devuelve

Siempre 0.

Siempre 0.

Deshabilita timers/interrupts, apaga drivers y salidas sin rampa.

##### Atención

Uso ante condiciones de falla. Requiere nueva orden de arranque.

Definición en la línea [857](#) del archivo [GestorSVM.c](#).

#### 10.13.4.3. GestorSVM\_GetAcel()

```
int GestorSVM_GetAcel ()
```

Obtiene la aceleración actual [Hz/seg].

#### 10.13.4.4. GestorSVM\_GetDesacel()

```
int GestorSVM_GetDesacel ()
```

Obtiene la desaceleración actual [Hz/seg].

#### 10.13.4.5. GestorSVM\_GetDir()

```
int GestorSVM_GetDir ()
```

Devuelve sentido de giro (1 horario, -1 antihorario).

Obtiene el sentido de giro actual (1 horario, -1 antihorario).

Definición en la línea 902 del archivo [GestorSVM.c](#).

#### 10.13.4.6. GestorSVM\_GetFrec()

```
int GestorSVM_GetFrec (
    void )
```

Devuelve la frecuencia de referencia configurada (Hz).

Lee la frecuencia objetivo de referencia (no escalada).

Devuelve

[frecuenciaReferencia](#).

Frecuencia de referencia [Hz].

Nota

Si querés la frecuencia *instantánea* en rampa, podrías exponer otra API.

Definición en la línea 887 del archivo [GestorSVM.c](#).

#### 10.13.4.7. GestorSVM\_Init()

```
void GestorSVM_Init (
    void )
```

Inicializa el gestor SVM y su estado interno.

Pone a cero variables internas, limpia buffers de cálculo y deja flags en estado consistente (motor detenido, sin cambios pendientes).

#### 10.13.4.8. GestorSVM\_MotorStart()

```
int GestorSVM_MotorStart (
    void )
```

Arranca el motor: habilita drivers, inicia timers y rampa hacia [frecuenciaReferencia](#).

Inicia la marcha: arranca rampa de aceleración hacia [frecReferencia](#).

##### Devuelve

- 0 si arranca; 1 si ya estaba en marcha.
- 0 si inicia correctamente, 1 si ya estaba en marcha.

Habilita drivers, inicia timers (cálculo y switching), precarga ticks, y pone flags para rampa ascendente. La frecuencia objetivo proviene de [GestorSVM\\_SetFrec](#) o [GestorSVM\\_SetConfiguration](#).

Definición en la línea 794 del archivo [GestorSVM.c](#).

#### 10.13.4.9. GestorSVM\_MotorStop()

```
int GestorSVM_MotorStop (
    void )
```

Ordena frenado con rampa de [desaceleracion](#) hasta 0 Hz.

Ordena frenado controlado (rampa de desacel hasta 0 Hz).

##### Devuelve

- 0 si acepta; 1 si ya estaba detenido.
- 0 si se acepta la orden, 1 si ya estaba detenido.

No corta drivers instantáneamente: mantiene el switching hasta llegar a 0 Hz, luego apaga salidas y notifica al gestor de estados.

Definición en la línea 831 del archivo [GestorSVM.c](#).

#### 10.13.4.10. GestorSVM\_SetAcel()

```
int GestorSVM_SetAcel (
    int acel)
```

Actualiza la aceleración dinámica [Hz/seg].

##### Parámetros

<i>acel</i>	Nueva aceleración.
-------------	--------------------

##### Devuelve

- 0 OK; -1 fuera de rango; -2 si hay cambio de velocidad en curso.

#### 10.13.4.11. GestorSVM\_SetConfiguration()

```
void GestorSVM_SetConfiguration (
    ConfiguracionSVM * configuracion)
```

Carga la configuración base de SVM y prepara tablas de BSRR por cuadrante.

Carga la configuración base del SVM.

#### Parámetros

---

<i>configuracion</i>	Puntero a <a href="#">ConfiguracionSVM</a> .
<i>configuracion</i>	Puntero a <a href="#">ConfiguracionSVM</a> con freq_switch, freqReferencia, <a href="#">direccionRotacion</a> , acel y des-acel.

Aplica freq\_switch, rampas y sentido; y propaga freqReferencia al pipeline de cambio de velocidad. Internamente recalcula lookups/BSRR por cuadrante.

#### Atención

Validar rangos antes de invocar si la config proviene de UI/externo.

Definición en la línea [585](#) del archivo [GestorSVM.c](#).

#### 10.13.4.12. GestorSVM\_SetDecel()

```
int GestorSVM_SetDecel (
    int nuevaDecel)
```

Actualiza desacelerada [Hz/s].

Actualiza la desaceleración dinámica [Hz/seg].

#### Devuelve

0 OK; -1 fuera de rango; -2 si hay rampa activa.

#### Parámetros

<i>decel</i>	Nueva desaceleración.
--------------	-----------------------

#### Devuelve

0 OK; -1 fuera de rango; -2 si hay cambio de velocidad en curso.

Definición en la línea [778](#) del archivo [GestorSVM.c](#).

#### 10.13.4.13. GestorSVM\_SetDir()

```
int GestorSVM_SetDir (
    int dir)
```

Cambia el sentido de giro si el motor está detenido.

Cambia el sentido de giro (solo con motor detenido).

#### Parámetros

---

<i>dir</i>	0 o 1 (mapeo a antihorario/horario).
------------	--------------------------------------

**Devuelve**

0 OK; -1 fuera de rango; -2 si motor en marcha o rampa activa.

**Parámetros**

<i>dir</i>	0 o 1 (mapea a antihorario/horario según tu implementación).
------------	--

**Devuelve**

- 0: Modificado.
- -1: Fuera de rango.
- -2: Rechazado (motor en marcha o cambio en curso).

Recalcula la tabla BSRR por cuadrante para invertir U/V.

Definición en la línea [717](#) del archivo [GestorSVM.c](#).

**10.13.4.14. GestorSVM\_SetFrec()**

```
int GestorSVM_SetFrec (
    int freq)
```

Solicita nueva frecuencia objetivo (Hz). Inicia rampa si el motor está en marcha.

Solicita una nueva frecuencia objetivo.

**Parámetros**

<i>freq</i>	Frecuencia objetivo [Hz].
-------------	---------------------------

**Devuelve**

0 si aceptada (motor detenido), 1 si aceptada con rampa en curso; -1 fuera de rango; -2 misma que la actual.

**Parámetros**

<i>freq</i>	Frecuencia objetivo [Hz].
-------------	---------------------------

**Devuelve**

- 0: Aceptada con motor detenido (queda como referencia).
- 1: Aceptada con motor en marcha (inicia rampa).
- -1: Fuera de rango ([FERC\\_OUT\\_MIN..FERC\\_OUT\\_MAX](#)).
- -2: Igual a la frecuencia actual (sin cambios).

Calcula delta por ciclo a partir de acel/ desacel y freq\_switch, y actualiza parámetros sombra para el lazo de cálculo.

Definición en la línea [668](#) del archivo [GestorSVM.c](#).

## 10.14. GestorSVM.h

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

```

00001 #ifndef GESTOR_SVM_GESTORSVM_H_
00002 #define GESTOR_SVM_GESTORSVM_H_
00003
00013 typedef struct ValoresSwitching {
00014     char cuadrante;
00015     char flagSwitch[3];
00016     int ticks1[2];
00017     int ticks2[2];
00018     int ticks3[2];
00019     int contador;
00020 } ValoresSwitching;
00021
00031 typedef struct ConfiguracionSVM {
00032     int freq_switch;
00033     int freqReferencia;
00034     int direccionRotacion;
00035     int acel;
00036     int desacel;
00037 } ConfiguracionSVM;
00038
00046 typedef enum {
00047     ORDEN_SWITCH_1_UP    = 0,
00048     ORDEN_SWITCH_2_UP    = 1,
00049     ORDEN_SWITCH_3_UP    = 2,
00050     ORDEN_SWITCH_3_DOWN  = 3,
00051     ORDEN_SWITCH_2_DOWN  = 4,
00052     ORDEN_SWITCH_1_DOWN  = 5,
00053 } OrdenSwitch;
00054
00063 typedef enum {
00064     SWITCH_INT_CH1 = 0,
00065     SWITCH_INT_CH2,
00066     SWITCH_INT_CH3,
00067     SWITCH_INT_RESET,
00068     SWITCH_INT_CLEAN,
00069 } SwitchInterruptType;
00070
00071 /* ----- Parámetros de operación por defecto / límites ----- */
00072
00073 #define FREC_SWITCH          2511
00074 #define FREC_REFERENCIA       50
00075 #define DIRECCION_ROTACION   1
00076 #define RESOLUCION_TIMER      255
00077 #define ACCELERACION_DEFAULT  5
00078
00079 #define FERC_OUT_MIN          1
00080 #define FERC_OUT_MAX          150
00081 #define DESACELERACION_DEFAULT 3
00082 #define ACCELERACION_MAXIMA    50
00083 #define ACCELERACION_MINIMA    1
00084 #define DESACELERACION_MAXIMA  50
00085 #define DESACELERACION_MINIMA  1
00086
00094 void GestorSVM_Init();
00095
00106 void GestorSVM_SetConfiguration(ConfiguracionSVM* configuracion);
00107
00108 /* ----- API de control en tiempo de ejecución ----- */
00109
00119 int GestorSVM_MotorStart();
00120
00129 int GestorSVM_MotorStop();
00130
00139 int GestorSVM_Estop();
00140
00154 int GestorSVM_SetFrec(int freq);
00155
00167 int GestorSVM_SetDir(int dir);
00168
00175 int GestorSVM_SetAcel(int acel);
00176
00183 int GestorSVM_SetDecel(int decel);
00184
00191 int GestorSVM_GetFrec();
00192
00198 int GestorSVM_GetAcel();
00199
00204 int GestorSVM_GetDesacel();
00205
00211 int GestorSVM_GetDir();
00212
00223 void GestorSVM_CalcInterrupt();

```

```
00224
00225 #endif /* GESTOR_SVM_GESTORSVM_H_ */
```

## 10.15. Referencia del archivo Modules/Gestor\_Timers/GestorTimers.c

```
#include "../Gestor_Timers/GestorTimers.h"
#include "../Inc/main.h"
```

### Funciones

- void [GestorTimers\\_Init](#) (TIM\_HandleTypeDef \*\_hTimerSwitch, TIM\_HandleTypeDef \*\_hTimerCalc)
- void [GestorTimers\\_IniciarTimerSVM](#) ()
- void [GestorTimers\\_DetenerTimerSVM](#) ()

### Variables

- TIM\_HandleTypeDef \* [hTimerSwitch](#)
- TIM\_HandleTypeDef \* [hTimerCalc](#)

### 10.15.1. Documentación de funciones

#### 10.15.1.1. [GestorTimers\\_DetenerTimerSVM\(\)](#)

```
void GestorTimers_DetenerTimerSVM ()
```

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

#### 10.15.1.2. [GestorTimers\\_IniciarTimerSVM\(\)](#)

```
void GestorTimers_IniciarTimerSVM ()
```

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

#### 10.15.1.3. [GestorTimers\\_Init\(\)](#)

```
void GestorTimers_Init (
    TIM_HandleTypeDef * _hTimerSwitch,
    TIM_HandleTypeDef * _hTimerCalc)
```

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

## 10.15.2. Documentación de variables

### 10.15.2.1. hTimerCalc

TIM\_HandleTypeDef\* hTimerCalc

Definición en la línea 12 del archivo [GestorTimers.c](#).

### 10.15.2.2. hTimerSwitch

TIM\_HandleTypeDef\* hTimerSwitch

Definición en la línea 11 del archivo [GestorTimers.c](#).

## 10.16. GestorTimers.c

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

```

00001 #include "../Gestor_Timers/GestorTimers.h"
00002
00003 #include "../Inc/main.h"
00004
00005 // Este gestor va a mantener los timers. A este le vamos a pedir que nos inicie
00006 // y detenga los timers.
00007 // Los posibles timers ya estan establecidos
00008
00009
00010 // Dos punteros a las variables de timer
00011 TIM_HandleTypeDef* hTimerSwitch;
00012 TIM_HandleTypeDef* hTimerCalc;
00013
00014 void GestorTimers_Init(TIM_HandleTypeDef* _hTimerSwitch, TIM_HandleTypeDef* _hTimerCalc) {
00015     hTimerSwitch = _hTimerSwitch;
00016     hTimerCalc = _hTimerCalc;
00017 }
00018
00019
00020 void GestorTimers_IniciarTimerSVM() {
00021
00022     // Seteo de la prioridad del timer 2 a la segunda mas alta (1)
00023     HAL_NVIC_SetPriority(TIM2_IRQn, 1, 0);
00024     HAL_NVIC_EnableIRQ(TIM2_IRQn);
00025
00026     // Seteo de la prioridad del timer 3 a la prioridad mas alta (0)
00027     HAL_NVIC_SetPriority(TIM3_IRQn, 0, 0);
00028     HAL_NVIC_EnableIRQ(TIM3_IRQn);
00029
00030     // Se inician el timer 2
00031     HAL_TIM_IC_Start_IT(hTimerCalc, TIM_CHANNEL_1);
00032
00033     // Se inician los 4 canales del timer 3
00034     HAL_TIM_IC_Start_IT(hTimerSwitch, TIM_CHANNEL_1);
00035     HAL_TIM_IC_Start_IT(hTimerSwitch, TIM_CHANNEL_2);
00036     HAL_TIM_IC_Start_IT(hTimerSwitch, TIM_CHANNEL_3);
00037     HAL_TIM_IC_Start_IT(hTimerSwitch, TIM_CHANNEL_4);
00038 }
00039
00040 void GestorTimers_DetenerTimerSVM() {
00041
00042     // Se detiene el timer de switcheo
00043     HAL_TIM_IC_Stop_IT(hTimerSwitch, TIM_CHANNEL_1);
00044     HAL_TIM_IC_Stop_IT(hTimerSwitch, TIM_CHANNEL_2);
00045     HAL_TIM_IC_Stop_IT(hTimerSwitch, TIM_CHANNEL_3);
00046     HAL_TIM_IC_Stop_IT(hTimerSwitch, TIM_CHANNEL_4);
00047
00048     __HAL_TIM_SET_COUNTER(hTimerSwitch, 0);
00049
00050
00051     // Se detiene el timer de calculo
00052     HAL_TIM_IC_Stop_IT(hTimerCalc, TIM_CHANNEL_1);
00053     __HAL_TIM_SET_COUNTER(hTimerCalc, 0);
00054
00055 }
```

## 10.17. Referencia del archivo Modules/Gestor\_Timers/GestorTimers.h

### Enumeraciones

- enum TimersEnum { SVM\_Timer = 0 , Dinamic\_Timer }

### Funciones

- void GestorTimers\_Init ()
- void GestorTimers\_IniciarTimerSVM ()
- void GestorTimers\_DetenerTimerSVM ()

### 10.17.1. Documentación de enumeraciones

#### 10.17.1.1. TimersEnum

```
enum TimersEnum
```

##### Valores de enumeraciones

SVM_Timer	
Dinamic_Timer	

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

### 10.17.2. Documentación de funciones

#### 10.17.2.1. GestorTimers\_DetenerTimerSVM()

```
void GestorTimers_DetenerTimerSVM ()
```

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

#### 10.17.2.2. GestorTimers\_IniciarTimerSVM()

```
void GestorTimers_IniciarTimerSVM ()
```

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

#### 10.17.2.3. GestorTimers\_Init()

```
void GestorTimers_Init ()
```

## 10.18. GestorTimers.h

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

```

00001 /*
00002  * GestorTimers.h
00003 *
00004  * Created on: Aug 10, 2025
00005  * Author: elian
00006 */
00007
00008 #ifndef GESTOR_TIMERS_GESTORTIMERS_H_
00009 #define GESTOR_TIMERS_GESTORTIMERS_H_
00010
00011
00012
00013 typedef enum {
00014     SVM_Timer = 0,
00015     Dinamic_Timer
00016 } TimersEnum;
00017
00018 void GestorTimers_Init();
00019
00020 void GestorTimers_IniciarTimerSVM();
00021 void GestorTimers_DetenerTimerSVM();
00022
00023 #endif /* GESTOR_TIMERS_GESTORTIMERS_H_ */

```

## 10.19. Referencia del archivo Modules/SPI\_Interfase/SPIModule.c

```

#include <stdio.h>
#include <string.h>
#include "main.h"
#include "../Gestor_Estados/GestorEstados.h"

```

### defines

- `#define SPI_BUF_SIZE 16`
- `#define SPI_TRANSMITION_SIZE 4`
- `#define CMD_LEN 3`

### Funciones

- `static void SPI_ProcesarComando (uint8_t *buffer, int cantBytes, uint8_t *bufferResponse)`  
*Procesa un comando recibido por SPI y construye la respuesta.*
- `void HAL_SPI_TxRxCpltCallback (SPI_HandleTypeDef *_hspi)`  
*Callback de HAL llamado al completar una transacción TxRx por DMA.*
- `void SPI_Init (void *_hspi)`  
*Inicializa el módulo SPI esclavo con DMA y deja activa la primera transacción.*

### Variables

- `uint8_t rxDMABuffer [SPI_BUF_SIZE]`
- `uint8_t txDMABuffer [SPI_BUF_SIZE]`
- `volatile uint8_t rxBuffer [SPI_BUF_SIZE]`
- `volatile int rxIndex = 0`
- `volatile uint8_t tx_buffer [CMD_LEN] = {0xAA, 0xBB, 0xCC}`
- `volatile uint8_t rx_index = 0`
- `SPI_HandleTypeDef * hspi`

### 10.19.1. Documentación de «define»

#### 10.19.1.1. CMD\_LEN

```
#define CMD_LEN 3
```

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

#### 10.19.1.2. SPI\_BUF\_SIZE

```
#define SPI_BUF_SIZE 16
```

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

#### 10.19.1.3. SPI\_TRANSMITION\_SIZE

```
#define SPI_TRANSMITION_SIZE 4
```

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

### 10.19.2. Documentación de funciones

#### 10.19.2.1. HAL\_SPI\_TxRxCpltCallback()

```
void HAL_SPI_TxRxCpltCallback (
    SPI_HandleTypeDef * _hspi)
```

Callback de HAL llamado al completar una transacción TxRx por DMA.

- Busca el delimitador ';' en rxDMABuffer para conocer longitud de comando.
- Construye una respuesta por defecto (ERR\_NO\_COMMAND) si el frame es inválido.
- Llama a [SPI\\_ProcesarComando\(\)](#) si hay un payload válido.
- Copia la respuesta a txDMABuffer para la PRÓXIMA transacción.
- Re-arma la transacción DMA continua con HAL\_SPI\_TransmitReceive\_DMA().

Definición en la línea 223 del archivo [SPIModule.c](#).

#### 10.19.2.2. SPI\_Init()

```
void SPI_Init (
    void * hspi)
```

Inicializa el módulo SPI esclavo con DMA y deja activa la primera transacción.

- Configura prioridades/enable de IRQ para DMA1 Channel 4/5 (Tx/Rx).
- Limpia buffers RX/TX y precarga una respuesta por defecto ([SPI\\_RESPONSE\\_OK](#) ; ).
- Llama a HAL\_SPI\_TransmitReceive\_DMA() para iniciar el ciclo continuo DMA (el tamaño de paquete es el configurado en el fuente, p.ej. [SPI\\_TRANSMITION\\_SIZE](#)).
- El parseo y la construcción de [SPI\\_Response](#) se realizan en el callback [HAL\\_SPI\\_TxRxCpltCallback](#), invocando [GestorEstados\\_Action](#) según corresponda.

### Parámetros

---

in	<i>hspi</i>	Handler de SPI inicializado por HAL (tipo SPI_HandleTypeDef*, p.ej. & <i>hspi2</i> ).
----	-------------	---

Ver también

[GestorEstados\\_Action](#)

[SPI\\_Request](#)

[SPI\\_Response](#)

Definición en la línea [257](#) del archivo [SPIModule.c](#).

### 10.19.2.3. SPI\_ProcesarComando()

```
void SPI_ProcesarComando (
    uint8_t * buffer,
    int cantBytes,
    uint8_t * bufferResponse) [static]
```

Procesa un comando recibido por SPI y construye la respuesta.

#### Parámetros

<i>buffer</i>	Puntero al buffer recibido (sin el ';' final).
<i>cantBytes</i>	Cantidad de bytes válidos en buffer (antes de ';').
<i>bufferResponse</i>	Puntero a un arreglo destino (mín. 4 bytes) donde se escribe la respuesta en formato [RESP][';'][dato1][dato2].

#### Nota

La respuesta se deja en *bufferResponse*; el callback de DMA la copiará a *txDMABuffer*.

Para GET\_FREC se empaquetan hasta 2 bytes (valor  $\leq 511$  según tu lógica actual).

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

### 10.19.3. Documentación de variables

#### 10.19.3.1. hspi

```
SPI_HandleTypeDef* hspi
```

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

#### 10.19.3.2. rx\_index

```
volatile uint8_t rx_index = 0
```

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

### 10.19.3.3. rxBuffer

```
volatile uint8_t rxBuffer[SPI_BUF_SIZE]
```

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

### 10.19.3.4. rxDMABuffer

```
uint8_t rxDMABuffer[SPI_BUF_SIZE]
```

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

### 10.19.3.5. rxIndex

```
volatile int rxIndex = 0
```

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

### 10.19.3.6. tx\_buffer

```
volatile uint8_t tx_buffer[CMD_LEN] = {0xAA, 0xBB, 0xCC}
```

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

### 10.19.3.7. txDMABuffer

```
uint8_t txDMABuffer[SPI_BUF_SIZE]
```

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

## 10.20. SPIModule.c

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

```
00001 /*
00002 *   SPIModule.c
00003 *
00004 *   Created on: 11 ago. 2025
00005 *       Author: elian
00006 */
00007
00008 #include <stdio.h>
00009 #include <string.h>
00010 #include "main.h"
00011 #include "../Gestor_Estados/GestorEstados.h"
00012
00013 /* Tamaños de buffers y frame SPI */
00014 #define SPI_BUF_SIZE          16 // Tamaño del buffer circular DMA RX/TX
00015 #define SPI_TRANSMISSION_SIZE  4 // Longitud de transacción DMA (bytes)
00016 #define CMD_LEN                3 // Cantidad de bytes que envía el master en un comando típico
00017
00018 /* Buffers para DMA (HW lee/escribe aquí directamente) */
00019 uint8_t rxDMABuffer[SPI_BUF_SIZE]; // RX por DMA (lectura directa desde periférico)
00020 uint8_t txDMABuffer[SPI_BUF_SIZE]; // TX por DMA (próxima respuesta a enviar)
00021
00022 /* Buffers auxiliares opcionales (no usados con DMA continuo) */
00023 volatile uint8_t rxBuffer[SPI_BUF_SIZE];
```

```

00024 volatile int rxIndex = 0;
00025
00026 /* Respuesta inicial dummy (no usada en flujo actual, se mantiene por compatibilidad) */
00027 volatile uint8_t tx_buffer[CMD_LEN] = {0xAA, 0xBB, 0xCC};
00028 volatile uint8_t rx_index = 0;
00029
00030 /* Handle del periférico SPI (inyectado desde fuera con SPI_Init) */
00031 SPI_HandleTypeDef hspi;
00032
00043 static void SPI_ProcesarComando(uint8_t* buffer, int cantBytes, uint8_t* bufferResponse) {
00044     int resp;
00045     int val;
00046
00047     // Selección por comando (primer byte del buffer de respuesta
00048     bufferResponse[0] = '\0';
00049     switch (buffer[0]) {
00050         case SPI_REQUEST_START:
00051             resp = GestorEstados_Action(ACTION_START, 0);
00052
00053             if(resp == ACTION_RESP_OK) {
00054                 bufferResponse[0] = SPI_RESPONSE_OK;
00055             } else if(resp == ACTION_RESP_MOVING) {
00056                 bufferResponse[0] = SPI_RESPONSE_ERR_MOVING;
00057             } else if(resp == ACTION_RESP_EMERGENCY_ACTIVE) {
00058                 bufferResponse[0] = SPI_RESPONSE_ERR_EMERGENCY_ACTIVE;
00059             } else {
00060                 bufferResponse[0] = SPI_RESPONSE_ERR;
00061             }
00062
00063             bufferResponse[1] = ';';
00064             return;
00065
00066         case SPI_REQUEST_STOP:
00067             resp = GestorEstados_Action(ACTION_STOP, 0);
00068
00069             if(resp == ACTION_RESP_OK) {
00070                 bufferResponse[0] = SPI_RESPONSE_OK;
00071             } else if(resp == ACTION_RESP_NOT_MOVING) {
00072                 bufferResponse[0] = SPI_RESPONSE_ERR_NOT_MOVING;
00073             } else {
00074                 bufferResponse[0] = SPI_RESPONSE_ERR;
00075             }
00076
00077             bufferResponse[1] = ';';
00078             return;
00079
00080         case SPI_REQUEST_EMERGENCY:
00081             GestorEstados_Action(ACTION_EMERGENCY, 0);
00082             bufferResponse[0] = SPI_RESPONSE_OK;
00083             bufferResponse[1] = ';';
00084             return;
00085
00086         case SPI_REQUEST_SET_FREC:
00087             val = (uint8_t)buffer[1] + (uint8_t)buffer[2];
00088             resp = GestorEstados_Action(ACTION_SET_FREC, val);
00089
00090             if(resp == ACTION_RESP_OK) {
00091                 bufferResponse[0] = SPI_RESPONSE_OK;
00092             } else if(resp == ACTION_RESP_OUT_RANGE) {
00093                 bufferResponse[0] = SPI_RESPONSE_ERR_DATA_OUT_RANGE;
00094             } else if(resp == ACTION_RESP_MOVING) {
00095                 bufferResponse[0] = SPI_RESPONSE_ERR_MOVING;
00096             } else {
00097                 bufferResponse[0] = SPI_RESPONSE_ERR;
00098             }
00099
00100             bufferResponse[1] = ';';
00101             return;
00102
00103         case SPI_REQUEST_SET_ACCEL:
00104             resp = GestorEstados_Action(ACTION_SET_ACCEL, (uint8_t)buffer[1]);
00105
00106             if(resp == ACTION_RESP_OK) {
00107                 bufferResponse[0] = SPI_RESPONSE_OK;
00108             } else if(resp == ACTION_RESP_OUT_RANGE) {
00109                 bufferResponse[0] = SPI_RESPONSE_ERR_DATA_OUT_RANGE;
00110             } else if(resp == ACTION_RESP_MOVING) {
00111                 bufferResponse[0] = SPI_RESPONSE_ERR_MOVING;
00112             } else {
00113                 bufferResponse[0] = SPI_RESPONSE_ERR;
00114             }
00115
00116             bufferResponse[1] = ';';
00117             return;
00118
00119         case SPI_REQUEST_SET_DESACEL:
00120             resp = GestorEstados_Action(ACTION_SET_DESACEL, (uint8_t)buffer[1]);

```

```
00121     if(resp == ACTION_RESP_OK) {
00122         bufferResponse[0] = SPI_RESPONSE_OK;
00123     } else if(resp == ACTION_RESP_OUT_RANGE) {
00124         bufferResponse[0] = SPI_RESPONSE_ERR_DATA_OUT_RANGE;
00125     } else if(resp == ACTION_RESP_MOVING) {
00126         bufferResponse[0] = SPI_RESPONSE_ERR_MOVING;
00127     } else {
00128         bufferResponse[0] = SPI_RESPONSE_ERR;
00129     }
00130
00131     bufferResponse[1] = ';';
00132     return;
00133
00134
00135 case SPI_REQUEST_SET_DIR:
00136     val = (uint8_t)buffer[1];
00137     resp = GestorEstados_Action(ACTION_SET_DIR, val);
00138
00139     if(resp == ACTION_RESP_OK) {
00140         bufferResponse[0] = SPI_RESPONSE_OK;
00141     } else if(resp == ACTION_RESP_OUT_RANGE) {
00142         bufferResponse[0] = SPI_RESPONSE_ERR_DATA_OUT_RANGE;
00143     } else if(resp == ACTION_RESP_MOVING) {
00144         bufferResponse[0] = SPI_RESPONSE_ERR_MOVING;
00145     } else {
00146         bufferResponse[0] = SPI_RESPONSE_ERR;
00147     }
00148
00149     bufferResponse[1] = ';';
00150     return;
00151
00152
00153 case SPI_REQUEST_GET_FREC:
00154     /* Lee valor actual de frecuencia desde el SVM */
00155     val = GestorSVM_GetFrec();
00156     bufferResponse[0] = SPI_RESPONSE_OK;
00157     /* Empaquetado simple en 2 bytes (LOW/HIGH "capado") */
00158     if (val > 255) {
00159         bufferResponse[1] = 255;
00160         bufferResponse[2] = (uint8_t)(val - 255);
00161     } else {
00162         bufferResponse[1] = (uint8_t)val;
00163         bufferResponse[2] = 0;
00164     }
00165     bufferResponse[3] = ';';
00166     return;
00167
00168 case SPI_REQUEST_GET_ACCEL:
00169     bufferResponse[0] = SPI_RESPONSE_OK;
00170     bufferResponse[1] = GestorSVM_GetAcel();
00171     bufferResponse[2] = 0;
00172     bufferResponse[3] = ';';
00173     return;
00174
00175 case SPI_REQUEST_GET_DESACEL:
00176     bufferResponse[0] = SPI_RESPONSE_OK;
00177     bufferResponse[1] = GestorSVM_GetDesacel();
00178     bufferResponse[2] = 0;
00179     bufferResponse[3] = ';';
00180     return;
00181
00182 case SPI_REQUEST_GET_DIR:
00183     bufferResponse[0] = SPI_RESPONSE_OK;
00184     bufferResponse[1] = GestorSVM_GetDir();
00185     bufferResponse[2] = 0;
00186     bufferResponse[3] = ';';
00187     return;
00188
00189 case SPI_REQUEST_IS_STOP:
00190     bufferResponse[0] = SPI_RESPONSE_OK;
00191     bufferResponse[1] = (uint8_t)GestorEstados_Action(ACTION_IS_MOTOR_STOP, 0);
00192     bufferResponse[2] = 0;
00193     bufferResponse[3] = ';';
00194     return;
00195
00196 case SPI_REQUEST_RESPONSE:
00197     bufferResponse[0] = SPI_RESPONSE_OK;
00198     bufferResponse[1] = ';';
00199     return;
00200
00201 default:
00202     /* Comando desconocido */
00203     bufferResponse[0] = SPI_RESPONSE_ERR_CMD_UNKNOWN;
00204     bufferResponse[1] = ';';
00205     return;
00206
00207 /* Fallback (no deberia alcanzarse) */
```

```

00208     bufferResponse[0] = SPI_RESPONSE_ERR;
00209     bufferResponse[1] = ';';
00210 }
00211
00223 void HAL_SPI_TxRxCpltCallback(SPI_HandleTypeDef *_hspi) {
00224     int len = 0;
00225     int found = 0;
00226
00227     if (_hspi->Instance == SPI2) {
00228         /* Buscar fin de comando ';' en el buffer de RX */
00229         for (int i = 0; i < SPI_BUF_SIZE; i++) {
00230             if (rxDMABuffer[i] == ';') {
00231                 found = 1;
00232                 len = i;           // bytes válidos (sin incluir ';')
00233                 break;
00234             }
00235         }
00236
00237         /* Respuesta por defecto: "ERR_NO_COMMAND;" */
00238         uint8_t resp[4] = { SPI_RESPONSE_ERR_NO_COMMAND, ';', 0, 0 };
00239
00240         if (found && len > 0) {
00241             SPI_ProcesarComando(rxDMABuffer, len, resp);
00242         }
00243
00244         txDMABuffer[0] = resp[0];
00245         txDMABuffer[1] = resp[1];
00246         txDMABuffer[2] = resp[2];
00247         txDMABuffer[3] = resp[3];
00248
00249         /* Limpiar RX para próxima captura */
00250         memset(rxDMABuffer, 0, sizeof(rxDMABuffer));
00251
00252         /* Reiniciar ciclo DMA full-duplex (no bloqueante) */
00253         HAL_SPI_TransmitReceive_DMA(hspi, txDMABuffer, rxDMABuffer, SPI_TRANSITION_SIZE);
00254     }
00255 }
00256
00257 void SPI_Init(void* _hspi) {
00258     /* Prioridades/enable de DMA para SPI2: TX (CH4), RX (CH5) */
00259     HAL_NVIC_SetPriority(DMA1_Channel4_IRQn, 2, 0);
00260     HAL_NVIC_EnableIRQ(DMA1_Channel4_IRQn);
00261
00262     HAL_NVIC_SetPriority(DMA1_Channel5_IRQn, 2, 0);
00263     HAL_NVIC_EnableIRQ(DMA1_Channel5_IRQn);
00264
00265     /* Guardar handle de SPI */
00266     hspi = (SPI_HandleTypeDef*)_hspi;
00267
00268     /* Inicialización básica de buffers */
00269     memset(rxDMABuffer, 0, sizeof(rxDMABuffer));
00270     memset(txDMABuffer, 0, sizeof(txDMABuffer));
00271
00272     /* Respuesta inicial por defecto: "OK;" (se enviará en la primera TxRx) */
00273     txDMABuffer[0] = SPI_RESPONSE_OK;
00274     txDMABuffer[1] = ';';
00275
00276     /* Iniciar ciclo DMA full-duplex no bloqueante */
00277     HAL_SPI_TransmitReceive_DMA(hspi, txDMABuffer, rxDMABuffer, SPI_TRANSITION_SIZE);
00278 }

```

## 10.21. Referencia del archivo Modules/SPI\_Interfase/SPIModule.h

Interfaz del módulo SPI (esclavo) con DMA.

### Enumeraciones

- enum SPI\_Request {
 SPI\_REQUEST\_START = 10, SPI\_REQUEST\_STOP, SPI\_REQUEST\_SET\_FREQ, SPI\_REQUEST\_SET\_ACCEL
 ,
 SPI\_REQUEST\_SET\_DESACCEL, SPI\_REQUEST\_SET\_DIR, SPI\_REQUEST\_GET\_FREQ, SPI\_REQUEST\_GET\_ACCEL
 ,
 SPI\_REQUEST\_GET\_DESACCEL, SPI\_REQUEST\_GET\_DIR, SPI\_REQUEST\_IS\_STOP, SPI\_REQUEST\_EMERGENCY
 ,
 SPI\_REQUEST\_RESPONSE = 0x50
 }

*Comandos aceptados por el esclavo a través de SPI.*

```
■ enum SPI_Response {
    SPI_RESPONSE_OK = 0xFF, 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_ERR_EMERGENCY_ACTIVE, SPI_LAST_VALUE
}
```

*Códigos de respuesta emitidos por el esclavo STM32.*

## Funciones

- void [SPI\\_Init](#) (void \*[hspi](#))  
*Inicializa el módulo SPI esclavo con DMA y deja activa la primera transacción.*

### 10.21.1. Descripción detallada

Interfaz del módulo SPI (esclavo) con DMA.

El maestro (ESP32) envía solicitudes [SPI\\_Request](#) y el STM32 responde con un código [SPI\\_Response](#) en la **siguiente** transacción DMA.

Este módulo actúa como capa de transporte sobre la lógica de estados implementada en [GestorEstados\\_Action](#) (ver [SystemAction](#) y [SystemState](#)).

#### Formato típico de tramas

- **Master → Slave:** CMD [DATOS...] ';' (ver [SPI\\_Request](#))
- **Slave → Master:** RESP [';'] [opcional: datos] (ver [SPI\\_Response](#))

Ver también

[GestorEstados\\_Action](#)  
[SystemAction](#)  
[SystemState](#)  
[SystemActionResponse](#)

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

### 10.21.2. Documentación de enumeraciones

#### 10.21.2.1. SPI\_Request

```
enum SPI_Request
```

Comandos aceptados por el esclavo a través de SPI.

Cada entrada se mapea a una acción de [SystemAction](#) procesada por [GestorEstados\\_Action](#). Los comandos \*\*← SET\_\*\*\* llevan datos; \*\*GET\_\*\*\* devuelven valores.

#### Valores de enumeraciones

---

SPI_REQUEST_START	
SPI_REQUEST_STOP	Solicita arranque → equivale a <a href="#">ACTION_START</a> .
SPI_REQUEST_SET_FREC	Solicita parada → equivale a <a href="#">ACTION_STOP</a> .
SPI_REQUEST_SET_ACCEL	Setea frecuencia de régimen → <a href="#">ACTION_SET_FREC</a> (requiere datos).
SPI_REQUEST_SET_DESACEL	Setea aceleración → <a href="#">ACTION_SET_ACCEL</a> (requiere datos).
SPI_REQUEST_SET_DIR	Setea desaceleración → <a href="#">ACTION_SET_DESACEL</a> (requiere datos).
SPI_REQUEST_GET_FREC	Setea dirección de giro → <a href="#">ACTION_SET_DIR</a> (requiere datos).
SPI_REQUEST_GET_ACCEL	Consulta frecuencia actual → lectura sin cambio de <a href="#">SystemState</a> .
SPI_REQUEST_GET_DESACEL	Consulta aceleración actual.
SPI_REQUEST_GET_DIR	Consulta desaceleración actual.
SPI_REQUEST_IS_STOP	Consulta dirección actual.
SPI_REQUEST_EMERGENCY	Consulta si el motor está detenido → usa <a href="#">ACTION_IS_MOTOR_STOP</a> .
SPI_REQUEST_RESPONSE	Fuerza emergencia → <a href="#">ACTION_EMERGENCY</a> .

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

#### 10.21.2.2. SPI\_Response

```
enum SPI_Response
```

Códigos de respuesta emitidos por el esclavo STM32.

Conceptualmente mapean los resultados de [GestorEstados\\_Action](#) (ver [SystemActionResponse](#)) hacia el enlace SPI. En comandos \*\*GET\_\*\*\* pueden incluir bytes de datos adicionales.

#### Valores de enumeraciones

SPI_RESPONSE_OK	
SPI_RESPONSE_ERR	Operación válida/exitosa → similar a <a href="#">ACTION_RESP_OK</a> .
SPI_RESPONSE_ERR_CMD_UNKNOWN	Error genérico → similar a <a href="#">ACTION_RESP_ERR</a> .
SPI_RESPONSE_ERR_NO_COMMAND	Comando desconocido (no mapea a acción válida).
SPI_RESPONSE_ERR_MOVING	Trama sin comando o sin ':'.
SPI_RESPONSE_ERR_NOT_MOVING	Incompatible por movimiento → similar a <a href="#">ACTION_RESP_MOVING</a> .
SPI_RESPONSE_ERR_DATA_MISSING	Incompatible por estar detenido → similar a <a href="#">ACTION_RESP_NOT_MOVING</a> .
SPI_RESPONSE_ERR_DATA_INVALID	Faltan datos en un comando que los requiere.
SPI_RESPONSE_ERR_DATA_OUT_RANGE	Datos inválidos.
SPI_RESPONSE_ERR_EMERGENCY_ACTIVE	Datos fuera de rango → similar a <a href="#">ACTION_RESP_OUT_RANGE</a> .
SPI_LAST_VALUE	No permitido en emergencia → similar a <a href="#">ACTION_RESP_EMERGENCY_ACTIVE</a> . Marcador final (no usar como respuesta).

Definición en la línea 53 del archivo [SPIModule.h](#).

### 10.21.3. Documentación de funciones

#### 10.21.3.1. SPI\_Init()

```
void SPI_Init (
    void * _hspi)
```

Inicializa el módulo SPI esclavo con DMA y deja activa la primera transacción.

- Configura prioridades/enable de IRQ para DMA1 Channel 4/5 (Tx/Rx).
- Limpia buffers RX/TX y precarga una respuesta por defecto ([SPI\\_RESPONSE\\_OK](#)).
- Llama a [HAL\\_SPI\\_TransmitReceive\\_DMA\(\)](#) para iniciar el ciclo continuo DMA (el tamaño de paquete es el configurado en el fuente, p.ej. [SPI\\_TRANSMITION\\_SIZE](#)).
- El parseo y la construcción de [SPI\\_Response](#) se realizan en el callback [HAL\\_SPI\\_TxRxCpltCallback](#), invocando [GestorEstados\\_Action](#) según corresponda.

#### Parámetros

in	<i>hspi</i>	Handler de SPI inicializado por HAL (tipo <a href="#">SPI_HandleTypeDef</a> *, p.ej. & <a href="#">hspi2</a> ).
----	-------------	---

#### Ver también

[GestorEstados\\_Action](#)  
[SPI\\_Request](#)  
[SPI\\_Response](#)

Definición en la línea [257](#) del archivo [SPIModule.c](#).

## 10.22. SPIModule.h

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

```
00001
00019
00020 #ifndef SPI_MODULE_H_
00021 #define SPI_MODULE_H_
00022
00030 typedef enum {
00031     SPI_REQUEST_START      = 10,
00032     SPI_REQUEST_STOP,
00033     SPI_REQUEST_SET_FREC,
00034     SPI_REQUEST_SET_ACCEL,
00035     SPI_REQUEST_SET_DESACEL,
00036     SPI_REQUEST_SET_DIR,
00037     SPI_REQUEST_GET_FREC,
00038     SPI_REQUEST_GET_ACCEL,
00039     SPI_REQUEST_GET_DESACEL,
00040     SPI_REQUEST_GET_DIR,
00041     SPI_REQUEST_IS_STOP,
00042     SPI_REQUEST_EMERGENCY,
00043     SPI_REQUEST_RESPONSE   = 0x50
00044 } SPI_Request;
00045
00053 typedef enum {
00054     SPI_RESPONSE_OK = 0xFF,
00055     SPI_RESPONSE_ERR = 0xA0,
00056     SPI_RESPONSE_ERR_CMD_UNKNOWN,
```

```

00057     SPI_RESPONSE_ERR_NO_COMMAND,
00058     SPI_RESPONSE_ERR_MOVING,
00059     SPI_RESPONSE_ERR_NOT_MOVING,
00060     SPI_RESPONSE_ERR_DATA_MISSING,
00061     SPI_RESPONSE_ERR_DATA_INVALID,
00062     SPI_RESPONSE_ERR_DATA_OUT_RANGE,
00063     SPI_RESPONSE_ERR_EMERGENCY_ACTIVE,
00064     SPI_LAST_VALUE
00065 } SPI_Response;
00066
00084 void SPI_Init(void* hspi);
00085
00086 #endif /* SPI_MODULE_H */

```

## 10.23. Referencia del archivo Src/main.c

Punto de entrada del firmware y configuración de periféricos.

```

#include "main.h"
#include <stdio.h>
#include "../Modules/Gestor_SVM/GestorSVM.h"
#include "../Modules/Gestor_Timers/GestorTimers.h"
#include "../Modules/Gestor_Estados/GestorEstados.h"
#include "../Modules/SPI_Interfase/SPIModule.h"

```

### Funciones

- static void **SystemClock\_Config** (void)  
*Configura el clock del sistema a 72 MHz desde HSE=8 MHz con PLLx9.*
- static void **MX\_GPIO\_Init** (void)  
*Inicializa GPIOA y GPIOB.*
- static void **MX\_DMA\_Init** (void)  
*Habilita el clock del DMA1 y sus interrupciones de canal 4 y 5.*
- static void **MX\_TIM3\_Init** (void)  
*Función inicialización del timer 3 (SVM).*
- static void **MX\_TIM2\_Init** (void)  
*Función de inicialización del timer 2 (Pre cálculo).*
- static void **MX\_USART1\_UART\_Init** (void)  
*Inicialización de la UART 1 (DEBUG).*
- static void **MX\_SPI2\_Init** (void)  
*SPI2 Initialization Function (HMI).*
- int **main** (void)  
*Función de entrada del sistema.*
- void **Error\_Handler** (void)  
*Manejador global de errores fatales.*

### Variables

- SPI\_HandleTypeDef hspi2
- DMA\_HandleTypeDef hdma\_spi2\_tx
- DMA\_HandleTypeDef hdma\_spi2\_rx
- TIM\_HandleTypeDef htim2
- TIM\_HandleTypeDef htim3
- USART\_HandleTypeDef huart1

### 10.23.1. Descripción detallada

Punto de entrada del firmware y configuración de periféricos.

Inicializa reloj de sistema, GPIO, DMA, SPI2 (esclavo), USART1 (debug), y TIM2/TIM3 (cálculo y switching). Integra módulos GestorSVM, GestorTimers, GestorEstados y SPI module. Disposición de transistores, los transistores son los Q1, Q2, Q3, Q4, Q5 y Q6 | | | Q1 Q2 Q3 | | | | | Q4 Q5 Q6 | | | Como esto está controlado por drivers de mosfets, entonces vamos a poner un 1 para activar el transistor superior y un cero para el transistor inferior. Vamos a tener ocho vectores en que vamos a trabajar: v0, v1, v2, v3, v4, v5, v6 y v7. Siendo el v0 y v7 dos vectores nulos. Cada uno de estos cuadrantes tiene un valor que indica el valor de los transistores:

- v0 = 000,
- v1 = 001,
- v2 = 010,
- v3 = 011,
- v4 = 100,
- v5 = 101,
- v6 = 110,
- v7 = 111. Ahora vamos a detallar por cuadrante como nos va a quedar y que cambia en cada cuadrante
  - Quad 1: v0, v1, v3, v7 => P3: High, P2: High, P1: High, P0: Low, P2: Low, P3: Low
  - Quad 2: v0, v2, v3, v7 => P2: High, P3: High, P1: High, P0: Low, P3: Low, P2: Low
  - Quad 3: v0, v2, v6, v7 => P2: High, P1: High, P3: High, P0: Low, P1: Low, P2: Low
  - Quad 4: v0, v4, v6, v7 => P1: High, P2: High, P3: High, P0: Low, P2: Low, P1: Low
  - Quad 5: v0, v4, v5, v7 => P1: High, P3: High, P2: High, P0: Low, P3: Low, P1: Low
  - Quad 6: v0, v1, v5, v7 => P3: High, P1: High, P2: High, P0: Low, P1: Low, P3: Low

Definición en el archivo [main.c](#).

### 10.23.2. Documentación de funciones

#### 10.23.2.1. Error\_Handler()

```
void Error_Handler (
    void )
```

Manejador global de errores fatales.

Manejador genérico de errores.

Detiene la ejecución en un bucle y puede ser utilizado para reportar/diagnosticar condiciones críticas. Implementado en [main.c](#).

Deshabilita IRQs y entra en bucle infinito para provocar reset por IWDG si está activo.

Definición en la línea [375](#) del archivo [main.c](#).

### 10.23.2.2. main()

```
int main (
    void )
```

Función de entrada del sistema.

La función main inicializa la estructura de configuración, inicializa los periféricos y el gestor de estados. Al terminar su trabajo queda en un while(1) con la sentencia WFI para reducir el consumo de CPU. 1) HAL\_Init() 2) **SystemClock\_Config()** 3) Carga configuración SVM (frecuencia de switching, ref, etc.). 4) Inicializa manejador de timers (GestorTimers\_Init). 5) Inicializa periféricos (GPIO, DMA, TIM3, USART1, TIM2, SPI2) y driver SPI. 6) Notifica fin de init al Gestor de Estados (ACTION\_INIT\_DONE). 7) Entra en lazo con WFI para ahorrar CPU, atendiendo a interrupciones.

#### Valores devueltos

<i>int</i>	Sin uso
------------	---------

Configuración del MCU. Reset of all peripherals, Initializes the Flash interface and the Systick.

Configuración del módulo SVM

Frecuencia de 2.511kHz para el switch del timer 3 entre subida y bajada del timer 3

Frecuencia de salida

Sentido horario

Aceleración del sistema por default [Hz/seg]

Desaceleración del sistema por default [Hz/seg]

Definición en la línea [132](#) del archivo [main.c](#).

### 10.23.2.3. MX\_DMA\_Init()

```
void MX_DMA_Init (
    void ) [static]
```

Habilita el clock del DMA1 y sus interrupciones de canal 4 y 5.

Habilita el reloj del **DMA1** y registra en el NVIC las IRQ de **DMA1\_Channel4** y **DMA1\_Channel5** con prioridad 2. En F1, estos canales suelen mapearse a **SPI2\_RX (Ch4)** y **SPI2\_TX (Ch5)**, por lo que esta rutina deja listo el DMA para que llamadas como `HAL_SPI_TransmitReceive_DMA()` configuren y lancen las transferencias. No configura direcciones, tamaños ni modos del DMA; solo habilita el clock y las IRQ.

#### Nota

Anteriormente la prioridad era 0. Luego pasó a 2 ya que en el SPI se configura en 2.

Definición en la línea [337](#) del archivo [main.c](#).

#### 10.23.2.4. MX\_GPIO\_Init()

```
void MX_GPIO_Init (
    void ) [static]
```

Inicializa GPIOA y GPIOB.

Configura los pines de los puertos A y B como salidas push pull, sin pull up y como pines de baja frecuencia. Los pines configurados son los que controlan el modulador y salidas (leds de estado). Las entradas digitales (termoswitch y botón de stop) no son configuradas ya que serán señales controladas por el HMI.

Definición en la línea [345](#) del archivo [main.c](#).

#### 10.23.2.5. MX\_SPI2\_Init()

```
void MX_SPI2_Init (
    void ) [static]
```

SPI2 Initialization Function (HMI).

Configura al SPI2 como esclavo, dependiendo de la comunicación con el maestro, el ESP32 para poder trascender información. Inicializa un tamaño de 8 bits de datos, con dos líneas de comunicación (MISO y MOSI) comenzando el byte por bit más significativo. No utiliza el módulo CRC, el latch del dato se toma en el flanco decreciente del clock y la línea en reposo queda en estado alto. El pin de selección de esclavo es utilizado por el maestro para iniciar la comunicación.

Definición en la línea [208](#) del archivo [main.c](#).

#### 10.23.2.6. MX\_TIM2\_Init()

```
void MX_TIM2_Init (
    void ) [static]
```

Función de inicialización del timer 2 (Pre cálculo).

El timer funcionará a 10KHz tratando de adelantar los cálculos del timer 3, permitiendo agilizar la carga de sus contadores y evitando que haya problemas en la señal de salida.

Definición en la línea [225](#) del archivo [main.c](#).

#### 10.23.2.7. MX\_TIM3\_Init()

```
void MX_TIM3_Init (
    void ) [static]
```

Función inicialización del timer 3 (SVM).

El timer 3 es utilizado para ejecutar el switching de los pines del variador de frecuencia. Trabaja con un timer de 0 a 256 utilizando el clock interno como fuente de conteo dando un período de 398.22us ( $f = 2511\text{Hz}$ ) y 4 canales de interrupción CCR1-4 que se irán corriendo conforme vaya avanzando la señal y cambiando la tensión de salida buscada de acuerdo a lo que dicte el estado del módulo SVM.

Definición en la línea [264](#) del archivo [main.c](#).

### 10.23.2.8. MX\_USART1\_UART\_Init()

```
void MX_USART1_UART_Init (
    void ) [static]
```

Inicialización de la UART 1 (DEBUG).

El periférico se utilizará como puerto de debug. Solo tendrá pines de transmisión y recepción sin control de flujo; con configuración 115200,8,N,1

Definición en la línea [323](#) del archivo [main.c](#).

### 10.23.2.9. SystemClock\_Config()

```
void SystemClock_Config (
    void ) [static]
```

Configura el clock del sistema a 72 MHz desde HSE=8 MHz con PLL×9.

Activa HSE, configura PLL (source=HSE, mul=×9) y selecciona SYSCLK=PLL. AHB=72 MHz, APB2=72 MHz (div1), APB1=36 MHz (div2), FLASH\_LATENCY=2.

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

## 10.23.3. Documentación de variables

### 10.23.3.1. hdma\_spi2\_rx

```
DMA_HandleTypeDef hdma_spi2_rx
```

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

### 10.23.3.2. hdma\_spi2\_tx

```
DMA_HandleTypeDef hdma_spi2_tx
```

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

### 10.23.3.3. hspi2

```
SPI_HandleTypeDef hspi2
```

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

### 10.23.3.4. htim2

```
TIM_HandleTypeDef htim2
```

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

### 10.23.3.5. htim3

```
TIM_HandleTypeDef htim3
```

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

### 10.23.3.6. huart1

```
UART_HandleTypeDef huart1
```

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

## 10.24. main.c

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

```
00001
00033
00034 #include "main.h"
00035 #include <stdio.h>
00036
00037 #include "../Modules/Gestor_SVM/GestorSVM.h"
00038 #include "../Modules/Gestor_Timers/GestorTimers.h"
00039 #include "../Modules/Gestor_Estados/GestorEstados.h"
00040 #include "../Modules/SPI_Interfase/SPIModule.h"
00041
00042 SPI_HandleTypeDef hspi2;
00043 DMA_HandleTypeDef hdma_spi2_tx;
00044 DMA_HandleTypeDef hdma_spi2_rx;
00045 TIM_HandleTypeDef htim2;
00046 TIM_HandleTypeDef htim3;
00047 UART_HandleTypeDef huart1;
00048
00055 static void SystemClock_Config(void);
00056
00064 static void MX_GPIO_Init(void);
00065
00076 static void MX_DMA_Init(void);
00077
00085 static void MX_TIM3_Init(void);
00086
00094 static void MX_TIM2_Init(void);
00095
00103 static void MX_USART1_UART_Init(void);
00104
00112 static void MX_SPI2_Init(void);
00113
00132 int main(void) {
00133
00134     HAL_Init();
00135     SystemClock_Config();
00136
00137
00138     ConfiguracionSVM config = {
00139         .frec_switch = 2511,
00140         .frecReferencia = 50,
00141         .direccionRotacion = 1,
00142         .acel = 5,
00143         .desacel = 3,
00144     };
00145
00146 // Cargamos los numeros de los pines
00147 config.puerto_señal_pierna[0] = GPIO_PIN_1;
00148 config.puerto_señal_pierna[1] = GPIO_PIN_3;
00149 config.puerto_señal_pierna[2] = GPIO_PIN_5;
00150 config.puerto_encen_pierna[0] = GPIO_PIN_2;
00151 config.puerto_encen_pierna[1] = GPIO_PIN_4;
00152 config.puerto_encen_pierna[2] = GPIO_PIN_6;
00153 GestorSVM_SetConfiguration(&config);
00154
00155 // Initialize all configured peripherals
00156 MX_GPIO_Init();
00157 MX_DMA_Init();
```

```

00158     MX_TIM3_Init();
00159     MX_TIM2_Init();
00160     MX_USART1_UART_Init();
00161     MX_SPI2_Init();
00162     SPI_Init(&hspi2);
00163
00164     // Inicio del gestor de timers
00165     GestorTimers_Init(&htim3, &htim2);
00166
00167     // Informamos al gestor de estados que finalizo la inicializacion
00168     // Esta debe ser la ultima llama de funcion del init
00169     GestorEstados_Action(ACTION_INIT_DONE, 0);
00170     printf("Config done\n");
00171
00172     __HAL_DBGMCU_FREEZE_TIM3();
00173     __HAL_DBGMCU_FREEZE_TIM2();
00174
00175     while (1) {
00176         __WFI();
00177     }
00178 }
00179
00180 static void SystemClock_Config(void) {
00181     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
00182     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
00183
00184     // Initializes the RCC Oscillators according to the specified parameters in the RCC_OscInitTypeDef structure.
00185     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
00186     RCC_OscInitStruct.HSEState = RCC_HSE_ON;
00187     RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
00188     RCC_OscInitStruct.HSISite = RCC_HSI_ON;
00189     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
00190     RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
00191     RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
00192     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK) {
00193         Error_Handler();
00194     }
00195
00196     // Initializes the CPU, AHB and APB buses clocks
00197     RCC_ClkInitStruct.ClockType =
00198         RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
00199     RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
00200     RCC_ClkInitStruct.AHBC1KDivider = RCC_SYSCLK_DIV1;
00201     RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
00202     RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
00203
00204     if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK) {
00205         Error_Handler();
00206     }
00207
00208 static void MX_SPI2_Init(void) {
00209     hspi2.Instance = SPI2;
00210     hspi2.Init.Mode = SPI_MODE_SLAVE;
00211     hspi2.Init.Direction = SPI_DIRECTION_2LINES;
00212     hspi2.Init.DataSize = SPI_DATASIZE_8BIT;
00213     hspi2.Init.CLKPolarity = SPI_POLARITY_LOW;
00214     hspi2.Init.CLKPhase = SPI_PHASE_1EDGE;
00215     hspi2.Init.NSS = SPI_NSS_HARD_INPUT;
00216     hspi2.Init.FirstBit = SPI_FIRSTBIT_MSB;
00217     hspi2.Init.TIMode = SPI_TIMODE_DISABLE;
00218     hspi2.Init.CRCCalculation = SPI_CRCALCULATION_DISABLE;
00219     hspi2.Init.CRCPolynomial = 10;
00220     if (HAL_SPI_Init(&hspi2) != HAL_OK) {
00221         Error_Handler();
00222     }
00223 }
00224
00225 static void MX_TIM2_Init(void) {
00226
00227     TIM_ClockConfigTypeDef sClockSourceConfig = {0};
00228     TIM_MasterConfigTypeDef sMasterConfig = {0};
00229     TIM_OC_InitTypeDef sConfigOC = {0};
00230
00231     htim2.Instance = TIM2;
00232     htim2.Init.Prescaler = 0;
00233     htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
00234     htim2.Init.Period = 7199;
00235     htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
00236     htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
00237     if (HAL_TIM_Base_Init(&htim2) != HAL_OK) {
00238         Error_Handler();
00239     }
00240
00241     sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
00242     if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK) {

```

```
00243     Error_Handler();
00244 }
00245 if (HAL_TIM_OC_Init(&htim2) != HAL_OK) {
00246     Error_Handler();
00247 }
00248 sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
00249 sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
00250 if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK) {
00251     Error_Handler();
00252 }
00253 }
00254 sConfigOC.OCMode = TIM_OCMODE_TIMING;
00255 sConfigOC.Pulse = 5;
00256 sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
00257 sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
00258 if (HAL_TIM_OC_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_1) != HAL_OK) {
00259     Error_Handler();
00260 }
00261 }
00262 }
00263
00264 static void MX_TIM3_Init(void) {
00265     TIM_ClockConfigTypeDef sClockSourceConfig = {0};
00266     TIM_MasterConfigTypeDef sMasterConfig = {0};
00267     TIM_OC_InitTypeDef sConfigOC = {0};
00268
00269     htim3.Instance = TIM3;
00270     htim3.Init.Prescaler = 55;
00271     htim3.Init.CounterMode = TIM_COUNTERMODE_CENTERALIGNED3;
00272     htim3.Init.Period = 256;
00273     htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
00274     htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
00275     if (HAL_TIM_Base_Init(&htim3) != HAL_OK) {
00276         Error_Handler();
00277     }
00278
00279     sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
00280     if (HAL_TIM_ConfigClockSource(&htim3, &sClockSourceConfig) != HAL_OK) {
00281         Error_Handler();
00282     }
00283     if (HAL_TIM_OC_Init(&htim3) != HAL_OK) {
00284         Error_Handler();
00285     }
00286
00287     sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
00288     sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
00289     if (HAL_TIMEx_MasterConfigSynchronization(&htim3, &sMasterConfig) != HAL_OK) {
00290         Error_Handler();
00291     }
00292
00293     sConfigOC.OCMode = TIM_OCMODE_TIMING;
00294     sConfigOC.Pulse = 0;
00295     sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
00296     sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
00297     if (HAL_TIM_OC_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
00298     {
00299         Error_Handler();
00300     }
00301     __HAL_TIM_ENABLE_OCxPRELOAD(&htim3, TIM_CHANNEL_1);
00302     sConfigOC.Pulse = 20;
00303     if (HAL_TIM_OC_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_2) != HAL_OK)
00304     {
00305         Error_Handler();
00306     }
00307     __HAL_TIM_ENABLE_OCxPRELOAD(&htim3, TIM_CHANNEL_2);
00308     sConfigOC.Pulse = 80;
00309     if (HAL_TIM_OC_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_3) != HAL_OK)
00310     {
00311         Error_Handler();
00312     }
00313     __HAL_TIM_ENABLE_OCxPRELOAD(&htim3, TIM_CHANNEL_3);
00314     sConfigOC.Pulse = 240;
00315     if (HAL_TIM_OC_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_4) != HAL_OK)
00316     {
00317         Error_Handler();
00318     }
00319     __HAL_TIM_ENABLE_OCxPRELOAD(&htim3, TIM_CHANNEL_4);
00320
00321 }
00322
00323 static void MX_USART1_UART_Init(void) {
00324     huart1.Instance = USART1;
00325     huart1.Init.BaudRate = 115200;
00326     huart1.Init.WordLength = UART_WORDLENGTH_8B;
00327     huart1.Init.StopBits = UART_STOPBITS_1;
00328     huart1.Init.Parity = UART_PARITY_NONE;
00329     huart1.Init.Mode = UART_MODE_TX_RX;
```

```

00330     huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
00331     huart1.Init.OverSampling = UART_OVERSAMPLING_16;
00332     if (HAL_UART_Init(&huart1) != HAL_OK) {
00333         Error_Handler();
00334     }
00335 }
00336
00337 static void MX_DMA_Init(void) {
00338     __HAL_RCC_DMA1_CLK_ENABLE();
00339     HAL_NVIC_SetPriority(DMA1_Channel14_IRQn, 2, 0);
00340     HAL_NVIC_EnableIRQ(DMA1_Channel14_IRQn);
00341     HAL_NVIC_SetPriority(DMA1_Channel15_IRQn, 2, 0);
00342     HAL_NVIC_EnableIRQ(DMA1_Channel15_IRQn);
00343 }
00344
00345 static void MX_GPIO_Init(void) {
00346     GPIO_InitTypeDef GPIO_InitStruct = {0};
00347
00348     /* GPIO Ports Clock Enable */
00349     __HAL_RCC_GPIOD_CLK_ENABLE();
00350     __HAL_RCC_GPIOA_CLK_ENABLE();
00351     __HAL_RCC_GPIOB_CLK_ENABLE();
00352
00353     HAL_GPIO_WritePin(GPIOA, GPIO_U_IN|GPIO_U_SD|GPIO_V_IN|GPIO_V_SD|GPIO_W_IN|GPIO_W_SD,
00354     GPIO_PIN_RESET); //|GPIO_TERMO_SWITCH|GPIO_STOP_BUTTON, GPIO_PIN_RESET);
00355     HAL_GPIO_WritePin(GPIOB, GPIO_LED_STATE|GPIO_LED_ERROR, GPIO_PIN_RESET);
00356
00357     GPIO_InitStruct.Pin =
00358         GPIO_U_IN|GPIO_U_SD|GPIO_V_IN|GPIO_V_SD|GPIO_W_IN|GPIO_W_SD|GPIO_TERMO_SWITCH|GPIO_STOP_BUTTON;
00359     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
00360     GPIO_InitStruct.Pull = GPIO_NOPULL;
00361     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
00362     HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
00363
00364     GPIO_InitStruct.Pin = GPIO_LED_STATE|GPIO_LED_ERROR;
00365     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
00366     GPIO_InitStruct.Pull = GPIO_NOPULL;
00367     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
00368     HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
00369 }
00370
00371 void Error_Handler(void) {
00372     __disable_irq();
00373     while (1) {
00374     }
00375 }
00376
00377 #ifdef USE_FULL_ASSERT
00378 void assert_failed(uint8_t *file, uint32_t line);
00379 void assert_failed(uint8_t *file, uint32_t line) {
00380     printf("Wrong parameters value: file%s on line%d\r\n", file, line)
00381 }
00382 #endif /* USE_FULL_ASSERT */

```

## 10.25. Referencia del archivo

**C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDe←  
Frecuencia/Software/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"

```

## 10.25 Referencia del archivo

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/main.d97

```
#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"

## 10.25.1. 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](#).

## 10.25.2. Documentación de funciones

### 10.25.2.1. 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](#).

### 10.25.3. Documentación de variables

#### 10.25.3.1. TAG

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

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

## 10.26. 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", 4096,
00056                             NULL, 9, NULL, 0);
00056     xTaskCreatePinnedToCore(GPIO_interrupt_attendance_task, "GPIO_interrupt_attendance_task", 4096,
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 }
```

## Grupo5-VariadorDeFrecuencia/Software/Software

### ESP32/SPI\_ESP32\_TestModule/main.c

```
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "SPI_Module.h"
#include "UART_Module.h"
#include <string.h>
```

#### Funciones

- void [UART\\_SPI\\_TestModule \(\)](#)
- void [app\\_main \(void\)](#)
- void [UART\\_SPI\\_TestModule \(void \\*arg\)](#)

#### 10.27.1. Documentación de funciones

##### 10.27.1.1. app\_main()

```
void app_main (
    void )
```

Definición en la línea [10](#) del archivo [main.c](#).

##### 10.27.1.2. UART\_SPI\_TestModule() [1/2]

```
void UART_SPI_TestModule ()
```

##### 10.27.1.3. UART\_SPI\_TestModule() [2/2]

```
void UART_SPI_TestModule (
    void * arg)
```

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

## 10.28. main.c

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

```

00001 #include "freertos/FreeRTOS.h"
00002 #include "freertos/task.h"
00003 #include "SPI_Module.h"
00004 #include "UART_Module.h"
00005 #include <string.h>
00006
00007
00008 void UART_SPI_TestModule();
00009
00010 void app_main(void) {
00011
00012     UART_Init();
00013     SPI_Init();
00014     xTaskCreate(UART_SPI_TestModule, "uart_task", 4096, NULL, 5, NULL);
00015
00016 }
00017
00018 void UART_SPI_TestModule(void *arg) {
00019
00020     int uartBuffer[6];
00021
00022     int comando;
00023     int setVal;
00024     int getVal;
00025     SPI_Response retVal;
00026
00027     while (1) {
00028
00029         printf("\nIngresar comando: ");
00030         fflush(stdout);
00031
00032         memset(uartBuffer, 0, sizeof(uartBuffer));
00033
00034         UART_GetComando(uartBuffer);
00035
00036         // El comando siempre esta en el primer elemento
00037         comando = uartBuffer[0];
00038         // El valor si esta, es el segundo elemento
00039         setVal = uartBuffer[1];
00040
00041         printf("\n[Main] Uart comand: %d, uartSetVal: %d\n", comando, setVal);
00042
00043         retVal = SPI_SendRequest(comando, setVal, &getVal);
00044
00045         printf("[Main] RetVal: %d\n", retVal);
00046
00047         switch(retVal) {
00048             case SPI_RESPONSE_OK:
00049                 printf("Todo OK - RetVal: %d\n", getVal);
00050                 break;
00051             case SPI_RESPONSE_ERR:
00052                 printf("Error en mensaje\n");
00053                 break;
00054             case SPI_RESPONSE_ERR_CMD_UNKNOWN:
00055                 printf("Error: Comando desconocido\n");
00056                 break;
00057             case SPI_RESPONSE_ERR_NO_COMMAND:
00058                 printf("Error: Sin comando\n");
00059                 break;
00060             case SPI_RESPONSE_ERR_MOVING:
00061                 printf("Error: Motor en movimiento\n");
00062                 break;
00063             case SPI_RESPONSE_ERR_NOT_MOVING:
00064                 printf("Error: Motor detenido\n");
00065                 break;
00066             case SPI_RESPONSE_ERR_DATA_MISSING:
00067                 printf("Error: Falta de datos\n");
00068                 break;
00069             case SPI_RESPONSE_ERR_DATA_INVALID:
00070                 printf("Error: Datos invalidos\n");
00071                 break;
00072             case SPI_RESPONSE_ERR_DATA_OUT_RANGE:
00073                 printf("Error: Datos fuera de rango\n");
00074                 break;
00075             case SPI_RESPONSE_ERR_EMERGENCY_ACTIVE:
00076                 printf("Error: Emergencia activa\n");
00077                 break;
00078             default:
00079                 printf("Error: Codigo de respuesta desconocido\n");
00080                 break;
00081         }
00082 }
```

```
00083
00084      }
00085
00086  }
00087
```

## 10.29. Referencia del archivo Src/stm32f1xx\_hal\_msp.c

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

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

### Funciones

- void [HAL\\_MspInit](#) (void)
- void [HAL\\_SPI\\_MspInit](#) (SPI\_HandleTypeDef \*hspi)  
*SPI MSP Initialization This function configures the hardware resources used in this example.*
- void [HAL\\_SPI\\_MspDeInit](#) (SPI\_HandleTypeDef \*hspi)  
*SPI MSP De-Initialization This function freeze the hardware resources used in this example.*
- void [HAL\\_TIM\\_Base\\_MspInit](#) (TIM\_HandleTypeDef \*htim\_base)  
*TIM\_Base MSP Initialization This function configures the hardware resources used in this example.*
- void [HAL\\_TIM\\_Base\\_MspDeInit](#) (TIM\_HandleTypeDef \*htim\_base)  
*TIM\_Base MSP De-Initialization This function freeze the hardware resources used in this example.*
- void [HAL\\_UART\\_MspInit](#) (UART\_HandleTypeDef \*huart)  
*UART MSP Initialization This function configures the hardware resources used in this example.*
- void [HAL\\_UART\\_MspDeInit](#) (UART\_HandleTypeDef \*huart)  
*UART MSP De-Initialization This function freeze the hardware resources used in this example.*

### Variables

- DMA\_HandleTypeDef [hdma\\_spi2\\_tx](#)
- DMA\_HandleTypeDef [hdma\\_spi2\\_rx](#)

### 10.29.1. Descripción detallada

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

#### Atención

Copyright (c) 2025 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

Definición en el archivo [stm32f1xx\\_hal\\_msp.c](#).

## 10.29.2. Documentación de funciones

### 10.29.2.1. HAL\_MspInit()

```
void HAL_MspInit (
    void )
```

Initializes the Global MSP.

Definición en la línea 27 del archivo [stm32f1xx\\_hal\\_msp.c](#).

### 10.29.2.2. HAL\_SPI\_MspDeInit()

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

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

#### Parámetros

<i>hspi</i>	SPI handle pointer
-------------	--------------------

#### Valores devueltos

<i>None</i>	
-------------	--

SPI2 GPIO Configuration PB12 -----> SPI2\_NSS PB13 -----> SPI2\_SCK PB14 -----> SPI2\_MISO PB15 -----> SPI2\_MOSI

Definición en la línea 101 del archivo [stm32f1xx\\_hal\\_msp.c](#).

### 10.29.2.3. HAL\_SPI\_MspInit()

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

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

#### Parámetros

<i>hspi</i>	SPI handle pointer
-------------	--------------------

#### Valores devueltos

<i>None</i>	
-------------	--

SPI2 GPIO Configuration PB12 -----> SPI2\_NSS PB13 -----> SPI2\_SCK PB14 -----> SPI2\_MISO PB15 -----> SPI2\_MOSI

Definición en la línea 39 del archivo [stm32f1xx\\_hal\\_msp.c](#).

#### 10.29.2.4. HAL\_TIM\_Base\_MspDeInit()

```
void HAL_TIM_Base_MspDeInit (
    TIM_HandleTypeDef * htim_base)
```

TIM\_Base MSP De-Initialization This function freeze the hardware resources used in this example.

##### Parámetros

<i>htim_base</i>	TIM_Base handle pointer
------------------	-------------------------

##### Valores devueltos

<i>None</i>	
-------------	--

Definición en la línea 144 del archivo [stm32f1xx\\_hal\\_msp.c](#).

#### 10.29.2.5. HAL\_TIM\_Base\_MspInit()

```
void HAL_TIM_Base_MspInit (
    TIM_HandleTypeDef * htim_base)
```

TIM\_Base MSP Initialization This function configures the hardware resources used in this example.

##### Parámetros

<i>htim_base</i>	TIM_Base handle pointer
------------------	-------------------------

##### Valores devueltos

<i>None</i>	
-------------	--

Definición en la línea 126 del archivo [stm32f1xx\\_hal\\_msp.c](#).

#### 10.29.2.6. HAL\_UART\_MspDeInit()

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

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

##### Parámetros

<i>huart</i>	UART handle pointer
--------------	---------------------

##### Valores devueltos

None	
------	--

Definición en la línea 187 del archivo [stm32f1xx\\_hal\\_msp.c](#).

#### 10.29.2.7. HAL\_UART\_MspInit()

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

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

##### Parámetros

<i>huart</i>	UART handle pointer
--------------	---------------------

##### Valores devueltos

None	
------	--

Definición en la línea 160 del archivo [stm32f1xx\\_hal\\_msp.c](#).

### 10.29.3. Documentación de variables

#### 10.29.3.1. hdma\_spi2\_rx

```
DMA_HandleTypeDef hdma_spi2_rx [extern]
```

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

#### 10.29.3.2. hdma\_spi2\_tx

```
DMA_HandleTypeDef hdma_spi2_tx [extern]
```

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

## 10.30. stm32f1xx\_hal\_msp.c

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

```

00001
00018
00019 #include "main.h"
00020
00021 extern DMA_HandleTypeDef hdma_spi2_tx;
00022 extern DMA_HandleTypeDef hdma_spi2_rx;
00023
00027 void HAL_MspInit(void) {
00028     __HAL_RCC_AFIO_CLK_ENABLE();
00029     __HAL_RCC_PWR_CLK_ENABLE();
00030     __HAL_AFIO_REMAP_SWJ_NOJTAG();
00031 }
00032
00039 void HAL_SPI_MspInit(SPI_HandleTypeDef* hspi) {
00040     GPIO_InitTypeDef GPIO_InitStruct = {0};
00041     if(hspi->Instance==SPI2) {
00042         __HAL_RCC_SPI2_CLK_ENABLE();
00043
00044         __HAL_RCC_GPIOB_CLK_ENABLE();
00045         GPIO_InitStruct.Pin = GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_15;
00046         GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
00047         GPIO_InitStruct.Pull = GPIO_NOPULL;
00048         HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
00049
00050         GPIO_InitStruct.Pin = GPIO_PIN_14;
00051         GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
00052         GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
00053         HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
00054
00055         hdma_spi2_tx.Instance = DMA1_Channel15;
00056         hdma_spi2_tx.Init.Direction = DMA_MEMORY_TO_PERIPH;
00057         hdma_spi2_tx.Init.PeriphInc = DMA_PINC_DISABLE;
00058         hdma_spi2_tx.Init.MemInc = DMA_MINC_ENABLE;
00059         hdma_spi2_tx.Init.PeriphDataAlignment = DMA_PDATAALIGN_BYTE;
00060         hdma_spi2_tx.Init.MemDataAlignment = DMA_MDATAALIGN_BYTE;
00061         hdma_spi2_tx.Init.Mode = DMA_NORMAL;
00062         hdma_spi2_tx.Init.Priority = DMA_PRIORITY_LOW;
00063         if (HAL_DMA_Init(&hdma_spi2_tx) != HAL_OK) {
00064             Error_Handler();
00065         }
00066
00067         __HAL_LINKDMA(hspi,hdmatx,hdma_spi2_tx);
00068
00069         hdma_spi2_rx.Instance = DMA1_Channel14;
00070         hdma_spi2_rx.Init.Direction = DMA_PERIPH_TO_MEMORY;
00071         hdma_spi2_rx.Init.PeriphInc = DMA_PINC_DISABLE;
00072         hdma_spi2_rx.Init.MemInc = DMA_MINC_ENABLE;
00073         hdma_spi2_rx.Init.PeriphDataAlignment = DMA_PDATAALIGN_BYTE;
00074         hdma_spi2_rx.Init.MemDataAlignment = DMA_MDATAALIGN_BYTE;
00075         hdma_spi2_rx.Init.Mode = DMA_NORMAL;
00076         hdma_spi2_rx.Init.Priority = DMA_PRIORITY_LOW;
00077         if (HAL_DMA_Init(&hdma_spi2_rx) != HAL_OK) {
00078             Error_Handler();
00079         }
00080
00081         __HAL_LINKDMA(hspi,hdmarx,hdma_spi2_rx);
00082
00083         HAL_NVIC_SetPriority(SPI2_IRQn, 0, 0);
00084         HAL_NVIC_EnableIRQ(SPI2_IRQn);
00085     }
00086
00087 }
00088
00089
00090 }
00091
00092
00093 }
00094
00101 void HAL_SPI_MspDeInit(SPI_HandleTypeDef* hspi) {
00102     if(hspi->Instance==SPI2) {
00103         __HAL_RCC_SPI2_CLK_DISABLE();
00104
00105         HAL_GPIO_DeInit(GPIOB, GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15);
00106
00107         HAL_DMA_DeInit(hspi->hdmatx);
00108         HAL_DMA_DeInit(hspi->hdmarx);
00109
00110         HAL_NVIC_DisableIRQ(SPI2_IRQn);
00111     }
00112
00113 }
00114
00115
00116 }
00117
00118 }
00119
00126 void HAL_TIM_Base_MspInit(TIM_HandleTypeDef* htim_base) {
00127     if(htim_base->Instance==TIM2) {
00128         __HAL_RCC_TIM2_CLK_ENABLE();
00129         HAL_NVIC_SetPriority(TIM2_IRQn, 0, 0);
00130         HAL_NVIC_EnableIRQ(TIM2_IRQn);
00131     } else if(htim_base->Instance==TIM3) {

```

```

00132     __HAL_RCC_TIM3_CLK_ENABLE();
00133     HAL_NVIC_SetPriority(TIM3_IRQn, 0, 0);
00134     HAL_NVIC_EnableIRQ(TIM3_IRQn);
00135 }
00136 }
00137
00144 void HAL_TIM_Base_MspDeInit(TIM_HandleTypeDef* htim_base) {
00145     if(htim_base->Instance==TIM2) {
00146         __HAL_RCC_TIM2_CLK_DISABLE();
00147         HAL_NVIC_DisableIRQ(TIM2_IRQn);
00148     } else if(htim_base->Instance==TIM3) {
00149         __HAL_RCC_TIM3_CLK_DISABLE();
00150         HAL_NVIC_DisableIRQ(TIM3_IRQn);
00151     }
00152 }
00153
00160 void HAL_UART_MspInit(UART_HandleTypeDef* huart) {
00161     GPIO_InitTypeDef GPIO_InitStruct = {0};
00162     if(huart->Instance==USART1) {
00163         __HAL_RCC_USART1_CLK_ENABLE();
00164         __HAL_RCC_GPIOB_CLK_ENABLE();
00165
00166         GPIO_InitStruct.Pin = GPIO_PIN_6;
00167         GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
00168         GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
00169         HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
00170
00171         GPIO_InitStruct.Pin = GPIO_PIN_7;
00172         GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
00173         GPIO_InitStruct.Pull = GPIO_NOPULL;
00174         HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
00175
00176         __HAL_AFIO_REMAP_USART1_ENABLE();
00177     }
00178
00179 }
00180
00187 void HAL_UART_MspDeInit(UART_HandleTypeDef* huart) {
00188     if(huart->Instance==USART1) {
00189         __HAL_RCC_USART1_CLK_DISABLE();
00190
00191         HAL_GPIO_DeInit(GPIOB, GPIO_PIN_6|GPIO_PIN_7);
00192     }
00193 }
00194 }
```

## 10.31. Referencia del archivo Src/stm32f1xx\_it.c

Interrupt Service Routines.

```
#include "main.h"
#include "stm32f1xx_it.h"
#include "../Modules/Gestor_SVM/GestorSVM.h"
```

### Funciones

- void **NMI\_Handler** (void)
 

*This function handles Non maskable interrupt.*
- void **HardFault\_Handler** (void)
 

*This function handles Hard fault interrupt.*
- void **MemManage\_Handler** (void)
 

*This function handles Memory management fault.*
- void **BusFault\_Handler** (void)
 

*This function handles Prefetch fault, memory access fault.*
- void **UsageFault\_Handler** (void)
 

*This function handles Undefined instruction or illegal state.*
- void **SVC\_Handler** (void)

- This function handles System service call via SWI instruction.
- void [DebugMon\\_Handler](#) (void)
  - This function handles Debug monitor.
- void [PendSV\\_Handler](#) (void)
  - This function handles Pendable request for system service.
- void [SysTick\\_Handler](#) (void)
  - This function handles System tick timer.
- void [DMA1\\_Channel4\\_IRQHandler](#) (void)
  - This function handles DMA1 channel4 global interrupt.
- void [DMA1\\_Channel5\\_IRQHandler](#) (void)
  - This function handles DMA1 channel5 global interrupt.
- void [TIM2\\_IRQHandler](#) (void)
  - This function handles TIM2 global interrupt.
- void [TIM3\\_IRQHandler](#) (void)
  - This function handles TIM3 global interrupt.
- void [SPI2\\_IRQHandler](#) (void)
  - This function handles SPI2 global interrupt.

## Variables

- DMA\_HandleTypeDef [hdma\\_spi2\\_tx](#)
- DMA\_HandleTypeDef [hdma\\_spi2\\_rx](#)
- SPI\_HandleTypeDef [hspi2](#)
- TIM\_HandleTypeDef [htim2](#)
- TIM\_HandleTypeDef [htim3](#)

### 10.31.1. Descripción detallada

Interrupt Service Routines.

#### Atención

Copyright (c) 2025 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

Definición en el archivo [stm32f1xx\\_it.c](#).

### 10.31.2. Documentación de funciones

#### 10.31.2.1. BusFault\_Handler()

```
void BusFault_Handler (
    void )
```

This function handles Prefetch fault, memory access fault.

Definición en la línea 62 del archivo [stm32f1xx\\_it.c](#).

### 10.31.2.2. DebugMon\_Handler()

```
void DebugMon_Handler ( void )
```

This function handles Debug monitor.

Definición en la línea 89 del archivo [stm32f1xx\\_it.c](#).

### 10.31.2.3. DMA1\_Channel4\_IRQHandler()

```
void DMA1_Channel4_IRQHandler ( void )
```

This function handles DMA1 channel4 global interrupt.

Definición en la línea 110 del archivo [stm32f1xx\\_it.c](#).

### 10.31.2.4. DMA1\_Channel5\_IRQHandler()

```
void DMA1_Channel5_IRQHandler ( void )
```

This function handles DMA1 channel5 global interrupt.

Definición en la línea 117 del archivo [stm32f1xx\\_it.c](#).

### 10.31.2.5. HardFault\_Handler()

```
void HardFault_Handler ( void )
```

This function handles Hard fault interrupt.

Definición en la línea 42 del archivo [stm32f1xx\\_it.c](#).

### 10.31.2.6. MemManage\_Handler()

```
void MemManage_Handler ( void )
```

This function handles Memory management fault.

Definición en la línea 52 del archivo [stm32f1xx\\_it.c](#).

### 10.31.2.7. NMI\_Handler()

```
void NMI_Handler ( void )
```

This function handles Non maskable interrupt.

Definición en la línea 32 del archivo [stm32f1xx\\_it.c](#).

### 10.31.2.8. PendSV\_Handler()

```
void PendSV_Handler (
    void )
```

This function handles Pendable request for system service.

Definición en la línea 96 del archivo [stm32f1xx\\_it.c](#).

### 10.31.2.9. SPI2\_IRQHandler()

```
void SPI2_IRQHandler (
    void )
```

This function handles SPI2 global interrupt.

Definición en la línea 139 del archivo [stm32f1xx\\_it.c](#).

### 10.31.2.10. SVC\_Handler()

```
void SVC_Handler (
    void )
```

This function handles System service call via SWI instruction.

Definición en la línea 82 del archivo [stm32f1xx\\_it.c](#).

### 10.31.2.11. SysTick\_Handler()

```
void SysTick_Handler (
    void )
```

This function handles System tick timer.

Definición en la línea 103 del archivo [stm32f1xx\\_it.c](#).

### 10.31.2.12. TIM2\_IRQHandler()

```
void TIM2_IRQHandler (
    void )
```

This function handles TIM2 global interrupt.

Definición en la línea 124 del archivo [stm32f1xx\\_it.c](#).

### 10.31.2.13. TIM3\_IRQHandler()

```
void TIM3_IRQHandler (
    void )
```

This function handles TIM3 global interrupt.

Definición en la línea 132 del archivo [stm32f1xx\\_it.c](#).

#### 10.31.2.14. UsageFault\_Handler()

```
void UsageFault_Handler (
    void )
```

This function handles Undefined instruction or illegal state.

Definición en la línea [72](#) del archivo [stm32f1xx\\_it.c](#).

### 10.31.3. Documentación de variables

#### 10.31.3.1. hdma\_spi2\_rx

```
DMA_HandleTypeDef hdma_spi2_rx [extern]
```

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

#### 10.31.3.2. hdma\_spi2\_tx

```
DMA_HandleTypeDef hdma_spi2_tx [extern]
```

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

#### 10.31.3.3. hspi2

```
SPI_HandleTypeDef hspi2 [extern]
```

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

#### 10.31.3.4. htim2

```
TIM_HandleTypeDef htim2 [extern]
```

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

#### 10.31.3.5. htim3

```
TIM_HandleTypeDef htim3 [extern]
```

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

## 10.32. stm32f1xx\_it.c

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

```
00001 /* USER CODE BEGIN Header */
00017
00018 #include "main.h"
00019 #include "stm32f1xx_it.h"
00020
00021 #include "../Modules/Gestor_SVM/GestorSVM.h"
00022
00023 extern DMA_HandleTypeDef hdma_spi2_tx;
00024 extern DMA_HandleTypeDef hdma_spi2_rx;
00025 extern SPI_HandleTypeDef hspi2;
00026 extern TIM_HandleTypeDef htim2;
00027 extern TIM_HandleTypeDef htim3;
00028
00029 void NMI_Handler(void) {
00030
00031     while (1) {
00032
00033 }
00034
00035 }
00036
00037 }
00038
00039 void HardFault_Handler(void) {
00040
00041     while (1) {
00042
00043 }
00044
00045 }
00046
00047 }
00048
00049 void MemManage_Handler(void) {
00050
00051     while (1) {
00052
00053 }
00054
00055 }
00056
00057 }
00058
00059 void BusFault_Handler(void) {
00060
00061     while (1) {
00062
00063 }
00064
00065 }
00066
00067 }
00068
00069 void UsageFault_Handler(void) {
00070
00071     while (1) {
00072
00073 }
00074
00075 }
00076
00077 }
00078
00079 void SVC_Handler(void) {
00080
00081 }
00082
00083 void DebugMon_Handler(void) {
00084
00085 }
00086
00087 void PendSV_Handler(void) {
00088
00089 }
00090
00091
00092
00093 void SysTick_Handler(void) {
00094     HAL_IncTick();
00095 }
00096
00097
00098 }
00099
00100 void DMA1_Channel4_IRQHandler(void) {
00101     HAL_DMA_IRQHandler(&hdma_spi2_rx);
00102 }
00103
00104
00105
00106
00107 void DMA1_Channel5_IRQHandler(void) {
00108     HAL_DMA_IRQHandler(&hdma_spi2_tx);
00109 }
00110
00111
00112
00113
00114 void TIM2_IRQHandler(void) {
00115     GestorSVM_CalcInterrupt();
00116     HAL_TIM_IRQHandler(&htim2);
00117 }
00118
00119
00120
00121
00122 void TIM3_IRQHandler(void) {
00123     HAL_TIM_IRQHandler(&htim3);
00124 }
00125
00126
00127
00128
00129 void SPI2_IRQHandler(void) {
```

```
00140     HAL_SPI_IRQHandler(&hspi2);  
00141 }
```

### 10.33. Referencia del archivo Src/syscalls.c

STM32CubeIDE Minimal System calls file.

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

#### Funciones

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

#### Variables

- char \*\* `environ` = `__env`

### 10.33.1. Descripción detallada

STM32CubeIDE Minimal System calls file.

#### Autor

Auto-generated by STM32CubeIDE

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

#### Atención

Copyright (c) 2020-2022 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

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

### 10.33.2. Documentación de funciones

#### 10.33.2.1. \_\_attribute\_\_()

```
__attribute__ (
    weak) )
```

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

#### 10.33.2.2. \_\_io\_getchar()

```
int __io_getchar (
    void) [extern]
```

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

#### 10.33.2.3. \_\_io\_putchar()

```
int __io_putchar (
    int ch) [extern]
```

#### 10.33.2.4. \_close()

```
int _close (
    int file)
```

Definición en la línea [92](#) del archivo [syscalls.c](#).

### 10.33.2.5. `_execve()`

```
int _execve (
    char * name,
    char ** argv,
    char ** env)
```

Definición en la línea 169 del archivo [syscalls.c](#).

### 10.33.2.6. `_exit()`

```
void _exit (
    int status)
```

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

### 10.33.2.7. `_fork()`

```
int _fork (
    void )
```

Definición en la línea 163 del archivo [syscalls.c](#).

### 10.33.2.8. `_fstat()`

```
int _fstat (
    int file,
    struct stat * st)
```

Definición en la línea 99 del archivo [syscalls.c](#).

### 10.33.2.9. `_getpid()`

```
int _getpid (
    void )
```

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

### 10.33.2.10. `_isatty()`

```
int _isatty (
    int file)
```

Definición en la línea 106 del archivo [syscalls.c](#).

**10.33.2.11. \_kill()**

```
int _kill (
    int pid,
    int sig)
```

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

**10.33.2.12. \_link()**

```
int _link (
    char * old,
    char * new)
```

Definición en la línea 155 del archivo [syscalls.c](#).

**10.33.2.13. \_lseek()**

```
int _lseek (
    int file,
    int ptr,
    int dir)
```

Definición en la línea 112 del archivo [syscalls.c](#).

**10.33.2.14. \_open()**

```
int _open (
    char * path,
    int flags,
    ...)
```

Definición en la línea 120 del archivo [syscalls.c](#).

**10.33.2.15. \_stat()**

```
int _stat (
    char * file,
    struct stat * st)
```

Definición en la línea 148 del archivo [syscalls.c](#).

**10.33.2.16. \_times()**

```
int _times (
    struct tms * buf)
```

Definición en la línea 142 del archivo [syscalls.c](#).

### 10.33.2.17. `_unlink()`

```
int _unlink (
    char * name)
```

Definición en la línea 135 del archivo [syscalls.c](#).

### 10.33.2.18. `_wait()`

```
int _wait (
    int * status)
```

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

### 10.33.2.19. `initialise_monitor_handles()`

```
void initialise_monitor_handles ()
```

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

## 10.33.3. Documentación de variables

### 10.33.3.1. `environ`

```
char** environ = __env
```

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

## 10.34. `syscalls.c`

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

```
00001
00022
00023 /* Includes */
00024 #include <sys/stat.h>
00025 #include <stdlib.h>
00026 #include <errno.h>
00027 #include <stdio.h>
00028 #include <signal.h>
00029 #include <time.h>
00030 #include <sys/time.h>
00031 #include <sys/times.h>
00032
00033
00034 /* Variables */
00035 extern int __io_putchar(int ch) __attribute__((weak));
00036 extern int __io_getchar(void) __attribute__((weak));
00037
00038
00039 char *__env[1] = { 0 };
00040 char **environ = __env;
00041
00042
00043 /* Functions */
00044 void initialise_monitor_handles()
00045 {
00046 }
```

```
00048 int __getpid(void)
00049 {
00050     return 1;
00051 }
00052
00053 int __kill(int pid, int sig)
00054 {
00055     (void)pid;
00056     (void)sig;
00057     errno = EINVAL;
00058     return -1;
00059 }
00060
00061 void __exit (int status)
00062 {
00063     __kill(status, -1);
00064     while (1) {} /* Make sure we hang here */
00065 }
00066
00067 __attribute__((weak)) int __read(int file, char *ptr, int len)
00068 {
00069     (void)file;
00070     int DataIdx;
00071
00072     for (DataIdx = 0; DataIdx < len; DataIdx++)
00073     {
00074         *ptr++ = __io_getchar();
00075     }
00076
00077     return len;
00078 }
00079
00080 __attribute__((weak)) int __write(int file, char *ptr, int len)
00081 {
00082     (void)file;
00083     int DataIdx;
00084
00085     for (DataIdx = 0; DataIdx < len; DataIdx++)
00086     {
00087         __io_putchar(*ptr++);
00088     }
00089     return len;
00090 }
00091
00092 int __close(int file)
00093 {
00094     (void)file;
00095     return -1;
00096 }
00097
00098
00099 int __fstat(int file, struct stat *st)
00100 {
00101     (void)file;
00102     st->st_mode = S_IFCHR;
00103     return 0;
00104 }
00105
00106 int __isatty(int file)
00107 {
00108     (void)file;
00109     return 1;
00110 }
00111
00112 int __lseek(int file, int ptr, int dir)
00113 {
00114     (void)file;
00115     (void)ptr;
00116     (void)dir;
00117     return 0;
00118 }
00119
00120 int __open(char *path, int flags, ...)
00121 {
00122     (void)path;
00123     (void)flags;
00124     /* Pretend like we always fail */
00125     return -1;
00126 }
00127
00128 int __wait(int *status)
00129 {
00130     (void)status;
00131     errno = ECHILD;
00132     return -1;
00133 }
00134
```

```

00135 int _unlink(char *name)
00136 {
00137     (void)name;
00138     errno = ENOENT;
00139     return -1;
00140 }
00141
00142 int _times(struct tms *buf)
00143 {
00144     (void)buf;
00145     return -1;
00146 }
00147
00148 int _stat(char *file, struct stat *st)
00149 {
00150     (void)file;
00151     st->st_mode = S_IFCHR;
00152     return 0;
00153 }
00154
00155 int _link(char *old, char *new)
00156 {
00157     (void)old;
00158     (void)new;
00159     errno = EMLINK;
00160     return -1;
00161 }
00162
00163 int _fork(void)
00164 {
00165     errno = EAGAIN;
00166     return -1;
00167 }
00168
00169 int _execve(char *name, char **argv, char **env)
00170 {
00171     (void)name;
00172     (void)argv;
00173     (void)env;
00174     errno = ENOMEM;
00175     return -1;
00176 }

```

## 10.35. Referencia del archivo Src/sysmem.c

STM32CubelDE System Memory calls file.

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

### Funciones

- void \* [\\_sbrk](#) (ptrdiff\_t incr)  
*\_sbrk() allocates memory to the newlib heap and is used by malloc and others from the C library*

### Variables

- static uint8\_t \* [\\_sbrk\\_heap\\_end](#) = NULL

### 10.35.1. Descripción detallada

STM32CubeIDE System Memory calls file.

#### Autor

Generated by STM32CubeIDE

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

#### Atención

Copyright (c) 2022 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

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

### 10.35.2. Documentación de funciones

#### 10.35.2.1. `_sbrk()`

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

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

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

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

#### Parámetros

<code>incr</code>	Memory size
-------------------	-------------

#### Devuelve

Pointer to allocated memory

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

### 10.35.3. Documentación de variables

#### 10.35.3.1. \_\_sbrk\_heap\_end

```
uint8_t* __sbrk_heap_end = NULL [static]
```

Pointer to the current high watermark of the heap usage

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

## 10.36. sysmem.c

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

```
00001
00022
00023 /* Includes */
00024 #include <errno.h>
00025 #include <stdint.h>
00026
00030 static uint8_t * __sbrk_heap_end = NULL;
00031
00053 void *_sbrk(ptrdiff_t incr)
00054 {
00055     extern uint8_t _end; /* Symbol defined in the linker script */
00056     extern uint8_t _estack; /* Symbol defined in the linker script */
00057     extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker script */
00058     const uint32_t stack_limit = (uint32_t)&_estack - (uint32_t)&_Min_Stack_Size;
00059     const uint8_t *max_heap = (uint8_t *)stack_limit;
00060     uint8_t *prev_heap_end;
00061
00062     /* Initialize heap end at first call */
00063     if (NULL == __sbrk_heap_end)
00064     {
00065         __sbrk_heap_end = &_end;
00066     }
00067
00068     /* Protect heap from growing into the reserved MSP stack */
00069     if (__sbrk_heap_end + incr > max_heap)
00070     {
00071         errno = ENOMEM;
00072         return (void *)-1;
00073     }
00074
00075     prev_heap_end = __sbrk_heap_end;
00076     __sbrk_heap_end += incr;
00077
00078     return (void *)prev_heap_end;
00079 }
```

## 10.37. Referencia del archivo Src/system\_stm32f1xx.c

CMSIS Cortex-M3 Device Peripheral Access Layer System Source File.

```
#include "stm32f1xx.h"
```

#### defines

- `#define HSE_VALUE 8000000U`
- `#define HSI_VALUE 8000000U`

## Funciones

- void [SystemInit](#) (void)  
*Setup the microcontroller system Initialize the Embedded Flash Interface, the PLL and update the SystemCoreClock variable.*
- void [SystemCoreClockUpdate](#) (void)  
*Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.*

## Variables

- uint32\_t [SystemCoreClock](#) = 8000000
- const uint8\_t [AHBPrescTable](#) [16U] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
- const uint8\_t [APBPrescTable](#) [8U] = {0, 0, 0, 0, 1, 2, 3, 4}

### 10.37.1. Descripción detallada

CMSIS Cortex-M3 Device Peripheral Access Layer System Source File.

#### Autor

MCD Application Team

1. This file provides two functions and one global variable to be called from user application:
  - [SystemInit\(\)](#): Setups the system clock (System clock source, PLL Multiplier factors, AHB/APBx prescalers and Flash settings). This function is called at startup just after reset and before branch to main program. This call is made inside the "startup\_stm32f1xx\_xx.s" file.
  - SystemCoreClock variable: Contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.
  - [SystemCoreClockUpdate\(\)](#): Updates the variable SystemCoreClock and must be called whenever the core clock is changed during program execution.
2. After each device reset the HSI (8 MHz) is used as system clock source. Then [SystemInit\(\)](#) function is called, in "startup\_stm32f1xx\_xx.s" file, to configure the system clock before to branch to main program.
3. The default value of HSE crystal is set to 8 MHz (or 25 MHz, depending on the product used), refer to "HSE↔\_VALUE". When HSE is used as system clock source, directly or through PLL, and you are using different crystal you have to adapt the HSE value to your own configuration.

#### Atención

Copyright (c) 2017-2021 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

Definición en el archivo [system\\_stm32f1xx.c](#).

## 10.38. system\_stm32f1xx.c

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

```

00001
00045
00049
00053
00057
00058 #include "stm32f1xx.h"
00059
00063
00067
00071
00075
00076 #if !defined (HSE_VALUE)
00077     #define HSE_VALUE           80000000U
00079 #endif /* HSE_VALUE */
00080
00081 #if !defined (HSI_VALUE)
00082     #define HSI_VALUE           80000000U
00084 #endif /* HSI_VALUE */
00085
00087 #if defined(STM32F100xE) || defined(STM32F101xE) || defined(STM32F101xG) || defined(STM32F103xE) ||
00088     defined(STM32F103xG)
00089 /* #define DATA_IN_ExtSRAM */
00090 #endif /* STM32F100xE || STM32F101xE || STM32F101xG || STM32F103xE || STM32F103xG */
00091 /* Note: Following vector table addresses must be defined in line with linker
00092     configuration. */
00096 /* #define USER_VECT_TAB_ADDRESS */
00097
00098 #if defined(USER_VECT_TAB_ADDRESS)
00101 /* #define VECT_TAB_SRAM */
00102 #if defined(VECT_TAB_SRAM)
00103 #define VECT_TAB_BASE_ADDRESS    SRAM_BASE
00105 #define VECT_TAB_OFFSET        0x00000000U
00107 #else
00108 #define VECT_TAB_BASE_ADDRESS    FLASH_BASE
00110 #define VECT_TAB_OFFSET        0x00000000U
00112 #endif /* VECT_TAB_SRAM */
00113 #endif /* USER_VECT_TAB_ADDRESS */
00114
00115 /*****
```

---

```

00116
00120
00124
00128
00132
00133 /* This variable is updated in three ways:
00134     1) by calling CMSIS function SystemCoreClockUpdate()
00135     2) by calling HAL API function HAL_RCC_GetHCLKFreq()
00136     3) each time HAL_RCC_ClockConfig() is called to configure the system clock frequency
00137         Note: If you use this function to configure the system clock; then there
00138             is no need to call the 2 first functions listed above, since SystemCoreClock
00139             variable is updated automatically.
00140 */
00141 uint32_t SystemCoreClock = 8000000;
00142 const uint8_t AHBPrescTable[16U] = {0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9};
00143 const uint8_t APBPrescTable[8U] = {0, 0, 0, 1, 2, 3, 4};
00144
00148
00152
00153 #if defined(STM32F100xE) || defined(STM32F101xE) || defined(STM32F101xG) || defined(STM32F103xE) ||
00154     defined(STM32F103xG)
00155 #ifdef DATA_IN_ExtSRAM
00156     static void SystemInit_ExtMemCtl(void);
00157 #endif /* DATA_IN_ExtSRAM */
00158 #endif /* STM32F100xE || STM32F101xE || STM32F101xG || STM32F103xE || STM32F103xG */
00162
00166
00175 void SystemInit (void)
00176 {
00177 #if defined(STM32F100xE) || defined(STM32F101xE) || defined(STM32F101xG) || defined(STM32F103xE) ||
00178     defined(STM32F103xG)
00179     SystemInit_ExtMemCtl();
00180 #endif /* DATA_IN_ExtSRAM */
00181 }
00182
00183 /* Configure the Vector Table location -----*/
00184 #if defined(USER_VECT_TAB_ADDRESS)
00185     SCB->VTOR = VECT_TAB_BASE_ADDRESS | VECT_TAB_OFFSET; /* Vector Table Relocation in Internal SRAM. */
00186 #endif /* USER_VECT_TAB_ADDRESS */
00187 }
```

```

00188
00189     void SystemCoreClockUpdate (void)
00190 {
00191     uint32_t tmp = 0U, pllmull = 0U, pllsource = 0U;
00192
00193 #if defined(STM32F105xC) || defined(STM32F107xC)
00194     uint32_t prediv1source = 0U, prediv1factor = 0U, prediv2factor = 0U, pllmull = 0U;
00195 #endif /* STM32F105xC */
00196
00197 #if defined(STM32F100xB) || defined(STM32F100xE)
00198     uint32_t prediv1factor = 0U;
00199 #endif /* STM32F100xB or STM32F100xE */
00200
00201     /* Get SYSCLK source -----*/
00202     tmp = RCC->CFGR & RCC_CFGR_SWS;
00203
00204     switch (tmp)
00205     {
00206         case 0x00U: /* HSI used as system clock */
00207             SystemCoreClock = HSI_VALUE;
00208             break;
00209         case 0x04U: /* HSE used as system clock */
00210             SystemCoreClock = HSE_VALUE;
00211             break;
00212         case 0x08U: /* PLL used as system clock */
00213
00214             /* Get PLL clock source and multiplication factor -----*/
00215             pllmull = RCC->CFGR & RCC_CFGR_PLLMULL;
00216             pllsource = RCC->CFGR & RCC_CFGR_PLLSRC;
00217
00218 #if !defined(STM32F105xC) && !defined(STM32F107xC)
00219             pllmull = (pllmull >> 18U) + 2U;
00220
00221             if (pllsource == 0x00U)
00222             {
00223                 /* HSI oscillator clock divided by 2 selected as PLL clock entry */
00224                 SystemCoreClock = (HSI_VALUE >> 1U) * pllmull;
00225             }
00226             else
00227             {
00228 #if defined(STM32F100xB) || defined(STM32F100xE)
00229                 prediv1factor = (RCC->CFGR2 & RCC_CFGR2_PREDIV1) + 1U;
00230                 /* HSE oscillator clock selected as PREDIV1 clock entry */
00231                 SystemCoreClock = (HSE_VALUE / prediv1factor) * pllmull;
00232             #else
00233                 /* HSE selected as PLL clock entry */
00234                 if ((RCC->CFGR & RCC_CFGR_PLLXTPRE) != (uint32_t)RESET)
00235                     /* HSE oscillator clock divided by 2 */
00236                     SystemCoreClock = (HSE_VALUE >> 1U) * pllmull;
00237                 else
00238                     {
00239                         SystemCoreClock = HSE_VALUE * pllmull;
00240                     }
00241             #endif
00242         }
00243     }
00244
00245     #else
00246
00247         pllmull = pllmull >> 18U;
00248
00249         if (pllmull != 0x0DU)
00250         {
00251             pllmull += 2U;
00252         }
00253         else
00254             /* PLL multiplication factor = PLL input clock * 6.5 */
00255             pllmull = 13U / 2U;
00256
00257         if (pllsource == 0x00U)
00258         {
00259             /* HSI oscillator clock divided by 2 selected as PLL clock entry */
00260             SystemCoreClock = (HSI_VALUE >> 1U) * pllmull;
00261         }
00262         else
00263             /* PREDIV1 selected as PLL clock entry */
00264
00265             /* Get PREDIV1 clock source and division factor */
00266             prediv1source = RCC->CFGR2 & RCC_CFGR2_PREDIV1SRC;
00267             prediv1factor = (RCC->CFGR2 & RCC_CFGR2_PREDIV1) + 1U;
00268
00269             if (prediv1source == 0U)
00270             {
00271                 /* HSE oscillator clock selected as PREDIV1 clock entry */
00272                 SystemCoreClock = (HSE_VALUE / prediv1factor) * pllmull;
00273             }
00274             else
00275                 /* PLL2 clock selected as PREDIV1 clock entry */
00276
00277     }
00278
00279 #endif
00280
00281     pllmull = pllmull >> 18U;
00282
00283     if (pllmull != 0x0DU)
00284     {
00285         pllmull += 2U;
00286     }
00287     else
00288         /* PLL multiplication factor = PLL input clock * 6.5 */
00289         pllmull = 13U / 2U;
00290
00291     if (pllsource == 0x00U)
00292     {
00293         /* HSI oscillator clock divided by 2 selected as PLL clock entry */
00294         SystemCoreClock = (HSI_VALUE >> 1U) * pllmull;
00295     }
00296     else
00297         /* PREDIV1 selected as PLL clock entry */
00298
00299         /* Get PREDIV1 clock source and division factor */
00300         prediv1source = RCC->CFGR2 & RCC_CFGR2_PREDIV1SRC;
00301         prediv1factor = (RCC->CFGR2 & RCC_CFGR2_PREDIV1) + 1U;
00302
00303         if (prediv1source == 0U)
00304         {
00305             /* HSE oscillator clock selected as PREDIV1 clock entry */
00306             SystemCoreClock = (HSE_VALUE / prediv1factor) * pllmull;
00307         }
00308         else
00309             /* PLL2 clock selected as PREDIV1 clock entry */
00310
00311
00312
00313
00314
00315
00316
00317
00318
00319
00320
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01840
01841
01842
01843
01844
01845
01846
0
```

```

00310
00311     /* Get PREDIV2 division factor and PLL2 multiplication factor */
00312     prediv2factor = ((RCC->CFGR2 & RCC_CFGR2_PREDIV2) » 4U) + 1U;
00313     pll2mull = ((RCC->CFGR2 & RCC_CFGR2_PLL2MUL) » 8U) + 2U;
00314     SystemCoreClock = ((HSE_VALUE / prediv2factor) * pll2mull) / prediv1factor * pllmull;
00315 }
00316 }
00317 #endif /* STM32F105xC */
00318     break;
00319
00320     default:
00321     SystemCoreClock = HSI_VALUE;
00322     break;
00323 }
00324
00325 /* Compute HCLK clock frequency -----*/
00326 /* Get HCLK prescaler */
00327 tmp = AHBPrescTable[((RCC->CFGR & RCC_CFGR_HPRE) » 4U)];
00328 /* HCLK clock frequency */
00329 SystemCoreClock »= tmp;
00330 }
00331
00332 #if defined(STM32F100xE) || defined(STM32F101xE) || defined(STM32F101xG) || defined(STM32F103xE) ||
defined(STM32F103xG)
00333 #ifdef DATA_IN_ExtSRAM
00334 void SystemInit_ExtMemCtl(void)
00335 {
00336     __IO uint32_t tmpreg;
00337
00338     /* Enable FSMC clock */
00339     RCC->AHBENR = 0x00000114U;
00340
00341     /* Delay after an RCC peripheral clock enabling */
00342     tmpreg = READ_BIT(RCC->AHBENR, RCC_AHBENR_FSMCEN);
00343
00344     /* Enable GPIOD, GPIOE, GPIOF and GPIOG clocks */
00345     RCC->APB2ENR = 0x000001E0U;
00346
00347     /* Delay after an RCC peripheral clock enabling */
00348     tmpreg = READ_BIT(RCC->APB2ENR, RCC_APB2ENR_IOPDEN);
00349
00350     (void) (tmpreg);
00351
00352     /* ----- SRAM Data lines, NOE and NWE configuration -----*/
00353     /*----- SRAM Address lines configuration -----*/
00354     /*----- NOE and NWE configuration -----*/
00355     /*----- NE3 configuration -----*/
00356     /*----- NBL0, NBL1 configuration -----*/
00357
00358     GPIOD->CRL = 0x44BB44BBU;
00359     GPIOD->CRH = 0xBFFFFFFFU;
00360
00361     GPIOE->CRL = 0xB44444BBU;
00362     GPIOE->CRH = 0xBFFFFFFFU;
00363
00364     GPIOF->CRL = 0x44BBBBBBBBU;
00365     GPIOF->CRH = 0xB4444444U;
00366
00367     GPIOG->CRL = 0x44BBBBBBBBU;
00368     GPIOG->CRH = 0x444B4B444U;
00369
00370     /*----- FSMC Configuration -----*/
00371     /*----- Enable FSMC Bank1_SRAM Bank -----*/
00372
00373     FSMC_Bank1->BTCR[4U] = 0x000001091U;
00374     FSMC_Bank1->BTCR[5U] = 0x000110212U;
00375
00376 #endif /* DATA_IN_ExtSRAM */
00377 #endif /* STM32F100xE || STM32F101xE || STM32F101xG || STM32F103xE || STM32F103xG */
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403

```

## 10.39. Referencia del archivo

**C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDe  
Frecuencia/Software/LVFV\_ESP32/main/adc/adc.c**

Funcion de inicialización y tarea lectura del ADC.

```
#include <stdio.h>
#include <math.h>
#include "freertos/FreeRTOS.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

### 10.39.1. Descripción detallada

Funcion de inicialización y tarea lectura del ADC.

**Autor**

Andrenacci - Carra

**Versión**

2.0

**Fecha**

2025-11-06

Definición en el archivo [adc.c](#).

### 10.39.2. Documentación de «define»

#### 10.39.2.1. ADC\_ATTEN\_USED

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

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

#### 10.39.2.2. ADC\_CH\_GPIO34

```
#define ADC_CH_GPIO34 ADC_CHANNEL_6
```

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

#### 10.39.2.3. ADC\_CH\_GPIO35

```
#define ADC_CH_GPIO35 ADC_CHANNEL_7
```

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

#### 10.39.2.4. ADC\_CH\_GPIO36

```
#define ADC_CH_GPIO36 ADC_CHANNEL_0
```

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

#### 10.39.2.5. ADC\_CH\_GPIO39

```
#define ADC_CH_GPIO39 ADC_CHANNEL_3
```

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

#### 10.39.2.6. ADC\_PERIOD\_MS

```
#define ADC_PERIOD_MS 100
```

**Parámetros**

---

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

### Valores devueltos

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

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

#### 10.39.2.7. ADC\_UNIT\_USED

```
#define ADC_UNIT_USED ADC_UNIT_1
```

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

### 10.39.3. Documentación de funciones

#### 10.39.3.1. 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	uint	Unidad del ADC utilizada. Puede tomar valores ADC_UNIT_1 o ADC_UNIT_2
in	ch	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	atten	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	out	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](#).

#### **10.39.3.2. 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](#).

#### **10.39.3.3. 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](#).

#### **10.39.3.4. 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 encola 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"><li>• true: Si el sistema está inicializado y las mediciones pueden ser obtenidas</li><li>• false: Si el sistema todavía no inició y las mediciones que pudo haber obtenido fueron inválidas</li></ul> |
|---|--|

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

#### 10.39.4. Documentación de variables

##### 10.39.4.1. bus\_meas\_evt\_queue

```
QueueHandle_t bus_meas_evt_queue = NULL
```

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

##### 10.39.4.2. calibration\_3V3\_source

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

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

##### 10.39.4.3. calibration\_3V3\_source\_ok

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

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

##### 10.39.4.4. calibration\_5V\_source

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

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

##### 10.39.4.5. calibration\_5V\_source\_ok

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

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

##### 10.39.4.6. calibration\_bus\_voltage

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

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

#### 10.39.4.7. calibration\_bus\_voltage\_ok

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

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

#### 10.39.4.8. calibration\_current

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

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

#### 10.39.4.9. calibration\_current\_ok

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

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

#### 10.39.4.10. s\_adc

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

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

#### 10.39.4.11. TAG

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

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

### 10.40. adc.c

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

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

```

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

```

```

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

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

## 10.41. Referencia del archivo

### C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/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.*

#### 10.41.1. 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](#).

## 10.41.2. Documentación de funciones

### 10.41.2.1. 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](#).

### 10.41.2.2. 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](#).

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

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

## 10.42. 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
```

## 10.43. Referencia del archivo

[C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDe←  
Frecuencia/Software/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\_QUADRATIC\_VARIABLE 1
- #define FREC\_LINE\_IDX VARIABLE\_FIRST

- #define VBUS\_LINE\_INDX VARIABLE\_SECOND
- #define IBUS\_LINE\_INDX VARIABLE\_THIRD
- #define HINI\_LINE\_INDX VARIABLE\_FOURTH
- #define HFIN\_LINE\_INDX VARIABLE\_FIFTH
- #define EDIT\_FREQUENCY\_INDX VARIABLE\_FIRST
- #define EDIT\_ACCELERATION\_INDX VARIABLE\_SECOND
- #define EDIT\_DESACCELERATION\_INDX VARIABLE\_THIRD
- #define EDIT\_INPUT\_VARIATION\_INDX VARIABLE\_FOURTH
- #define EDIT\_TIME\_LINE\_INDX VARIABLE\_SECOND
- #define EDIT\_HINI\_LINE\_INDX VARIABLE\_THIRD
- #define EDIT\_HFIN\_LINE\_INDX VARIABLE\_FOURTH
- #define EDIT\_VBUS\_LINE\_INDX VARIABLE\_SECOND
- #define EDIT\_IBUS\_LINE\_INDX VARIABLE\_THIRD

## Enumeraciones

- enum screen\_selected\_e {
 SCREEN\_MAIN , SCREEN\_SELECT\_VARIABLE , SCREEN\_TIME\_EDIT , SCREEN\_SECURITY\_EDIT ,
 SCREEN\_FREQUENCY\_EDIT }

*Listado de posibles pantallas que puede ver el usuario.*

## Funciones

- static void i2c\_master\_init (void)
 

*Inicializa el bus I2C para comunicarse con el display SH1106.*
- static void sh1106\_clear\_buffer ()
 

*Vacia el buffer de la pantalla para iniciar un nuevo dibujo.*
- static esp\_err\_t sh1106\_refresh ()
 

*Función que imprime la pantalla con los elementos agregados por el usuario.*
- static void sh1106\_print\_frame ()
 

*Función que agrega al buffer de la pantalla un marco fijo que incluye la hora, la línea horizontal divisoria y el logo de la UTN.*
- static void sh1106\_time\_edit\_variables (time\_settings\_SH1106\_t \*variables)
 

*Imprime en pantalla las variables de tiempo a editar: Hora del sistema, hora de inicio y hora de fin.*
- static void sh1106\_security\_edit\_variables (security\_settings\_SH1106\_t \*variables)
 

*Imprime en pantalla las variables de seguridad a editar: Tensión mínima en la bus de contínua y corriente máxima en el bus de continua.*
- static void sh1106\_frequency\_edit\_variables (frequency\_settings\_SH1106\_t \*variables)
 

*Imprime en pantalla las variables de frecuencia a editar: Frecuencia de régimen, aceleración, desaceleración y variación de las entradas aisladas para seleccionar la velocidad de régimen.*
- static void sh1106\_select\_edit\_variables (sh1106\_variable\_lines\_e variable)
 

*Permite al usuario seleccionar entre las tres pantallas de edición de variables: Frecuencia, tiempo y seguridad.*
- static void sh1106\_splash\_screen ()
 

*Pantalla de splash que se imprime al iniciar el sistema.*
- static void sh1106\_main\_screen ()
 

*Pantalla principal del sistema. Allí se muestra la hora del sistema, la frecuencia a la que se encuentra el variador, la hora de inicio y fin de movimiento del motor.*
- static void sh1106\_print\_emergency ()
 

*Entrega el valor de frecuencia de régimen seteado por el usuario.*
- uint16\_t get\_system\_frequency ()
 

*Entrega el valor de aceleración seteado por el usuario.*
- uint16\_t get\_system\_acceleration ()
 

*Entrega el valor de desaceleración seteado por el usuario.*

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

## Variables

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

### 10.43.1. 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`.

### 10.43.2. Documentación de «define»

#### 10.43.2.1. EDIT\_ACCELERATION\_INDX

```
#define EDIT_ACCELERATION_INDX VARIABLE_SECOND
```

Definición en la línea 49 del archivo `display.c`.

#### 10.43.2.2. EDIT\_DESACCELERATION\_INDX

```
#define EDIT_DESACCELERATION_INDX VARIABLE_THIRD
```

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

#### 10.43.2.3. EDIT\_FREQUENCY\_INDX

```
#define EDIT_FREQUENCY_INDX VARIABLE_FIRST
```

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

#### 10.43.2.4. EDIT\_HFIN\_LINE\_INDX

```
#define EDIT_HFIN_LINE_INDX VARIABLE_FOURTH
```

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

#### 10.43.2.5. EDIT\_HINI\_LINE\_INDX

```
#define EDIT_HINI_LINE_INDX VARIABLE_THIRD
```

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

#### 10.43.2.6. EDIT\_IBUS\_LINE\_INDX

```
#define EDIT_IBUS_LINE_INDX VARIABLE_THIRD
```

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

#### 10.43.2.7. EDIT\_INPUT\_VARIATION\_INDX

```
#define EDIT_INPUT_VARIATION_INDX VARIABLE_FOURTH
```

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

#### 10.43.2.8. EDIT\_TIME\_LINE\_INDX

```
#define EDIT_TIME_LINE_INDX VARIABLE_SECOND
```

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

#### 10.43.2.9. EDIT\_VBUS\_LINE\_INDX

```
#define EDIT_VBUS_LINE_INDX VARIABLE_SECOND
```

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

#### 10.43.2.10. FREC\_LINE\_INDX

```
#define FREC_LINE_INDX VARIABLE_FIRST
```

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

#### 10.43.2.11. FREQUENCY CUADRATIC VARIABLE

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

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

#### 10.43.2.12. FREQUENCY\_LINEAR\_VARIABLE

```
#define FREQUENCY_LINEAR_VARIABLE 0
```

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

#### 10.43.2.13. HFIN\_LINE\_INDX

```
#define HFIN_LINE_INDX VARIABLE_FIFTH
```

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

#### 10.43.2.14. HINI\_LINE\_INDX

```
#define HINI_LINE_INDX VARIABLE_FOURTH
```

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

#### 10.43.2.15. I2C\_DISPLAY\_MASTER\_NUM

```
#define I2C_DISPLAY_MASTER_NUM I2C_NUM_0
```

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

#### 10.43.2.16. I2C\_MASTER\_FREQ\_HZ

```
#define I2C_MASTER_FREQ_HZ 400000
```

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

#### 10.43.2.17. I2C\_MASTER\_RX\_BUF\_DISABLE

```
#define I2C_MASTER_RX_BUF_DISABLE 0
```

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

#### 10.43.2.18. I2C\_MASTER\_SCL\_IO

```
#define I2C_MASTER_SCL_IO 5
```

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

#### 10.43.2.19. I2C\_MASTER\_SDA\_IO

```
#define I2C_MASTER_SDA_IO 18
```

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

#### 10.43.2.20. I2C\_MASTER\_TX\_BUF\_DISABLE

```
#define I2C_MASTER_TX_BUF_DISABLE 0
```

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

#### 10.43.2.21. IBUS\_LINE\_INDX

```
#define IBUS_LINE_INDX VARIABLE_THIRD
```

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

#### 10.43.2.22. VBUS\_LINE\_INDX

```
#define VBUS_LINE_INDX VARIABLE_SECOND
```

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

### 10.43.3. Documentación de enumeraciones

#### 10.43.3.1. 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](#).

#### 10.43.4. Documentación de funciones

##### 10.43.4.1. DisplayEventPost()

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

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

##### Valores devueltos

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

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

##### 10.43.4.2. 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](#).

##### 10.43.4.3. 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](#).

##### 10.43.4.4. 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](#).

#### 10.43.4.5. `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](#).

#### 10.43.4.6. `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](#).

#### 10.43.4.7. `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](#).

#### 10.43.4.8. `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](#).

#### 10.43.4.9. 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](#).

#### 10.43.4.10. sh1106\_print\_emergency()

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

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

#### 10.43.4.11. 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](#).

#### 10.43.4.12. 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](#).

#### 10.43.4.13. 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](#).

#### 10.43.4.14. 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](#).

#### 10.43.4.15. 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](#).

#### 10.43.4.16. 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](#).

#### 10.43.4.17. 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](#).

#### 10.43.4.18. 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](#).

### 10.43.5. Documentación de variables

#### 10.43.5.1. blink

```
uint8_t blink [static]
```

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

#### 10.43.5.2. button\_evt\_queue

```
QueueHandle_t button_evt_queue
```

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

#### 10.43.5.3. 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](#).

#### 10.43.5.4. oled

```
sh1106_t oled
```

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

#### 10.43.5.5. screen\_displayed

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

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

### 10.43.5.6. system\_frequency\_settings

```
frequency_settings_t system_frequency_settings
```

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

### 10.43.5.7. system\_seccurity\_settings

```
seccurity_settings_t system_seccurity_settings
```

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

### 10.43.5.8. system\_time\_settings

```
time_settings_t system_time_settings
```

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

### 10.43.5.9. TAG

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

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

## 10.44. 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     uint8_t new_button; // Variable Queue
00525
00526     sh1106_variable_lines_e top_variable = first, bottom_variable = fifth; // Identificador de la
00527     variable seleccionada
00528     sh1106_variable_lines_e variable_lines = VARIABLE_FIRST;
00529
00530     uint8_t top_multiplier = 100, bottom_multiplier = 1; // Incremental de la
00531     variable seleccionada
00532     uint8_t multiplier = 1;
00533
00534     uint32_t edit_variable_min = 0, edit_variable_max = 0; // Variable
00535     seleccionada
00536     uint16_t *edit_variable = NULL;
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     time_settings_SH1106_t time_edit;
00595     security_settings_SH1106_t seccurity_edit;
00596     frequency_settings_SH1106_t frequency_edit;
00597
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             }
00726             top_multiplier = 1;
00727             bottom_multiplier = 1;
00728         } else {
00729             edit = 1;
00730             multiplier = 1;
00731             top_multiplier = 100;
00732             bottom_multiplier = 1;
00733         }
00734     } else if ( screen_displayed == SCREEN_SECURITY_EDIT ) {
00735         if ( edit_variable == NULL ) {
00736             if ( variable_lines == EDIT_VBUS_LINE_INDX ) {
00737                 edit_variable = (uint16_t*)
00738                     &(security_edit.security_settings.vbus_min);
00739                 edit_variable_min = 250;
00740                 edit_variable_max = 360;
00741             } else if ( variable_lines == EDIT_IBUS_LINE_INDX ) {
00742                 edit_variable = (uint16_t*)
00743                     &(security_edit.security_settings.ibus_max);
00744                 edit_variable_min = 500;
00745                 edit_variable_max = 2000;
00746             }
00747             edit = 1;
00748             multiplier = 1;
00749             top_multiplier = 100;
00750             bottom_multiplier = 1;
00751         } else {
00752             edit = 0;
00753             edit_variable = NULL;
00754             edit_variable_min = 0;
```

```

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

```

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

## 10.45. Referencia del archivo

**C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDe←  
Frecuencia/Software/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 régimen 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.*

### 10.45.1. 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](#).

## 10.45.2. Documentación de funciones

### 10.45.2.1. DisplayEventPost()

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

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

#### Valores devueltos

---

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

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

#### 10.45.2.2. `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](#).

#### 10.45.2.3. `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](#).

#### 10.45.2.4. `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](#).

#### 10.45.2.5. `sh1106_init()`

```
esp_err_t sh1106_init ()
```

Función que inicializa el display para comenzar a imprimir.

##### Valores devueltos

---

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

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

#### 10.45.2.6. `system_variables_save()`

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

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

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

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

#### Parámetros

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

#### Devuelve

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

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

#### 10.45.2.7. `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](#).

## 10.46. 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

```

## 10.47. Referencia del archivo C:/Users/User/Desktop/ProyectoFinal- Grupo5-VariadorDeFrecuencia/Software/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 \[14\] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x70}](#)
- 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 \[14\] = {0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x00,0x00,0x18,0x18}](#)
- 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 \[\]\[14\]](#)

### 10.47.1. 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](#).

## 10.47.2. Documentación de «typedef»

### 10.47.2.1. bitmap\_t

```
typedef struct bitmap_t bitmap_t
```

## 10.47.3. Documentación de funciones

### 10.47.3.1. 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](#).

### 10.47.3.2. 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](#).

### 10.47.3.3. 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

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](#).

#### 10.47.3.4. 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](#).

#### 10.47.3.5. sh1106\_draw\_text()

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

##### Parámetros

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

Definición en la línea 219 del archivo [sh1106\\_graphics.c](#).

#### 10.47.3.6. sh1106\_draw\_utn\_logo()

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

##### Parámetros

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

Definición en la línea 313 del archivo [sh1106\\_graphics.c](#).

#### 10.47.4. Documentación de variables

##### 10.47.4.1. 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](#).

##### 10.47.4.2. 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](#).

##### 10.47.4.3. 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](#).

##### 10.47.4.4. 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](#).

##### 10.47.4.5. 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](#).

##### 10.47.4.6. 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](#).

#### 10.47.4.7. 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](#).

#### 10.47.4.8. 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](#).

#### 10.47.4.9. 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},  
    {0x33,0x33,0x33,0x1E,0x0C,0x0C,0x0C},  
    {0x3F,0x03,0x06,0x0C,0x18,0x30,0x3F}  
}
```

Definición en la línea 33 del archivo [sh1106\\_graphics.c](#).

#### 10.47.4.10. 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](#).

#### 10.47.4.11. 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](#).

#### 10.47.4.12. 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](#).

#### 10.47.4.13. font8x14\_close\_excl

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

Definición en la línea 30 del archivo [sh1106\\_graphics.c](#).

#### 10.47.4.14. font8x14\_close\_quest

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

Definición en la línea 28 del archivo [sh1106\\_graphics.c](#).

#### 10.47.4.15. font8x14\_comma

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

Definición en la línea 26 del archivo [sh1106\\_graphics.c](#).

#### 10.47.4.16. font8x14\_dot

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

Definición en la línea 24 del archivo [sh1106\\_graphics.c](#).

#### 10.47.4.17. 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](#).

#### 10.47.4.18. 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](#).

#### 10.47.4.19. font8x14\_rowmajor

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

##### Valor inicial:

```
= {
    {0x1E,0x1E,0x33,0x33,0x33,0x33,0x3F,0x3F,0x33,0x33,0x33,0x33,0x33,0x33,0x33},
    {0x3E,0x3E,0x33,0x33,0x33,0x33,0x3E,0x3E,0x33,0x33,0x33,0x3E,0x3E},
    {0x1E,0x1E,0x33,0x33,0x30,0x30,0x30,0x30,0x30,0x33,0x33,0x1E,0x1E},
    {0x3C,0x3C,0x36,0x36,0x33,0x33,0x33,0x33,0x33,0x36,0x36,0x3C,0x3C},
    {0x3F,0x3F,0x30,0x30,0x30,0x30,0x3E,0x30,0x30,0x30,0x30,0x3F,0x3F},
    {0x3F,0x3F,0x30,0x30,0x30,0x30,0x3E,0x3E,0x30,0x30,0x30,0x30,0x30},
    {0x1E,0x1E,0x33,0x30,0x30,0x37,0x37,0x33,0x33,0x33,0x1F,0x1F},
    {0x33,0x33,0x33,0x33,0x33,0x3F,0x3F,0x33,0x33,0x33,0x33,0x33},
    {0x1E,0x1E,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x1E,0x1E},
    {0x0F,0x0F,0x06,0x06,0x06,0x06,0x06,0x06,0x06,0x36,0x36,0x1C,0x1C},
    {0x33,0x33,0x36,0x36,0x3C,0x3C,0x38,0x38,0x3C,0x3C,0x36,0x36,0x33,0x33},
    {0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x3F,0x3F},
    {0x33,0x33,0x3F,0x3F,0x3F,0x3F,0x33,0x33,0x33,0x33,0x33,0x33,0x33},
    {0x33,0x33,0x3B,0x3B,0x3F,0x3F,0x37,0x33,0x33,0x33,0x33,0x33,0x33},
    {0x1E,0x1E,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x1E,0x1E},
    {0x3E,0x3E,0x33,0x33,0x33,0x33,0x3E,0x3E,0x30,0x30,0x30,0x30,0x30},
    {0x1E,0x1E,0x33,0x33,0x33,0x33,0x33,0x37,0x37,0x36,0x36,0x1D,0x1D},
    {0x3E,0x3E,0x33,0x33,0x33,0x33,0x3E,0x3E,0x3C,0x3C,0x36,0x36,0x33,0x33},
    {0x1E,0x1E,0x33,0x33,0x30,0x30,0x1E,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](#).

#### 10.47.4.20. font8x14\_rowminor

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

##### Valor inicial:

```
= {  
    {0x00, 0x00, 0x00, 0x00, 0x3C, 0x3C, 0x06, 0x06, 0x3E, 0x3E, 0x66, 0x66, 0x3E, 0x3E},  
    {0x60, 0x60, 0x60, 0x60, 0x7C, 0x7C, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x7C, 0x7C},  
    {0x00, 0x00, 0x00, 0x00, 0x3C, 0x3C, 0x66, 0x66, 0x60, 0x60, 0x66, 0x66, 0x3C, 0x3C},  
    {0x06, 0x06, 0x06, 0x06, 0x3E, 0x3E, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x3E, 0x3E},  
    {0x00, 0x00, 0x00, 0x00, 0x3C, 0x3C, 0x66, 0x66, 0x7E, 0x7E, 0x60, 0x60, 0x3C, 0x3C},  
    {0x1C, 0x1C, 0x30, 0x30, 0x30, 0x30, 0x7C, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30},  
    {0x00, 0x00, 0x3E, 0x3E, 0x66, 0x66, 0x66, 0x3E, 0x06, 0x06, 0x7C, 0x7C},  
    {0x60, 0x60, 0x60, 0x60, 0x7C, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66},  
    {0x18, 0x18, 0x00, 0x00, 0x38, 0x38, 0x18, 0x18, 0x18, 0x18, 0x18, 0x3C, 0x3C},  
    {0x06, 0x06, 0x00, 0x00, 0x06, 0x06, 0x06, 0x06, 0x66, 0x66, 0x3C, 0x3C},  
    {0x60, 0x60, 0x60, 0x60, 0x66, 0x6C, 0x6C, 0x78, 0x78, 0x6C, 0x6C, 0x66, 0x66},  
    {0x38, 0x38, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x3C, 0x3C},  
    {0x00, 0x00, 0x00, 0x00, 0x6C, 0x6C, 0x7E, 0x7E, 0x7E, 0x6C, 0x6C, 0x6C},  
    {0x00, 0x00, 0x00, 0x00, 0x7C, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66},  
    {0x00, 0x00, 0x00, 0x00, 0x3C, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x3C},  
    {0x00, 0x00, 0x7C, 0x7C, 0x66, 0x66, 0x66, 0x7C, 0x60, 0x60, 0x60, 0x60},  
    {0x00, 0x00, 0x3E, 0x3E, 0x66, 0x66, 0x66, 0x3E, 0x06, 0x06, 0x06, 0x06},  
    {0x00, 0x00, 0x00, 0x00, 0x6C, 0x6C, 0x76, 0x76, 0x60, 0x60, 0x60, 0x60, 0x60},  
    {0x00, 0x00, 0x00, 0x00, 0x3E, 0x60, 0x60, 0x3C, 0x06, 0x06, 0x7C, 0x7C},  
    {0x30, 0x30, 0x30, 0x30, 0x7C, 0x30, 0x30, 0x30, 0x30, 0x30, 0x1C, 0x1C},  
    {0x00, 0x00, 0x00, 0x00, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x3E, 0x3E},  
    {0x00, 0x00, 0x00, 0x00, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x3C, 0x18},  
    {0x00, 0x00, 0x00, 0x00, 0x66, 0x66, 0x66, 0x66, 0x7E, 0x7E, 0x6C, 0x6C},  
    {0x00, 0x00, 0x00, 0x00, 0x66, 0x66, 0x66, 0x3C, 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](#).

#### 10.47.4.21. font8x14\_rownumber

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

##### Valor inicial:

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

#### **10.47.4.22. 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](#).

#### **10.47.4.23. 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](#).

#### **10.47.4.24. logo16\_utn\_bmp**

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

Definición en la línea 14 del archivo [sh1106\\_graphics.c](#).

#### **10.47.4.25. slash**

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

Definición en la línea 16 del archivo [sh1106\\_graphics.c](#).

## 10.48. 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] =
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
// 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] =
{0x00,0x38,0x38,0x00,0x00,0x00,0x00,0x00,0x38,0x38,0x00,0x00,0x00,0x00,0x00};
// Dos puntos 8 x 14
00023 static const uint8_t font5x7_dot[7] = {0x00,0x00,0x00,0x00,0x00,0x18,0x18};
// Punto 5 x 7
00024 static const uint8_t font8x14_dot[14] =
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x38};
// Punto 8 x 14
00025 static const uint8_t font5x7_comma[7] = {0x00,0x00,0x00,0x00,0x18,0x18,0x30};
// Coma 5 x 7
00026 static const uint8_t font8x14_comma[14] =
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x38,0x70};
// 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,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
// Menos 8 x 14
00033 static const uint8_t font5x7_rowmajor[] [7] = {
// Alfabeto mayúsculas 5 x 7
00034     // A-Z
00035     {0x1E,0x33,0x33,0x3F,0x33,0x33,0x33},
// A
00036     {0x3E,0x33,0x33,0x3E,0x33,0x33,0x3E},
// B
00037     {0x1E,0x33,0x30,0x30,0x30,0x33,0x1E},
// C
00038     {0x3C,0x36,0x33,0x33,0x36,0x3C},
// D
00039     {0x3F,0x30,0x30,0x3E,0x30,0x30,0x3F},
// E
00040     {0x3F,0x30,0x30,0x3E,0x30,0x30,0x30},
// F
00041     {0x1E,0x33,0x30,0x37,0x33,0x33,0x1F},
// G
00042     {0x33,0x33,0x33,0x3F,0x33,0x33,0x33},
// H
00043     {0x1E,0x0C,0x0C,0x0C,0x0C,0x0C,0x1E},
// I
00044     {0x0F,0x06,0x06,0x06,0x06,0x36,0x1C},
// J
00045     {0x33,0x36,0x3C,0x38,0x3C,0x36,0x33},
// K
00046     {0x30,0x30,0x30,0x30,0x30,0x30,0x3F},
// L
}

```

```

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

```

// 0
00092 {0x0C, 0x1C, 0x0C, 0x0C, 0x0C, 0x1E},
// 1
00093 {0x1E, 0x33, 0x03, 0x0E, 0x18, 0x30, 0x3F},
// 2
00094 {0x1E, 0x33, 0x03, 0x0E, 0x03, 0x33, 0x1E},
// 3
00095 {0x06, 0x0E, 0x1E, 0x36, 0x3F, 0x06, 0x06},
// 4
00096 {0x3F, 0x30, 0x3E, 0x03, 0x03, 0x33, 0x1E},
// 5
00097 {0x1E, 0x30, 0x3E, 0x33, 0x33, 0x33, 0x1E},
// 6
00098 {0x3F, 0x03, 0x06, 0x0C, 0x18, 0x18, 0x18},
// 7
00099 {0x1E, 0x33, 0x33, 0x1E, 0x33, 0x33, 0x1E},
// 8
00100 {0x1E, 0x33, 0x33, 0x1F, 0x03, 0x03, 0x1E}
// 9
00101 };
00102 static const uint8_t font8x14_rowmajor[][14] = {
    // Alfabeto mayúsculas 8 x 14
00103 {0x1E, 0x1E, 0x33, 0x33, 0x33, 0x3F, 0x3F, 0x33, 0x33, 0x33, 0x33, 0x33, 0x33},
    // A
00104 {0x3E, 0x3E, 0x33, 0x33, 0x33, 0x3E, 0x3E, 0x33, 0x33, 0x33, 0x3E, 0x3E},
    // B
00105 {0x1E, 0x1E, 0x33, 0x33, 0x30, 0x30, 0x30, 0x30, 0x30, 0x33, 0x33, 0x1E}, 
    // 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, 0x33, 0x37, 0x37, 0x36, 0x36, 0x1D, 0x1D},
    // Q
00120 {0x3E, 0x3E, 0x33, 0x33, 0x33, 0x3E, 0x3E, 0x3C, 0x3C, 0x36, 0x36, 0x33, 0x33},
    // R
00121 {0x1E, 0x1E, 0x33, 0x33, 0x30, 0x1E, 0x1E, 0x03, 0x03, 0x33, 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, 0x0C},
    // Y
00128 {0x3F, 0x3F, 0x03, 0x03, 0x06, 0x06, 0x0C, 0x0C, 0x18, 0x18, 0x30, 0x30, 0x3F, 0x3F},
    // Z
00129 };
00130 static const uint8_t font8x14_rowminor[][14] = {
    // Alfabeto minúsculas 8 x 14
00131 {0x00, 0x00, 0x00, 0x00, 0x3C, 0x3C, 0x06, 0x06, 0x3E, 0x3E, 0x66, 0x66, 0x3E, 0x3E},
    // a
00132 {0x60, 0x60, 0x60, 0x60, 0x7C, 0x7C, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x7C, 0x7C},
    // b
00133 {0x00, 0x00, 0x00, 0x00, 0x3C, 0x3C, 0x66, 0x66, 0x60, 0x60, 0x66, 0x66, 0x3C, 0x3C},
    // c
00134 {0x06, 0x06, 0x06, 0x06, 0x3E, 0x3E, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x3E, 0x3E},
    // d
00135 {0x00, 0x00, 0x00, 0x00, 0x3C, 0x3C, 0x66, 0x66, 0x7E, 0x7E, 0x60, 0x60, 0x3C, 0x3C},
    // e

```

```

00136     {0x1C,0x1C,0x30,0x30,0x30,0x30,0x7C,0x7C,0x30,0x30,0x30,0x30,0x30,0x30},  

00137     // f  

00138     {0x00,0x00,0x3E,0x3E,0x66,0x66,0x66,0x66,0x3E,0x06,0x06,0x7C,0x7C},  

00139     // g  

00140     {0x60,0x60,0x60,0x60,0x7C,0x7C,0x66,0x66,0x66,0x66,0x66,0x66,0x66,0x66},  

00141     // h  

00142     {0x18,0x18,0x00,0x00,0x38,0x38,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x3C,0x3C},  

00143     // i  

00144     {0x06,0x06,0x00,0x00,0x06,0x06,0x06,0x06,0x06,0x66,0x66,0x66,0x3C,0x3C},  

00145     // j  

00146     {0x60,0x60,0x60,0x66,0x66,0x6C,0x6C,0x78,0x78,0x6C,0x6C,0x66,0x66},  

00147     // k  

00148     {0x38,0x38,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x3C,0x3C},  

00149     // l  

00150     {0x00,0x00,0x00,0x00,0x6C,0x6C,0x7E,0x7E,0x7E,0x6C,0x6C,0x6C,0x6C},  

00151     // m  

00152     {0x00,0x00,0x00,0x00,0x7C,0x7C,0x66,0x66,0x66,0x66,0x66,0x66,0x66,0x66},  

00153     // n  

00154     {0x00,0x00,0x7C,0x7C,0x66,0x66,0x66,0x7C,0x7C,0x60,0x60,0x60,0x60},  

00155     // o  

00156     {0x00,0x00,0x3E,0x3E,0x60,0x60,0x3C,0x3C,0x06,0x06,0x7C,0x7C},  

00157     // p  

00158 static const uint8_t font8x14_rownumber[][14] = {  

00159     // Números 8 x 14  

00160     {0x3C,0x66,0xC3,0xC3,0xC3,0xC3,0xC3,0xC3,0xC3,0xC3,0x66,0x3C,0x00},  

00161     // 0  

00162     {0x18,0x38,0x78,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x7E,0x00},  

00163     // 1  

00164     {0x3C,0x66,0xC3,0x03,0x06,0x0C,0x18,0x30,0x60,0xC0,0xC3,0xFF,0xFE,0x00},  

00165     // 2  

00166     {0x7E,0xC3,0x03,0x06,0x3C,0x06,0x03,0x03,0xC3,0x66,0x3C,0x00},  

00167     // 3  

00168     {0x06,0x0E,0x1E,0x36,0x66,0xC6,0x86,0xFF,0xFF,0x06,0x06,0x06,0x00},  

00169     // 4  

00170     {0xFE,0xC0,0xC0,0xFC,0xC6,0x03,0x03,0x03,0xC3,0x66,0x3C,0x00},  

00171     // 5  

00172     {0x3C,0x66,0xC3,0xC0,0xC0,0xFC,0xC6,0xC3,0xC3,0xC3,0x66,0x3C,0x00},  

00173     // 6  

00174     {0x0F,0xC3,0x03,0x06,0x0C,0x18,0x30,0x60,0x60,0x60,0x00},  

00175     // 7  

00176     {0x3C,0x66,0xC3,0xC3,0x66,0x3C,0xC3,0xC3,0x66,0x3C,0x00},  

00177     // 8  

00178     {0x3C,0x66,0xC3,0xC3,0x67,0x3F,0x03,0x03,0xC3,0x66,0x3C,0x00}  

00179     // 9  

00180 };  

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     } 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 }

```

## 10.49. Referencia del archivo C:/Users/User/Desktop/ProyectoFinal- Grupo5-VariadorDeFrecuencia/Software/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)

### 10.49.1. 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](#).

### 10.49.2. Documentación de «typedef»

#### 10.49.2.1. sh1106\_t

```
typedef struct sh1106_t sh1106_t
```

### 10.49.3. Documentación de funciones

#### 10.49.3.1. sh1106\_draw\_arrow()

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

#### Parámetros

<a href="#">out</a>	<a href="#">oled</a>	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](#).

#### 10.49.3.2. 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](#).

#### 10.49.3.3. 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](#).

#### 10.49.3.4. 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](#).

#### 10.49.3.5. 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](#).

## 10.50. sh1106\_graphics.h

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

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

## 10.51. Referencia del archivo

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDe  
Frecuencia/Software/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.*

## 10.51.1. 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](#).

## 10.51.2. Documentación de «define»

### 10.51.2.1. CMD\_CONTROL

```
#define CMD_CONTROL 0x00
```

Definición en la línea 16 del archivo [sh1106\\_i2c.c](#).

### 10.51.2.2. DATA\_CONTROL

```
#define DATA_CONTROL 0x40
```

Definición en la línea 17 del archivo [sh1106\\_i2c.c](#).

### 10.51.2.3. I2C\_DISPLAY

```
#define I2C_DISPLAY I2C_NUM_0
```

Definición en la línea 14 del archivo [sh1106\\_i2c.c](#).

### 10.51.2.4. SH1106\_ADDR

```
#define SH1106_ADDR 0x3C
```

Definición en la línea 15 del archivo [sh1106\\_i2c.c](#).

## 10.51.3. Documentación de funciones

### 10.51.3.1. 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](#).

## 10.52. 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     return ret;

```

## 10.53. Referencia del archivo

[C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDe←  
Frecuencia/Software/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.*

### 10.53.1. 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](#).

### 10.53.2. Documentación de enumeraciones

#### 10.53.2.1. 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](#).

### 10.53.3. Documentación de funciones

#### 10.53.3.1. 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](#).

## 10.54. 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
```

## 10.55. Referencia del archivo

**C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDe←  
Frecuencia/Software/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/FreRTOS.h"
#include "freertos/semphr.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include <inttypes.h>
#include <stdio.h>
#include <stdlib.h>
#include "mcp23017_defs.h"
#include "./MCP23017.h"
#include "./io_control.h"
#include "../system/sysAdmin.h"
#include "../system/sysControl.h"
```

### defines

- #define INT\_A\_PIN GPIO\_NUM\_22
- #define BUZZER\_PIN 0
- #define RELAY\_PIN 1
- #define FREQ\_SEL\_1\_PIN 2
- #define FREQ\_SEL\_2\_PIN 3
- #define FREQ\_SEL\_3\_PIN 4
- #define STOP\_PIN 5
- #define TERMO\_SW\_PIN 6
- #define FREQ\_SEL\_MASK ( (1 << FREQ\_SEL\_3\_PIN) | (1 << FREQ\_SEL\_2\_PIN) | (1 << FREQ\_SEL\_1\_PIN) )
- #define TERMO\_SW\_MASK (1 << TERMO\_SW\_PIN)
- #define STOP\_SW\_MASK (1 << STOP\_PIN)
- #define INT\_B\_PIN GPIO\_NUM\_23
- #define MATRIX\_SW1 0x17
- #define MATRIX\_SW2 0x27
- #define MATRIX\_SW3 0x1B
- #define MATRIX\_SW4 0x2B
- #define MATRIX\_SW5 0x1D
- #define MATRIX\_SW6 0x2D
- #define MATRIX\_SW7 0x1E
- #define MATRIX\_SW8 0x2E
- #define SWITCH\_BUTTON\_BACK MATRIX\_SW1
- #define SWITCH\_BUTTON\_UP MATRIX\_SW2
- #define SWITCH\_BUTTON\_LEFT MATRIX\_SW3
- #define SWITCH\_BUTTON\_RIGHT MATRIX\_SW4

- #define SWITCH\_BUTTON\_DOWN MATRIX\_SW5
- #define SWITCH\_BUTTON\_SAVE MATRIX\_SW6
- #define SWITCH\_BUTTON\_OK MATRIX\_SW7
- #define SWITCH\_BUTTON\_MENU MATRIX\_SW8
- #define STOP\_BUTTON\_PIN GPIO\_NUM\_16
- #define START\_BUTTON\_PIN GPIO\_NUM\_17
- #define PWM\_GPIO 4
- #define PWM\_FREQ\_HZ 1000
- #define PWM\_TIMER LEDC\_TIMER\_0
- #define PWM\_MODE LEDC\_LOW\_SPEED\_MODE
- #define PWM\_CHANNEL LEDC\_CHANNEL\_0
- #define PWM\_RES LEDC\_TIMER\_10\_BIT
- #define PWM\_STEP\_MS 250
- #define DUTY\_MIN\_PCT 54
- #define DUTY\_MAX\_PCT 98

## Funciones

- static esp\_err\_t freq\_0\_10V\_output ()  
*Inicializa el timer para el correcto funcionamiento del PWM que permite obtener la salida de 0 a 10V analógicos.*
- static esp\_err\_t gpio\_init\_interrupts (void)  
*Inicializa las interrupciones para INTA y INTB del MCP23017 y para los botones aislados stop y start.*
- static void MCP23017\_buzzer\_control (void \*arg)  
*Tarea que espera comandos a través de la cola buzzer\_evt\_queue para hacer sonar al buzzer.*
- static void MCP23017\_keyboard\_control (void \*arg)  
*Tarea que controla las salidas del MCP23017 para hacer funcionar al teclado.*
- static void MCP23017\_relay\_control (void \*arg)  
*Tarea que espera comandos a través de la cola relay\_evt\_queue para activar o desactivar el relay.*
- void IRAM\_ATTR gpio\_isr\_handler (void \*arg)
- 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 milisegundos.*

## Variables

- static const char \* TAG = "IO\_CTRL"
- QueueHandle\_t buzzer\_evt\_queue = NULL
- QueueHandle\_t relay\_evt\_queue = NULL
- QueueHandle\_t GPIO\_evt\_queue = NULL
- uint8\_t mcp\_porta
- uint8\_t mcp\_portb

### 10.55.1. Descripción detallada

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

#### Autor

Andrenacci - Carra

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

#### Versión

2.0

#### Fecha

2025-11-06

Definición en el archivo [io\\_control.c](#).

### 10.55.2. Documentación de «define»

#### 10.55.2.1. 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](#).

#### 10.55.2.2. 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](#).

#### 10.55.2.3. 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](#).

#### 10.55.2.4. 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](#).

#### 10.55.2.5. 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](#).

#### 10.55.2.6. 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](#).

#### 10.55.2.7. 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](#).

#### 10.55.2.8. 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](#).

#### 10.55.2.9. 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](#).

### 10.55.2.10. 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](#).

### 10.55.2.11. 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](#).

### 10.55.2.12. 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](#).

### 10.55.2.13. 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](#).

### 10.55.2.14. 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](#).

### 10.55.2.15. 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](#).

### 10.55.2.16. 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](#).

### 10.55.2.17. 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](#).

### 10.55.2.18. 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](#).

### 10.55.2.19. 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](#).

### 10.55.2.20. 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](#).

### 10.55.2.21. 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](#).

### 10.55.2.22. 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](#).

### 10.55.2.23. PWM\_STEP\_MS

```
#define PWM_STEP_MS 250
```

Cambio cada 500 ms

Definición en la línea [82](#) del archivo [io\\_control.c](#).

### 10.55.2.24. 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](#).

### 10.55.2.25. 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](#).

### 10.55.2.26. 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](#).

### 10.55.2.27. 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](#).

### 10.55.2.28. 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](#).

### 10.55.2.29. 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](#).

### 10.55.2.30. SWITCH\_BUTTON\_BACK

```
#define SWITCH_BUTTON_BACK MATRIX_SW1
```

Asignación de boton al SW1 representado por 0x17

Definición en la línea 64 del archivo [io\\_control.c](#).

### 10.55.2.31. SWITCH\_BUTTON\_DOWN

```
#define SWITCH_BUTTON_DOWN MATRIX_SW5
```

Asignación de boton al SW5 representado por 0x1D

Definición en la línea 68 del archivo [io\\_control.c](#).

### 10.55.2.32. SWITCH\_BUTTON\_LEFT

```
#define SWITCH_BUTTON_LEFT MATRIX_SW3
```

Asignación de boton al SW3 representado por 0x1B

Definición en la línea 66 del archivo [io\\_control.c](#).

### 10.55.2.33. SWITCH\_BUTTON\_MENU

```
#define SWITCH_BUTTON_MENU MATRIX_SW8
```

Asignación de boton al SW8 representado por 0x2E

Definición en la línea 71 del archivo [io\\_control.c](#).

### 10.55.2.34. 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](#).

### 10.55.2.35. 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](#).

### 10.55.2.36. 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](#).

### 10.55.2.37. 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](#).

### 10.55.2.38. 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](#).

### 10.55.2.39. 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](#).

## 10.55.3. Documentación de funciones

### 10.55.3.1. BuzzerEventPost()

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

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

#### Parámetros

---

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

Devuelve

- ESP\_OK Evento creado exitosamente
- ESP\_FAIL Error al crear el evento

Definición en la línea 601 del archivo [io\\_control.c](#).

### 10.55.3.2. 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](#).

### 10.55.3.3. 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](#).

#### 10.55.3.4. GPIO\_interrupt\_attendance\_task()

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

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

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

Definición en la línea [386](#) del archivo [io\\_control.c](#).

#### 10.55.3.5. gpio\_isr\_handler()

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

Definición en la línea [164](#) del archivo [io\\_control.c](#).

#### 10.55.3.6. 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](#).

#### 10.55.3.7. 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](#).

#### 10.55.3.8. 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](#).

#### 10.55.3.9. RelayEventPost()

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

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

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

#### Parámetros

---

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

Devuelve

- ESP\_OK Evento creado exitosamente
- ESP\_FAIL Error al crear el evento

Definición en la línea 593 del archivo [io\\_control.c](#).

#### 10.55.3.10. 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](#).

### 10.55.4. Documentación de variables

#### 10.55.4.1. 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](#).

#### 10.55.4.2. 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](#).

#### 10.55.4.3. mcp\_porta

```
uint8_t mcp_porta
```

Estado de entradas del puerto A del MCP23017

Definición en la línea [90](#) del archivo [io\\_control.c](#).

#### 10.55.4.4. mcp\_portb

```
uint8_t mcp_portb
```

Estado de entradas del puerto B del MCP23017

Definición en la línea [91](#) del archivo [io\\_control.c](#).

#### 10.55.4.5. 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](#).

#### 10.55.4.6. 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](#).

### 10.56. io\_control.c

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

```
00001
00012 #include "sdkconfig.h"
00013
00014 #include "esp_err.h"
00015 #include "esp_log.h"
00016
00017 #include "driver/gpio.h"
00018 #include "driver/ledc.h"
00019 #include "display/display.h"
00020
00021 #include "freertos/FreeRTOS.h"
00022 #include "freertos/semphr.h"
00023 #include "freertos/task.h"
00024 #include "freertos/queue.h"
00025
00026 #include <inttypes.h>
00027 #include <stdio.h>
00028 #include <stdlib.h>
00029
00030 #include "mcp23017_defs.h"
00031
00032 #include "./MCP23017.h"
```

```
00033 #include "./io_control.h"
00034 #include "../system/sysAdmin.h"
00035 #include "../system/sysControl.h"
00036
00037 static const char *TAG = "IO_CTRL";
00038
00039 #define INT_A_PIN           GPIO_NUM_22
00040
00041 #define BUZZER_PIN          0
00042 #define RELAY_PIN            1
00043 #define FREQ_SEL_1_PIN        2
00044 #define FREQ_SEL_2_PIN        3
00045 #define FREQ_SEL_3_PIN        4
00046 #define STOP_PIN              5
00047 #define TERMO_SW_PIN          6
00048
00049 #define FREQ_SEL_MASK        ((1 << FREQ_SEL_3_PIN) | (1 << FREQ_SEL_2_PIN) | (1 <<
00050   FREQ_SEL_1_PIN))
00050 #define TERMO_SW_MASK         (1 << TERMO_SW_PIN)
00051 #define STOP_SW_MASK          (1 << STOP_PIN)
00052
00053 #define INT_B_PIN           GPIO_NUM_23
00054
00055 #define MATRIX_SW1            0x17
00056 #define MATRIX_SW2            0x27
00057 #define MATRIX_SW3            0x1B
00058 #define MATRIX_SW4            0x2B
00059 #define MATRIX_SW5            0x1D
00060 #define MATRIX_SW6            0x2D
00061 #define MATRIX_SW7            0x1E
00062 #define MATRIX_SW8            0x2E
00063
00064 #define SWITCH_BUTTON_BACK    MATRIX_SW1
00065 #define SWITCH_BUTTON_UP      MATRIX_SW2
00066 #define SWITCH_BUTTON_LEFT    MATRIX_SW3
00067 #define SWITCH_BUTTON_RIGHT   MATRIX_SW4
00068 #define SWITCH_BUTTON_DOWN    MATRIX_SW5
00069 #define SWITCH_BUTTON_SAVE    MATRIX_SW6
00070 #define SWITCH_BUTTON_OK      MATRIX_SW7
00071 #define SWITCH_BUTTON_MENU    MATRIX_SW8
00072
00073 #define STOP_BUTTON_PIN        GPIO_NUM_16
00074 #define START_BUTTON_PIN       GPIO_NUM_17
00075
00076 #define PWM_GPIO              4
00077 #define PWM_FREQ_HZ            1000
00078 #define PWM_TIMER             LEDC_TIMER_0
00079 #define PWM_MODE               LEDC_LOW_SPEED_MODE
00080 #define PWM_CHANNEL            LEDC_CHANNEL_0
00081 #define PWM_RES                LEDC_TIMER_10_BIT
00082 #define PWM_STEP_MS             250
```

```
00083 #define DUTY_MIN_PCT          54
00084 #define DUTY_MAX_PCT          98
00085
00086 QueueHandle_t buzzer_evt_queue = NULL;
00087 QueueHandle_t relay_evt_queue = NULL;
00088 QueueHandle_t GPIO_evt_queue = NULL;
00089
00090 uint8_t mcp_porta;
00091 uint8_t mcp_portb;
00092
00104 static esp_err_t freq_0_10V_output();
00105
00121 static esp_err_t gpio_init_interrupts(void);
00122
00131 static void MCP23017_buzzer_control(void* arg);
00132
00141 static void MCP23017_keyboard_control(void* arg);
00142
00151 static void MCP23017_relay_control(void* arg);
00152
00153
00164 void IRAM_ATTR gpio_isr_handler(void* arg) {
00165     uint32_t gpio_num = (uint32_t) arg;
00166     BaseType_t xHigherPriorityTaskWoken = pdFALSE;
00167
00168     xQueueSendFromISR(GPIO_evt_queue, &gpio_num, &xHigherPriorityTaskWoken);
00169     if (xHigherPriorityTaskWoken) {
00170         portYIELD_FROM_ISR();
00171     }
00172 }
00173
00174 static esp_err_t freq_0_10V_output() {
00175     esp_err_t retval;
00176     // Configurar el temporizador del PWM
00177     ledc_timer_config_t ledc_timer = {
00178         .speed_mode      = PWM_MODE,
00179         .timer_num       = PWM_TIMER,
00180         .duty_resolution = PWM_RES,
00181         .freq_hz         = PWM_FREQ_HZ,
00182         .clk_cfg         = LEDC_AUTO_CLK
00183     };
00184     retval = ledc_timer_config(&ledc_timer);
00185
00186     if (retval != ESP_OK) {
00187         ESP_LOGE(TAG, "Error al configurar el timer de la salida 0-10V");
00188         return retval;
00189     }
00190
00191     // Configurar el canal
00192     ledc_channel_config_t ledc_channel = {
00193         .gpio_num        = PWM_GPIO,
00194         .speed_mode      = PWM_MODE,
00195         .channel         = PWM_CHANNEL,
00196         .timer_sel       = PWM_TIMER,
00197         .duty            = 0,
00198         .hpoint          = 0
00199     };
00200
00201     retval = ledc_channel_config(&ledc_channel);
00202
00203     if (retval != ESP_OK) {
00204         ESP_LOGE(TAG, "Error al configurar el canal de la salida 0-10V");
00205         return retval;
00206     }
00207
00208     return ESP_OK;
00209 }
00210
00211 static esp_err_t gpio_init_interrupts(void) {
00212     esp_err_t retval;
00213     gpio_config_t io_conf_stop_start;
00214
00215     io_conf_stop_start.io_conf =
00216
00217     if (GPIO_evt_queue == NULL) {
00218         GPIO_evt_queue = xQueueCreate(1, sizeof(uint32_t));
00219         if (GPIO_evt_queue == NULL) {
```

```

00220         ESP_LOGE(TAG, "No se pudo crear la cola de interrupciones");
00221         return ESP_FAIL;
00222     }
00223 }
00224
00225     retval = gpio_install_isr_service(0);
00226     if (retval != ESP_OK) {
00227         ESP_LOGE(TAG, "Error al configurar la interrupción de GPIO para INTA y INTB del MCP23017");
00228         return retval;
00229     }
00230
00231 // Configuración de pines
00232 io_conf.intr_type = GPIO_INTR_NEGEDGE;
00233 io_conf.mode = GPIO_MODE_INPUT;
00234 io_conf.pin_bit_mask = (1ULL<<INT_A_PIN) | (1ULL<<INT_B_PIN);
00235 io_conf.pull_down_en = GPIO_PULLDOWN_DISABLE;
00236 io_conf.pull_up_en = GPIO_PULLUP_DISABLE;
00237
00238     retval = gpio_config(&io_conf);
00239     if (retval != ESP_OK) {
00240         ESP_LOGE(TAG, "Error al configurar los GPIO");
00241         return retval;
00242     }
00243
00244 // Asocio pines a la rutina ISR
00245     retval = gpio_isr_handler_add(INT_A_PIN, gpio_isr_handler, (void*) (uintptr_t) INT_A_PIN);
00246     if (retval != ESP_OK) {
00247         ESP_LOGE(TAG, "Error al configurar la interrupción de GPIO para pin INTA del MCP23017");
00248         return retval;
00249     }
00250
00251     retval = gpio_isr_handler_add(INT_B_PIN, gpio_isr_handler, (void*) (uintptr_t) INT_B_PIN);
00252     if (retval != ESP_OK) {
00253         ESP_LOGE(TAG, "Error al configurar la interrupción de GPIO para pin INTB del MCP23017");
00254         return retval;
00255     }
00256
00257     ESP_LOGI(TAG, "Interrupciones GPIO configuradas en %d (Int A MCP) y %d (Int B MCP)", INT_A_PIN,
00258             INT_B_PIN);
00259
00260 // Configuración de pines
00261 io_conf_stop_start.intr_type = GPIO_INTR_ANYEDGE;
00262 io_conf_stop_start.mode = GPIO_MODE_INPUT;
00263 io_conf_stop_start.pin_bit_mask = (1ULL<<STOP_BUTTON_PIN) | (1ULL<<START_BUTTON_PIN);
00264 io_conf_stop_start.pull_down_en = GPIO_PULLDOWN_DISABLE;
00265 io_conf_stop_start.pull_up_en = GPIO_PULLUP_DISABLE;
00266
00267     retval = gpio_config(&io_conf_stop_start);
00268     if (retval != ESP_OK) {
00269         ESP_LOGE(TAG, "Error al configurar los GPIO para botones aislados stop y start");
00270         return retval;
00271     }
00272
00273 // Asocio pines a la rutina ISR
00274     retval = gpio_isr_handler_add(STOP_BUTTON_PIN, gpio_isr_handler, (void*) (uintptr_t)
00275             STOP_BUTTON_PIN);
00276     if (retval != ESP_OK) {
00277         ESP_LOGE(TAG, "Error al configurar la interrupción de GPIO para botones aislados stop");
00278         return retval;
00279     }
00280
00281     retval = gpio_isr_handler_add(START_BUTTON_PIN, gpio_isr_handler, (void*) (uintptr_t)
00282             START_BUTTON_PIN);
00283     if (retval != ESP_OK) {
00284         ESP_LOGE(TAG, "Error al configurar la interrupción de GPIO para botones aislados start");
00285         return retval;
00286     }
00287
00288     ESP_LOGI(TAG, "Interrupciones GPIO configuradas en %d (Stop) y %d (Start)", STOP_BUTTON_PIN,
00289             START_BUTTON_PIN);
00290     return ESP_OK;
00291 }
00292
00293 static void MCP23017_buzzer_control(void* arg) {
00294     uint32_t delay_time;
00295     uint16_t buzzer_time;
00296
00297     if (buzzer_evt_queue == NULL) {
00298         buzzer_evt_queue = xQueueCreate(10, sizeof(uint32_t));
00299         if (buzzer_evt_queue == NULL) {
00300             ESP_LOGE(TAG, "No se pudo crear la cola de buzzer");
00301             return;
00302         }
00303     }
00304
00305     mcp_write_output_pin(PORTA, 0, 0);

```

```

00303     for(;;) {
00304         if(xQueueReceive(buzzer_evt_queue, &delay_time, portMAX_DELAY)) {
00305             buzzer_time = (uint16_t) delay_time;
00306             mcp_write_output_pin(PORTA, 0, 1);
00307             vTaskDelay(pdMS_TO_TICKS(buzzer_time));
00308             mcp_write_output_pin(PORTA, 0, 0);
00309         }
00310     }
00311 }
00312
00313 static void MCP23017_keyboard_control(void* arg) {
00314     ESP_LOGI(TAG, "Iniciando tarea de control de teclado");
00315
00316     for(;;) {
00317         vTaskDelay(pdMS_TO_TICKS(100));
00318         if ( mcp_portb & 0x10 ) {
00319             mcp_portb = mcp_portb & (~0x10);
00320             mcp_portb = mcp_portb | (0x20);
00321         } else {
00322             mcp_portb = mcp_portb & (~0x20);
00323             mcp_portb = mcp_portb | (0x10);
00324         }
00325         if ( ( mcp_portb & 0x0F ) != 0x0F ) {
00326             // ESP_LOGI(TAG, "Puerto B: %X. Botón pulsado...", mcp_portb);
00327             mcp_read_port(PORTB, &mcp_portb);
00328             continue;
00329         }
00330         mcp_write_output_port(PORTB, mcp_portb);
00331     }
00332 }
00333
00334 static void MCP23017_relay_control(void* arg) {
00335
00336     uint8_t io_num;
00337
00338     if (relay_evt_queue == NULL) {
00339         relay_evt_queue = xQueueCreate(1, sizeof(uint8_t));
00340         if (relay_evt_queue == NULL) {
00341             ESP_LOGE(TAG, "No se pudo crear la cola de relay");
00342             return;
00343         }
00344     }
00345
00346     mcp_write_output_pin(PORTA, 1, 0);
00347     for(;;) {
00348         if(xQueueReceive(relay_evt_queue, &io_num, portMAX_DELAY)) {
00349             ESP_LOGI(TAG, "New relay state: %d", io_num);
00350             if ( io_num == 1 ) {
00351                 mcp_write_output_pin(PORTA, 1, 1);
00352             } else {
00353                 mcp_write_output_pin(PORTA, 1, 0);
00354             }
00355         }
00356     }
00357 }
00358
00359 esp_err_t set_freq_output(uint16_t freq)
00360 {
00361     const uint16_t F_MAX      = 150;    // Hz
00362     const uint16_t DUTY_MAX   = 98;     // %
00363     const uint16_t DUTY_MIN   = 54;     // %
00364     const uint32_t PWM_MAX   = 1023;   // ticks (10 bits)
00365
00366     if (freq > F_MAX) freq = F_MAX;
00367
00368     // duty_pct = 98 - round( (44*freq)/150 )
00369     uint16_t drop = (uint16_t)((44u * freq + 75u) / 150u); // +75 para redondeo
00370     int duty_pct = (int)DUTY_MAX - (int)drop;
00371
00372     if (duty_pct < DUTY_MIN) duty_pct = DUTY_MIN;
00373     if (duty_pct > DUTY_MAX) duty_pct = DUTY_MAX;
00374
00375     // ticks = round( duty_pct/100 * 1023 )
00376     uint32_t ticks = (uint32_t)((duty_pct * PWM_MAX + 50u) / 100u);
00377
00378     if (ledc_get_duty(PWM_MODE, PWM_CHANNEL) != ticks) {
00379         ESP_ERROR_CHECK(ledc_set_duty(PWM_MODE, PWM_CHANNEL, ticks));
00380         ESP_ERROR_CHECK(ledc_update_duty(PWM_MODE, PWM_CHANNEL));
00381         ESP_LOGI("PWM", "Freq=%u Hz, Duty=%d % (ticks=%lu)", freq, duty_pct, (unsigned long)ticks);
00382     }
00383     return ESP_OK;
00384 }
00385
00386 void GPIO_interrupt_attendance_task(void* arg) {
00387     uint32_t io_num;
00388     uint32_t last_interrupt = 0;
00389     uint8_t scratch_data;

```

```

00390     uint8_t speed_selector;
00391
00392     if (MCP23017_INIT() == ESP_OK) {
00393         ESP_LOGI(TAG, "MCP23017 Tarea iniciada correctamente");
00394     } else {
00395         ESP_LOGE(TAG, "Error al inicializar el MCP23017");
00396         ESP_ERROR_CHECK(ESP_FAIL);
00397     }
00398
00399     gpio_init_interrupts();
00400     freq_0_10V_output();
00401     set_freq_output(0);
00402
00403     xTaskCreate(MCP23017_buzzer_control, "MCP23017_buzzer_control", 2048, NULL, 10, NULL);
00404     xTaskCreate(MCP23017_relay_control, "MCP23017_relay_control", 2048, NULL, 10, NULL);
00405     xTaskCreate(MCP23017_keyboard_control, "MCP23017_keyboard_control", 2048, NULL, 10, NULL);
00406
00407     vTaskDelay(pdMS_TO_TICKS(100));
00408
00409     for (;;) {
00410         system_status_t s_e;
00411         get_status(&s_e);
00412         set_freq_output(s_e.frequency);
00413         if (xQueueReceive(GPIO_evt_queue, &io_num, pdMS_TO_TICKS(50))) {
00414             if (io_num == INT_A_PIN) {
00415                 ESP_ERROR_CHECK_WITHOUT_ABORT(mcp_interrupt_flag(PORTA, &scratch_data));
00416                 ESP_ERROR_CHECK_WITHOUT_ABORT(mcp_read_port(PORTA, &mcp_porta));
00417
00418                 if (scratch_data & TERMO_SW_MASK) {
00419                     if (mcp_porta & TERMO_SW_MASK) {
00420                         last_interrupt = TERMO_SW_RELEASED;
00421                     } else {
00422                         last_interrupt = TERMO_SW_PRESSED;
00423                     }
00424                 } else if (scratch_data & FREQ_SEL_MASK) {
00425                     mcp_porta = ~mcp_porta;
00426                     speed_selector = mcp_porta & FREQ_SEL_MASK;
00427
00428 // 00ss ss00
00429 // 0000 ssss
00430                     speed_selector = (speed_selector » (FREQ_SEL_1_PIN));
00431
00432                     last_interrupt = SPEED_SELECTOR_0 + speed_selector;
00433                 } else if (scratch_data & STOP_SW_MASK) {
00434                     if (mcp_porta & STOP_SW_MASK) {
00435                         last_interrupt = STOP_RELEASED;
00436                     } else {
00437                         last_interrupt = STOP_PRESSED;
00438                     }
00439
00440                 } else if (io_num == INT_B_PIN) {
00441                     ESP_ERROR_CHECK_WITHOUT_ABORT(mcp_read_port(PORTB, &mcp_portb));
00442                     switch (mcp_portb) {
00443                         case SWITCH_BUTTON_MENU:
00444                             last_interrupt = BUTTON_MENU;
00445                             break;
00446                         case SWITCH_BUTTON_DOWN:
00447                             last_interrupt = BUTTON_DOWN;
00448                             break;
00449                         case SWITCH_BUTTON_RIGHT:
00450                             last_interrupt = BUTTON_RIGHT;
00451                             break;
00452                         case SWITCH_BUTTON_LEFT:
00453                             last_interrupt = BUTTON_LEFT;
00454                             break;
00455                         case SWITCH_BUTTON_UP:
00456                             last_interrupt = BUTTON_UP;
00457                             break;
00458                         case SWITCH_BUTTON_OK:
00459                             last_interrupt = BUTTON_OK;
00460                             break;
00461                         case SWITCH_BUTTON_SAVE:
00462                             last_interrupt = BUTTON_SAVE;
00463                             break;
00464                         case SWITCH_BUTTON_BACK:
00465                             last_interrupt = BUTTON_BACK;
00466                             break;
00467
00468                     }
00469                 } else if (io_num == STOP_BUTTON_PIN) {
00470                     if (gpio_get_level(io_num)) {
00471                         last_interrupt = EMERGENCI_STOP_PRESSED;
00472                     } else {
00473                         last_interrupt = EMERGENCI_STOP_RELEASED;
00474                     }
00475                 } else if (io_num == START_BUTTON_PIN) {
00476                     if (!gpio_get_level(io_num)) {
00477                         last_interrupt = START_PRESSED;
00478                     } else {
00479                         last_interrupt = START_RELEASED;
00480                     }
00481
00482             }
00483         }
00484     }
00485
00486     vTaskDelete(NULL);
00487
00488     return NULL;
00489 }

```

```
00475         }
00476     }
00477     continue;
00478 }
00479 mcp_read_port (PORTA, &mcp_porta);
00480 mcp_read_port (PORTB, &mcp_portb);
00481
00482 switch (last_interrupt) {
00483     case BUTTON_MENU:
00484         if ( mcp_portb == SWITCH_BUTTON_MENU ) {
00485             DisplayEventPost (last_interrupt);
00486             BuzzerEventPost ( 50 );
00487         }
00488         break;
00489     case BUTTON_DOWN:
00490         if ( mcp_portb == SWITCH_BUTTON_DOWN ) {
00491             DisplayEventPost (last_interrupt);
00492             BuzzerEventPost ( 50 );
00493         }
00494         break;
00495     case BUTTON_RIGHT:
00496         if ( mcp_portb == SWITCH_BUTTON_RIGHT ) {
00497             DisplayEventPost (last_interrupt);
00498             BuzzerEventPost ( 50 );
00499         }
00500         break;
00501     case BUTTON_LEFT:
00502         if ( mcp_portb == SWITCH_BUTTON_LEFT ) {
00503             DisplayEventPost (last_interrupt);
00504             BuzzerEventPost ( 50 );
00505         }
00506         break;
00507     case BUTTON_UP:
00508         if ( mcp_portb == SWITCH_BUTTON_UP ) {
00509             DisplayEventPost (last_interrupt);
00510             BuzzerEventPost ( 50 );
00511         }
00512         break;
00513     case BUTTON_OK:
00514         if ( mcp_portb == SWITCH_BUTTON_OK ) {
00515             DisplayEventPost (last_interrupt);
00516             BuzzerEventPost ( 50 );
00517         }
00518         break;
00519     case BUTTON_SAVE:
00520         if ( mcp_portb == SWITCH_BUTTON_SAVE ) {
00521             DisplayEventPost (last_interrupt);
00522             BuzzerEventPost ( 50 );
00523         }
00524         break;
00525     case BUTTON_BACK:
00526         if ( mcp_portb == SWITCH_BUTTON_BACK ) {
00527             DisplayEventPost (last_interrupt);
00528             BuzzerEventPost ( 50 );
00529         }
00530         break;
00531     case TERMO_SW_PRESSED:
00532         if ( !(mcp_porta & TERMO_SW_MASK) ) {
00533             SystemEventPost (TERMO_SW_PRESSED);
00534         }
00535         break;
00536     case TERMO_SW_RELEASED:
00537         if ( mcp_porta & TERMO_SW_MASK ) {
00538             SystemEventPost (TERMO_SW_RELEASED);
00539         }
00540         break;
00541     case STOP_PRESSED:
00542         if ( !(mcp_porta & STOP_SW_MASK) ) {
00543             SystemEventPost (STOP_PRESSED);
00544         }
00545         break;
00546     case STOP_RELEASED:
00547         if ( mcp_porta & STOP_SW_MASK ) {
00548             SystemEventPost (STOP_RELEASED);
00549         }
00550         break;
00551     case SPEED_SELECTOR_0:
00552     case SPEED_SELECTOR_1:
00553     case SPEED_SELECTOR_2:
00554     case SPEED_SELECTOR_3:
00555     case SPEED_SELECTOR_4:
00556     case SPEED_SELECTOR_5:
00557     case SPEED_SELECTOR_6:
00558     case SPEED_SELECTOR_7:
00559     case SPEED_SELECTOR_8:
00560     case SPEED_SELECTOR_9:
00561         mcp_porta = ~mcp_porta;
```

```

00562         speed_selector = mcp_porta & FREQ_SEL_MASK;
00563 // 00ss ss00
00564 // 0000 ssss
00565         speed_selector = ( speed_selector » ( FREQ_SEL_1_PIN ) );
00566     }
00567     break;
00568     case EMERGENCI_STOP_PRESSED:
00569         if ( gpio_get_level(STOP_BUTTON_PIN) ) {
00570             SystemEventPost(EMERGENCI_STOP_PRESSED);
00571         }
00572     break;
00573     case EMERGENCI_STOP_RELEASED:
00574         if ( !gpio_get_level(STOP_BUTTON_PIN) ) {
00575             SystemEventPost(EMERGENCI_STOP_RELEASED);
00576         }
00577     break;
00578     case START_PRESSED:
00579         if ( !gpio_get_level(START_BUTTON_PIN) ) {
00580             SystemEventPost(START_PRESSED);
00581         }
00582     break;
00583     case START_RELEASED:
00584         if ( gpio_get_level(START_BUTTON_PIN) ) {
00585             SystemEventPost(START_RELEASED);
00586         }
00587     break;
00588 }
00589     last_interrupt = 0;
00590 }
00591 }
00592
00593 esp_err_t RelayEventPost( uint8_t relay_event ) {
00594     uint8_t event = relay_event;
00595     if ( event == 1 || event == 0 ) {
00596         return xQueueSend(relay_evt_queue, &event, 0);
00597     }
00598     return ESP_FAIL;
00599 }
00600
00601 esp_err_t BuzzerEventPost( uint32_t buzzer_time_on ) {
00602     if ( buzzer_time_on > 3000 ) {
00603         buzzer_time_on = 3000;
00604     }
00605     return xQueueSend(buzzer_evt_queue, &buzzer_time_on, 0);
00606 }
```

## 10.57. Referencia del archivo

### C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/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 active o desactive de acuerdo a la variable relay\_event.*

- esp\_err\_t **BuzzerEventPost** (uint32\_t buzzer\_time\_on)

*Envía una señal para hacer sonar el buzzer durante buzzer\_time\_on milisegundos.*

### 10.57.1. 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](#).

### 10.57.2. Documentación de funciones

#### 10.57.2.1. BuzzerEventPost()

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

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

#### Parámetros

in	<code>buzzer_time_on</code>	Tiempo en mili segundos durante el que el buzzer sonará.
----	-----------------------------	--

#### Devuelve

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

Definición en la línea 601 del archivo [io\\_control.c](#).

#### 10.57.2.2. GPIO\_interrupt\_attendance\_task()

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

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

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

Definición en la línea 386 del archivo [io\\_control.c](#).

### 10.57.2.3. RelayEventPost()

```
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`.

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 593 del archivo [io\\_control.c](#).

#### 10.57.2.4. `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](#).

## 10.58. `io_control.h`

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

```
00001
00009 #ifndef __IO_CONTROL_H__
00010
00011 #define __IO_CONTROL_H__
00012
00021 void GPIO_interrupt_attendance_task(void* arg);
00022
00037 esp_err_t set_freq_output(uint16_t freq);
00038
00053 esp_err_t RelayEventPost( uint8_t relay_event );
00054
00067 esp_err_t BuzzerEventPost( uint32_t buzzer_time_on );
00068
00069 #endif
```

## 10.59. Referencia del archivo

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/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/semphr.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

**10.59.1. 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](#).

**10.59.2. Documentación de «define»****10.59.2.1. 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](#).

### 10.59.2.2. 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](#).

### 10.59.2.3. 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](#).

### 10.59.2.4. 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](#).

### 10.59.2.5. 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](#).

### 10.59.2.6. 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](#).

### 10.59.2.7. 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](#).

### 10.59.3. Documentación de funciones

#### 10.59.3.1. 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](#).

#### 10.59.3.2. 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](#).

#### 10.59.3.3. 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](#).

#### 10.59.3.4. MCP23017\_INIT()

```
esp_err_t MCP23017_INIT (
    void )
```

Función dedicada a inicializar y configurar el MCP23017. El chip utiliza I<sup>2</sup>C.

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 I<sup>2</sup>C 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](#).

#### 10.59.3.5. 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 I<sup>2</sup>C 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](#).

### 10.59.3.6. `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](#).

**10.59.3.7. 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](#).

**10.59.3.8. 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](#).

**10.59.3.9. 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](#).

**10.59.3.10. 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](#).

#### 10.59.3.11. 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](#).

#### 10.59.4. Documentación de variables

##### 10.59.4.1. TAG

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

Varaible que se imprime en los ESP\_LOG

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

#### 10.59.4.2. 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](#).

## 10.60. 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/sempwr.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_OPCODE__, 0);
00151     i2c_master_write_byte(cmd, register_address, true);
00152     i2c_master_start(cmd);
00153     i2c_master_write_byte(cmd, __MCP23017_READ_OPCODE__, 0);
00154     i2c_master_read_byte(cmd, data, I2C_MASTER_NACK);
00155     i2c_master_stop(cmd);
00156     ret = i2c_master_cmd_begin(I2C_IO_MASTER_NUM, cmd, pdMS_TO_TICKS(I2C_MASTER_TIMEOUT_MS));
00157     i2c_cmd_link_delete(cmd);
00158     return ret;
00159 }
00160
00161 static esp_err_t mcp_register_write(uint8_t register_address, uint8_t data) {
00162     esp_err_t ret = ESP_OK;
00163     i2c_cmd_handle_t cmd = i2c_cmd_link_create();
00164     i2c_master_start(cmd);
00165     i2c_master_write_byte(cmd, __MCP23017_WRITE_OPCODE__, 0);
00166     i2c_master_write_byte(cmd, register_address, true);
00167     i2c_master_write_byte(cmd, data, true);
00168     i2c_master_stop(cmd);
00169     ret = i2c_master_cmd_begin(I2C_IO_MASTER_NUM, cmd, pdMS_TO_TICKS(I2C_MASTER_TIMEOUT_MS));
00170     i2c_cmd_link_delete(cmd);
00171     return ret;
00172 }
00173
00174 esp_err_t MCP23017_INIT( void ) {
00175     esp_err_t ret = ESP_OK;
00176     ESP_LOGI(TAG, "Iniciando modulo");
00177     if ( i2c_master_init() == ESP_OK ) {
00178         ESP_LOGI(TAG, "I2C Inicializado correctamente");
00179     } else {
00180         ESP_LOGE(TAG, "Error al inicializar I2C");
00181         return ESP_FAIL;
00182     }
00183     MCP23017_IOCON_t iocon;
00184     MCP23017_GPPU_t gppua, gppub;
00185     MCP23017_IODIR_t iodira, iodirb;
00186     MCP23017_GPINEN_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__; los puertos como si guesen de 8 bits y no de 16
00200     gppua.all = __MCP23017_GPPU_PULL_UP_DISABLE__;
00201     gppub.all = __MCP23017_GPPU_PULL_UP_DISABLE__;
00202     gppub.bits.PU0 = __MCP23017_GPPU_PULL_UP_ENABLE__;
00203     gppub.bits.PU1 = __MCP23017_GPPU_PULL_UP_ENABLE__;
00204     gppub.bits.PU2 = __MCP23017_GPPU_PULL_UP_ENABLE__;
00205     gppub.bits.PU3 = __MCP23017_GPPU_PULL_UP_ENABLE__;
00206
00207     iodira.all = 0x00;
00208     iodira.bits.IO0 = __MCP23017_IODIR_OUTPUT__;
00209     iodira.bits.IO1 = __MCP23017_IODIR_OUTPUT__;
00210     iodira.bits.IO2 = __MCP23017_IODIR_INPUT__;
00211     iodira.bits.IO3 = __MCP23017_IODIR_INPUT__;
00212     iodira.bits.IO4 = __MCP23017_IODIR_INPUT__;
00213     iodira.bits.IO5 = __MCP23017_IODIR_INPUT__;
00214     iodira.bits.IO6 = __MCP23017_IODIR_INPUT__;
00215     iodira.bits.IO7 = __MCP23017_IODIR_OUTPUT__;
00216
00217     iodirb.all = 0x00;
00218     iodirb.bits.IO0 = __MCP23017_IODIR_INPUT__;
00219     iodirb.bits.IO1 = __MCP23017_IODIR_INPUT__;
00220     iodirb.bits.IO2 = __MCP23017_IODIR_INPUT__;
00221     iodirb.bits.IO3 = __MCP23017_IODIR_INPUT__;
00222     iodirb.bits.IO4 = __MCP23017_IODIR_OUTPUT__;
00223     iodirb.bits.IO5 = __MCP23017_IODIR_OUTPUT__;
00224     iodirb.bits.IO6 = __MCP23017_IODIR_OUTPUT__;
00225     iodirb.bits.IO7 = __MCP23017_IODIR_OUTPUT__;
00226     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 }
```

## 10.61. Referencia del archivo

**C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDe←  
Frecuencia/Software/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 escribe el registro OLAT del port del MCP23017.*

### 10.61.1. 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](#).

## 10.61.2. Documentación de funciones

### 10.61.2.1. MCP23017\_INIT()

```
esp_err_t MCP23017_INIT (
    void )
```

Función dedicada a inicializar y configurar el MCP23017. El chip utiliza I2C.

Inicializa el MCP23017. 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](#).

### 10.61.2.2. 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](#).

### 10.61.2.3. 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

<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 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](#).

### 10.61.2.4. 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

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

### 10.61.2.5. 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](#).

### 10.61.2.6. 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](#).

## 10.62. 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

```

## 10.63. Referencia del archivo C:/Users/User/Desktop/ProyectoFinal- Grupo5-VariadorDeFrecuencia/Software/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\\_INTCON\\_t](#)
- union [MCP23017\\_IOCON\\_t](#)
- union [MCP23017\\_INTF\\_t](#)
- union [MCP23017\\_INTCAP\\_t](#)

defines

- #define \_\_MCP23017\_BANK\_SEQUENTIAL\_\_ 0
- #define \_\_MCP23017\_BANK\_SPLIT\_\_ 1
- #define \_\_MCP23017\_MIRROR\_UNCONNECTED\_\_ 0
- #define \_\_MCP23017\_MIRROR\_CONNECTED\_\_ 1
- #define \_\_MCP23017\_SEQOP\_ENABLED\_\_ 0
- #define \_\_MCP23017\_SEQOP\_DISABLED\_\_ 1
- #define \_\_MCP23017\_DISSLW\_DISABLED\_\_ 0
- #define \_\_MCP23017\_DISSLW\_ENABLED\_\_ 1
- #define \_\_MCP23017\_ODR\_OPEN\_DRAIN\_OUTPUT\_\_ 1
- #define \_\_MCP23017\_ODR\_ACTIVE\_DRIVER\_OUTPUT\_\_ 0
- #define \_\_MCP23017\_INTPOL\_POLARITY\_LOW\_\_ 1
- #define \_\_MCP23017\_INTPOL\_POLARITY\_HIGH\_\_ 0
- #define \_\_MCP23017\_GPPU\_PULL\_UP\_ENABLE\_\_ 1
- #define \_\_MCP23017\_GPPU\_PULL\_UP\_DISABLE\_\_ 0
- #define \_\_MCP23017\_IODIR\_INPUT\_\_ 1
- #define \_\_MCP23017\_IODIR\_OUTPUT\_\_ 0
- #define \_\_MCP23017\_GPINTEN\_ENABLE\_\_ 1
- #define \_\_MCP23017\_GPINTEN\_DISABLE\_\_ 0
- #define \_\_MCP23017\_INTCON\_DEFVAL\_\_ 1
- #define \_\_MCP23017\_INTCON\_CHANGE\_\_ 0
- #define \_\_MCP23017\_WRITE\_\_ 0
- #define \_\_MCP23017\_READ\_\_ 1
- #define \_\_MCP23017\_POSITIVE\_LOGIC\_\_ 0
- #define \_\_MCP23017\_OPPOSITE\_LOGIC\_\_ 1
- #define \_\_MCP23017\_INTERRUPT\_ENABLE\_\_ 0
- #define \_\_MCP23017\_INTERRUPT\_DISABLE\_\_ 1
- #define \_\_MCP23017\_ADDRESS\_\_ 0b010
- #define \_\_MCP23017\_READ\_OPCODE\_\_ 0b0100 << 4 | \_\_MCP23017\_ADDRESS\_\_ << 1 | \_\_MCP23017\_READ\_\_
- #define \_\_MCP23017\_WRITE\_OPCODE\_\_ 0b0100 << 4 | \_\_MCP23017\_ADDRESS\_\_ << 1 | \_\_MCP23017\_WRITE\_\_
- #define \_\_MCP23017\_FUNC\_MODE\_\_ \_\_MCP23017\_BANK\_SEQUENTIAL\_\_
- #define MCP23017\_IODIRA\_REGISTER 0x00
- #define MCP23017\_IODIRB\_REGISTER 0x01
- #define MCP23017\_IPOLA\_REGISTER 0x02
- #define MCP23017\_IPOLB\_REGISTER 0x03
- #define MCP23017\_GPINTENA\_REGISTER 0x04
- #define MCP23017\_GPINTENB\_REGISTER 0x05
- #define MCP23017\_DEFVALA\_REGISTER 0x06
- #define MCP23017\_DEFVALB\_REGISTER 0x07
- #define MCP23017\_INTCONA\_REGISTER 0x08
- #define MCP23017\_INTCONB\_REGISTER 0x09
- #define MCP23017\_IOCON\_REGISTER 0x0A
- #define MCP23017\_GPPUA\_REGISTER 0x0C
- #define MCP23017\_GPPUB\_REGISTER 0x0D
- #define MCP23017\_INTFA\_REGISTER 0x0E
- #define MCP23017\_INTFB\_REGISTER 0x0F
- #define MCP23017\_INTCAPA\_REGISTER 0x10
- #define MCP23017\_INTCAPB\_REGISTER 0x11
- #define MCP23017\_GPIOA\_REGISTER 0x12
- #define MCP23017\_GPIOB\_REGISTER 0x13
- #define MCP23017\_OLATA\_REGISTER 0x14
- #define MCP23017\_OLATB\_REGISTER 0x15

## Enumeraciones

- enum `mcp_port_e` { `PORTA` = 0 , `PORTB` = 1 }

*Selector de puertos del MCP23017. Puede ser PORTA (0) o PORTB (1). De esta manera el usuario se 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.*

### 10.63.1. Documentación de «define»

#### 10.63.1.1. \_\_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](#).

#### 10.63.1.2. \_\_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](#).

#### 10.63.1.3. \_\_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](#).

#### 10.63.1.4. \_\_MCP23017\_DISSLW\_DISABLED\_\_

```
#define __MCP23017_DISSLW_DISABLED__ 0
```

Slew rate deshabilitado

Definición en la línea 20 del archivo [mcp23017\\_defs.h](#).

#### 10.63.1.5. \_\_MCP23017\_DISSLW\_ENABLED\_\_

```
#define __MCP23017_DISSLW_ENABLED__ 1
```

Slew rate habilitado

Definición en la línea 21 del archivo [mcp23017\\_defs.h](#).

#### 10.63.1.6. \_\_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](#).

#### 10.63.1.7. \_\_MCP23017\_GPINTEN\_DISABLE\_\_

```
#define __MCP23017_GPINTEN_DISABLE__ 0
```

Interrupción de GPIO deshabilitada

Definición en la línea 36 del archivo [mcp23017\\_defs.h](#).

#### 10.63.1.8. \_\_MCP23017\_GPINTEN\_ENABLE\_\_

```
#define __MCP23017_GPINTEN_ENABLE__ 1
```

Interrupción de GPIO habilitada

Definición en la línea 35 del archivo [mcp23017\\_defs.h](#).

#### 10.63.1.9. \_\_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](#).

#### 10.63.1.10. \_\_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](#).

#### 10.63.1.11. \_\_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](#).

#### 10.63.1.12. \_\_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](#).

#### 10.63.1.13. \_\_MCP23017\_INTERRUPT\_DISABLE\_\_

```
#define __MCP23017_INTERRUPT_DISABLE__ 1
```

Habilitación de la interrupción del GPIO

Definición en la línea 48 del archivo [mcp23017\\_defs.h](#).

#### 10.63.1.14. \_\_MCP23017\_INTERRUPT\_ENABLE\_\_

```
#define __MCP23017_INTERRUPT_ENABLE__ 0
```

Deshabilitación de la interrupción del GPIO

Definición en la línea 47 del archivo [mcp23017\\_defs.h](#).

#### 10.63.1.15. \_\_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](#).

#### 10.63.1.16. \_\_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](#).

#### 10.63.1.17. \_\_MCP23017\_IODIR\_INPUT\_\_

```
#define __MCP23017_IODIR_INPUT__ 1
```

GPIO configurada como entrada

Definición en la línea 32 del archivo [mcp23017\\_defs.h](#).

#### 10.63.1.18. \_\_MCP23017\_IODIR\_OUTPUT\_\_

```
#define __MCP23017_IODIR_OUTPUT__ 0
```

GPIO configurada como salida

Definición en la línea 33 del archivo [mcp23017\\_defs.h](#).

#### 10.63.1.19. \_\_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](#).

#### 10.63.1.20. \_\_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](#).

#### 10.63.1.21. \_\_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](#).

#### 10.63.1.22. \_\_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](#).

#### 10.63.1.23. \_\_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](#).

#### 10.63.1.24. \_\_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](#).

#### 10.63.1.25. \_\_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](#).

#### 10.63.1.26. \_\_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](#).

#### 10.63.1.27. \_\_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](#).

#### 10.63.1.28. \_\_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](#).

#### 10.63.1.29. \_\_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](#).

#### 10.63.1.30. \_\_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](#).

#### 10.63.1.31. MCP23017\_DEFVALA\_REGISTER

```
#define MCP23017_DEFVALA_REGISTER 0x06
```

Dirección de memoria del registro DEFVALA

Definición en la línea 63 del archivo [mcp23017\\_defs.h](#).

#### 10.63.1.32. MCP23017\_DEFVALB\_REGISTER

```
#define MCP23017_DEFVALB_REGISTER 0x07
```

Dirección de memoria del registro DEFVALB

Definición en la línea 64 del archivo [mcp23017\\_defs.h](#).

#### 10.63.1.33. MCP23017\_GPINTENA\_REGISTER

```
#define MCP23017_GPINTENA_REGISTER 0x04
```

Dirección de memoria del registro GPINTENA

Definición en la línea 61 del archivo [mcp23017\\_defs.h](#).

#### 10.63.1.34. MCP23017\_GPINTENB\_REGISTER

```
#define MCP23017_GPINTENB_REGISTER 0x05
```

Dirección de memoria del registro GPINTENB

Definición en la línea 62 del archivo [mcp23017\\_defs.h](#).

#### 10.63.1.35. MCP23017\_GPIOA\_REGISTER

```
#define MCP23017_GPIOA_REGISTER 0x12
```

Dirección de memoria del registro GPIOA

Definición en la línea 75 del archivo [mcp23017\\_defs.h](#).

### 10.63.1.36. MCP23017\_GPIOB\_REGISTER

```
#define MCP23017_GPIOB_REGISTER 0x13
```

Dirección de memoria del registro GPIOB

Definición en la línea 76 del archivo [mcp23017\\_defs.h](#).

### 10.63.1.37. MCP23017\_GPPUA\_REGISTER

```
#define MCP23017_GPPUA_REGISTER 0x0C
```

Dirección de memoria del registro GPPUA

Definición en la línea 69 del archivo [mcp23017\\_defs.h](#).

### 10.63.1.38. MCP23017\_GPPUB\_REGISTER

```
#define MCP23017_GPPUB_REGISTER 0x0D
```

Dirección de memoria del registro GPPUB

Definición en la línea 70 del archivo [mcp23017\\_defs.h](#).

### 10.63.1.39. MCP23017\_INTCAPA\_REGISTER

```
#define MCP23017_INTCAPA_REGISTER 0x10
```

Dirección de memoria del registro INTCAPA

Definición en la línea 73 del archivo [mcp23017\\_defs.h](#).

### 10.63.1.40. MCP23017\_INTCAPB\_REGISTER

```
#define MCP23017_INTCAPB_REGISTER 0x11
```

Dirección de memoria del registro INTCAPB

Definición en la línea 74 del archivo [mcp23017\\_defs.h](#).

### 10.63.1.41. MCP23017\_INTCONA\_REGISTER

```
#define MCP23017_INTCONA_REGISTER 0x08
```

Dirección de memoria del registro INTCONA

Definición en la línea 65 del archivo [mcp23017\\_defs.h](#).

#### 10.63.1.42. MCP23017\_INTCONB\_REGISTER

```
#define MCP23017_INTCONB_REGISTER 0x09
```

Dirección de memoria del registro INTCONB

Definición en la línea 66 del archivo [mcp23017\\_defs.h](#).

#### 10.63.1.43. MCP23017\_INTFA\_REGISTER

```
#define MCP23017_INTFA_REGISTER 0x0E
```

Dirección de memoria del registro INTFA

Definición en la línea 71 del archivo [mcp23017\\_defs.h](#).

#### 10.63.1.44. MCP23017\_INTFB\_REGISTER

```
#define MCP23017_INTFB_REGISTER 0x0F
```

Dirección de memoria del registro INTFB

Definición en la línea 72 del archivo [mcp23017\\_defs.h](#).

#### 10.63.1.45. MCP23017\_IOCON\_REGISTER

```
#define MCP23017_IOCON_REGISTER 0x0A
```

Dirección de memoria del registro IOCON

Definición en la línea 67 del archivo [mcp23017\\_defs.h](#).

#### 10.63.1.46. MCP23017\_IODIRA\_REGISTER

```
#define MCP23017_IODIRA_REGISTER 0x00
```

Dirección de memoria del registro IODIRA

Definición en la línea 57 del archivo [mcp23017\\_defs.h](#).

#### 10.63.1.47. MCP23017\_IODIRB\_REGISTER

```
#define MCP23017_IODIRB_REGISTER 0x01
```

Dirección de memoria del registro IODIRB

Definición en la línea 58 del archivo [mcp23017\\_defs.h](#).

#### 10.63.1.48. MCP23017\_IPOLA\_REGISTER

```
#define MCP23017_IPOLA_REGISTER 0x02
```

Dirección de memoria del registro IPOLA

Definición en la línea [59](#) del archivo [mcp23017\\_defs.h](#).

#### 10.63.1.49. MCP23017\_IPOLB\_REGISTER

```
#define MCP23017_IPOLB_REGISTER 0x03
```

Dirección de memoria del registro IPOLB

Definición en la línea [60](#) del archivo [mcp23017\\_defs.h](#).

#### 10.63.1.50. MCP23017\_OLATA\_REGISTER

```
#define MCP23017_OLATA_REGISTER 0x14
```

Dirección de memoria del registro OLATA

Definición en la línea [77](#) del archivo [mcp23017\\_defs.h](#).

#### 10.63.1.51. MCP23017\_OLATB\_REGISTER

```
#define MCP23017_OLATB_REGISTER 0x15
```

Dirección de memoria del registro OLATB

Definición en la línea [78](#) del archivo [mcp23017\\_defs.h](#).

### 10.63.2. Documentación de enumeraciones

#### 10.63.2.1. mcp\_port\_e

```
enum mcp_port_e
```

Selector de puertos del MCP23017. Puede ser PORTA (0) o PORTB (1). De esta manera el usuario se abraza en conocer como se llaman los registros del chip, solo con las funciones específicas de lectura y escritura, datos, pines y esta definición de puerto.

#### Valores de enumeraciones

PORTE	Puerto A del MCP23017
PORTB	Puerto B del MCP23017

Definición en la línea [340](#) del archivo [mcp23017\\_defs.h](#).

## 10.64. 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
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

```

## 10.65. Referencia del archivo

**C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDe←  
Frecuencia/Software/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_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 }`

### 10.65.1. 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](#).

## 10.65.2. Documentación de «define»

### 10.65.2.1. LINE\_INCREMENT

```
#define LINE_INCREMENT 9
```

Definición en la línea 15 del archivo [LVFV\\_system.h](#).

### 10.65.2.2. SH1106\_SIZE\_1

```
#define SH1106_SIZE_1 0
```

Definición en la línea 124 del archivo [LVFV\\_system.h](#).

### 10.65.2.3. SH1106\_SIZE\_2

```
#define SH1106_SIZE_2 1
```

Definición en la línea 125 del archivo [LVFV\\_system.h](#).

### 10.65.2.4. VARIABLE\_EIGHTH

```
#define VARIABLE_EIGHTH 83
```

Definición en la línea 24 del archivo [LVFV\\_system.h](#).

### 10.65.2.5. VARIABLE\_FIFTH

```
#define VARIABLE_FIFTH 56
```

Definición en la línea 21 del archivo [LVFV\\_system.h](#).

### 10.65.2.6. VARIABLE\_FIRST

```
#define VARIABLE_FIRST 20
```

Definición en la línea 17 del archivo [LVFV\\_system.h](#).

### 10.65.2.7. VARIABLE\_FOURTH

```
#define VARIABLE_FOURTH 47
```

Definición en la línea 20 del archivo [LVFV\\_system.h](#).

### 10.65.2.8. VARIABLE\_SECOND

```
#define VARIABLE_SECOND 29
```

Definición en la línea 18 del archivo [LVFV\\_system.h](#).

### 10.65.2.9. VARIABLE\_SEVENTH

```
#define VARIABLE_SEVENTH 74
```

Definición en la línea 23 del archivo [LVFV\\_system.h](#).

### 10.65.2.10. VARIABLE\_SIXTH

```
#define VARIABLE_SIXTH 65
```

Definición en la línea 22 del archivo [LVFV\\_system.h](#).

### 10.65.2.11. VARIABLE\_THIRD

```
#define VARIABLE_THIRD 38
```

Definición en la línea 19 del archivo [LVFV\\_system.h](#).

## 10.65.3. Documentación de «typedef»

### 10.65.3.1. frequency\_settings\_SH1106\_t

```
typedef struct frequency_settings_SH1106_t frequency_settings_SH1106_t
```

### 10.65.3.2. frequency\_settings\_t

```
typedef struct frequency_settings_t frequency_settings_t
```

### 10.65.3.3. seccurity\_settings\_SH1106\_t

```
typedef struct seccurity_settings_SH1106_t seccurity_settings_SH1106_t
```

### 10.65.3.4. seccurity\_settings\_t

```
typedef struct seccurity_settings_t seccurity_settings_t
```

### 10.65.3.5. sh1106\_variable\_lines\_e

```
typedef enum sh1106_variable_lines_e sh1106_variable_lines_e
```

### 10.65.3.6. system\_status\_e

```
typedef enum system_status_e system_status_e
```

### 10.65.3.7. system\_status\_t

```
typedef struct system_status_t system_status_t
```

### 10.65.3.8. time\_settings\_SH1106\_t

```
typedef struct time_settings_SH1106_t time_settings_SH1106_t
```

### 10.65.3.9. time\_settings\_t

```
typedef struct time_settings_t time_settings_t
```

## 10.65.4. Documentación de enumeraciones

### 10.65.4.1. sh1106\_variable\_lines\_e

```
enum sh1106_variable_lines_e
```

#### Valores de enumeraciones

first	
second	
third	
fourth	
fifth	
sixth	
seventh	

Definición en la línea 76 del archivo [LVFV\\_system.h](#).

### 10.65.4.2. system\_status\_e

```
enum system_status_e
```

Posibles estados general que puede tomar el sistema.

#### Valores de enumeraciones

---

SYSTEM_IDLE	
SYSTEM_ACCEL_DESACCEL	
SYSTEM_REGIME	
SYSTEM_BREAKING	
SYSTEM_EMERGENCY	
SYSTEM_EMERGENCY_OK	

Definición en la línea 67 del archivo [LVFV\\_system.h](#).

#### 10.65.4.3. 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](#).

## 10.66. LVFV\_system.h

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

```

00001
00009 #ifndef LVFV_SYSTEM_H
00010 #define LVFV_SYSTEM_H
00011
00012 #include <inttypes.h>
00013 #include <time.h>
00014
00015 #define LINE_INCREMENT 9
00016
00017 #define VARIABLE_FIRST 20
00018 #define VARIABLE_SECOND 29
00019 #define VARIABLE_THIRD 38
00020 #define VARIABLE_FOURTH 47
00021 #define VARIABLE_FIFTH 56
00022 #define VARIABLE_SIXTH 65
00023 #define VARIABLE_SEVENTH 74
00024 #define VARIABLE_EIGHTH 83
00025
00031 typedef enum {
00032     BUTTON_MENU = 1,
00033     BUTTON_OK,
00034     BUTTON_BACK,
00035     BUTTON_UP,
00036     BUTTON_DOWN,
00037     BUTTON_LEFT,
00038     BUTTON_RIGHT,
00039     BUTTON_SAVE,
00040     EMERGENCY_STOP_PRESSED,
00041     EMERGENCY_STOP_RELEASED,
00042     START_PRESSED,
00043     START_RELEASED,
00044     TERMO_SW_PRESSED,
00045     TERMO_SW_RELEASED,
00046     STOP_PRESSED,
00047     STOP_RELEASED,
00048     SPEED_SELECTOR_0,
00049     SPEED_SELECTOR_1,
00050     SPEED_SELECTOR_2,
00051     SPEED_SELECTOR_3,
00052     SPEED_SELECTOR_4,
00053     SPEED_SELECTOR_5,
00054     SPEED_SELECTOR_6,
00055     SPEED_SELECTOR_7,
00056     SPEED_SELECTOR_8,
00057     SPEED_SELECTOR_9,
00058     SECURITY_EXCEEDED,
00059     SECURITY_OK
00060 } systemSignal_e;
00061
00067 typedef enum system_status_e {

```

```

00068     SYSTEM_IDLE,
00069     SYSTEM_ACCEL_DESACCEL,
00070     SYSTEM_REGIME,
00071     SYSTEM_BREAKING,
00072     SYSTEM_EMERGENCY,
00073     SYSTEM_EMERGENCY_OK
00074 } system_status_e;
00075
00076 typedef enum sh1106_variable_lines_e{
00077     first = VARIABLE_FIRST,
00078     second = VARIABLE_SECOND,
00079     third = VARIABLE_THIRD,
00080     fourth = VARIABLE_FOURTH,
00081     fifth = VARIABLE_FIFTH,
00082     sixth = VARIABLE_SIXTH,
00083     seventh = VARIABLE_SEVENTH
00084 } sh1106_variable_lines_e;
00085
00086 typedef struct frequency_settings_t {
00087     uint16_t freq_regime;
00088     uint16_t acceleration;
00089     uint16_t desacceleration;
00090     uint16_t input_variable;
00091 } frequency_settings_t;
00092
00093 typedef struct time_settings_t {
00094     struct tm *time_system;
00095     struct tm *time_start;
00096     struct tm *time_stop;
00097 } time_settings_t;
00098
00099 typedef struct security_settings_t {
00100     uint16_t vbus_min;
00101     uint16_t ibus_max;
00102 } security_settings_t;
00103
00104 typedef struct system_status_t {
00105     uint16_t frequency;                                     // Frecuencia actual de giro del motor
00106     uint16_t frequency_destiny;                            // Frecuencia a la que terminará
00107     uint16_t vbus_min;                                      // Tensión mínima del bus de continua en
00108     la que será una condición buena                      // Corriente máxima del bus de continua en
00109     uint16_t ibus_max;                                     // Velocidad de aceleración del motor
00110     la que será una condición buena                      // Velocidad de desaceleración del motor
00111     uint16_t acceleration;                                // Índice de entrada aislada seleccionado
00112     configurada por el usuario                           // Estado general del sistema
00113     uint16_t desacceleration;                            // Estado actual de las señales de
00114     configurada por el usuario                           emergencia b0: Entrada emergencia; b1: Security; b3: Termo-switch
00115     uint8_t inputs_status;                               00116     system_status_e status;                             // Estado general del sistema
00117     uint8_t emergency_signals;                          00118     uint8_t emergency_signals;                         // Estado actual de las señales de
00119 } system_status_t;                                     emergencia b0: Entrada emergencia; b1: Security; b3: Termo-switch
00120
00121 /*
00122  * Definiciones para display
00123 */
00124 #define SH1106_SIZE_1          0
00125 #define SH1106_SIZE_2          1
00126
00127 typedef struct frequency_settings_SH1106_t {
00128     frequency_settings_t frequency_settings;
00129     sh1106_variable_lines_e *edit;
00130     uint8_t edit_variable;
00131     uint8_t *edit_flag;
00132     uint8_t *multiplier;
00133 } frequency_settings_SH1106_t;
00134
00135 typedef struct time_settings_SH1106_t {
00136     time_settings_t time_settings;
00137     sh1106_variable_lines_e *edit;
00138     uint8_t edit_variable;
00139     uint8_t *edit_flag;
00140     uint8_t *multiplier;
00141 } time_settings_SH1106_t;
00142
00143 typedef struct security_settings_SH1106_t {
00144     security_settings_t security_settings;
00145     sh1106_variable_lines_e *edit;
00146     uint8_t edit_variable;
00147     uint8_t *edit_flag;
00148     uint8_t *multiplier;
00149 } security_settings_SH1106_t;
00150
00151 #endif

```

## 10.67. Referencia del archivo

**C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDe←  
Frecuencia/Software/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 volátil.
- 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 volátil.
- esp\_err\_t **nvs\_init\_once** (void)
 

Inicializa la memoria NVS una única vez.
- esp\_err\_t **load\_variables** (**frequency\_settings\_t** \*frequency\_settings, **time\_settings\_t** \*time\_settings, **security\_settings\_t** \*security\_settings)
 

Carga desde NVS los parámetros de frecuencia, seguridad y ventanas horarias.
- esp\_err\_t **save\_variables** (**frequency\_settings\_t** \*frequency\_settings, **time\_settings\_t** \*time\_settings, **security\_settings\_t** \*security\_settings)
 

Guarda en NVS los parámetros de frecuencia, seguridad y ventanas horarias.

## Variables

- static const char \* TAG = "NVS"

### 10.67.1. 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](#).

### 10.67.2. Documentación de «define»

#### 10.67.2.1. load\_acceleration

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

##### Valor:

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

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

#### 10.67.2.2. load\_desacceleration

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

##### Valor:

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

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

### 10.67.2.3. `load_frequency`

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

**Valor:**

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

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

### 10.67.2.4. `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](#).

### 10.67.2.5. `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](#).

### 10.67.2.6. `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](#).

### 10.67.2.7. `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](#).

### 10.67.2.8. 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](#).

### 10.67.2.9. 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](#).

### 10.67.2.10. 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](#).

### 10.67.2.11. save\_acceleration

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

**Valor:**

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

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

### 10.67.2.12. save\_desacceleration

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

**Valor:**

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

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

### 10.67.2.13. `save_frequency`

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

**Valor:**

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

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

### 10.67.2.14. `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](#).

### 10.67.2.15. `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](#).

### 10.67.2.16. `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](#).

### 10.67.2.17. `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](#).

### 10.67.2.18. 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](#).

### 10.67.2.19. 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](#).

### 10.67.2.20. 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](#).

## 10.67.3. Documentación de funciones

### 10.67.3.1. 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	<i>defval</i>	Valor default que se utiliza en caso que nunca se haya creado la variable o si existe algún problema al abrir la NVS.
----	---------------	---

### Valores devueltos

- ESP\_OK: Si la lectura de la variable fue exitosa
  - ESP\_FAIL: si hay un interno error; generalmente ocurre cuando la memoria tiene una partición corrupta.
  - ESP\_ERR\_NVS\_NOT\_INITIALIZED: si la memoria no fue inicializada.
  - ESP\_ERR\_NVS\_PART\_NOT\_FOUND: si la partición con la etiqueta "nvs" no se encuentra.
  - ESP\_ERR\_NVS\_NOT\_FOUND: id namespace doesn't exist yet and mode is NVS\_READONLY.
  - ESP\_ERR\_NVS\_INVALID\_NAME: si el nombre del namespace no cumple con las restricciones.
  - ESP\_ERR\_NO\_MEM: en caso que la memoria no 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 [203](#) del archivo [nvs.c](#).

#### 10.67.3.2. `load_variables()`

```
esp_err_t load_variables (
    frequency_settings_t * frequency_settings,
    time_settings_t * time_settings,
    security_settings_t * security_settings)
```

Carga desde NVS los parámetros de frecuencia, seguridad y ventanas horarias.

Utiliza las funciones internas de lectura (`load_16`) para obtener cada parámetro persistido en NVS. Si alguna lectura falla, retorna el primer error encontrado. En caso de no existir una clave, se aplica el valor por defecto definido en cada macro de carga.

### Parámetros

out	<i>frequency_settings</i>	Estructura de parámetros de frecuencia a cargar.
out	<i>time_settings</i>	Estructura de parámetros de tiempo (ventanas start/stop) a cargar.
out	<i>security_settings</i>	Estructura de parámetros de seguridad (vbus/ibus) a cargar.

### Valores devueltos

- ESP\_OK: Si todas las lecturas fueron exitosas (o claves ausentes con default aplicado).
  - ESP\_FAIL: si hay un error interno.
  - ESP\_ERR\_NVS\_NOT\_INITIALIZED: si la NVS no está inicializada.
  - ESP\_ERR\_NVS\_PART\_NOT\_FOUND: si la partición "nvs" no se encuentra.
  - ESP\_ERR\_NVS\_INVALID\_NAME: si alguna clave no cumple restricciones.
  - ESP\_ERR\_NO\_MEM / ESP\_ERR\_NVS\_NOT\_ENOUGH\_SPACE: errores internos de la NVS.
  - ESP\_ERR\_NOT\_ALLOWED: si la partición es solo lectura.
  - ESP\_ERR\_INVALID\_ARG: si el handler interno de NVS es NULL.

Definición en la línea [245](#) del archivo [nvs.c](#).

#### 10.67.3.3. 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](#).

#### 10.67.3.4. 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](#).

**10.67.3.5. 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](#).

#### 10.67.4. Documentación de variables

##### 10.67.4.1. TAG

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

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

## 10.68. 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 }
```

## 10.69. Referencia del archivo

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/nvs/nvs.h

Declaración de funciones de lectura y escritura de memoria no volátil. Útiles para los reset y no tener que reconfigurar siempre las variables básicas del sistema.

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

### Funciones

- esp\_err\_t **nvs\_init\_once** (void)  
*Inicializa la memoria NVS una única vez.*
- esp\_err\_t **load\_variables** (**frequency\_settings\_t** \*frequency\_settings, **time\_settings\_t** \*time\_settings, **security\_settings\_t** \*security\_settings)  
*Carga desde NVS los parámetros de frecuencia, seguridad y ventanas horarias.*
- esp\_err\_t **save\_variables** (**frequency\_settings\_t** \*frequency\_settings, **time\_settings\_t** \*time\_settings, **security\_settings\_t** \*security\_settings)  
*Guarda en NVS los parámetros de frecuencia, seguridad y ventanas horarias.*

### 10.69.1. Descripción detallada

Declaración de funciones de lectura y escritura de memoria no volátil. Útiles para los reset y no tener que reconfigurar siempre las variables básicas del sistema.

#### Autor

Andrenacci - Carra

#### Versión

2.0

#### Fecha

2025-11-06

Definición en el archivo [nvs.h](#).

## 10.69.2. Documentación de funciones

### 10.69.2.1. 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>seccurity_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](#).

### 10.69.2.2. 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](#).

#### 10.69.2.3. 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](#).

## 10.70. 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 *security_settings);
00065
00093 esp_err_t save_variables(frequency_settings_t *frequency_settings, time_settings_t *time_settings,
    security_settings_t *security_settings);
00094
00095 #endif

```

## 10.71. Referencia del archivo

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDe←  
Frecuencia/Software/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.*

## 10.71 Referencia del archivo

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/rtc/rtc365

---

### 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

### 10.71.1. 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](#).

### 10.71.2. Documentación de funciones

#### 10.71.2.1. 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](#).

#### 10.71.2.2. 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](#).

#### 10.71.2.3. `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](#).

#### 10.71.2.4. `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](#).

#### 10.71.2.5. `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

## 10.71 Referencia del archivo

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/rtc/rtc367

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](#).

### 10.71.2.6. `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 <a href="#">time_settings_t</a> 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](#).

### 10.71.2.7. `setTime()`

```
esp_err_t setTime (
    struct tm * setting_time)
```

Inicializa el RTC en la hora cargada en `setting_time`.

No es importante el día configurado para el sistema, pero carga la hora 0 de `UNIX_TIME`.

Parámetros

in	<i>setting_time</i>	Esturctura tm que es cargada en el RTC del ESP32 según su año, mes, día, hora, minuto y segundo.
----	---------------------	--

Devuelve

- ESP\_OK si carga el horario correctamente.
- ESP\_ERR\_INVALID\_ARG en caso de recibir un puntero NULL

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

### 10.71.3. Documentación de variables

#### 10.71.3.1. alarm\_settings

```
time_settings_t alarm_settings [static]
```

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

#### 10.71.3.2. rtc\_alarms

```
rtc_alarms_t rtc_alarms [static]
```

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

#### 10.71.3.3. TAG

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

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

#### 10.71.3.4. time\_start

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

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

#### 10.71.3.5. time\_stop

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

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

## 10.72. rtc.c

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

```

00001
00002 #include <sys/time.h>
00003
00004
00005 #include "sdkconfig.h"
00006 #include "esp_err.h"
00007 #include "esp_timer.h"
00008 #include "esp_log.h"
00009
00010
00011
00012
00013
00014
00015
00016 #include "../system/sysControl.h"
00017 #include "RTC.h"
00018
00019
00020
00021
00022
00023
00024 typedef struct {
00025     esp_timer_handle_t start_timer;
00026     esp_timer_handle_t stop_timer;
00027 } rtc_alarms_t;
00028
00029 static const char *TAG = "RTC";
00030 static mensajes de logs
00031 static rtc_alarms_t rtc_alarms;
00032 static de alarma para inicio y fin de marcha
00033 static time_settings_t alarm_settings;
00034 static // Settings de horarios de inicio y fin de
00035
00036 static // Variable dedicada a la impresión de
00037 static // Declaración de contenedor de handlers
00038
00039 static // Settings de horarios de inicio y fin de
00040
00041
00042
00043
00044
00045
00046 static esp_err_t program_alarm(esp_timer_handle_t *timer_handle, void (*callback)(void *), int hour,
00047 int minute, const char *name);
00048
00049 static void alarm_start_cb(void *arg);
00050
00051 static void alarm_stop_cb(void *arg);
00052
00053
00054 esp_err_t initRTC() {
00055
00056     struct timeval tv = {
00057         .tv_sec = 0,
00058         .tv_usec = 0
00059     };
00060
00061     settimeofday(&tv, NULL); // Cargar en RTC interno
00062
00063     alarm_settings.time_start = &time_start;
00064     alarm_settings.time_stop = &time_stop;
00065
00066     return ESP_OK;
00067 }
00068
00069
00070 esp_err_t setTime(struct tm *setting_time) {
00071
00072     if ( setting_time == NULL ) {
00073         return ESP_ERR_INVALID_ARG;
00074     }
00075
00076     time_t now = mktime(setting_time); // Convierte a timestamp UNIX
00077
00078     struct timeval tv = {
00079         .tv_sec = now,
00080         .tv_usec = 0
00081     };
00082
00083     settimeofday(&tv, NULL); // Cargar en RTC interno
00084     return ESP_OK;
00085 }
00086
00087
00088 esp_err_t getTime(struct tm *current_timeinfo) {
00089
00090     time_t now;
00091
00092     if ( current_timeinfo == NULL ) {
00093         return ESP_ERR_INVALID_ARG;
00094     }
00095
00096     time(&now);
00097     localtime_r(&now, current_timeinfo);
00098
00099     return ESP_OK;
00100 }
00101
00102
00103 esp_err_t rtc_schedule_alarms(time_settings_t *time_settings) {
00104
00105     if ( time_settings != NULL ) {
00106         alarm_settings.time_start->tm_hour = time_settings->time_start->tm_hour;
00107
00108         alarm_settings.time_stop->tm_hour = time_settings->time_stop->tm_hour;
00109
00110         alarm_start_cb(alarm_settings.time_start);
00111
00112         alarm_stop_cb(alarm_settings.time_stop);
00113
00114         return ESP_OK;
00115     }
00116
00117     return ESP_ERR_INVALID_ARG;
00118 }
00119
00120
00121
00122
00123
00124
00125
00126
00127
00128
00129
00130
00131
00132
00133
00134
00135
00136
00137
00138
00139
00140

```

```

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")));

```

## 10.73 Referencia del archivo

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/main/rtc/rtc3.h

```
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 }
```

## 10.73. Referencia del archivo

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDe←  
Frecuencia/Software/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.*

### 10.73.1. 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](#).

## 10.73.2. Documentación de funciones

### 10.73.2.1. getTime()

```
esp_err_t getTime (
    struct tm * current_timeinfo)
```

Carga la hora actual 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](#).

### 10.73.2.2. 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](#).

### 10.73.2.3. 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, 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](#).

**10.73.2.4. 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](#).

**10.74. 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
```

## 10.75. Referencia del archivo

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDe←  
Frecuencia/Software/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/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "esp_system.h"
#include "SysAdmin.h"
#include "SysControl.h"
#include "../display/display.h"
```

### Funciones

- static void **accelerating** (void \*pvParameters)
 

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

*Tarea dedicada a decrementar la frecuencia que es impresa en el display de acuerdo a la desaceleración y hasta la frecuencia de régimen o 0Hz.*
- uint16\_t **engine\_start** ()
 

*Función que setea la frecuencia de régimen de acuerdo a lo configurado por el usuario y ejecuta la tarea accelerating.*
- uint16\_t **engine\_stop** ()
 

*Función que setea la frecuencia de régimen de acuerdo a lo configurado por el usuario y ejecuta la tarea desaccelerating.*
- bool **engine\_emergency\_stop\_release** (uint8\_t signal)
 

*Pasa a estado SYSTEM\_EMERGENCY\_OK si todas las señales de emergencia son 0 para que el usuario pueda enviar la señal de stop al STM32 necesaria para poder arrancar nuevamente el motor.*
- void **engine\_emergency\_stop** (uint8\_t signal)
 

*Pasa la frecuencia de régimen a 0Hz y pasa a estado SYSTEM\_EMERGENCY.*
- uint16\_t **change\_frequency** (uint8\_t speed\_selector)
 

*Cambia la frecuencias de régimen del motor de acuerdo a la entrada aislada que haya ingresado. Ejecuta accelerating o desaccelerating, de acuerdo a lo necesario.*
- esp\_err\_t **get\_status** (system\_status\_t \*s\_e)
 

*Obtiene una réplica del status del sistema.*
- system\_status\_t **update\_meas** (uint16\_t vbus\_meas, uint16\_t ibus\_meas)
 

*Actualiza las mediciones de tensión y corriente del bus de continua.*
- void **set\_frequency\_table** (uint16\_t input\_variable, uint16\_t freq\_regime)
 

*Carga en el vector de velocidades configurables por entradas, las diferentes frecuencias de trabajo.*
- void **set\_system\_settings** (frequency\_settings\_t \*f\_s, security\_settings\_t \*s\_s)
 

*Actualiza las variables de seguridad del sistema.*

## Variables

- static const char \* TAG = "sysAdmin"
- static `system_status_t` system\_status
- static `seccurity_settings_t` system\_seccurity\_settings
- static `uint16_t` frequency\_table [8]
- static `TaskHandle_t` accelerating\_handle = NULL
- static `TaskHandle_t` desaccelerating\_handle = NULL

### 10.75.1. 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](#).

### 10.75.2. Documentación de funciones

#### 10.75.2.1. 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 régimen.

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\_ACCEL\_DESACCEL a SYSTEM\_REGIME

#### Parámetros

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

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

#### 10.75.2.2. change\_frequency()

```
uint16_t change_frequency (
    uint8_t speed_selector)
```

Cambia la frecuencias de régimen del motor de acuerdo a la entrada aislada que haya ingresado. Ejecuta accelerating o desaccelerating, de acuerdo a lo necesario.

De acuerdo a la frecuencia de régimen elegida y el tipo de variación, la frecuencia de destino tendrá diferentes valores, siempre menores a las de régimen.

#### Parámetros

in	<i>speed_selector</i>	Índice dentro del array con frecuencias de trabajo
----	-----------------------	--

**Devuelve**

0Hz si *speed\_selector* fue mal pasada como argumento, sino la frecuencia de destino

Definición en la línea 151 del archivo [sysAdmin.c](#).

**10.75.2.3. 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](#).

**10.75.2.4. engine\_emergency\_stop()**

```
void engine_emergency_stop (
    uint8_t signal)
```

Pasa la frecuencia de regimen a 0Hz y pasa a estado SYSTEM\_EMERGENCY.

Termina las tareas accelerating y desaccelerating y guarda la señal que dispara la emergencia para contabilizar todos las señales vigentes

**Parámetros**

in	<i>signal</i>	Señal que genera el estado de emergencia
----	---------------	--

Definición en la línea 136 del archivo [sysAdmin.c](#).

**10.75.2.5. engine\_emergency\_stop\_release()**

```
bool engine_emergency_stop_release (
    uint8_t signal)
```

Pasa a estado SYSTEM\_EMERGENCY\_OK si todas las señales de emergencia son 0 para que el usuario pueda enviar la señal de stop al STM32 necesaria para poder arrancar nuevamente el motor.

**Parámetros**

in	signal	Señal que normaliza el estado de emergencia
----	--------	---

### Valores devueltos

true	si cambia a SYSTEM_EMERGENCY_OK; false si existe alguna señal aún activa
------	--

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

#### 10.75.2.6. 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](#).

#### 10.75.2.7. 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](#).

#### 10.75.2.8. get\_status()

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

Obtiene una réplica del status del sistema.

#### Parámetros

out	$s \leftarrow$	Puntero a la variable donde se devolverá la información de status
	$_e$	

**Devuelve**

- ESP\_ERR\_NOT\_ALLOWED: En caso de que *s\_e* sea un puntero a NULL
- ESP\_OK: Si la copia fue exitosa

Definición en la línea 185 del archivo [sysAdmin.c](#).

**10.75.2.9. 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 229 del archivo [sysAdmin.c](#).

**10.75.2.10. 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	$f \leftarrow$	Puntero a la estructura con las variables de frecuencia a actualizar
in	$s \leftarrow$	Puntero a la estructura con las variables de seguridad a actualizar

Definición en la línea 251 del archivo [sysAdmin.c](#).

### 10.75.2.11. 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"> <li>Tensión de bus de continua leído por el ADC</li> </ul>
in	<i>ibus_meas</i>	<ul style="list-style-type: none"> <li>Corriente de bus de continua leído por el ADC</li> </ul>

**Devuelve**

- SYSTEM\_IDLE: El sistema se encuentra en estado de reposo
- SYSTEM\_ACCEL\_DESACCEL: El sistema se encuentra acelerando o desacelerando
- SYSTEM\_REGIME: El sistema está con el motor girando a régimen
- SYSTEM\_BREAKING: El sistema está frenando
- SYSTEM\_EMERGENCY: El sistema entra por primera vez en estado de emergencia

Definición en la línea [197](#) del archivo [sysAdmin.c](#).

### 10.75.3. Documentación de variables

#### 10.75.3.1. accelerating\_handle

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

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

#### 10.75.3.2. desaccelerating\_handle

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

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

#### 10.75.3.3. frequency\_table

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

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

#### 10.75.3.4. system\_security\_settings

```
seccurity_settings_t system_seccurity_settings [static]
```

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

### 10.75.3.5. system\_status

```
system_status_t system_status [static]
```

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

### 10.75.3.6. TAG

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

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

## 10.76. sysAdmin.c

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

```
00001
00009 #include "esp_log.h"
00010 #include "esp_err.h"
00011
00012 #include "freertos/FreeRTOS.h"
00013 #include "freertos/task.h"
00014 #include "freertos/queue.h"
00015
00016 #include "esp_system.h"
00017
00018 #include "SysAdmin.h"
00019 #include "SysControl.h"
00020 #include "../display/display.h"
00021
00022 static const char *TAG = "sysAdmin";
00023 static system_status_t system_status;
00024 static security_settings_t system_seccurity_settings;
00025 static uint16_t frequency_table[8];
00026 static TaskHandle_t accelerating_handle = NULL;
00027 static TaskHandle_t desaccelerating_handle = NULL;
00028
00029 static void accelerating(void *pvParameters );
00040
00051 static void desaccelerating( void *pvParameters );
00052
00053 static void accelerating(void *pvParameters ) {
00054     // uint16_t acceleration = *((uint16_t*)pvParameters);
00055     vTaskDelay(pdMS_TO_TICKS(200));
00056     while( system_status.frequency != system_status.frequency_destiny ) {
00057         if ( (system_status.frequency + system_status.acceleration) > system_status.frequency_destiny
00058             ) {
00059             system_status.frequency = system_status.frequency_destiny;
00060         } else {
00061             system_status.frequency += system_status.acceleration;
00062         }
00063         vTaskDelay(pdMS_TO_TICKS(1000));
00064     }
00065     system_status.status = SYSTEM_REGIME;
00066     accelerating_handle = NULL;
00067     vTaskDelete(NULL);
00068
00069 static void desaccelerating( void *pvParameters ) {
00070     // uint16_t desacceleration = *((uint16_t*)pvParameters);
00071     vTaskDelay(pdMS_TO_TICKS(200));
00072     while( system_status.frequency != system_status.frequency_destiny ) {
00073         if ( system_status.frequency - system_status.desacceleration < system_status.frequency_destiny
00074             ) {
00075             system_status.frequency = system_status.frequency_destiny;
00076         } else if ( system_status.frequency > system_status.desacceleration ) {
00077             system_status.frequency -= system_status.desacceleration;
00078         } else {
00079             system_status.frequency = 0;
00080         }
00081         vTaskDelay(pdMS_TO_TICKS(1000));
00082     }
00082     if ( system_status.status != SYSTEM_EMERGENCY ) {
```

```

00083     if( system_status.status != SYSTEM_BREAKING ) {
00084         system_status.status = SYSTEM_REGIME;
00085     } else {
00086         system_status.status = SYSTEM_IDLE;
00087     }
00088 }
00089 desaccelerating_handle = NULL;
00090 vTaskDelete(NULL);
00091 }
00092
00093 uint16_t engine_start() {
00094     if ( system_status.status != SYSTEM_EMERGENCY ) {
00095         system_status.frequency_destiny = frequency_table[system_status.inputs_status];
00096         system_status.status = SYSTEM_ACCEL_DESACCEL;
00097         if ( desaccelerating_handle != NULL ) {
00098             vTaskDelete( desaccelerating_handle );
00099             desaccelerating_handle = NULL;
00100         }
00101         xTaskCreatePinnedToCore(accelerating, "accelerating", 1024, (void*)
00102             &system_status.acceleration, 9, &accelerating_handle, 1);
00103     }
00104     return system_status.frequency_destiny;
00105 }
00106 uint16_t engine_stop() {
00107     if ( system_status.status == SYSTEM_EMERGENCY ) {
00108     } else if ( system_status.status == SYSTEM_EMERGENCY_OK ) {
00109         system_status.status = SYSTEM_IDLE;
00110     } else {
00111         system_status.acceleration = get_system_acceleration();
00112         system_status.desacceleration = get_system_desacceleration();
00113         system_status.frequency_destiny = 0;
00114         if ( accelerating_handle != NULL ) {
00115             vTaskDelete( accelerating_handle );
00116             accelerating_handle = NULL;
00117         }
00118         system_status.status = SYSTEM_BREAKING;
00119         xTaskCreatePinnedToCore(desaccelerating, "desaccelerating", 1024, (void*)
00120             &system_status.desacceleration, 9, &desaccelerating_handle, 1);
00121     }
00122     return system_status.frequency_destiny;
00123 }
00124 bool engine_emergency_stop_release(uint8_t signal) {
00125     bool retval = false;
00126     system_status.emergency_signals &= ~signal;
00127     if ( system_status.emergency_signals == 0 ) {
00128         system_status.status = SYSTEM_EMERGENCY_OK;
00129         retval = true;
00130     } else {
00131         ESP_LOGI(TAG, "Estado de las señales de emergencia:%d", system_status.emergency_signals );
00132     }
00133     return retval;
00134 }
00135
00136 void engine_emergency_stop(uint8_t signal) {
00137     if ( accelerating_handle != NULL ) {
00138         vTaskDelete( accelerating_handle );
00139         accelerating_handle = NULL;
00140     }
00141     if ( desaccelerating_handle != NULL ) {
00142         vTaskDelete( desaccelerating_handle );
00143         desaccelerating_handle = NULL;
00144     }
00145     system_status.frequency_destiny = 0;
00146     system_status.frequency = 0;
00147     system_status.status = SYSTEM_EMERGENCY;
00148     system_status.emergency_signals |= signal;
00149 }
00150
00151 uint16_t change_frequency(uint8_t speed_slector) {
00152     if (speed_slector > 8 ) {
00153         ESP_LOGE(TAG, "El numero del indice del selector de velocidad es muy grande%d",
00154             speed_slector);
00155         return 0;
00156     }
00157     system_status.inputs_status = speed_slector;
00158     if ( system_status.frequency > frequency_table[system_status.inputs_status] ) {
00159         if ( accelerating_handle != NULL ) {
00160             vTaskDelete( accelerating_handle );
00161             accelerating_handle = NULL;
00162         }
00163         uint16_t desacceleration = get_system_desacceleration();
00164         system_status.frequency_destiny = frequency_table[system_status.inputs_status];
00165         if ( desaccelerating_handle == NULL ) {
00166             xTaskCreatePinnedToCore(desaccelerating, "desaccelerating", 1024, (void*)

```

```

        &desacceleration, 9, &desaccelerating_handle, 1);
00167    }
00168    ESP_LOGI( TAG, "Cambiando la frecuencia de regimen a %d. Desacelerando",
00169    system_status.frequency_destiny);
00170    } else if ( system_status.frequency < frequency_table[system_status.inputs_status] ) {
00171        if ( desaccelerating_handle != NULL ) {
00172            vTaskDelete( desaccelerating_handle );
00173            desaccelerating_handle = NULL;
00174        }
00175        uint16_t acceleration = get_system_acceleration();
00176        system_status.frequency_destiny = frequency_table[system_status.inputs_status];
00177        if ( accelerating_handle == NULL ) {
00178            xTaskCreatePinnedToCore(accelerating, "accelerating", 1024, (void*) &(acceleration), 9,
00179            &accelerating_handle, 1);
00180        }
00181        ESP_LOGI( TAG, "Cambiando la frecuencia de regimen a %d. Acelerando",
00182        system_status.frequency_destiny);
00183    }
00184
00185 esp_err_t get_status(system_status_t *s_e) {
00186    if (s_e == NULL)
00187        return ESP_ERR_NOT_ALLOWED;
00188    s_e->frequency = system_status.frequency;
00189    s_e->frequency_destiny = system_status.frequency_destiny;
00190    s_e->vbus_min = system_status.vbus_min;
00191    s_e->ibus_max = system_status.ibus_max;
00192    s_e->inputs_status = system_status.inputs_status;
00193    s_e->status = system_status.status;
00194    return ESP_OK;
00195 }
00196
00197 system_status_t update_meas(uint16_t vbus_meas, uint16_t ibus_meas) {
00198    bool in_emergency = false;
00199    system_status.vbus_min = vbus_meas;
00200    system_status.ibus_max = ibus_meas;
00201
00202    if ( system_status.ibus_max > system_security_settings.ibus_max ) {
00203        in_emergency = true;
00204        if ( system_status.status != SYSTEM_EMERGENCY ) {
00205            ESP_LOGE( TAG, "Disparo de emergencia por sobre corriente.");
00206        }
00207    }
00208
00209    if ( system_status.vbus_min < system_security_settings.vbus_min ) {
00210        in_emergency = true;
00211        if ( system_status.status != SYSTEM_EMERGENCY ) {
00212            ESP_LOGE( TAG, "Disparo de emergencia por baja tensión.");
00213        }
00214    }
00215
00216    if ( in_emergency == true ) {
00217        if ( system_status.status != SYSTEM_EMERGENCY ) {
00218            SystemEventPost(SECURITY_EXCEEDED);
00219            ESP_LOGI( TAG, "Mando senal.");
00220        }
00221    } else if ( system_status.status == SYSTEM_EMERGENCY && ( system_status.emergency_signals & 0b010
00222 ) ) {
00223        SystemEventPost(SECURITY_OK);
00224        ESP_LOGI( TAG, "Salgo de emergencia por seguridad.");
00225    }
00226
00227    return system_status.status;
00228 }
00229 void set_frequency_table( uint16_t input_variable, uint16_t freq_regime ) {
00230    if ( input_variable == 1 ) { //lineal
00231        uint16_t frequency_gap = freq_regime / 8;
00232        frequency_table[7] = frequency_gap;
00233        for (uint8_t i = 6; i > 0; i-- ) {
00234            frequency_table[i] = frequency_table[i + 1] + frequency_gap;
00235            ESP_LOGI(TAG, "frequency_table[%d] = %d", i, frequency_table[i]);
00236        }
00237        frequency_table[0] = freq_regime;
00238        ESP_LOGI(TAG, "frequency_table[0] = %d", frequency_table[0]);
00239    } else if ( input_variable == 2 ) { //Cuadratica
00240        for (uint8_t i = 0; i < 8; i++ ) {
00241            uint32_t num = (uint32_t)freq_regime * (uint32_t)(i + 1) * (uint32_t)(i + 1);
00242            uint16_t fi = (uint16_t)((num + 32) / 64);
00243            if (fi == 0)
00244                fi = 1;
00245            frequency_table[7 - i] = fi;
00246            ESP_LOGI(TAG, "frequency_table[%d] = %d", 7 - i, frequency_table[7 - i]);
00247        }
00248    }

```

```

00249 }
00250
00251 void set_system_settings( frequency_settings_t *f_s, security_settings_t *s_s ) {
00252     system_status.acceleration = f_s->acceleration;
00253     system_status.desacceleration = f_s->desacceleration;
00254     system_seccurity_settings.vbus_min = s_s->vbus_min;
00255     system_seccurity_settings.ibus_max = s_s->ibus_max;
00256     set_frequency_table( f_s->input_variable, f_s->freq_regime );
00257 }
```

## 10.77. Referencia del archivo

**C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDe←  
Frecuencia/Software/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.*
- **bool engine\_emergency\_stop\_release (uint8\_t signal)**  
*Pasa a estado SYSTEM\_EMERGENCY\_OK si todas las señales de emergencia son 0 para que el usuario pueda enviar la señal de stop al STM32 necesaria para poder arrancar nuevamente el motor.*
- **void engine\_emergency\_stop (uint8\_t signal)**  
*Pasa la frecuencia de regimen a 0Hz y pasa a estado SYSTEM\_EMERGENCY.*
- **uint16\_t change\_frequency (uint8\_t speed\_slector)**  
*Cambia la frecuencias de régimen del motor de acuerdo a la entrada aislada que haya ingresado. Ejecuta accelerating o desaccelerating, de acuerdo a lo necesario.*
- **system\_status\_e update\_meas (uint16\_t vbus\_meas, uint16\_t ibus\_meas)**  
*Actualiza las mediciones de tensión y corriente del bus de continua.*
- **void set\_frequency\_table (uint16\_t input\_variable, uint16\_t freq\_regime)**  
*Carga en el vector de velocidades configurables por entradas, las diferentes frecuencias de trabajo.*
- **void set\_system\_settings (frequency\_settings\_t \*f\_s, security\_settings\_t \*s\_s)**  
*Actualiza las variables de seguridad del sistema.*

### 10.77.1. 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](#).

## 10.77.2. Documentación de funciones

### 10.77.2.1. change\_frequency()

```
uint16_t change_frequency (
    uint8_t speed_selector)
```

Cambia la frecuencias de régimen del motor de acuerdo a la entrada aislada que haya ingresado. Ejecuta accelerating o desaccelerating, de acuerdo a lo necesario.

De acuerdo a la frecuencia de régimen elegida y el tipo de variación, la frecuencia de destino tendrá diferentes valores, siempre menores a las de régimen.

#### Parámetros

in	<i>speed_selector</i>	Índice dentro del array con frecuencias de trabajo
----	-----------------------	--

#### Devuelve

0Hz si *speed\_selector* fue mal pasada como argumento, sino la frecuencia de destino

Definición en la línea 151 del archivo [sysAdmin.c](#).

### 10.77.2.2. engine\_emergency\_stop()

```
void engine_emergency_stop (
    uint8_t signal)
```

Pasa la frecuencia de regimen a 0Hz y pasa a estado SYSTEM\_EMERGENCY.

Termina las tareas accelerating y desaccelerating y guarda la señal que dispara la emergencia para contabilizar todos las señales vigentes

#### Parámetros

in	<i>signal</i>	Señal que genera el estado de emergencia
----	---------------	--

Definición en la línea 136 del archivo [sysAdmin.c](#).

### 10.77.2.3. engine\_emergency\_stop\_release()

```
bool engine_emergency_stop_release (
    uint8_t signal)
```

Pasa a estado SYSTEM\_EMERGENCY\_OK si todas las señales de emergencia son 0 para que el usuario pueda enviar la señal de stop al STM32 necesaria para poder arrancar nuevamente el motor.

#### Parámetros

in	<i>signal</i>	Señal que normaliza el estado de emergencia
----	---------------	---

**Valores devueltos**

true	si cambia a SYSTEM_EMERGENCY_OK; false si existe alguna señal aún activa
------	--

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

**10.77.2.4. 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](#).

**10.77.2.5. 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](#).

**10.77.2.6. get\_status()**

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

Obtiene una réplica del status del sistema.

**Parámetros**

out	$s \leftarrow$ $_e$	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 185 del archivo [sysAdmin.c](#).

**10.77.2.7. set\_frequency\_table()**

```
void set_frequency_table (
    uint16_t input_variable,
    uint16_t freq_regime)
```

Carga en el vector de velocidades configurables por entradas, las diferentes frecuencias de trabajo.

La posición 0 del buffer es la frecuencia de regimen, la posición 7 es la frecuencia más baja distinta de cero

**Parámetros**

in	<i>input_variable</i>	Tipo de variación entre las diferentes entradas. 1 para variación lineal; 2 para variación cuadrática.
in	<i>freq_regime</i>	Frecuencia de regimen ingresada por el usuario

Definición en la línea 229 del archivo [sysAdmin.c](#).

**10.77.2.8. set\_system\_settings()**

```
void set_system_settings (
    frequency_settings_t * f_s,
    security_settings_t * s_s)
```

Actualiza las variables de seguridad del sistema.

**Parámetros**

in	$f \leftarrow$ $_s$	Puntero a la estructura con las variables de frecuencia a actualizar
in	$s \leftarrow$ $_s$	Puntero a la estructura con las variables de seguridad a actualizar

Definición en la línea 251 del archivo [sysAdmin.c](#).

### 10.77.2.9. 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"> <li>Tensión de bus de continua leído por el ADC</li> </ul>
in	<i>ibus_meas</i>	<ul style="list-style-type: none"> <li>Corriente de bus de continua leído por el ADC</li> </ul>

**Devuelve**

- SYSTEM\_IDLE: El sistema se encuentra en estado de reposo
- SYSTEM\_ACCEL\_DESACCEL: El sistema se encuentra acelerando o desacelerando
- SYSTEM\_REGIME: El sistema está con el motor girando a régimen
- SYSTEM\_BREAKING: El sistema está frenando
- SYSTEM\_EMERGENCY: El sistema entra por primera vez en estado de emergencia

Definición en la línea 197 del archivo [sysAdmin.c](#).

## 10.78. sysAdmin.h

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

```

00001
00009 #ifndef __SYS_ADMIN_H__
00010
00011 #define __SYS_ADMIN_H__
00012
00013 #include "../LVFV_system.h"
00014
00027 esp_err_t get_status(system_status_t *s_e);
00028
00038 uint16_t engine_start();
00039
00049 uint16_t engine_stop();
00050
00060 bool engine_emergency_stop_release(uint8_t signal);
00061
00071 void engine_emergency_stop(uint8_t signal);
00072
00085 uint16_t change_frequency(uint8_t speed_selector);
00086
00107 system_status_e update_meas(uint16_t vbus_meas, uint16_t ibus_meas);
00108
00122 void set_frequency_table( uint16_t input_variable, uint16_t freq_regime );
00123
00135 void set_system_settings( frequency_settings_t *f_s, security_settings_t *s_s );
00136
00137 #endif

```

## 10.79. Referencia del archivo

[C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\\_ESP32/main/system/sysControl.c](C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/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/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "driver/spi_master.h"
#include "driver/gpio.h"
#include "./io_control/io_control.h"
#include "../LVFV_system.h"
#include "./display/display.h"
#include "./adc/adc.h"
#include "./SysAdmin.h"
#include "./SysControl.h"
```

## defines

- #define PIN\_NUM\_CS 12 /\*\* @def PIN\_NUM\_CS @brief Chip select del SPI para comunicación con el STM32. IO12 \*/
- #define PIN\_NUM\_CLK 14 /\*\* @def PIN\_NUM\_CLK @brief Clock del SPI para comunicación con el STM32. IO14 \*/
- #define PIN\_NUM\_MISO 26 /\*\* @def PIN\_NUM\_MISO @brief master input slave output del SPI para comunicación con el STM32. IO26 \*/
- #define PIN\_NUM\_MOSI 25 /\*\* @def PIN\_NUM\_MOSI @brief master output slave input del SPI para comunicación con el STM32. IO25 \*/
- #define SPI\_HOST\_USED SPI2\_HOST /\*\* @def SPI\_HOST\_USED @brief don't know \*/
- #define SPI\_CLOCK\_HZ 1\*1000\*1000 /\*\* @def SPI\_CLOCK\_HZ @brief Velocidad de clock: 1 MHz \*/
- #define SPI\_QUEUE\_TX\_DEPTH 1 /\*\* @def SPI\_QUEUE\_TX\_DEPTH @brief Profundidad de comandos para el puerto SPI \*/

## Funciones

- static SPI\_Response SPI\_SendRequest (spi\_cmd\_item\_t \*spi\_cmd\_item)
 

*Envía el comando configurado en spi\_cmd\_item->request con el argumento spi\_cmd\_item->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

## 10.79.1. 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](#).

## 10.79.2. Documentación de «define»

### 10.79.2.1. 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](#).

### 10.79.2.2. 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](#).

### 10.79.2.3. 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](#).

### 10.79.2.4. 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](#).

### 10.79.2.5. 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](#).

### 10.79.2.6. 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](#).

### 10.79.2.7. 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](#).

## 10.79.3. Documentación de funciones

### 10.79.3.1. 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](#).

### 10.79.3.2. SPI\_Init()

```
esp_err_t SPI_Init (
    void )
```

Inicializa el módulo SPI del ESP32.

Inicializa el módulo.

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](#).

### 10.79.3.3. SPI\_SendRequest()

```
SPI_Response SPI_SendRequest (
    spi_cmd_item_t * spi_cmd_item) [static]
```

Envía el comando configurado en `spi_cmd_item->request` con el argumento `spi_cmd_item->setValue` en caso de ser necesario y obtiene los datos que responde el STM32 en `spi_cmd_item->getValue` si corresponde.

El ESP32 debe enviar comandos en dos instancias. La primera envía el comando deseado por el usuario, se detiene la ejecución durante 200 mili segundos y luego se envía un nuevo comando para consultarle al STM32 si el comando enviado por el usuario fue recibido correctamente o no.

#### Parámetros

<code>in, out</code>	<code>spi_cdm_item</code>	Estructura de datos con los parámetros necesarios para llevar a cabo la comunicación. Allí se almacena el comando, valores a enviar y valores recibidos.
----------------------	---------------------------	--

Devuelve

- SPI\_RESPONSE\_ERR: Error en la transmisión del comando por problemas de inicialización del handler. El comando no sale del ESP32
- SPI\_RESPONSE\_ERR\_CMD\_UNKNOWN: El comando enviado no está dentro los posibles
- SPI\_RESPONSE\_ERR\_MOVING: Se está intentando enviar un comando de start con el motor en movimiento
- SPI\_RESPONSE\_ERR\_NOT\_MOVING: Se está intentando enviar un comando de stop con el motor detenido
- SPI\_RESPONSE\_ERR\_DATA\_MISSING: Faltó enviar un dato dentro de la trama
- SPI\_RESPONSE\_ERR\_DATA\_INVALID: Los datos enviados como argumento dentro del comando están fuera de rango
- SPI\_RESPONSE\_OK: La comunicación entre ESP32 y STM32 fue exitosa

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

#### 10.79.3.4. SystemEventPost()

```
esp_err_t SystemEventPost (
    systemSignal_e event)
```

Encola comandos en la cola system\_event\_queue.

##### Parámetros

in	event	Evento que se desea encolar en el sistema
----	-------	---

##### Devuelve

- pdTRUE Si el comando se posteó correctamente
- Cualquier otra respuesta implica que la cola está llena

Definición en la línea [385](#) del archivo [sysControl.c](#).

### 10.79.4. Documentación de variables

#### 10.79.4.1. spi\_handle

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

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

#### 10.79.4.2. system\_event\_queue

```
QueueHandle_t system_event_queue = NULL
```

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

#### 10.79.4.3. TAG

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

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

## 10.80. sysControl.c

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

```

00001
00009 #include <stdio.h>
00010 #include <string.h>
00011
00012 #include "esp_log.h"
00013 #include "esp_err.h"
00014
00015 #include "freertos/FreeRTOS.h"
00016 #include "freertos/task.h"
00017 #include "freertos/queue.h"
00018
00019 #include "driver/spi_master.h"
00020 #include "driver/gpio.h"
00021
00022 #include "./io_control/io_control.h"
00023 #include "../LVFV_system.h"
00024 #include "./display/display.h"
00025 #include "./adc/adc.h"
00026 #include "./SysAdmin.h"
00027 #include "./SysControl.h"
00028
00029 // ===== Pines =====
00030 #define PIN_NUM_CS 12
00031 #define PIN_NUM_CLK 14
00032 #define PIN_NUM_MISO 26
00033 #define PIN_NUM_MOSI 25
00034
00035 // ===== SPI =====
00036 #define SPI_HOST_USED SPI2_HOST
00037 #define SPI_CLOCK_HZ 1*1000*1000
00038 #define SPI_QUEUE_TX_DEPTH 1
00039
00040 static const char *TAG = "sysControl";
00041
00042 QueueHandle_t system_event_queue = NULL;
00043
00044 static spi_device_handle_t spi_handle = NULL; // Handler del puerto SPI para
    comunicación con el STM32. Inicializado en esp_err_t SPI_Init(void)
00045
00046 static SPI_Response SPI_SendRequest(spi_cmd_item_t *spi_cmd_item);
00047
00048 static SPI_Response SPI_SendRequest(spi_cmd_item_t *spi_cmd_item) {
00049     uint8_t tx_buffer[4];
00050     uint8_t rx_buffer[4];
00051     esp_err_t ret;
00052     int flagDevolverValor; // Se necesita devolver el valor por parametro
00053
00054     ESP_LOGI(TAG, "[SPI Module] Request %d", spi_cmd_item->request);
00055
00056     // Chequeo de errores fuera de rango y comando desconocido
00057     if(spi_cmd_item->setValue > 512 || spi_cmd_item->setValue < 0) {
00058         return SPI_RESPONSE_ERR_DATA_INVALID;
00059     }
00060
00061     if(spi_cmd_item->request < SPI_REQUEST_START || spi_cmd_item->request > SPI_REQUEST_EMERGENCY) {
00062         ESP_LOGI(TAG, "[SPI Module] Comando desconocido\n");
00063         return SPI_RESPONSE_ERR_CMD_UNKNOWN;
00064     }
00065
00066     // Es de los comandos que deben devolver un valor por parametro?
00067     if(spi_cmd_item->request >= SPI_REQUEST_GET_FREC && spi_cmd_item->request <= SPI_REQUEST_IS_STOP)
00068     {
00069         flagDevolverValor = 1;
00070     }else {
00071         flagDevolverValor = 0;
00072     }
00073
00074     // Se arma la cadena a enviar
00075     tx_buffer[0] = spi_cmd_item->request;
00076
00077     if(spi_cmd_item->setValue > 255) {
00078         tx_buffer[1] = 255;
00079         tx_buffer[2] = spi_cmd_item->setValue - 255;
00080         tx_buffer[3] = ';';
00081     }else {
00082         tx_buffer[1] = spi_cmd_item->setValue;
00083         tx_buffer[2] = 0;
00084         tx_buffer[3] = ';';
00085     }
00086
00087
00088
00089
00090
00091
00092
00093
00094
00095
00096
00097
00098
00099
00100
00101
00102
00103
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
00142 ESP_LOGI(TAG, "rx_buffer[0]: %d", rx_buffer[0]);
00143 if(flagDevolverValor && rx_buffer[0] == SPI_RESPONSE_OK) {
00144
00145     spi_cmd_item->getValue = rx_buffer[1] + rx_buffer[2];
00146     ESP_LOGI(TAG, "RxValores: %d - %d", rx_buffer[1], rx_buffer[2]);
00147
00148 }else {
00149     spi_cmd_item->getValue = 0;
00150 }
00151
00152 return rx_buffer[0];
00153 }
00154
00155 esp_err_t SPI_Init(void) {
00156     spi_bus_config_t buscfg = {
00157         .misio_io_num = PIN_NUM_MISO,
00158         .mosi_io_num = PIN_NUM_MOSI,
00159         .sclk_io_num = PIN_NUM_CLK,
00160         .quadwp_io_num = -1,
00161         .quadhd_io_num = -1,
00162         .max_transfer_sz = 32
00163     };
00164
00165     spi_device_interface_config_t devcfg = {
00166         .clock_speed_hz = SPI_CLOCK_HZ,
00167         .mode = 0,                                // SPI mode 0
00168         .spics_io_num = PIN_NUM_CS,                // CS pin
00169         .queue_size = 1,                            // Sin dummy bits
00170         .flags = SPI_DEVICE_NO_DUMMY
00171     };
00172
00173 // Inicializar bus SPI
00174 esp_err_t ret = spi_bus_initialize(VSPI_HOST, &buscfg, 1);
00175 if (ret != ESP_OK) {
00176     return ret;
00177 }
00178 ESP_ERROR_CHECK(ret);
00179
00180 // Añadir dispositivo
00181 ret = spi_bus_add_device(VSPI_HOST, &devcfg, &spi_handle);
00182 ESP_ERROR_CHECK(ret);
00183 return ESP_OK;
00184 }
00185
00186 void SPI_communication(void *arg) {
00187     spi_cmd_item_t item;
00188
00189     systemSignal_e new_button;
00190
00191     if (SPI_Init() != ESP_OK) {
00192         ESP_LOGE(TAG, "SPI_Init falló; no creo SPI_communication");
00193         ESP_ERROR_CHECK(ESP_FAIL);
00194     }

```

```

00194     }
00195
00196     if (system_event_queue == NULL) {
00197         system_event_queue = xQueueCreate(1, sizeof(systemSignal_e));
00198         if (system_event_queue == NULL) {
00199             ESP_LOGE(TAG, "No se pudo crear la cola de interrupciones");
00200             ESP_ERROR_CHECK(ESP_FAIL);
00201         }
00202     }
00203
00204     vTaskDelay(pdMS_TO_TICKS(4000));
00205
00206     do {
00207         item.request = SPI_REQUEST_STOP;
00208         item.setValue = 0;
00209         item.getValue = 0;
00210         vTaskDelay(pdMS_TO_TICKS(100));
00211     } while ( SPI_SendRequest(&item) != SPI_RESPONSE_ERR_NOT_MOVING );
00212
00213     for ( uint32_t i = 0;; i++ ) {
00214         if ( readADC() ) {
00215             while( xQueueReceive( system_event_queue, &new_button, pdMS_TO_TICKS(0) ) );
00216             SystemEventPost(STOP_PRESSED);
00217             break;
00218         }
00219         vTaskDelay(pdMS_TO_TICKS(100));
00220     }
00221
00222     SystemEventPost(STOP_PRESSED);
00223
00224     ESP_LOGI(TAG, "SPI_communication lista. Esperando comandos...");
00225
00226     while (1) {
00227         system_status_t s_e;
00228         get_status( &s_e );
00229         readADC();
00230         if ( xQueueReceive( system_event_queue, &new_button, pdMS_TO_TICKS(20) ) ) {
00231             switch ( new_button ) {
00232                 case EMERGENCI_STOP_PRESSED:
00233                     if ( s_e.status != SYSTEM_EMERGENCY ) {
00234                         ESP_LOGI(TAG, "Botón de EMERGENCIA presionado");
00235                         item.request = SPI_REQUEST_EMERGENCY;
00236                         item.setValue = 0;
00237                         item.getValue = 0;
00238                         SPI_SendRequest(&item);
00239                         RelayEventPost( 1 );
00240                     } else {
00241                         ESP_LOGI(TAG, "El sistema ya se encontraba en emergencia");
00242                     }
00243                     engine_emergency_stop(0b001);
00244                     break;
00245                 case SECURITY_EXCEEDED:
00246                     if ( s_e.status != SYSTEM_EMERGENCY ) {
00247                         ESP_LOGI(TAG, "Corriente elevada o tensión reducida");
00248                         item.request = SPI_REQUEST_EMERGENCY;
00249                         item.setValue = 0;
00250                         item.getValue = 0;
00251                         SPI_SendRequest(&item);
00252                         RelayEventPost( 1 );
00253                     } else {
00254                         ESP_LOGI(TAG, "El sistema ya se encontraba en emergencia");
00255                     }
00256                     engine_emergency_stop(0b010);
00257                     break;
00258                 case TERMO_SW_PRESSED:
00259                     if ( s_e.status != SYSTEM_EMERGENCY ) {
00260                         ESP_LOGI(TAG, "Termoswitch activo");
00261                         item.request = SPI_REQUEST_EMERGENCY;
00262                         item.setValue = 0;
00263                         item.getValue = 0;
00264                         SPI_SendRequest(&item);
00265                         RelayEventPost( 1 );
00266                     } else {
00267                         ESP_LOGI(TAG, "El sistema ya se encontraba en emergencia");
00268                     }
00269                     engine_emergency_stop(0b100);
00270                     break;
00271                 case EMERGENCI_STOP_RELEASED:
00272                     ESP_LOGI(TAG, "Botón de EMERGENCIA liberado");
00273                     engine_emergency_stop_release(0b001);
00274                     break;
00275                 case SECURITY_OK:
00276                     ESP_LOGI(TAG, "Tensión y corriente normalizadas");
00277                     engine_emergency_stop_release(0b010);
00278                     break;
00279                 case TERMO_SW_RELEASED:
00280                     ESP_LOGI(TAG, "Termoswitch desactivado");
00281             }
00282         }
00283     }
00284 }
```

```

00281         engine_emergency_stop_release(0b100);
00282         break;
00283     case STOP_PRESSED:
00284         if ( s_e.status == SYSTEM_EMERGENCY ) {
00285             ESP_LOGI(TAG, "Primero salir de estado de emergencia");
00286         } else if ( s_e.status == SYSTEM_ACCEL_DESACCEL || s_e.status == SYSTEM_REGIME ) {
00287             engine_stop();
00288             ESP_LOGI(TAG, "Botón de Parada presionado");
00289             item.request = SPI_REQUEST_STOP;
00290             item.setValue = 0;
00291             item.getValue = 0;
00292             SPI_SendRequest(&item);
00293         } else if ( s_e.status == SYSTEM_EMERGENCY_OK ) {
00294             engine_stop();
00295             ESP_LOGI(TAG, "Botón de Parada presionado - Liberando estado de emergencia");
00296             item.request = SPI_REQUEST_STOP;
00297             item.setValue = 0;
00298             item.getValue = 0;
00299             SPI_SendRequest(&item);
00300             RelayEventPost( 0 );
00301         }
00302         break;
00303     case START_PRESSED:
00304         if ( s_e.status == SYSTEM_IDLE ) {
00305             ESP_LOGI(TAG, "Botón de Inicio presionado");
00306             SPI_Commando_response;
00307
00308             item.request = SPI_REQUEST_SET_FREQ;
00309             item.setValue = get_system_frequency();
00310             item.getValue = 0;
00311             SPI_Commando_response = SPI_SendRequest(&item);
00312             if ( SPI_Commando_response != SPI_RESPONSE_OK ) {
00313                 ESP_LOGE(TAG, "Error cargando la frecuencia");
00314                 break;
00315             }
00316
00317             item.request = SPI_REQUEST_SET_ACCEL;
00318             item.setValue = get_system_acceleration();
00319             item.getValue = 0;
00320             SPI_Commando_response = SPI_SendRequest(&item);
00321             if ( SPI_Commando_response != SPI_RESPONSE_OK ) {
00322                 ESP_LOGE(TAG, "Error cargando la aceleracion");
00323                 break;
00324             }
00325
00326             item.request = SPI_REQUEST_SET_DESACEL;
00327             item.setValue = get_system_desacceleration();
00328             item.getValue = 0;
00329             SPI_Commando_response = SPI_SendRequest(&item);
00330             if ( SPI_Commando_response != SPI_RESPONSE_OK ) {
00331                 ESP_LOGE(TAG, "Error cargando la desaceleracion");
00332                 break;
00333             }
00334
00335             item.request = SPI_REQUEST_START;
00336             item.setValue = 0;
00337             item.getValue = 0;
00338             SPI_Commando_response = SPI_SendRequest(&item);
00339             if ( SPI_Commando_response != SPI_RESPONSE_OK ) {
00340                 ESP_LOGE(TAG, "Error arrancando el motor");
00341                 break;
00342             }
00343             uint16_t dest = engine_start();
00344             ESP_LOGI(TAG, "Motor arrancado. Frecuencia destino:%d", dest);
00345         } else {
00346             ESP_LOGE(TAG, "El motor debe estar detenido para poder arrancarlo");
00347         }
00348         break;
00349     case START_RELEASED:
00350         ESP_LOGI(TAG, "Botón de Inicio liberado");
00351         break;
00352     case SPEED_SELECTOR_0:
00353     case SPEED_SELECTOR_1:
00354     case SPEED_SELECTOR_2:
00355     case SPEED_SELECTOR_3:
00356     case SPEED_SELECTOR_4:
00357     case SPEED_SELECTOR_5:
00358     case SPEED_SELECTOR_6:
00359     case SPEED_SELECTOR_7:
00360     case SPEED_SELECTOR_8:
00361     case SPEED_SELECTOR_9:
00362
00363         if ( s_e.status == SYSTEM_REGIME || s_e.status == SYSTEM_ACCEL_DESACEL ) {
00364             ESP_LOGI(TAG, "Cambio de velocidad:%d", new_button - SPEED_SELECTOR_0);
00365             uint8_t old_input_status = s_e.inputs_status;
00366             item.request = SPI_REQUEST_SET_FREQ;
00367             item.setValue = change_frequency( new_button - SPEED_SELECTOR_0 );

```

```

00368         item.getValue = 0;
00369         if ( SPI_SendRequest(&item) != SPI_RESPONSE_OK ) {
00370             change_frequency( old_input_status );
00371             ESP_LOGE(TAG, "El STM32 no respondio correctamente");
00372             xQueueSend(system_event_queue, &new_button, pdMS_TO_TICKS(1000));
00373         }
00374     } else {
00375         ESP_LOGI(TAG, "No puede cambiar la velocidad, status =%d", s_e.status);
00376     }
00377     break;
00378 default:
00379     break;
00380 }
00381 }
00382 }
00383 }
00384
00385 esp_err_t SystemEventPost(systemSignal_e event) {
00386     if (system_event_queue == NULL) {
00387         return ESP_FAIL;
00388     }
00389     return xQueueSend(system_event_queue, &event, 0);
00390 }
```

## 10.81. Referencia del archivo

### C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/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_FREQ`, `SPI_REQUEST_SET_ACCEL`,
 , `SPI_REQUEST_SET_DEACCEL`, `SPI_REQUEST_SET_DIR`, `SPI_REQUEST_GET_FREQ`, `SPI_REQUEST_GET_ACCEL`,
 , `SPI_REQUEST_GET_DEACCEL`, `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.*

### 10.81.1. 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](#).

### 10.81.2. Documentación de enumeraciones

#### 10.81.2.1. 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](#).

### 10.81.2.2. 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](#).

### 10.81.3. Documentación de funciones

#### 10.81.3.1. 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](#).

### 10.81.3.2. 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](#).

### 10.81.3.3. SystemEventPost()

```
esp_err_t SystemEventPost (
    systemSignal_e event)
```

Encola comandos en la cola system\_event\_queue.

Parámetros

in	<code>event</code>	Evento que se desea encolar en el sistema
----	--------------------	---

Devuelve

- pdTRUE Si el comando se posteó correctamente
- Cualquier otra respuesta implica que la cola está llena

Definición en la línea 385 del archivo [sysControl.c](#).

## 10.82. 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

```

## 10.83. Referencia del archivo

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDe←  
Frecuencia/Software/LVFV\_ESP32/main/wifi/form.c

### Variables

- const char \* [HTML\\_FORM](#)

## 10.83.1. Documentación de variables

### 10.83.1.1. HTML\_FORM

```
const char* HTML_FORM
```

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

## 10.84. form.c

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

```
00001 const char *HTML_FORM =
00002 "<!DOCTYPE html>"
00003 "<html lang=\"es\">"
00004 "<head>"
00005 "    <meta charset=\"UTF-8\">"
00006 "    <meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">"
00007 "    <title>Configuración del Variador</title>"
00008 "    <style>"
00009 "        body{font-family:system-ui,-apple-system,Segoe
UI,Roboto,sans-serif;background:#f8f9fa;margin:0;}"
00010 "        .container{max-width:960px;margin:0 auto;padding:1rem;}"
00011 "        h1{font-size:1.8rem;margin-bottom:1.5rem;}"
00012 "        .row{display:flex;flex-wrap:wrap;margin:-0.5rem;}"
00013 "        .col-md-4{flex:0 0 100%;max-width:100%;padding:0.5rem;}"
00014 "        @media (min-width:768px){.col-md-4{flex:0 0 33.3333%;max-width:33.3333%;}}"
00015 "        .form-label{display:block;font-weight:600;margin-bottom:0.25rem;font-size:0.95rem;}"
00016 "        .form-control,.form-select{display:block;width:100%;padding:0.375rem 0.75rem;}"
00017 "            font-size:1rem;line-height:1.5;color:#212529;background-color:#fff;}"
00018 "            border:1px solid #ced4da;border-radius:0.25rem;box-sizing:border-box;}"
00019 "            .form-control:focus,.form-select:focus{outline:2px solid #0d6efd33;outline-offset:1px;}"
00020 "        .btn{display:inline-block;font-weight:400;line-height:1.5;text-align:center;}"
00021 "            text-decoration:none;border:1px solid transparent;padding:0.375rem 0.75rem;}"
00022 "            font-size:1rem;border-radius:0.25rem;cursor:pointer;margin:1.5rem;}"
00023 "            .btn-primary{color:#fff;background-color:#0d6efd;border-color:#0d6efd;width:100%;}"
00024 "            .btn-primary:hover{background-color:#0b5ed7;border-color:#0a58ca;}"
00025 "            .btn-success{color:#fff;background-color:#198754;border-color:#198754;}"
00026 "            .btn-danger{color:#fff;background-color:#dc3545;border-color:#dc3545;}"
00027 "            .btn-success:hover{background-color:#157347;border-color:#146c43;}"
00028 "            .btn-danger:hover{background-color:#bb2d3b;border-color:#b02a37;}"
00029 "            .mb-4{margin-bottom:1.5rem;}"
00030 "            .mt-4{margin-top:1.5rem;}"
00031 "            .py-4{padding-top:1.5rem;padding-bottom:1.5rem;}"
00032 "            .me-2{margin-right:0.5rem;}"
00033 "        </style>"
00034 "</head>"
00035 "<body class=\"bg-light\">"
00036 "<div class=\"container py-4\">"
00037 "    <h1 class=\"mb-4\">Configuración del Variador</h1>"
00038 "    <form id=\"config-form\" method=\"POST\" action=\"/save\" class=\"row g-3\">"
00039
00040 "        <div class=\"col-md-4\">
00041 "            <label class=\"form-label\">Frecuencia de operación (Hz)</label>
00042 "            <input type=\"number\" class=\"form-control\" name=\"frequency\" min=\"1\" max=\"150\" required>
00043 "        </div>
00044
00045 "        <div class=\"col-md-4\">
00046 "            <label class=\"form-label\">Aceleración (Hz/s)</label>
00047 "            <input type=\"number\" class=\"form-control\" name=\"accel\" min=\"1\" max=\"50\" required>
00048 "        </div>
00049
00050 "        <div class=\"col-md-4\">
00051 "            <label class=\"form-label\">Desaceleración (Hz/s)</label>
00052 "            <input type=\"number\" class=\"form-control\" name=\"decel\" min=\"1\" max=\"50\" required>
00053 "        </div>
00054
00055 "        <div class=\"col-md-4\">
00056 "            <label class=\"form-label\">Variación de las entradas</label>
00057 "            <select class=\"form-select\" name=\"variation\" required>
00058 "                <option value=\"1\">Lineal</option>
00059 "                <option value=\"2\">Cuadrática</option>
00060 "            </select>
00061 "        </div>
00062
00063 "        <div class=\"col-md-4\">
```

```
00064 "      <label class=\"form-label\">Corriente máx. bus DC (mA)</label>"  
00065 "      <input type=\"number\" class=\"form-control\" name=\"imax\" min=\"500\" max=\"2000\" required>"  
00066 "    </div>"  
00067  
00068 "    <div class=\"col-md-4\">  
00069 "      <label class=\"form-label\">Tensión máx. bus DC (V)</label>"  
00070 "      <input type=\"number\" class=\"form-control\" name=\"vmin\" min=\"250\" max=\"350\" required>"  
00071 "    </div>"  
00072  
00073 "    <div class=\"col-md-4\">  
00074 "      <label class=\"form-label\">Hora del sistema (HH:mm:ss)</label>"  
00075 "      <input type=\"text\" class=\"form-control\" name=\"sys_time\" placeholder=\"12:34:56\" required>"  
00076 "    </div>"  
00077  
00078 "    <div class=\"col-md-4\">  
00079 "      <label class=\"form-label\">Horario de arranque (HH:mm)</label>"  
00080 "      <input type=\"text\" class=\"form-control\" name=\"start_time\" placeholder=\"08:00\" required>"  
00081 "    </div>"  
00082  
00083 "    <div class=\"col-md-4\">  
00084 "      <label class=\"form-label\">Horario de parada (HH:mm)</label>"  
00085 "      <input type=\"text\" class=\"form-control\" name=\"stop_time\" placeholder=\"18:00\" required>"  
00086 "    </div>"  
00087  
00088 "    <div class=\"col-12 mt-4\">  
00089 "      <button type=\"submit\" class=\"btn btn-primary me-2\" name=\"cmd\" value=\"save\">Guardar  
configuración</button>"  
00090 "      <button type=\"submit\" class=\"btn btn-success me-2\" name=\"cmd\" value=\"start\">Arrancar  
motor</button>"  
00091 "      <button type=\"submit\" class=\"btn btn-danger\" name=\"cmd\" value=\"stop\">Parar  
motor</button>"  
00092 "    </div>"  
00093 "  </form>"  
00094 "</div>"  
00095  
00096 "<script>"  
00097 "  const formId = 'config-form';"  
00098 "  const storageKey = 'variador_form_state';"  
00099 "  function saveFormState() {"  
00100 "    const form = document.getElementById(formId);"  
00101 "    if (!form) return;"  
00102 "    const data = {};"  
00103 "    Array.from(form.elements).forEach(el => {"  
00104 "      if (!el.name) return;"  
00105 "      if (el.type === 'checkbox' || el.type === 'radio') {"  
00106 "        data[el.name] = el.checked;"  
00107 "      } else {"  
00108 "        data[el.name] = el.value;"  
00109 "      }"  
00110 "    });"  
00111 "    try {"  
00112 "      localStorage.setItem(storageKey, JSON.stringify(data));"  
00113 "    } catch (e) {"  
00114 "      console.log('No se pudo guardar estado:', e);"  
00115 "    }"  
00116 "  }"  
00117 "  function loadFormState() {"  
00118 "    const form = document.getElementById(formId);"  
00119 "    if (!form) return;"  
00120 "    const raw = localStorage.getItem(storageKey);"  
00121 "    if (!raw) return;"  
00122 "    try {"  
00123 "      const data = JSON.parse(raw);"  
00124 "      Array.from(form.elements).forEach(el => {"  
00125 "        if (!el.name || !(el.name in data)) return;"  
00126 "        if (el.name === 'cmd') return;"  
00127 "        if (el.type === 'checkbox' || el.type === 'radio') {"  
00128 "          el.checked = !!data[el.name];"  
00129 "        } else {"  
00130 "          el.value = data[el.name];"  
00131 "        }"  
00132 "      }";"  
00133 "    } catch (e) {"  
00134 "      console.log('No se pudo leer estado:', e);"  
00135 "    }"  
00136 "  }"  
00137 "  window.addEventListener('load', () => {"  
00138 "    loadFormState();"  
00139 "    const form = document.getElementById(formId);"  
00140 "    if (!form) return;"  
00141 "    form.addEventListener('input', saveFormState);"  
00142 "  });"  
00143 "</script>"  
00144  
00145 "</body>"
```

```
00146 "</html>";
```

## 10.85. Referencia del archivo

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDe←  
Frecuencia/Software/LVFV\_ESP32/main/wifi/form.h

### Variables

- const char \* [HTML\\_FORM](#)

### 10.85.1. Documentación de variables

#### 10.85.1.1. HTML\_FORM

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

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

## 10.86. 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
```

## 10.87. Referencia del archivo

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDe←  
Frecuencia/Software/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"

### 10.87.1. Documentación de funciones

#### 10.87.1.1. [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](#).

### 10.87.1.2. hex2int()

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

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

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

### 10.87.1.3. 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](#).

### 10.87.1.4. 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](#).

### 10.87.1.5. start\_webserver()

```
httpd_handle_t start_webserver (
    void )
```

Arranca el servidor HTTP y registra endpoints.

- GET "/" → formulario
- POST "/save" → procesa y responde

Definición en la línea 282 del archivo [wifi.c](#).

### 10.87.1.6. 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](#).

### 10.87.1.7. wifi\_init\_softap()

```
void wifi_init_softap (
    void )
```

Configura ESP-NETIF + Wi-Fi en modo AP y arranca SSID/clave. (Canal 1, máx 1 cliente, WPA/WPA2).

Definición en la línea 307 del archivo [wifi.c](#).

## 10.87.2. Documentación de variables

### 10.87.2.1. TAG

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

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

## 10.88. wifi.c

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

```
00001
00009 #include <stdio.h>
00010 #include <string.h>
00011 #include <stdlib.h>
00012 #include <ctype.h>
00013
00014 #include "freertos/FreeRTOS.h"
00015 #include "freertos/task.h"
00016
00017 #include "esp_event.h"
00018 #include "esp_log.h"
00019 #include "nvs_flash.h"
00020 // #include "mdns.h"
00021
00022 #include "esp_netif.h"
00023
00024 // Tus headers con las structs y funciones:
00025 #include "../LVFV_system.h"
00026 #include "../system/sysControl.h"
00027 #include "../display/display.h"
00028 #include "../wifi.h"
00029 #include "form.h"
00030
00031 static const char *TAG = "WEB_CFG";
00032
00033 /* ----- Helpers para parsear POST x-www-form-urlencoded ----- */
00034
00040 static int hex2int(char c);
00041
00049 static void url_decode(char *dst, const char *src);
00050
00063 static bool get_param_value(const char *body, const char *key, char *dest, size_t dest_len);
00064
00070 static esp_err_t root_get_handler(httpd_req_t *req);
00071
00089 static esp_err_t save_post_handler(httpd_req_t *req);
```

```

00090
00091 static int hex2int(char c) {
00092     if (c >= '0' && c <= '9') return c - '0';
00093     if (c >= 'a' && c <= 'f') return 10 + (c - 'a');
00094     if (c >= 'A' && c <= 'F') return 10 + (c - 'A');
00095     return 0;
00096 }
00097
00098 static void url_decode(char *dst, const char *src) {
00099     while (*src) {
00100         if (*src == '+') {
00101             *dst++ = ' ';
00102             src++;
00103         } else if (*src == '%' && isxdigit((unsigned char)src[1]) && isxdigit((unsigned char)src[2])) {
00104             int hi = hex2int(src[1]);
00105             int lo = hex2int(src[2]);
00106             *dst++ = (char)((hi << 4) | lo);
00107             src += 3;
00108         } else {
00109             *dst++ = *src++;
00110         }
00111     }
00112     *dst = '\0';
00113 }
00114
00115 static bool get_param_value(const char *body, const char *key, char *dest, size_t dest_len) {
00116     size_t key_len = strlen(key);
00117     const char *p = body;
00118
00119     while (p && *p) {
00120         const char *eq = strchr(p, '=');
00121         if (!eq) break;
00122
00123         // Nombre de parámetro
00124         size_t this_key_len = (size_t)(eq - p);
00125
00126         if (this_key_len == key_len && strncmp(p, key, key_len) == 0) {
00127             // Encontramos la clave. Buscar fin del valor (& o final de cadena)
00128             const char *val_start = eq + 1;
00129             const char *amp = strchr(val_start, '&');
00130             size_t val_len = amp ? (size_t)(amp - val_start) : strlen(val_start);
00131             if (val_len >= dest_len) val_len = dest_len - 1;
00132
00133             char encoded[256];
00134             if (val_len >= sizeof(encoded)) val_len = sizeof(encoded) - 1;
00135
00136             memcpy(encoded, val_start, val_len);
00137             encoded[val_len] = '\0';
00138
00139             url_decode(dest, encoded);
00140             return true;
00141         }
00142
00143         // Ir al siguiente par key=value
00144         const char *amp = strchr(eq + 1, '&');
00145         if (!amp) break;
00146         p = amp + 1;
00147     }
00148
00149     return false;
00150 }
00151
00152 static esp_err_t root_get_handler(httpd_req_t *req) {
00153     httpd_resp_set_type(req, "text/html");
00154     httpd_resp_send(req, HTML_FORM, HTTPD_RESP_USE_STRLEN);
00155     return ESP_OK;
00156 }
00157
00158 static esp_err_t save_post_handler(httpd_req_t *req) {
00159     char body[512];
00160     char cmd[16] = {0};
00161
00162     time_settings_SH1106_t time_edit;
00163     security_settings_SH1106_t seccurity_edit;
00164     frequency_settings_SH1106_t frequency_edit;
00165
00166     if (req->content_len >= sizeof(body)) {
00167         ESP_LOGW(TAG, "Body demasiado grande (%d)", req->content_len);
00168         httpd_resp_send_err(req, HTTPD_500_INTERNAL_SERVER_ERROR, "Body too large");
00169         return ESP_FAIL;
00170     }
00171
00172     int received = httpd_req_recv(req, body, req->content_len);
00173     if (received <= 0) {
00174         ESP_LOGW(TAG, "Error recibiendo body");
00175         httpd_resp_send_err(req, HTTPD_500_INTERNAL_SERVER_ERROR, "Receive error");
}

```

```

00176     return ESP_FAIL;
00177 }
00178 body[received] = '\0';
00179
00180 ESP_LOGI(TAG, "Body: %s", body);
00181
00182 char buf[64];
00183
00184 /* ----- Leer parámetros ----- */
00185
00186 int frequency = 0, accel = 0, decel = 0, variation = 0;
00187 int imax = 0, vmin = 0;
00188 char sys_time_str[16] = {0};
00189 char start_time_str[16] = {0};
00190 char stop_time_str[16] = {0};
00191
00192 if (get_param_value(body, "frequency", buf, sizeof(buf))) {
00193     frequency = atoi(buf);
00194     ESP_LOGI(TAG, "Frecuencia: %d", frequency);
00195 }
00196 if (get_param_value(body, "accel", buf, sizeof(buf))) {
00197     accel = atoi(buf);
00198     ESP_LOGI(TAG, "Aceleración: %d", accel);
00199 }
00200 if (get_param_value(body, "decel", buf, sizeof(buf))) {
00201     decel = atoi(buf);
00202     ESP_LOGI(TAG, "Desaceleración: %d", decel);
00203 }
00204 if (get_param_value(body, "imax", buf, sizeof(buf))) {
00205     imax = atoi(buf);
00206     ESP_LOGI(TAG, "Corriente maxima: %d", imax);
00207 }
00208 if (get_param_value(body, "vmin", buf, sizeof(buf))) {
00209     vmin = atoi(buf);
00210     ESP_LOGI(TAG, "Tension minima: %d", vmin);
00211 }
00212 if (get_param_value(body, "variation", buf, sizeof(buf))) {
00213     variation = atoi(buf);
00214     ESP_LOGI(TAG, "Variacion de entradas %d", variation);
00215 }
00216 get_param_value(body, "sys_time", sys_time_str, sizeof(sys_time_str));
00217 get_param_value(body, "start_time", start_time_str, sizeof(start_time_str));
00218 get_param_value(body, "stop_time", stop_time_str, sizeof(stop_time_str));
00219
00220 /* ----- Parsear horas ----- */
00221
00222 int sys_h=0, sys_m=0, sys_s=0;
00223 int start_h=0, start_m=0;
00224 int stop_h=0, stop_m=0;
00225
00226 if (sscanf(sys_time_str, "%2d:%2d:%2d", &sys_h, &sys_m, &sys_s) != 3) {
00227     ESP_LOGW(TAG, "Hora de sistema inválida: %s", sys_time_str);
00228 }
00229 if (sscanf(start_time_str, "%2d:%2d", &start_h, &start_m) != 2) {
00230     ESP_LOGW(TAG, "Hora de arranque inválida: %s", start_time_str);
00231 }
00232 if (sscanf(stop_time_str, "%2d:%2d", &stop_h, &stop_m) != 2) {
00233     ESP_LOGW(TAG, "Hora de parada inválida: %s", stop_time_str);
00234 }
00235 ESP_LOGI(TAG, "Hora de sistema: %02d:%02d:%02d", sys_h, sys_m, sys_s);
00236 ESP_LOGI(TAG, "Hora de arranque: %02d:%02d", start_h, start_m);
00237 ESP_LOGI(TAG, "Hora de parada: %02d:%02d", stop_h, stop_m);
00238
00239 struct tm time_system;
00240 time_system.tm_hour = sys_h;
00241 time_system.tm_min = sys_m;
00242 time_system.tm_sec = sys_s;
00243 struct tm time_start;
00244 time_start.tm_hour = start_h;
00245 time_start.tm_min = start_m;
00246 struct tm time_stop;
00247 time_stop.tm_hour = stop_h;
00248 time_stop.tm_min = stop_m;
00249
00250 frequency_edit.frequency_settings.freq_regime = frequency;
00251 frequency_edit.frequency_settings.acceleration = accel;
00252 frequency_edit.frequency_settings.desacceleration = decel;
00253 frequency_edit.frequency_settings.input_variable = variation;
00254
00255 security_edit.security_settings.ibus_max = imax;
00256 security_edit.security_settings.vbus_min = vmin;
00257
00258 time_edit.time_settings.time_system = &time_system;
00259 time_edit.time_settings.time_start = &time_start;
00260 time_edit.time_settings.time_stop = &time_stop;
00261
00262

```

```

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

```

## 10.89. Referencia del archivo

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDe←  
Frecuencia/Software/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).*

### 10.89.1. Documentación de funciones

#### 10.89.1.1. `start_webserver()`

```
httpd_handle_t start_webserver (
    void )
```

Arranca el servidor HTTP y registra endpoints.

- GET "/" → formulario
- POST "/save" → procesa y responde

Definición en la línea 282 del archivo [wifi.c](#).

#### 10.89.1.2. `wifi_init_softap()`

```
void wifi_init_softap (
    void )
```

Configura ESP-NETIF + Wi-Fi en modo AP y arranca SSID/clave. (Canal 1, máx 1 cliente, WPA/WPA2).

Definición en la línea 307 del archivo [wifi.c](#).

## 10.90. 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
```

## 10.91. Referencia del archivo

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDe←  
Frecuencia/Software/LVFV\_ESP32/pytest\_hello\_world.py

### Espacios de nombres

- namespace [pytest\\_hello\\_world](#)

### Funciones

- None [test\\_hello\\_world](#) (IdfDut dut, Callable[..., None] log\_minimum\_free\_heap\_size)
- None [test\\_hello\\_world\\_linux](#) (IdfDut dut)
- None [test\\_hello\\_world\\_macos](#) (IdfDut dut)
- None [verify\\_elf\\_sha256\\_embedding](#) (QemuApp app, str sha256\_reported)
- None [test\\_hello\\_world\\_host](#) (QemuApp app, QemuDut dut)

## 10.92. pytest\_hello\_world.py

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

```

00001 # SPDX-FileCopyrightText: 2022-2025 Espressif Systems (Shanghai) CO LTD
00002 # SPDX-License-Identifier: CC0-1.0
00003 import hashlib
00004 import logging
00005 from typing import Callable
00006
00007 import pytest
00008 from pytest_embedded_idf.dut import IdfDut
00009 from pytest_embedded_idf.utils import idf_parametrize
00010 from pytest_embedded_qemu.app import QemuApp
00011 from pytest_embedded_qemu.dut import QemuDut
00012
00013
00014 @pytest.mark.generic
00015 @idf_parametrize('target', ['supported_targets', 'preview_targets'], indirect=['target'])
00016 def test_hello_world(dut: IdfDut, log_minimum_free_heap_size: Callable[..., None]) -> None:
00017     dut.expect('Hello world!')
00018     log_minimum_free_heap_size()
00019
00020
00021 @pytest.mark.host_test
00022 @idf_parametrize('target', ['linux'], indirect=['target'])
00023 def test_hello_world_linux(dut: IdfDut) -> None:
00024     dut.expect('Hello world!')
00025
00026
00027 @pytest.mark.host_test
00028 @pytest.mark.macos_shell
00029 @idf_parametrize('target', ['linux'], indirect=['target'])
00030 def test_hello_world_macos(dut: IdfDut) -> None:
00031     dut.expect('Hello world!')
00032
00033
00034 def verify_elf_sha256_embedding(app: QemuApp, sha256_reported: str) -> None:
00035     sha256 = hashlib.sha256()
00036     with open(app.elf_file, 'rb') as f:
00037         sha256.update(f.read())
00038     sha256_expected = sha256.hexdigest()
00039
00040     logging.info(f'ELF file SHA256: {sha256_expected}')
00041     logging.info(f'ELF file SHA256 (reported by the app): {sha256_reported}')
00042
00043     # the app reports only the first several hex characters of the SHA256, check that they match
00044     if not sha256_expected.startswith(sha256_reported):
00045         raise ValueError('ELF file SHA256 mismatch')
00046
00047
00048 @pytest.mark.host_test
00049 @pytest.mark.qemu
00050 @idf_parametrize('target', ['esp32', 'esp32c3'], indirect=['target'])
00051 def test_hello_world_host(app: QemuApp, dut: QemuDut) -> None:
00052     sha256_reported = dut.expect(r'ELF file SHA256:\s+([a-f0-9]+)').group(1).decode('utf-8')
00053     verify_elf_sha256_embedding(app, sha256_reported)
00054
00055     dut.expect('Hello world!')

```

## 10.93 Referencia del archivo

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/README.md 15

## 10.93. Referencia del archivo

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/LVFV\_ESP32/README.md

## 10.94. Referencia del archivo C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/Software\_ESP32/SPI\_ESP32\_TestModule/SPI\_Module.c

```
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/spi_master.h"
#include "driver/gpio.h"
#include "esp_log.h"
#include <stdio.h>
#include <string.h>
#include "SPI_Module.h"
```

### defines

- #define SPI\_REQUEST\_RESPONSE 0x50
- #define PIN\_NUM\_MISO 19
- #define PIN\_NUM\_MOSI 23
- #define PIN\_NUM\_CLK 18
- #define PIN\_NUM\_CS 5

### Funciones

- void SPI\_Init ()  
*Inicializa el módulo SPI del ESP32.*
- SPI\_Response SPI\_SendRequest (SPI\_Request request, int setValue, int \*getValue)  
*Ejecuta una consulta y devuelve la respuesta.*

### Variables

- spi\_device\_handle\_t spi\_handle

## 10.94.1. Documentación de «define»

### 10.94.1.1. PIN\_NUM\_CLK

```
#define PIN_NUM_CLK 18
```

Definición en la línea 14 del archivo [SPI\\_Module.c](#).

#### 10.94.1.2. PIN\_NUM\_CS

```
#define PIN_NUM_CS 5
```

Definición en la línea 15 del archivo [SPI\\_Module.c](#).

#### 10.94.1.3. PIN\_NUM\_MISO

```
#define PIN_NUM_MISO 19
```

Definición en la línea 12 del archivo [SPI\\_Module.c](#).

#### 10.94.1.4. PIN\_NUM\_MOSI

```
#define PIN_NUM_MOSI 23
```

Definición en la línea 13 del archivo [SPI\\_Module.c](#).

#### 10.94.1.5. SPI\_REQUEST\_RESPONSE

```
#define SPI_REQUEST_RESPONSE 0x50
```

Definición en la línea 10 del archivo [SPI\\_Module.c](#).

### 10.94.2. Documentación de funciones

#### 10.94.2.1. SPI\_Init()

```
void SPI_Init (
    void )
```

Inicializa el módulo SPI del ESP32.

Inicializa el módulo.

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 20 del archivo [SPI\\_Module.c](#).

#### 10.94.2.2. SPI\_SendRequest()

```
SPI_Response SPI_SendRequest (
    SPI_Request request,
    int setValue,
    int * getValue)
```

Ejecuta una consulta y devuelve la respuesta.

#### Parámetros

---

<i>request</i>	En este se carga el tipo de consulta que se requiere
<i>setValue</i>	Dependiendo del tipo de request se necesita un valor o no. Si es un setFrec le debemos enviar en este campo la frecuencia deseada. Si se quiere retornar un valor es indistinto el valor que se ingrese a este campo.
<i>getValue</i>	Este campo se utiliza para retornar un valor en caso que corresponda, manejar los punteros respectivamente, el programa espera espacio para un solo int en el que se guardara informacion de retorno aparte de la respuesta return de la funcion.

**Devuelve**

En esta se devuelve una de la lista posible de respuestas [SPI\\_Response](#)

Definición en la línea 47 del archivo [SPI\\_Module.c](#).

### 10.94.3. Documentación de variables

#### 10.94.3.1. spi\_handle

```
spi_device_handle_t spi_handle
```

Definición en la línea 18 del archivo [SPI\\_Module.c](#).

## 10.95. SPI\_Module.c

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

```
00001 #include "freertos/FreeRTOS.h"
00002 #include "freertos/task.h"
00003 #include "driver/spi_master.h"
00004 #include "driver/gpio.h"
00005 #include "esp_log.h"
00006 #include <stdio.h>
00007 #include <string.h>
00008 #include "SPI_Module.h"
00009
00010 #define SPI_REQUEST_RESPONSE 0x50 // Comando para recibir la respuesta
00011
00012 #define PIN_NUM_MISO 19      // rojo
00013 #define PIN_NUM_MOSI 23     // marron
00014 #define PIN_NUM_CLK 18      // naranja
00015 #define PIN_NUM_CS 5        // amarillo
00016
00017
00018 spi_device_handle_t spi_handle;
00019
00020 void SPI_Init() {
00021     spi_bus_config_t buscfg = {
00022         .miso_io_num = PIN_NUM_MISO,
00023         .mosi_io_num = PIN_NUM_MOSI,
00024         .sclk_io_num = PIN_NUM_CLK,
00025         .quadwp_io_num = -1,
00026         .quadhd_io_num = -1,
00027         .max_transfer_sz = 32
00028     };
00029
00030     spi_device_interface_config_t devcfg = {
00031         .clock_speed_hz = 1*1000*1000,           // 1 MHz
00032         .mode = 0,                            // SPI mode 0
00033         .spics_io_num = PIN_NUM_CS,            // CS pin
00034         .queue_size = 1,                      // Sin dummy bits
00035     };
00036 }
00037
```

```

00038 // Inicializar bus SPI
00039 esp_err_t ret = spi_bus_initialize(VSPI_HOST, &buscfg, 1);
00040 ESP_ERROR_CHECK(ret);
00041
00042 // Añadir dispositivo
00043 ret = spi_bus_add_device(VSPI_HOST, &devcfg, &spi_handle);
00044 ESP_ERROR_CHECK(ret);
00045 }
00046
00047 SPI_Response SPI_SendRequest(SPI_Request request, int setValue, int* getValue) {
00048
00049     uint8_t tx_buffer[4];
00050     uint8_t rx_buffer[4];
00051     esp_err_t ret;
00052     int flagDevolverValor;      // Se necesita devolver el valor por parametro
00053
00054     printf("[SPI Module] Request %d\n", request);
00055
00056     // Chequeo de errores fuera de rango y comando desconocido
00057     if(setValue > 512 || setValue < 0) {
00058         return SPI_RESPONSE_ERR_DATA_INVALID;
00059     }
00060
00061     if(request < SPI_REQUEST_START || request >= SPI_REQUEST_LAST) {
00062         printf("[SPI Module] Comando desconocido\n");
00063         return SPI_RESPONSE_ERR_CMD_UNKNOWN;
00064     }
00065
00066     // Es de los comandos que deben devolver un valor por parametro?
00067     if(request >= SPI_REQUEST_GET_FREC && request <= SPI_REQUEST_IS_STOP) {
00068         flagDevolverValor = 1;
00069     } else {
00070         flagDevolverValor = 0;
00071     }
00072
00073     // Se arma la cadena a enviar
00074     tx_buffer[0] = request;
00075
00076     if(setValue > 255) {
00077         tx_buffer[1] = 255;
00078         tx_buffer[2] = setValue - 255;
00079         tx_buffer[3] = ';';
00080     } else {
00081         tx_buffer[1] = setValue;
00082         tx_buffer[2] = 0;
00083         tx_buffer[3] = ';';
00084     }
00085
00086
00087     // Limpiamos el buffer de recepcion
00088     memset(rx_buffer, 0, sizeof(rx_buffer));
00089
00090     // Siempre envio y recibo 4 bytes
00091     spi_transaction_t t = {
00092         .length = 8 * 4,
00093         .tx_buffer = tx_buffer,
00094         .rx_buffer = rx_buffer,
00095         .rxlength = 8 * 4,
00096     };
00097
00098     ret = spi_device_transmit(spi_handle, &t);
00099     if (ret != ESP_OK) {
00100         printf("[ESP32] Error en SPI transmit: %d\n", ret);
00101         return SPI_RESPONSE_ERR;
00102     }
00103
00104     tx_buffer[0] = SPI_REQUEST_RESPONSE;
00105     tx_buffer[1] = ';';
00106
00107     t.length = 8 * 4;
00108     t.tx_buffer = tx_buffer;
00109     t.rx_buffer = rx_buffer;
00110     t.rxlength = 8 * 4;
00111
00112     vTaskDelay(pdMS_TO_TICKS(200));
00113
00114     ret = spi_device_transmit(spi_handle, &t);
00115     if (ret != ESP_OK) {
00116         printf("[ESP32] Error en SPI transmit: %d\n", ret);
00117         return SPI_RESPONSE_ERR;
00118     }
00119
00120
00121
00122     if(flagDevolverValor && rx_buffer[0] == SPI_RESPONSE_OK) {
00123
00124         *getValue = rx_buffer[1] + rx_buffer[2];

```

## 10.96 Referencia del archivo

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/Software

ESP32/SPI\_ESP32\_TestModule/SPI\_Module.h

419

```
00125     printf("RxValores: %d - %d", rx_buffer[1], rx_buffer[2]);  
00126  
00127     }else {  
00128         *getValue = 0;  
00129     }  
00130  
00131     return rx_buffer[0];  
00132 }  
00133
```

## 10.96. Referencia del archivo C:/Users/User/Desktop/ProyectoFinal- Grupo5-VariadorDeFrecuencia/Software/Software ESP32/SPI\_ESP32\_TestModule/SPI\_Module.h

### Enumeraciones

- enum SPI\_Request {  
 SPI\_REQUEST\_START = 10 , SPI\_REQUEST\_STOP = 11 , SPI\_REQUEST\_SET\_FREC = 12 ,  
 SPI\_REQUEST\_SET\_ACCEL = 13 ,  
 SPI\_REQUEST\_SET\_DESACEL = 14 , SPI\_REQUEST\_SET\_DIR = 15 , SPI\_REQUEST\_GET\_FREC = 16  
 , SPI\_REQUEST\_GET\_ACCEL = 17 ,  
 SPI\_REQUEST\_GET\_DESACEL = 18 , SPI\_REQUEST\_GET\_DIR = 19 , SPI\_REQUEST\_IS\_STOP = 20 ,  
 SPI\_REQUEST\_EMERGENCY = 21 ,  
 SPI\_REQUEST\_LAST = 22 }
- enum SPI\_Response {  
 SPI\_RESPONSE\_OK = 0xFF , SPI\_RESPONSE\_ERR = 0xA0 , SPI\_RESPONSE\_ERR\_CMD\_UNKNOWN  
 = 0xA1 , SPI\_RESPONSE\_ERR\_NO\_COMMAND = 0xA2 ,  
 SPI\_RESPONSE\_ERR\_MOVING = 0xA3 , SPI\_RESPONSE\_ERR\_NOT\_MOVING = 0xA4 , SPI\_RESPONSE\_ERR\_DATA\_INVALID  
 = 0xA5 , SPI\_RESPONSE\_ERR\_DATA\_INVALID = 0xA6 ,  
 SPI\_RESPONSE\_ERR\_DATA\_OUT\_RANGE = 0xA7 , SPI\_RESPONSE\_ERR\_EMERGENCY\_ACTIVE =  
 0xA8 , SPI\_LAST\_VALUE }

### Funciones

- void SPI\_Init (void)  
*Inicializa el módulo.*
- SPI\_Response SPI\_SendRequest (SPI\_Request request, int setValue, int \*getValue)  
*Ejecuta una consulta y devuelve la respuesta.*

### 10.96.1. Documentación de enumeraciones

#### 10.96.1.1. SPI\_Request

enum SPI\_Request

##### Valores de enumeraciones

SPI_REQUEST_START	
SPI_REQUEST_STOP	
SPI_REQUEST_SET_FREC	

SPI_REQUEST_SET_ACCEL	
SPI_REQUEST_SET_DESACEL	
SPI_REQUEST_SET_DIR	
SPI_REQUEST_GET_FREC	
SPI_REQUEST_GET_ACCEL	
SPI_REQUEST_GET_DESACEL	
SPI_REQUEST_GET_DIR	
SPI_REQUEST_IS_STOP	
SPI_REQUEST_EMERGENCY	
SPI_REQUEST_LAST	

Definición en la línea 14 del archivo [SPI\\_Module.h](#).

#### 10.96.1.2. SPI\_Response

enum [SPI\\_Response](#)

##### Valores de enumeraciones

SPI_RESPONSE_OK	
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_ERR_EMERGENCY_ACTIVE	
SPI_LAST_VALUE	

Definición en la línea 30 del archivo [SPI\\_Module.h](#).

#### 10.96.2. Documentación de funciones

##### 10.96.2.1. SPI\_Init()

void SPI\_Init ()

Inicializa el módulo.

##### Parámetros

## 10.96 Referencia del archivo

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/Software

ESP32/SPI\_ESP32\_TestModule/SPI\_Module.h

421

void	
------	--

Devuelve

void

Inicializa el módulo.

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](#).

### 10.96.2.2. SPI\_SendRequest()

```
SPI_Response SPI_SendRequest (
    SPI_Request request,
    int setValue,
    int * getValue)
```

Ejecuta una consulta y devuelve la respuesta.

#### Parámetros

<i>request</i>	En este se carga el tipo de consulta que se requiere
<i>setValue</i>	Dependiendo del tipo de request se necesita un valor o no. Si es un setFreq le debemos enviar en este campo la frecuencia deseada. Si se quiere retornar un valor es indistinto el valor que se ingrese a este campo.
<i>getValue</i>	Este campo se utiliza para retornar un valor en caso que corresponda, manejar los punteros respectivamente, el programa espera espacio para un solo int en el que se guardara información de retorno aparte de la respuesta return de la función.

Devuelve

En esta se devuelve una de la lista posible de respuestas [SPI\\_Response](#)

Definición en la línea 47 del archivo [SPI\\_Module.c](#).

## 10.97. SPI\_Module.h

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

```

00001 #ifndef SPI_MODULE_H
00002 #define SPI_MODULE_H
00003
00004
00005 // =====
00006 // Definiciones y configuraciones
00007 // =====
00008
00009
00010 // =====
00011 // Tipos de datos
00012 // =====
00013
00014 typedef enum {
00015     SPI_REQUEST_START      = 10,
00016     SPI_REQUEST_STOP       = 11,
00017     SPI_REQUEST_SET_FREQ   = 12,
00018     SPI_REQUEST_SET_ACCEL  = 13,
00019     SPI_REQUEST_SET_DESACEL = 14,
00020     SPI_REQUEST_SET_DIR    = 15,
00021     SPI_REQUEST_GET_FREQ   = 16,
00022     SPI_REQUEST_GET_ACCEL  = 17,
00023     SPI_REQUEST_GET_DESACEL = 18,
00024     SPI_REQUEST_GET_DIR    = 19,
00025     SPI_REQUEST_IS_STOP    = 20,
00026     SPI_REQUEST_EMERGENCY  = 21,
00027     SPI_REQUEST_LAST       = 22,
00028 } SPI_Request;
00029
00030 typedef enum {
00031     SPI_RESPONSE_OK        = 0xFF,
00032     SPI_RESPONSE_ERR        = 0xA0,
00033     SPI_RESPONSE_ERR_CMD_UNKNOWN = 0xA1,           // Comando desconocido
00034     SPI_RESPONSE_ERR_NO_COMMAND = 0xA2,            // Llego mensaje pero sin comando
00035     SPI_RESPONSE_ERR_MOVING = 0xA3,              // Comando Start pero motor ya esta en movimiento
00036     SPI_RESPONSE_ERR_NOT_MOVING = 0xA4,           // Comando Stop pero motor no esta en movimiento
00037     SPI_RESPONSE_ERR_DATA_MISSING = 0xA5,          // Comando que requiere datos pero no llegaron
00038     SPI_RESPONSE_ERR_DATA_INVALID = 0xA6,          // Comando que tiene datos invalidos
00039     SPI_RESPONSE_ERR_DATA_OUT_RANGE = 0xA7,         // Comando con datos fuera de rango permitido
00040     SPI_RESPONSE_ERR_EMERGENCY_ACTIVE = 0xA8,        // Emergencia activa
00041     SPI_LAST_VALUE,                                // Los mensajes deben tener un valor consecutivo uno
00042     de otro
00043 } SPI_Response;
00044
00045 // =====
00046 // Funciones públicas
00047 // =====
00048
00055 void SPI_Init(void);
00056
00069 SPI_Response SPI_SendRequest(SPI_Request request, int setValue, int* getValue);
00070
00071 #endif // SPI_MODULE_H

```

## 10.98. Referencia del archivo C:/Users/User/Desktop/ProyectoFinal- Grupo5-VariadorDeFrecuencia/Software/Software ESP32/SPI\_ESP32\_TestModule/UART\_Module.c

```

#include "UART_Module.h"
#include "driver/uart.h"
#include "esp_log.h"
#include <stdio.h>
#include <string.h>
#include <stdbool.h>

```

- #define **BUF\_SIZE** 128
- #define **UART\_PORT** UART\_NUM\_0

## Funciones

- void **UART\_Init** ()
- int **UART\_GetComando** (int \*buffer\_out)

### 10.98.1. Documentación de «define»

#### 10.98.1.1. **BUF\_SIZE**

```
#define BUF_SIZE 128
```

Definición en la línea 8 del archivo [UART\\_Module.c](#).

#### 10.98.1.2. **UART\_PORT**

```
#define UART_PORT UART_NUM_0
```

Definición en la línea 9 del archivo [UART\\_Module.c](#).

## 10.98.2. Documentación de funciones

### 10.98.2.1. **UART\_GetComando()**

```
int UART_GetComando (
    int * buffer_out)
```

Definición en la línea 27 del archivo [UART\\_Module.c](#).

### 10.98.2.2. **UART\_Init()**

```
void UART_Init (
    void )
```

Definición en la línea 14 del archivo [UART\\_Module.c](#).

## 10.99. UART\_Module.c

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

```

00001 #include "UART_Module.h"
00002 #include "driver/uart.h"
00003 #include "esp_log.h"
00004 #include <stdio.h>
00005 #include <string.h>
00006 #include <stdbool.h>
00007
00008 #define BUF_SIZE 128
00009 #define UART_PORT UART_NUM_0
00010
00011
00012
00013
00014 void UART_Init() {
00015     uart_config_t uart_config = {
00016         .baud_rate = 115200,
00017         .data_bits = UART_DATA_8_BITS,
00018         .parity = UART_PARITY_DISABLE,
00019         .stop_bits = UART_STOP_BITS_1,
00020         .flow_ctrl = UART_HW_FLOWCTRL_DISABLE
00021     };
00022     uart_param_config(UART_PORT, &uart_config);
00023     uart_driver_install(UART_PORT, 1024, 0, 0, NULL, 0);
00024 }
00025
00026
00027 int UART_GetComando(int* buffer_out) {
00028     char aux[BUF_SIZE];
00029     char ch;
00030     int idx = 0;
00031     int idxOut = 0;
00032     int n, i;
00033
00034     while (1) {
00035
00036         ch = 0;
00037
00038         // Se lee un byte de la consola
00039         uart_read_bytes(UART_PORT, &ch, 1, portMAX_DELAY);
00040
00041
00042         if(ch != ' ' && ch != ';') {
00043             // Se usa un buffer auxiliar para guardar un numero en un solo byte
00044             aux[idx] = ch;
00045             aux[idx+1] = '\0';
00046             idx++;
00047
00048         }else {
00049             n = 0;
00050
00051             // Se calculo el numero del ultimo buffer auxiliar completado
00052             for(i = 0; i < idx; i++) {
00053                 n = n * 10 + (int)(aux[i] - '0'); // Convertir ASCII a número y acumular
00054             }
00055
00056             // Se chequea que no se vaya de los limites
00057             if(n > 512) {
00058                 n = 512;
00059             }else if(n < 0) {
00060                 n = 0;
00061             }
00062
00063             // Se guardan los numeros en nuestro buffer de retorno
00064             buffer_out[idxOut] = n;
00065             idx = 0;
00066             idxOut++;
00067
00068             // Si lleva un ; se devuelve lo que se tiene
00069             if(ch == ';') {
00070                 buffer_out[idxOut] = 0;
00071                 return 0;
00072             }
00073         }
00074     }
00075     return 1;
00076 }
00077
00078

```

## 10.100 Referencia del archivo

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/Software

ESP32/SPI\_ESP32\_TestModule/UART\_Module.h

10.100. Referencia del archivo C:/Users/User/Desktop/ProyectoFinal- 425

## Grupo5-VariadorDeFrecuencia/Software/Software

ESP32/SPI\_ESP32\_TestModule/UART\_Module.h

### Funciones

- void [UART\\_Init](#) (void)
- int [UART\\_GetComando](#) (int \*buffer\_out)

#### 10.100.1. Documentación de funciones

##### 10.100.1.1. [UART\\_GetComando\(\)](#)

```
int UART_GetComando (
    int * buffer_out)
```

Definición en la línea 27 del archivo [UART\\_Module.c](#).

##### 10.100.1.2. [UART\\_Init\(\)](#)

```
void UART_Init (
    void )
```

Definición en la línea 14 del archivo [UART\\_Module.c](#).

## 10.101. [UART\\_Module.h](#)

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

```
00001 #ifndef UART_MODULE_H
00002 #define UART_MODULE_H
00003
00004 // Funcion de inicializacion
00005 void UART\_Init(void);
00006
00007 // Funcion bloqueante que devuelve lo que se escribe en pantalla
00008 // Es bloqueante solo por propósitos de la prueba
00009 int UART\_GetComando(int* buffer_out);
00010
00011
00012 #endif
```

#### 10.102. Referencia del archivo C:/Users/User/Desktop/ProyectoFinal- 425 Grupo5-VariadorDeFrecuencia/Software/SVM Space Vector Modulation/SVM\_Calculos.py

### Espacios de nombres

- namespace [SVM\\_Calculos](#)

## Funciones

- `get_time_vector (Ts, M, angle)`
- `getSector (angle)`
- `getSecuencia (sector)`
- `getSecuenciaLabel (sector)`

## Variables

- int `SECTOR1` = 1
- int `SECTOR2` = 2
- int `SECTOR3` = 3
- int `SECTOR4` = 4
- int `SECTOR5` = 5
- int `SECTOR6` = 6
- int `Ts` = 1
- int `M` = 1
- list `vectors` = []
- int `current_angle` = 0

## 10.103. SVM\_Calculos.py

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

```

00001 import numpy as np
00002 import matplotlib.pyplot as plt
00003 from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
00004 import tkinter as tk
00005
00006 SECTOR1 = 1
00007 SECTOR2 = 2
00008 SECTOR3 = 3
00009 SECTOR4 = 4
00010 SECTOR5 = 5
00011 SECTOR6 = 6
00012
00013 # Este es el tiempo de sampleo
00014 Ts = 1
00015 # Indice de modulacion
00016 M = 1
00017
00018 # Lista para almacenar los vectores dibujados
00019 vectors = []
00020 current_angle = 0 # Ángulo inicial
00021
00022 # Funcion que devuelve los tiempos del sector
00023 # M toma un valor entre 0 y 1
00024 def get_time_vector(Ts, M, angle):
00025
00026     #sector = getSector(angle)
00027     angle = angle % 60
00028
00029     # Aca pongo mis constantes de calculo
00030     # Segun el paper t1 se cuenta a las dos porciones, por ello lo a
00031     # dividir por dos para que me quede mejor para mi definicion
00032     CONST_A = (3 * np.sqrt(3)) / (2 * np.pi) / 2
00033     CONST_B = 3 / np.pi / 2
00034
00035     #print(f"CONST_A: {CONST_A}, CONST_B: {CONST_B}")
00036
00037     varA = CONST_A * Ts
00038     varB = CONST_B * Ts
00039
00040     t0 = 0.0
00041     t1 = 0.0
00042     t2 = 0.0
00043
00044     t2 = varB * M * np.sin(np.radians(angle))
00045     t1 = varA * M * np.cos(np.radians(angle)) - (t2/2)

```

## 10.104 Referencia del archivo

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/SVM Space Vector

Modulation/SVM\_DiagramaPolar.py

427

```
00047
00048     # El tiempo de modulacion es igual a t1 + t2
00049     tMod = t1 + t2
00050
00051     t0 = Ts / 2 - tMod
00052
00053     return t0, t1, t2
00054
00055
00056
00057 # Devuelve el sector con un angulo dado desde 0 a 359
00058 def getSector(angle):
00059
00060     angle = angle % 360
00061
00062     if(angle >= 0 and angle < 60):
00063         return SECTOR1
00064     elif(angle >= 60 and angle < 120):
00065         return SECTOR2
00066     elif(angle >= 120 and angle < 180):
00067         return SECTOR3
00068     elif(angle >= 180 and angle < 240):
00069         return SECTOR4
00070     elif(angle >= 240 and angle < 300):
00071         return SECTOR5
00072     elif(angle >= 300 and angle < 360):
00073         return SECTOR6
00074
00075
00076
00077
00078 def getSecuencia(sector):
00079     if(sector == SECTOR1):
00080         return [0, 1, 2, 7, 7, 2, 1, 0]
00081     elif(sector == SECTOR2):
00082         return [0, 3, 2, 7, 7, 2, 3, 0]
00083     elif(sector == SECTOR3):
00084         return [0, 3, 4, 7, 7, 4, 3, 0]
00085     elif(sector == SECTOR4):
00086         return [0, 5, 4, 7, 7, 4, 5, 0]
00087     elif(sector == SECTOR5):
00088         return [0, 5, 6, 7, 7, 6, 5, 0]
00089     elif(sector == SECTOR6):
00090         return [0, 1, 6, 7, 7, 6, 1, 0]
00091
00092 def getSecuenciaLabel(sector):
00093     if(sector == SECTOR1):
00094         return "S0-S1-S2-S7-S7-S2-S1-S0"
00095     elif(sector == SECTOR2):
00096         return "S0-S3-S2-S7-S7-S2-S3-S0"
00097     elif(sector == SECTOR3):
00098         return "S0-S3-S4-S7-S7-S4-S3-S0"
00099     elif(sector == SECTOR4):
00100        return "S0-S5-S4-S7-S7-S4-S5-S0"
00101     elif(sector == SECTOR5):
00102        return "S0-S5-S6-S7-S7-S6-S5-S0"
00103     elif(sector == SECTOR6):
00104        return "S0-S1-S6-S7-S7-S6-S1-S0"
```

## 10.104. Referencia del archivo C:/Users/User/Desktop/ProyectoFinal- Grupo5-VariadorDeFrecuencia/Software/SVM Space Vector Modulation/SVM\_DiagramaPolar.py

### Espacios de nombres

- namespace [SVM\\_DiagramaPolar](#)

### Funciones

- [update\\_plot \(angle\)](#)
- [slider\\_update \(val\)](#)
- [animate \(\)](#)

## Variables

- list **vectors** = []
- int **current\_angle** = 0
- **root** = tk.Tk()
- **polar\_ax1**
- **polar\_ax2**
- **subplot\_kw**
- **figsize**
- **va**
- **canvas** = FigureCanvasTkAgg(fig, master=**root**)
- **canvas\_widget** = canvas.get\_tk\_widget()

## 10.105. SVM\_DiagramaPolar.py

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

```

00001 import numpy as np
00002 import matplotlib.pyplot as plt
00003 from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
00004 import tkinter as tk
00005
00006 # Lista para almacenar los vectores dibujados
00007 vectors = []
00008 current_angle = 0 # Ángulo inicial
00009
00010 # Función para actualizar el gráfico polar
00011 def update_plot(angle):
00012     global vectors
00013
00014     #current_angle = angle
00015     #v1, v2, v3 = calculate_modulation_index(angle)
00016
00017     # Eliminar solo los vectores existentes (sin borrar el resto del gráfico)
00018     for line in vectors:
00019         line.remove()
00020     vectors.clear()
00021
00022     magBase = 1.3
00023
00024     ang1 = 90
00025     ang2 = 210
00026     ang3 = 330
00027
00028     mag1 = magBase * np.cos(np.radians(ang1 - angle))
00029     mag2 = magBase * np.cos(np.radians(ang2 - angle))
00030     mag3 = magBase * np.cos(np.radians(ang3 - angle))
00031
00032
00033     magResultX = mag1 * np.cos(np.radians(ang1)) + mag2 * np.cos(np.radians(ang2)) + mag3 *
00034         np.cos(np.radians(ang3))
00035     magResultY = mag1 * np.sin(np.radians(ang1)) + mag2 * np.sin(np.radians(ang2)) + mag3 *
00036         np.sin(np.radians(ang3))
00037
00038     if mag1 < 0:
00039         ang1 += 180
00040         mag1 = -mag1
00041
00042     if mag2 < 0:
00043         ang2 += 180
00044         mag2 = -mag2
00045
00046     if mag3 < 0:
00047         ang3 += 180
00048         mag3 = -mag3
00049
00050     # Dibujar los nuevos vectores
00051     vector1 = polar_ax1.quiver(np.radians(ang1), 0, 0, mag1, angles='xy', scale_units='xy', scale=1,
00052         color='blue', alpha=0.7)
00053     vector2 = polar_ax1.quiver(np.radians(ang2), 0, 0, mag2, angles='xy', scale_units='xy', scale=1,
00054         color='green', alpha=0.7)
00055     vector3 = polar_ax1.quiver(np.radians(ang3), 0, 0, mag3, angles='xy', scale_units='xy', scale=1,
00056         color='red', alpha=0.7)

```

```

00054     result_vector = polar_ax1.quiver(np.radians(angle), 0, 0, magResult, angles='xy',
00055         scale_units='xy', scale=1, color='black')
00056
00057
00058     polar_ax1.set_theta_zero_location("N")
00059     polar_ax1.set_theta_direction(-1)
00060     polar_ax1.set_rticks([0.5, 1, 1.5, 2])
00061     polar_ax1.grid(True)
00062
00063
00064
00065     if(angle < 30) :
00066         mag1 = 0
00067     elif(angle >= 150 and angle < 210) :
00068         mag1 = 0
00069     elif(angle >= 330) :
00070         mag1 = 0
00071
00072     if(angle >= 90 and angle < 150) :
00073         mag2 = 0
00074     elif(angle >= 270 and angle < 330) :
00075         mag2 = 0
00076
00077     if(angle >= 30 and angle < 90) :
00078         mag3 = 0
00079     elif(angle >= 210 and angle < 270) :
00080         mag3 = 0
00081
00082
00083
00084     if(mag1 != 0) : # si el vector 1 no es cero entonces el otro tiene modulo mag2 + mag3
00085
00086         catetoX = mag1 * np.cos(np.radians(60)) + (mag2 + mag3)
00087         catetoY = mag1 * np.sin(np.radians(60))
00088         magResult = np.sqrt(catetoX**2 + catetoY**2)
00089
00090     elif(mag2 != 0) : # si el vector 2 no es cero entonces el otro tiene modulo mag1 + mag3
00091         catetoX = mag2 * np.cos(np.radians(60)) + (mag1 + mag3)
00092         catetoY = mag2 * np.sin(np.radians(60))
00093         magResult = np.sqrt(catetoX**2 + catetoY**2)
00094
00095
00096     vector4 = polar_ax2.quiver(np.radians(ang1), 0, 0, mag1, angles='xy', scale_units='xy', scale=1,
00097         color='blue', alpha=0.7)
00098     vector5 = polar_ax2.quiver(np.radians(ang2), 0, 0, mag2, angles='xy', scale_units='xy', scale=1,
00099         color='green', alpha=0.7)
00100     vector6 = polar_ax2.quiver(np.radians(ang3), 0, 0, mag3, angles='xy', scale_units='xy', scale=1,
00101         color='red', alpha=0.7)
00102     result_vector2 = polar_ax2.quiver(np.radians(angle), 0, 0, magResult, angles='xy',
00103         scale_units='xy', scale=1, color='black')
00104
00105
00106     # Actualizar lista de vectores
00107     vectors.extend([vector1, vector2, vector3, result_vector, vector4, vector5, vector6,
00108         result_vector2])
00109     canvas.draw()
00110
00111 # Función para sincronizar el slider y actualizar el gráfico
00112 def slider_update(val):
00113     angle = int(val)
00114     update_plot(angle)
00115
00116 # Función para incrementar el ángulo automáticamente
00117 def animate():
00118     global current_angle
00119     current_angle = (current_angle + 2) % 360 # Incrementar el ángulo, reiniciando a 0 después de 359
00120     #angle_slider.set(current_angle) # Sincronizar el slider con la animación
00121     update_plot(current_angle) # Redibujar el gráfico
00122     root.after(1, animate) # Llamar a esta función nuevamente después de 100 ms
00123
00124 # Crear ventana principal
00125 root = tk.Tk()
00126 root.title("Modulación SVM con Gráfico Polar")
00127
00128 # Crear una figura con dos graficos polares
00129 fig, (polar_ax1, polar_ax2) = plt.subplots(1, 2, subplot_kw={'projection': 'polar'}, figsize=(10, 5))
00130
00131 # Configurar el primer grafico polar
00132 polar_ax1.set_title("Grafico Polar 1", va='bottom')
00133 polar_ax1.set_theta_zero_location("N")
00134 polar_ax1.set_theta_direction(-1)
00135 polar_ax1.set_rticks([0.5, 1, 1.5, 2])

```

```

00135 polar_ax1.grid(True)
00136 polar_ax1.legend()
00137
00138 # Configurar el segundo grafico polar
00139 polar_ax2.set_title("Grafico Polar 2", va='bottom')
00140 polar_ax2.set_theta_zero_location("N")
00141 polar_ax2.set_theta_direction(-1)
00142 polar_ax2.set_rticks([0.5, 1, 1.5, 2])
00143 polar_ax2.grid(True)
00144
00145
00146
00147 canvas = FigureCanvasTkAgg(fig, master=root)
00148 canvas_widget = canvas.get_tk_widget()
00149 canvas_widget.pack()
00150
00151 # Crear un slider para cambiar el ángulo
00152 #angle_slider = tk.Scale(root, from_=0, to=360, orient="horizontal", label="Ángulo",
    command=slider_update)
00153 #angle_slider.pack()
00154
00155 # Iniciar la animación
00156 animate()
00157
00158 # Iniciar la interfaz gráfica
00159 root.mainloop()

```

## 10.106. Referencia del archivo C:/Users/User/Desktop/ProyectoFinal- Grupo5-VariadorDeFrecuencia/Software/SVM Space Vector Modulation/svm\_DiagramaSalidaTemporal.py

### Espacios de nombres

- namespace [svm\\_DiagramaSalidaTemporal](#)

### Funciones

- [generar\\_vector\\_salida\\_temporal \(Ts, M, angSwitch, tRise, tf\)](#)
- [calcular\\_fase \(t, out1, out2, out3, VBus\)](#)
- [calcular\\_rms \(signal\)](#)

### Variables

- int [SECTOR1](#) = 1
- int [SECTOR2](#) = 2
- int [SECTOR3](#) = 3
- int [SECTOR4](#) = 4
- int [SECTOR5](#) = 5
- int [SECTOR6](#) = 6
- int [Q1](#) = 0
- int [Q2](#) = 1
- int [Q3](#) = 2
- int [Ts](#) = 1
- int [M](#) = 1
- dict [SECTOR\\_VECTORS](#)
- int [angle\\_deg](#) = 45
- [t](#)
- [out1](#)
- [out2](#)

- out3
- angSwitch
- tRise
- tf
- t2
- faseRS
- faseST
- rms\_RS = calcular\_rms(faseRS)
- rms\_ST = calcular\_rms(faseST)
- figsize
- label

## 10.107. svm\_DiagramaSalidaTemporal.py

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

```

00001 import numpy as np
00002 import matplotlib.pyplot as plt
00003 from SVM_Calculos import getSector, get_time_vector, getSecuenciaLabel, getSecuencia
00004
00005
00006 SECTOR1 = 1
00007 SECTOR2 = 2
00008 SECTOR3 = 3
00009 SECTOR4 = 4
00010 SECTOR5 = 5
00011 SECTOR6 = 6
00012
00013 # Estas son la salida
00014 Q1 = 0
00015 Q2 = 1
00016 Q3 = 2
00017
00018 # Este es el tiempo de sampleo
00019 Ts = 1
00020 # Indice de modulacion
00021 M = 1
00022
00023 # Mapeo de vectores activos por sector
00024 # Cada entrada define el orden de vectores [V0, Vx, Vy, V7]
00025 SECTOR_VECTORS = {
00026     SECTOR1: [2, 1, 0],
00027     SECTOR2: [1, 2, 0],
00028     SECTOR3: [1, 0, 2],
00029     SECTOR4: [0, 1, 2],
00030     SECTOR5: [0, 2, 1],
00031     SECTOR6: [2, 0, 1],
00032 }
00033
00034
00035 # Generar la forma de onda de una salida
00036 def generar_vector_salida_temporal(Ts, M, angSwitch, tRise, tf) :
00037     t = []
00038     out1 = []
00039     out2 = []
00040     out3 = []
00041
00042     # Esta es la cantidad de switcheos que voy a tener
00043     cantSwitchs = int(tf / Ts)
00044
00045     # Angulo switch es el avance de angulo luego de todo un ciclo
00046     # de switch
00047
00048     # Estos son los tiempos y angulos al principio de mi iteracion
00049     # En cada paso del ciclo for voy a calcular un nuevo switcheo
00050     angAct = 0
00051     tiempoAct = 0
00052
00053     for i in range(cantSwitchs) :
00054
00055
00056
00057         # Voy a calcular el tiempo para cada paso
00058         t0, t1, t2 = get_time_vector(Ts, M, angAct)
00059         sector = getSector(angAct)
00060         ordenActivacion = SECTOR_VECTORS[sector]
```

```
00061     estadoOut = [0, 0, 0]
00062
00063
00064     # Pre paso 1
00065     tiempoAct += t0/2
00066     t.append(tiempoAct)
00067     out1.append(estadoOut[0])
00068     out2.append(estadoOut[1])
00069     out3.append(estadoOut[2])
00070
00071     # Paso 1
00072     estadoOut[ordenActivacion[0]] = 1
00073     t.append(tiempoAct + tRise)
00074     out1.append(estadoOut[0])
00075     out2.append(estadoOut[1])
00076     out3.append(estadoOut[2])
00077
00078     # Pre paso 2
00079     tiempoAct += t1
00080     t.append(tiempoAct)
00081     out1.append(estadoOut[0])
00082     out2.append(estadoOut[1])
00083     out3.append(estadoOut[2])
00084
00085     # Paso 2
00086     estadoOut[ordenActivacion[1]] = 1
00087     t.append(tiempoAct + tRise)
00088     out1.append(estadoOut[0])
00089     out2.append(estadoOut[1])
00090     out3.append(estadoOut[2])
00091
00092     # Pre paso 3
00093     tiempoAct += t2
00094     t.append(tiempoAct)
00095     out1.append(estadoOut[0])
00096     out2.append(estadoOut[1])
00097     out3.append(estadoOut[2])
00098
00099     # Paso 3
00100    estadoOut[ordenActivacion[2]] = 1
00101    t.append(tiempoAct + tRise)
00102    out1.append(estadoOut[0])
00103    out2.append(estadoOut[1])
00104    out3.append(estadoOut[2])
00105
00106     # Pre paso 4
00107     tiempoAct += t0
00108     t.append(tiempoAct)
00109     out1.append(estadoOut[0])
00110     out2.append(estadoOut[1])
00111     out3.append(estadoOut[2])
00112
00113     # Paso 4
00114    estadoOut[ordenActivacion[2]] = 0
00115    t.append(tiempoAct + tRise)
00116    out1.append(estadoOut[0])
00117    out2.append(estadoOut[1])
00118    out3.append(estadoOut[2])
00119
00120     # Pre paso 5
00121     tiempoAct += t2
00122     t.append(tiempoAct)
00123     out1.append(estadoOut[0])
00124     out2.append(estadoOut[1])
00125     out3.append(estadoOut[2])
00126
00127     # Paso 5
00128    estadoOut[ordenActivacion[1]] = 0
00129    t.append(tiempoAct + tRise)
00130    out1.append(estadoOut[0])
00131    out2.append(estadoOut[1])
00132    out3.append(estadoOut[2])
00133
00134     # Pre paso 6
00135     tiempoAct += t1
00136     t.append(tiempoAct)
00137     out1.append(estadoOut[0])
00138     out2.append(estadoOut[1])
00139     out3.append(estadoOut[2])
00140
00141     # Paso 6
00142    estadoOut[ordenActivacion[0]] = 0
00143    t.append(tiempoAct + tRise)
00144    out1.append(estadoOut[0])
00145    out2.append(estadoOut[1])
00146    out3.append(estadoOut[2])
00147
```

## 10.108 Referencia del archivo

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/SVM Space Vector

Modulation/svm\_interactivo.py

433

```
00148     tiempoAct += t0/2
00149     angAct += angSwitch
00150
00151     return t, out1, out2, out3
00152
00153
00154 def calcular_fase(t, out1, out2, out3, VBus) :
00155     faseRS = []
00156     faseST = []
00157
00158     for i in range(len(t)):
00159         faseRS.append((out1[i] - out2[i]) * VBus)
00160         faseST.append((out2[i] - out3[i]) * VBus)
00161
00162     return t, faseRS, faseST
00163
00164 def calcular_rms(signal):
00165     return np.sqrt(np.mean(np.square(signal)))
00166
00167 # Ejemplo de uso
00168 Ts = 1e-3 # 1 ms período de muestreo
00169 M = 0.8 # Índice de modulación
00170 angle_deg = 45 # Ángulo espacial
00171
00172 t, out1, out2, out3 = generar_vector_salida_temporal(Ts, M, angSwitch=1.8, tRise=1e-6, tf=0.5)
00173
00174 t2, faseRS, faseST = calcular_fase(t, out1, out2, out3, 1)
00175
00176 rms_RS = calcular_rms(faseRS)
00177 rms_ST = calcular_rms(faseST)
00178
00179 print(f"Valor RMS de fase RS: {rms_RS}")
00180 print(f"Valor RMS de fase ST: {rms_ST}")
00181
00182
00183
00184 # Plot
00185 plt.figure(figsize=(10,5))
00186
00187 # Subplot 1
00188 plt.subplot(2, 1, 1)
00189 plt.plot(t, out1, label="Out 1")
00190 plt.plot(t, out2, label="Out 2")
00191 plt.plot(t, out3, label="Out 3")
00192 plt.title("Forma de onda de SVM")
00193 plt.xlabel("Tiempo [s]")
00194 plt.ylabel("Valor logico de la salida")
00195 plt.grid(True)
00196 plt.legend()
00197
00198 # Subplot 2
00199 plt.subplot(2, 1, 2)
00200 plt.plot(t2, fasersRS, label="Dif R-S")
00201 plt.plot(t2, fasest, label="Dif S-T")
00202 plt.title("Valor de fase")
00203 plt.xlabel("Tiempo [s]")
00204 plt.ylabel("Valor lógico")
00205 plt.grid(True)
00206 plt.legend()
00207
00208 # Ajuste final
00209 plt.tight_layout()
00210 plt.show()
```

## 10.108. Referencia del archivo C:/Users/User/Desktop/ProyectoFinal- Grupo5-VariadorDeFrecuencia/Software/SVM Space Vector Modulation/svm\_interactivo.py

### Espacios de nombres

- namespace [svm\\_interactivo](#)

### Funciones

- [calculate\\_modulation\\_index \(angle\)](#)
- [update\\_plot \(angle\\_slider\)](#)

## Variables

- `root = tk.Tk()`
- `fig`
- `ax`
- `figsize`
- `angles = np.linspace(0, 360, 100)`
- `list modulation_indices = [calculate_modulation_index(a) for a in angles]`
- `label`
- `canvas = FigureCanvasTkAgg(fig, master=root)`
- `canvas_widget = canvas.get_tk_widget()`
- `angle_slider = tk.Scale(root, from_=0, to=360, orient="horizontal", label="Ángulo", command=lambda val: update_plot(angle_slider))`

## 10.109. svm\_interactivo.py

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

```

00001 import numpy as np
00002 import matplotlib.pyplot as plt
00003 from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
00004 import tkinter as tk
00005
00006 # Función para calcular el índice de modulación
00007 def calculate_modulation_index(angle):
00008     angle_rad = np.radians(angle)
00009     return np.sin(angle_rad) * np.sqrt(3)
00010
00011 # Función para actualizar el gráfico
00012 def update_plot(angle_slider):
00013     angle = float(angle_slider.get())
00014     mod_index = calculate_modulation_index(angle)
00015
00016     # Actualizar datos del gráfico
00017     ax.clear()
00018     ax.plot(angles, modulation_indices, label="Modulación")
00019     ax.scatter([angle], [mod_index], color='red', label=f"Índice = {mod_index:.2f}")
00020     ax.set_xlabel("Ángulo (grados)")
00021     ax.set_ylabel("Índice de Modulación")
00022     ax.set_title("Modulación SVM vs Ángulo")
00023     ax.legend()
00024     canvas.draw()
00025
00026 # Crear ventana principal
00027 root = tk.Tk()
00028 root.title("Modulación SVM")
00029
00030 # Crear una figura de Matplotlib
00031 fig, ax = plt.subplots(figsize=(6, 4))
00032 angles = np.linspace(0, 360, 100)
00033 modulation_indices = [calculate_modulation_index(a) for a in angles]
00034 ax.plot(angles, modulation_indices, label="Modulación")
00035 ax.set_xlabel("Ángulo (grados)")
00036 ax.set_ylabel("Índice de Modulación")
00037 ax.set_title("Modulación SVM vs Ángulo")
00038 ax.legend()
00039
00040 canvas = FigureCanvasTkAgg(fig, master=root)
00041 canvas_widget = canvas.get_tk_widget()
00042 canvas_widget.pack()
00043
00044 # Crear un slider para cambiar el ángulo
00045 angle_slider = tk.Scale(root, from_=0, to=360, orient="horizontal", label="Ángulo", command=lambda val: update_plot(angle_slider))
00046 angle_slider.pack()
00047
00048 # Iniciar la interfaz gráfica
00049 root.mainloop()

```

**10.110. Referencia del archivo C:/Users/User/Desktop/ProyectoFinal-**

## Grupo5-VariadorDeFrecuencia/Software/SVM Space Vector

### Modulation/SVM\_SwitchAnimacion.py

**Espacios de nombres**

- namespace [SVM\\_SwitchAnimacion](#)

**Variables**

- `screen_width`
- `screen_height`
- `screen = pygame.display.set_mode((screen_width, screen_height))`
- tuple `rect_color` = (255, 0, 0)
- tuple `background_color` = (128, 128, 128)
- tuple `text_color` = (255, 255, 255)
- tuple `circle_color` = (0, 0, 0)
- tuple `vector_color` = (100, 100, 100)
- tuple `black_color` = (0,0,0)
- tuple `green_color` = (0,255,0)
- tuple `red_color` = (255,0,0)
- tuple `blue_color` = (0,0,255)
- `rect_width`
- `rect_height`
- `rect_x`
- `rect_y`
- int `rect_speed` = 5
- `font_tiempo` = pygame.font.SysFont(None, 36)
- `font_caracteristicas` = pygame.font.SysFont(None, 20)
- int `line_spacing` = 30
- str `image_folder` = "image"
- list `image_files` = [f'E{i}.png' for i in range(8)]
- list `imageSwitch` = []
- `image_path` = os.path.join(image\_folder, file\_name)
- `image` = pygame.image.load(image\_path)
- `clock` = pygame.time.Clock()
- `start_time` = pygame.time.get\_ticks()
- tuple `circle_center` = (screen\_width // 2, screen\_height // 2)
- int `center_x` = screen\_width // 2
- int `center_y` = screen\_height // 2
- int `circle_radius` = 250
- int `vector_length` = 250
- int `num_vectors` = 6
- int `timeScale` = 100000000
- int `rotFreq` = 50
- int `swFreq` = 10000
- int `forward` = 1
- int `angSw` = rotFreq \* 360 / swFreq
- list `text_lines`
- list `comutacion_lineas` = []
- int `proxTiempoTimer` = 0
- int `ultTiempoTimer` = 0

```

■ int ang = 0
■ int sector = 0
■ int t0 = 0
■ int t1 = 0
■ int t2 = 0
■ str secuenciaLabel = ""
■ int numImageSwitch = 0
■ tuple elapsed_time = (pygame.time.get_ticks() - start_time)
■ microsTime = int(elapsed_time/timeScale * 1000000)
■ str time_text = f"Tiempo: {microsTime}us"
■ text_surface = font_tiempo.render(time_text, True, text_color)
■ tuple angle = (360 / num_vectors) * i
■ angle_rad = math.radians(angle)
■ tuple end_x = circle_center[0] + vector_length * math.cos(angle_rad)
■ tuple end_y = circle_center[1] + vector_length * math.sin(angle_rad)
■ int angMag = 250
■ int duty0 = 80
■ int duty1 = 60
■ int duty2 = 40
■ tuple tiempoEnCiclo = (microsTime - ultTiempoTimer) / 1000000
■ rendered_text = font_caracteristicas.render(f"Sector {sector}, Secuencia:", True, text_color)
■ width

```

## 10.111. SVM\_SwitchAnimacion.py

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

```

00001 import pygame
00002 import sys
00003 import os
00004 import math
00005 import numpy as np
00006 from SVM_Calculos import getSector, get_time_vector, getSecuenciaLabel, getSecuencia
00007
00008 #ejecutar con python prueba.py
00009
00010 # Inicializar pygame
00011 pygame.init()
00012
00013 # Configurar pantalla
00014 screen_width, screen_height = 800, 600
00015 screen = pygame.display.set_mode((screen_width, screen_height))
00016 pygame.display.set_caption("Modulacion de vector espacial")
00017
00018 # Colores
00019 rect_color = (255, 0, 0)
00020 background_color = (128, 128, 128)
00021 text_color = (255, 255, 255)
00022 circle_color = (0, 0, 0)
00023 vector_color = (100, 100, 100)
00024
00025 black_color = (0,0,0)
00026 green_color = (0,255,0)
00027 red_color = (255,0,0)
00028 blue_color = (0,0,255)
00029
00030 # Posicion del rectangulo
00031 rect_width, rect_height = 50, 50
00032 rect_x, rect_y = 100, 100
00033 rect_speed = 5
00034
00035 # Fuente para el texto
00036 font_tiempo = pygame.font.SysFont(None, 36)
00037 font_caracteristicas = pygame.font.SysFont(None, 20)
00038 line_spacing = 30
00039
00040 # Cargamos las imagenes
00041 image_folder = "image" # Carpeta donde estan las imagenes

```

```
00042 image_files = [f"E{i}.png" for i in range(8)] # Lista de nombres de archivos (E0.png, E1.png, ..., E7.png)
00043 # Cargar y escalar las imágenes
00044 imageSwitch = []
00045 for file_name in image_files:
00046     image_path = os.path.join(image_folder, file_name) # Crear la ruta completa
00047     try:
00048         image = pygame.image.load(image_path) # Cargar la imagen
00049         image = pygame.transform.scale(image, (110, 125)) # Escalar la imagen (ajusta el tamaño si es necesario)
00050         imageSwitch.append(image) # Agregar la imagen al vector
00051     except pygame.error as e:
00052         print(f"Error al cargar la imagen {file_name}: {e}")
00053
00054
00055 # Reloj para controlar FPS y calcular tiempo transcurrido
00056 clock = pygame.time.Clock()
00057 start_time = pygame.time.get_ticks()
00058
00059 # Posición del círculo
00060 circle_center = (screen_width // 2, screen_height // 2)
00061 center_x = screen_width // 2
00062 center_y = screen_height // 2
00063 circle_radius = 250
00064 vector_length = 250
00065 num_vectors = 6
00066
00067
00068
00069
00070
00071
00072 # Variables
00073 timeScale = 100000000 # Esta nos sirve para relentizar la animacion
00074 rotFreq = 50 # Esta es la frecuencia de rotacion del vector de tension
00075 swFreq = 10000 # Esta es la frecuencia de la conmutacion
00076 forward = 1 # Indica la direccion de rotacion, 1 para sentido horario visto del lado del eje del rotor """
00077
00078
00079 # Esto nos indica que angulo nos va a llevar un periodo de conmutacion
00080 angSw = rotFreq * 360 / swFreq
00081
00082
00083
00084 # Las caracteristicas de nuestro controlador
00085 text_lines = [
00086     f"Ang conmutacion: {angSw}",
00087     f"Frecuencia rotor: {rotFreq}",
00088     f"Frecuencia switch: {swFreq}"
00089 ]
00090
00091 # Variable donde guardamos nuestro orden de conmutacion
00092 conmutacion_lineas = []
00093
00094 # Variables globales
00095 proxTiempoTimer = 0
00096 ultTiempoTimer = 0
00097 ang = 0
00098 sector = 0
00099 t0 = 0
00100 t1 = 0
00101 t2 = 0
00102 secuenciaLabel = ""
00103 numImageSwitch = 0
00104
00105 # Bucle principal
00106 clock = pygame.time.Clock()
00107 while True:
00108     for event in pygame.event.get():
00109         if event.type == pygame.QUIT:
00110             pygame.quit()
00111             sys.exit()
00112
00113     # Calcular tiempo transcurrido en segundos
00114     elapsed_time = (pygame.time.get_ticks() - start_time)
00115     microsTime = int(elapsed_time/timeScale * 1000000)
00116     time_text = f"Tiempo: {microsTime}us"
00117     text_surface = font_tiempo.render(time_text, True, text_color)
00118
00119     # Rellenar el fondo con gris
00120     screen.fill(background_color)
00121
00122
00123     # Dibujar el círculo en el centro de la pantalla
00124     pygame.draw.circle(screen, circle_color, circle_center, circle_radius, 2)
00125     # Dibujar los vectores que parten desde el centro
```

```

00126     for i in range(num_vectors):
00127         angle = (360 / num_vectors) * i # Ángulo en grados
00128         angle_rad = math.radians(angle) # Convertir a radianes
00129         end_x = circle_center[0] + vector_length * math.cos(angle_rad)
00130         end_y = circle_center[1] + vector_length * math.sin(angle_rad)
00131         pygame.draw.line(screen, vector_color, circle_center, (end_x, end_y), 2)
00132
00133
00134
00135
00136
00137     angMag = 250
00138
00139     duty0 = 80
00140     duty1 = 60
00141     duty2 = 40
00142
00143
00144     # Vamos a simular un timer aca que arranca y termina en un periodo de conmutacion
00145     if(microsTime > proxTiempoTimer):
00146         ang = ((elapsed_time/timeScale) * 360 * rotFreq) % 360
00147         sector = getSector(ang)
00148         t0, t1, t2 = get_time_vector(1/swFreq, 1, ang)
00149         secuenciaLabel = getSecuenciaLabel(sector)
00150
00151         #print(f"SwitchTime: {1/swFreq}, T0: {t0}, T1: {t1}, T2: {t2}")
00152         #print(f"T1: {t0/2}, T2: {t0/2+t1}, T3: {t0/2+t1+t2}, T4: {t0+t1+t2}, T5: {3*t0/2+t1+t2}, T6:
00153         {3*t0/2+t1+t2*2}, T7: {3*t0/2+2*t1+2*t2}, T8: {2*t0+2*t1+2*t2}"")
00154
00155         # Aca tenemos que chequear de como sincronizar nuestro timer con el pwm
00156         proxTiempoTimer = microsTime + (1/swFreq) * 1000000      # Ambos deben estar en microsegundos
00157         ultTiempoTimer = microsTime
00158
00159         # Esto lo deberia hacer el PWM
00160         tiempoEnCiclo = (microsTime - ultTiempoTimer) / 1000000    # Este es el tiempo en que estoy del
00161         #print(f"tiempoEnCiclo: {tiempoEnCiclo}")
00162         #print(f"T1: {t0/2}, T2: {t0/2+t1}, T3: {t0/2+t1+t2}, T4: {t0+t1+t2}, T5: {3*t0/2+t1+t2}, T6:
00163         {3*t0/2+t1+t2*2}, T7: {3*t0/2+2*t1+2*t2}, T8: {2*t0+2*t1+2*t2}"")
00164         if(tiempoEnCiclo < t0/2) :
00165             numImageSwitch = getSecuencia(sector)[0]
00166             pygame.draw.circle(screen, green_color, circle_center, 10, 4)
00167         elif(tiempoEnCiclo < (t0/2+t1)) :
00168             numImageSwitch = getSecuencia(sector)[1]
00169             pygame.draw.line(screen, blue_color, circle_center, (angMag * np.cos(np.radians((sector-1) *
60)) + center_x, angMag * np.sin(np.radians((sector-1) * 60)) + center_y), 4)
00170         elif(tiempoEnCiclo < (t0/2+t1+t2)) :
00171             numImageSwitch = getSecuencia(sector)[2]
00172             pygame.draw.line(screen, red_color, circle_center, (angMag * np.cos(np.radians((sector) * 60)) +
center_x, angMag * np.sin(np.radians((sector) * 60)) + center_y), 4)
00173         elif(tiempoEnCiclo < (t0+t1+t2)) :
00174             numImageSwitch = getSecuencia(sector)[3]
00175             pygame.draw.circle(screen, green_color, circle_center, 10, 4)
00176         elif(tiempoEnCiclo < (3*t0/2+t1+t2)) :
00177             numImageSwitch = getSecuencia(sector)[4]
00178             pygame.draw.circle(screen, green_color, circle_center, 10, 4)
00179         elif(tiempoEnCiclo < (3*t0/2+t1+t2*2)) :
00180             numImageSwitch = getSecuencia(sector)[5]
00181             pygame.draw.line(screen, red_color, circle_center, (angMag * np.cos(np.radians((sector) * 60)) +
center_x, angMag * np.sin(np.radians((sector) * 60)) + center_y), 4)
00182         elif(tiempoEnCiclo < (3*t0/2+2*t1+2*t2)) :
00183             numImageSwitch = getSecuencia(sector)[6]
00184             pygame.draw.line(screen, blue_color, circle_center, (angMag * np.cos(np.radians((sector-1) *
60)) + center_x, angMag * np.sin(np.radians((sector-1) * 60)) + center_y), 4)
00185         elif(tiempoEnCiclo < (2*t0+2*t1+2*t2)) :
00186             numImageSwitch = getSecuencia(sector)[7]
00187             pygame.draw.circle(screen, green_color, circle_center, 10, 4)
00188
00189         # Dibujar el sector
00190         rendered_text = font_caracteristicas.render(f"Sector {sector}, Secuencia:", True, text_color)
00191         screen.blit(rendered_text, (10, 100))
00192         rendered_text = font_caracteristicas.render(secuenciaLabel, True, text_color)
00193         screen.blit(rendered_text, (10, 130))
00194
00195         # Dibujar secuencia
00196         screen.blit(imageSwitch[numImageSwitch], (20, 450))
00197
00198         # Dibujar T1 y T2
00199         rendered_text = font_caracteristicas.render(f"T0: {t0*swFreq* 100:.1f} %", True, text_color)
00200         screen.blit(rendered_text, (20, 150))
00201         pygame.draw.rect(screen, black_color, (28, 168, 44, 24), width=2)
00202         pygame.draw.rect(screen, red_color, (30, 170, t0*swFreq * 40, 20))
00203         rendered_text = font_caracteristicas.render(f"T1: {t1*swFreq* 100:.1f} %", True, text_color)
00204         screen.blit(rendered_text, (20, 200))
00205         pygame.draw.rect(screen, black_color, (28, 218, 44, 24), width=2)
00206         pygame.draw.rect(screen, red_color, (30, 220, t1*swFreq * 40, 20))

```

## 10.112 Referencia del archivo

C:/Users/User/Desktop/ProyectoFinal-Grupo5-VariadorDeFrecuencia/Software/SVM Space Vector

Modulation/SVM\_SwitchTime.py

439

```
00206     rendered_text = font_caracteristicas.render(f"T2: {t2*swFreq* 100:.1f} %", True, text_color)
00207     screen.blit(rendered_text, (20, 250))
00208     pygame.draw.rect(screen, black_color, (28, 268, 44, 24), width=2)
00209     pygame.draw.rect(screen, red_color, (30, 270, t2*swFreq * 40, 20))
00210
00211
00212     # Dibujar el vector de la magnitud de tension
00213     pygame.draw.line(screen, black_color, circle_center, (angMag * np.cos(np.radians(ang)) + center_x,
00214     angMag * np.sin(np.radians(ang)) + center_y), 2)
00215
00216
00217     # Dibujar el texto en la parte superior
00218     screen.blit(text_surface, (10, 10))
00219
00220
00221     # Dibujamos el periodo de conmutacion
00222     for i, line in enumerate(text_lines):
00223         rendered_text = font_caracteristicas.render(line, True, text_color)
00224         screen.blit(rendered_text, (600, 10 + i * line_spacing))
00225
00226     # Actualizar pantalla
00227     pygame.display.flip()
00228     clock.tick(60) # Limitar a 60 FPS
```

## 10.112. Referencia del archivo C:/Users/User/Desktop/ProyectoFinal- Grupo5-VariadorDeFrecuencia/Software/SVM Space Vector Modulation/SVM\_SwitchTime.py

### Espacios de nombres

- namespace [SVM\\_SwitchTime](#)

### Funciones

- [update\\_rectangle](#) (new\_x, new\_y)

### Variables

- [image](#) = np.random.random((100, 100))
- [fig](#)
- [ax](#)
- [im](#) = ax.imshow([image](#), cmap="gray")
- [rect](#) = patches.Rectangle((30, 30), 20, 20, linewidth=2, edgecolor='red', facecolor='none')
- [random\\_x](#) = random.randint(0, 80)
- [random\\_y](#) = random.randint(0, 80)

## 10.113. SVM\_SwitchTime.py

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

```
00001 import matplotlib.pyplot as plt
00002 import matplotlib.patches as patches
00003 import numpy as np
00004 import time
00005 import random
00006
00007 # Crear una imagen aleatoria como fondo
00008 image = np.random.random((100, 100))
00009
00010 # Crear la figura y el eje
```

```

00011 fig, ax = plt.subplots()
00012 im = ax.imshow(image, cmap="gray")
00013
00014 # Dibujar el rectángulo rojo
00015 rect = patches.Rectangle((30, 30), 20, 20, linewidth=2, edgecolor='red', facecolor='none')
00016 ax.add_patch(rect)
00017
00018 # Función para actualizar la posición del rectángulo
00019 def update_rectangle(new_x, new_y):
00020     rect.set_xy((new_x, new_y))
00021     fig.canvas.draw()
00022     fig.canvas.flush_events()
00023
00024 # Activar modo interactivo
00025 plt.ion()
00026 plt.show()
00027
00028 # Mover el rectángulo en un bucle
00029 for i in range(1000):
00030     time.sleep(0.01) # Esperar 0.5 segundos entre cada movimiento
00031     random_x = random.randint(0, 80) # Generar coordenada X aleatoria (ajustar a tus límites)
00032     random_y = random.randint(0, 80) # Generar coordenada Y aleatoria (ajustar a tus límites)
00033     update_rectangle(random_x, random_y) # Actualizar posición del rectángulo
00034
00035 plt.ioff()
00036 plt.show()

```

## 10.114. Referencia del archivo C:/Users/User/Desktop/ProyectoFinal- Grupo5-VariadorDeFrecuencia/Software/SVM Space Vector Modulation/svm\_v2.py

### Espacios de nombres

- namespace [svm\\_v2](#)

### Funciones

- [calculate\\_modulation\\_index \(angle\)](#)
- [update\\_plot1 \(angle\\_slider\)](#)
- [update\\_plot2 \(angle\\_slider\)](#)
- [update\\_plot \(angle\\_slider\)](#)
- [animate \(\)](#)

### Variables

- [root = tk.Tk\(\)](#)
- [fig](#)
- [polar\\_ax](#)
- [subplot\\_kw](#)
- [figsize](#)
- [va](#)
- list [vectors = \[\]](#)
- int [current\\_angle = 0](#)
- [canvas = FigureCanvasTkAgg\(fig, master=root\)](#)
- [canvas\\_widget = canvas.get\\_tk\\_widget\(\)](#)
- [angle\\_slider = tk.Scale\(root, from\\_=0, to=360, orient="horizontal", label="Ángulo", command=lambda val : update\\_plot\(angle\\_slider\)\)](#)

## 10.115. svm\_v2.py

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

```
00001 import numpy as np
00002 import matplotlib.pyplot as plt
00003 from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
00004 import tkinter as tk
00005
00006 # Función para calcular el índice de modulación
00007 def calculate_modulation_index(angle):
00008     angle_rad = np.radians(angle)
00009     return np.sin(angle_rad) * np.sqrt(3)
00010
00011 # Función para actualizar el gráfico polar
00012 def update_plot1(angle_slider):
00013     angle = float(angle_slider.get())
00014     mod_index = calculate_modulation_index(angle)
00015
00016     # Actualizar gráfico polar
00017     polar_ax.clear()
00018     polar_ax.set_theta_zero_location("N")
00019     polar_ax.set_theta_direction(-1)
00020     polar_ax.set_rticks([0.5, 1, 1.5, 2]) # Marcas radiales
00021     polar_ax.grid(True)
00022
00023     # Agregar vectores básicos (representan V1, V2, V3, etc.)
00024     #vectors_angles = [0, 120, 240]
00025     #for v_angle in vectors_angles:
00026         # polar_ax.quiver(np.radians(v_angle), 0, 0, 1, angles='xy', scale_units='xy', scale=1,
00027         # color='gray', alpha=0.5)
00028
00029     # Agregar vector resultante
00030     polar_ax.quiver(np.radians(angle), 0, 0, mod_index, angles='xy', scale_units='xy', scale=1,
00031     color='red')
00032     polar_ax.set_title("Gráfico Polar: Modulación SVM", va='bottom')
00033     canvas.draw()
00034
00035 def update_plot2(angle_slider):
00036     angle = float(angle_slider.get())
00037     mod_index = calculate_modulation_index(angle)
00038
00039     # Eliminar solo los vectores para redibujarlos
00040     for artist in polar_ax.lines + polar_ax.collections:
00041         artist.remove()
00042
00043     # Agregar vectores básicos (representan V1, V2, V3, etc.)
00044     vectors_angles = [0, 120, 240]
00045     for v_angle in vectors_angles:
00046         polar_ax.quiver(np.radians(v_angle), 0, 0, 1, angles='xy', scale_units='xy', scale=1,
00047         color='gray', alpha=0.5)
00048
00049     # Agregar vector resultante
00050     polar_ax.quiver(np.radians(angle), 0, 0, mod_index, angles='xy', scale_units='xy', scale=1,
00051     color='red')
00052     canvas.draw()
00053
00054 def update_plot(angle_slider):
00055     angle = float(angle_slider.get())
00056     global current_angle
00057     current_angle = angle
00058     mod_index = calculate_modulation_index(angle)
00059
00060     # Eliminar solo los vectores existentes (sin borrar el resto del gráfico)
00061     for line in vectors:
00062         line.remove()
00063     vectors.clear()
00064
00065     # Dibujar los nuevos vectores
00066     vector1 = polar_ax.quiver(0, 0, 1, 0, angles='xy', scale_units='xy', scale=1, color='blue',
00067     alpha=0.7)
00068     vector2 = polar_ax.quiver(2 * np.pi / 3, 0, 1, 0, angles='xy', scale_units='xy', scale=1,
00069     color='green', alpha=0.7)
00070     result_vector = polar_ax.quiver(np.radians(angle), 0, mod_index, 0, angles='xy', scale_units='xy',
00071     scale=1, color='red')
00072
00073     # Actualizar lista de vectores
00074     vectors.extend([vector1, vector2, result_vector])
00075
00076     polar_ax.set_theta_direction(-1)
00077     polar_ax.set_rticks([0.5, 1, 1.5, 2])
00078     polar_ax.grid(True)
00079
00080     canvas.draw()
```

```
00076
00077
00078 # Función para incrementar el ángulo automáticamente
00079 def animate():
00080     global current_angle
00081     current_angle = (current_angle + 1) % 360 # Incrementar el ángulo, reiniciando a 0 después de 359
00082     update_plot(current_angle) # Redibujar el gráfico
00083     root.after(100, animate) # Llamar a esta función nuevamente después de 100 ms
00084
00085
00086
00087
00088
00089 # Crear ventana principal
00090 root = tk.Tk()
00091 root.title("Modulación SVM con Gráfico Polar")
00092
00093 # Crear figura de Matplotlib para el gráfico polar
00094 fig, polar_ax = plt.subplots(subplot_kw={'projection': 'polar'}, figsize=(6, 6))
00095 polar_ax.set_theta_zero_location("N")
00096 polar_ax.set_theta_direction(-1)
00097 polar_ax.set_rticks([0.5, 1, 1.5, 2])
00098 polar_ax.grid(True)
00099 polar_ax.set_title("Gráfico Polar: Modulación SVM", va='bottom')
00100
00101
00102 # Lista para almacenar los vectores dibujados
00103 vectors = []
00104 current_angle = 0 # Ángulo inicial
00105
00106 canvas = FigureCanvasTkAgg(fig, master=root)
00107 canvas_widget = canvas.get_tk_widget()
00108 canvas_widget.pack()
00109
00110 # Crear un slider para cambiar el ángulo
00111 angle_slider = tk.Scale(root, from_=0, to=360, orient="horizontal", label="Ángulo", command=lambda
00112     val: update_plot(angle_slider))
00113 angle_slider.pack()
00114
00115 # Iniciar la animación
00116 animate()
00117
00118 # Iniciar la interfaz gráfica
root.mainloop()
```