Low Voltage Variable Frequency Drive

2.1

Generado por Doxygen 1.15.0

Capítulo 1

Jerarquía de directorios

1.1. Directorios

Capítulo 2

Índice de estructuras de datos

2.1. Estructuras de datos

Lista de estructuras con breves descripciones:

Capítulo 3

Índice de archivos

3.1. Lista de archivos

Lista de todos los archivos con breves descripciones:

Capítulo 4

Documentación de directorios

4.1. Referencia del directorio main/adc

Archivos

- archivo adc.c

  Funcion de inicialización y tarea lectura del ADC.
- archivo adc.h

  Declraración de funcion de inicialización y tarea lectura del ADC.

4.2. Referencia del directorio main/display

Archivos

- archivo display.c

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

  Declaración de funciones que de consulta de los seteos hechos por el usuario, tarea que controla el display y posteo de eventos para controlar el display.
- archivo sh1106_graphics.c

  Funciones que permiten operar sobre el display.
- archivo sh1106_graphics.h

  Declaración de funciones que permiten operar sobre el display.
- archivo sh1106_i2c.c

  Funciones de hardware para el puerto I2C.
- archivo sh1106_i2c.h

  Declaración de funciones de hardware para el puerto I2C.

4.3. Referencia del directorio main/io_control

Archivos

- archivo io_control.c

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

  Declaración de funciones de control del relay, buzzer y tarea del uso del expansor de entradas y salidas y entradas y salidas aisladas directas al ESP32.
- archivo MCP23017.c

  Declaración de funciones de inicialización, lectura y escritura a través del puerto I2C el dispositivo MCP23017.
- archivo MCP23017.h

  Declaración de funciones de inicialización, lectura y escritura a través del puerto I2C el dispositivo MCP23017.
- archivo mcp23017_defs.h

4.4. Referencia del directorio main

Directorios

- directorio adc
- directorio display
- directorio io_control
- directorio nvs
- directorio rtc
- directorio system
- directorio wifi

Archivos

- archivo LVFV_system.h

  Declaraciones de estructuras, variables, definiciones y estados generales del sistema utilizados en varios de los archivos.
- archivo main.c

  Inicialización de tareas y periféricos que no tienen una tarea específica. El resto de los periféricos se inicializan en las tareas que son utilizados.

4.5. Referencia del directorio main/nvs

Archivos

- archivo nvs.c

  Funciones de lectura y escritura de variables no volátiles. Carga en sistema además las variables de seguridad al momento de ser guardadas en la memoria.
- archivo nvs.h

  Declaración de funciones de lectura y escritura de memoria no volatil. Útiles para los reset y no tener que reconfigurar siempre las variables básicas del sistema.

4.6. Referencia del directorio main/rtc

Archivos

- archivo rtc.c

  Funciones de lectura y escritura del RTC y alarmas.
- archivo rtc.h

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

4.7. Referencia del directorio main/system

Archivos

- archivo sysAdmin.c

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

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

4.8. Referencia del directorio main/wifi

Archivos

- archivo form.c
- archivo form.h
- archivo wifi.c
- archivo wifi.h

Capítulo 5

Documentación de estructuras de datos

5.1. Referencia de la estructura bitmap_t

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

Campos de datos

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

5.1.1. Descripción detallada

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

Definición en la línea 176 del archivo sh1106_graphics.c.

5.1.2. Documentación de campos

5.1.2.1. bitmap

```
const uint8_t* bitmap
```

Puntero al caracter

Definición en la línea 177 del archivo sh1106_graphics.c.

5.1.2.2. h

```
uint8_t h
```

Alto del caracter

Definición en la línea 179 del archivo sh1106_graphics.c.

5.1.2.3. w

```
uint8_t w
```

Ancho del caracter

Definición en la línea 178 del archivo sh1106_graphics.c.

5.1.2.4. x

```
int x
```

Posición en X del caracter

Definición en la línea 180 del archivo sh1106_graphics.c.

5.1.2.5. y

```
int y
```

Posición en Y del caracter

Definición en la línea 181 del archivo sh1106_graphics.c.

La documentación de esta estructura está generada del siguiente archivo:

- main/display/sh1106_graphics.c

5.2. Referencia de la estructura frequency_settings_SH1106_t

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

Campos de datos

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

5.2.1. Descripción detallada

Definición en la línea 127 del archivo LVFV_system.h.

5.2.2. Documentación de campos

5.2.2.1. edit

```
sh1106_variable_lines_e* edit
```

Definición en la línea 129 del archivo LVFV_system.h.

5.2.2.2. edit_flag

```
uint8_t* edit_flag
```

Definición en la línea 131 del archivo LVFV_system.h.

5.2.2.3. edit_variable

```
uint8_t edit_variable
```

Definición en la línea 130 del archivo LVFV_system.h.

5.2.2.4. frequency_settings

```
frequency_settings_t frequency_settings
```

Definición en la línea 128 del archivo LVFV_system.h.

5.2.2.5. multiplier

```
uint8_t* multiplier
```

Definición en la línea 132 del archivo LVFV_system.h.

La documentación de esta estructura está generada del siguiente archivo:

- main/LVFV_system.h

5.3. Referencia de la estructura frequency_settings_t

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

Campos de datos

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

5.3.1. Descripción detallada

Definición en la línea 86 del archivo LVFV_system.h.

5.3.2. Documentación de campos

5.3.2.1. acceleration

```
uint16_t acceleration
```

Definición en la línea 88 del archivo LVFV_system.h.

5.3.2.2. desacceleration

```
uint16_t desacceleration
```

Definición en la línea 89 del archivo LVFV_system.h.

5.3.2.3. freq_regime

```
uint16_t freq_regime
```

Definición en la línea 87 del archivo LVFV_system.h.

5.3.2.4. input_variable

```
uint16_t input_variable
```

Definición en la línea 90 del archivo LVFV_system.h.

La documentación de esta estructura está generada del siguiente archivo:

- main/LVFV_system.h

5.4. Referencia de la unión MCP23017_DEFVAL_t

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

Campos de datos

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

5.4.1. Descripción detallada

Definición en la línea 237 del archivo mcp23017_defs.h.

5.4.2. Documentación de campos

5.4.2.1. all

```
uint8_t all
```

Unificación de la vaviable

Definición en la línea 238 del archivo mcp23017_defs.h.

5.4.2.2. [struct]

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

5.4.2.3. DEF0

```
uint8_t DEF0
```

Bit 0 de DEFVAL

Definición en la línea 240 del archivo mcp23017_defs.h.

5.4.2.4. DEF1

```
uint8_t DEF1
```

Bit 1 de DEFVAL

Definición en la línea 241 del archivo mcp23017_defs.h.

5.4.2.5. DEF2

```
uint8_t DEF2
```

Bit 2 de DEFVAL

Definición en la línea 242 del archivo mcp23017_defs.h.

5.4.2.6. DEF3

```
uint8_t DEF3
```

Bit 3 de DEFVAL

Definición en la línea 243 del archivo mcp23017_defs.h.

5.4.2.7. DEF4

```
uint8_t DEF4
```

Bit 4 de DEFVAL

Definición en la línea 244 del archivo mcp23017_defs.h.

5.4.2.8. DEF5

```
uint8_t DEF5
```

Bit 5 de DEFVAL

Definición en la línea 245 del archivo mcp23017_defs.h.

5.4.2.9. DEF6

```
uint8_t DEF6
```

Bit 6 de DEFVAL

Definición en la línea 246 del archivo mcp23017_defs.h.

5.4.2.10. DEF7

```
uint8_t DEF7
```

Bit 7 de DEFVAL

Definición en la línea 247 del archivo mcp23017_defs.h.

La documentación de esta unión está generada del siguiente archivo:

- main/io_control/mcp23017_defs.h

5.5. Referencia de la unión MCP23017_GPINTEN_t

`#include <mcp23017_defs.h>`

Campos de datos

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

5.5.1. Descripción detallada

Definición en la línea 153 del archivo mcp23017_defs.h.

5.5.2. Documentación de campos

5.5.2.1. all

`uint8_t all`

Unificación de la vaviable

Definición en la línea 154 del archivo mcp23017_defs.h.

5.5.2.2. [struct]

`struct { ... } bits`

5.5.2.3. GPINT0

`uint8_t GPINT0`

Bit 0 de GPINTEN

Definición en la línea 156 del archivo mcp23017_defs.h.

5.5.2.4. GPINT1

```
uint8_t GPINT1
```

Bit 1 de GPINTEN

Definición en la línea 157 del archivo mcp23017_defs.h.

5.5.2.5. GPINT2

```
uint8_t GPINT2
```

Bit 2 de GPINTEN

Definición en la línea 158 del archivo mcp23017_defs.h.

5.5.2.6. GPINT3

```
uint8_t GPINT3
```

Bit 3 de GPINTEN

Definición en la línea 159 del archivo mcp23017_defs.h.

5.5.2.7. GPINT4

```
uint8_t GPINT4
```

Bit 4 de GPINTEN

Definición en la línea 160 del archivo mcp23017_defs.h.

5.5.2.8. GPINT5

```
uint8_t GPINT5
```

Bit 5 de GPINTEN

Definición en la línea 161 del archivo mcp23017_defs.h.

5.5.2.9. GPINT6

```
uint8_t GPINT6
```

Bit 6 de GPINTEN

Definición en la línea 162 del archivo mcp23017_defs.h.

5.5.2.10. GPINT7

```
uint8_t GPINT7
```

Bit 7 de GPINTEN

Definición en la línea 163 del archivo mcp23017_defs.h.

La documentación de esta unión está generada del siguiente archivo:

- main/io_control/mcp23017_defs.h

5.6. Referencia de la unión MCP23017_GPIO_t

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

Campos de datos

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

5.6.1. Descripción detallada

Definición en la línea 195 del archivo mcp23017_defs.h.

5.6.2. Documentación de campos

5.6.2.1. all

```
uint8_t all
```

Unificación de la vaviable

Definición en la línea 196 del archivo mcp23017_defs.h.

5.6.2.2. [struct]

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

5.6.2.3. GP0

```
uint8_t GP0
```

Bit 0 de GPIO

Definición en la línea 198 del archivo mcp23017_defs.h.

5.6.2.4. GP1

```
uint8_t GP1
```

Bit 1 de GPIO

Definición en la línea 199 del archivo mcp23017_defs.h.

5.6.2.5. GP2

```
uint8_t GP2
```

Bit 2 de GPIO

Definición en la línea 200 del archivo mcp23017_defs.h.

5.6.2.6. GP3

```
uint8_t GP3
```

Bit 3 de GPIO

Definición en la línea 201 del archivo mcp23017_defs.h.

5.6.2.7. GP4

```
uint8_t GP4
```

Bit 4 de GPIO

Definición en la línea 202 del archivo mcp23017_defs.h.

5.6.2.8. GP5

```
uint8_t GP5
```

Bit 5 de GPIO

Definición en la línea 203 del archivo mcp23017_defs.h.

5.6.2.9. GP6

`uint8_t GP6`

Bit 6 de GPIO

Definición en la línea 204 del archivo mcp23017_defs.h.

5.6.2.10. GP7

`uint8_t GP7`

Bit 7 de GPIO

Definición en la línea 205 del archivo mcp23017_defs.h.

La documentación de esta unión está generada del siguiente archivo:

- main/io_control/mcp23017_defs.h

5.7. Referencia de la unión MCP23017_GPPU_t

`#include <mcp23017_defs.h>`

Campos de datos

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

5.7.1. Descripción detallada

Definición en la línea 174 del archivo mcp23017_defs.h.

5.7.2. Documentación de campos

5.7.2.1. all

`uint8_t all`

Unificación de la vaviable

Definición en la línea 175 del archivo mcp23017_defs.h.

5.7.2.2. [struct]

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

5.7.2.3. PU0

```
uint8_t PU0
```

Bit 0 de GPPU

Definición en la línea 177 del archivo mcp23017_defs.h.

5.7.2.4. PU1

```
uint8_t PU1
```

Bit 1 de GPPU

Definición en la línea 178 del archivo mcp23017_defs.h.

5.7.2.5. PU2

```
uint8_t PU2
```

Bit 2 de GPPU

Definición en la línea 179 del archivo mcp23017_defs.h.

5.7.2.6. PU3

```
uint8_t PU3
```

Bit 3 de GPPU

Definición en la línea 180 del archivo mcp23017_defs.h.

5.7.2.7. PU4

```
uint8_t PU4
```

Bit 4 de GPPU

Definición en la línea 181 del archivo mcp23017_defs.h.

5.7.2.8. PU5

```
uint8_t PU5
```

Bit 5 de GPPU

Definición en la línea 182 del archivo mcp23017_defs.h.

5.7.2.9. PU6

```
uint8_t PU6
```

Bit 6 de GPPU

Definición en la línea 183 del archivo mcp23017_defs.h.

5.7.2.10. PU7

```
uint8_t PU7
```

Bit 7 de GPPU

Definición en la línea 184 del archivo mcp23017_defs.h.

La documentación de esta unión está generada del siguiente archivo:

- main/io_control/mcp23017_defs.h

5.8. Referencia de la unión MCP23017_INTCAP_t

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

Campos de datos

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

5.8.1. Descripción detallada

Definición en la línea 321 del archivo mcp23017_defs.h.

5.8.2. Documentación de campos

5.8.2.1. all

```
uint8_t all
```

Unificación de la vaviable

Definición en la línea 322 del archivo mcp23017_defs.h.

5.8.2.2. [struct]

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

5.8.2.3. ICP0

```
uint8_t ICP0
```

Bit 0 de INTCAP

Definición en la línea 324 del archivo mcp23017_defs.h.

5.8.2.4. ICP1

```
uint8_t ICP1
```

Bit 1 de INTCAP

Definición en la línea 325 del archivo mcp23017_defs.h.

5.8.2.5. ICP2

```
uint8_t ICP2
```

Bit 2 de INTCAP

Definición en la línea 326 del archivo mcp23017_defs.h.

5.8.2.6. ICP3

```
uint8_t ICP3
```

Bit 3 de INTCAP

Definición en la línea 327 del archivo mcp23017_defs.h.

5.8.2.7. ICP4

`uint8_t ICP4`

Bit 4 de INTCAP

Definición en la línea 328 del archivo mcp23017_defs.h.

5.8.2.8. ICP5

`uint8_t ICP5`

Bit 5 de INTCAP

Definición en la línea 329 del archivo mcp23017_defs.h.

5.8.2.9. ICP6

`uint8_t ICP6`

Bit 6 de INTCAP

Definición en la línea 330 del archivo mcp23017_defs.h.

5.8.2.10. ICP7

`uint8_t ICP7`

Bit 7 de INTCAP

Definición en la línea 331 del archivo mcp23017_defs.h.

La documentación de esta unión está generada del siguiente archivo:

- main/io_control/mcp23017_defs.h

5.9. Referencia de la unión MCP23017_INTCON_t

`#include <mcp23017_defs.h>`

Campos de datos

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

5.9.1. Descripción detallada

Definición en la línea 258 del archivo mcp23017_defs.h.

5.9.2. Documentación de campos

5.9.2.1. all

```
uint8_t all
```

Unificación de la vaviable

Definición en la línea 259 del archivo mcp23017_defs.h.

5.9.2.2. [struct]

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

5.9.2.3. IOC0

```
uint8_t IOC0
```

Bit 0 de INTCON

Definición en la línea 261 del archivo mcp23017_defs.h.

5.9.2.4. IOC1

```
uint8_t IOC1
```

Bit 1 de INTCON

Definición en la línea 262 del archivo mcp23017_defs.h.

5.9.2.5. IOC2

```
uint8_t IOC2
```

Bit 2 de INTCON

Definición en la línea 263 del archivo mcp23017_defs.h.

5.9.2.6. IOC3

```
uint8_t IOC3
```

Bit 3 de INTCON

Definición en la línea 264 del archivo mcp23017_defs.h.

5.9.2.7. IOC4

`uint8_t IOC4`

Bit 4 de INTCON

Definición en la línea 265 del archivo mcp23017_defs.h.

5.9.2.8. IOC5

`uint8_t IOC5`

Bit 5 de INTCON

Definición en la línea 266 del archivo mcp23017_defs.h.

5.9.2.9. IOC6

`uint8_t IOC6`

Bit 6 de INTCON

Definición en la línea 267 del archivo mcp23017_defs.h.

5.9.2.10. IOC7

`uint8_t IOC7`

Bit 7 de INTCON

Definición en la línea 268 del archivo mcp23017_defs.h.

La documentación de esta unión está generada del siguiente archivo:

- main/io_control/mcp23017_defs.h

5.10. Referencia de la unión MCP23017_INTF_t

`#include <mcp23017_defs.h>`

Campos de datos

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

5.10.1. Descripción detallada

Definición en la línea 300 del archivo mcp23017_defs.h.

5.10.2. Documentación de campos

5.10.2.1. all

```
uint8_t all
```

Unificación de la vaviable

Definición en la línea 301 del archivo mcp23017_defs.h.

5.10.2.2. [struct]

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

5.10.2.3. INT0

```
uint8_t INT0
```

Bit 0 de INTF

Definición en la línea 303 del archivo mcp23017_defs.h.

5.10.2.4. INT1

```
uint8_t INT1
```

Bit 1 de INTF

Definición en la línea 304 del archivo mcp23017_defs.h.

5.10.2.5. INT2

```
uint8_t INT2
```

Bit 2 de INTF

Definición en la línea 305 del archivo mcp23017_defs.h.

5.10.2.6. INT3

```
uint8_t INT3
```

Bit 3 de INTF

Definición en la línea 306 del archivo mcp23017_defs.h.

5.10.2.7. INT4

`uint8_t INT4`

Bit 4 de INTF

Definición en la línea 307 del archivo mcp23017_defs.h.

5.10.2.8. INT5

`uint8_t INT5`

Bit 5 de INTF

Definición en la línea 308 del archivo mcp23017_defs.h.

5.10.2.9. INT6

`uint8_t INT6`

Bit 6 de INTF

Definición en la línea 309 del archivo mcp23017_defs.h.

5.10.2.10. INT7

`uint8_t INT7`

Bit 7 de INTF

Definición en la línea 310 del archivo mcp23017_defs.h.

La documentación de esta unión está generada del siguiente archivo:

- main/io_control/mcp23017_defs.h

5.11. Referencia de la unión MCP23017_IOCON_t

`#include <mcp23017_defs.h>`

Campos de datos

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

5.11.1. Descripción detallada

Definición en la línea 279 del archivo mcp23017_defs.h.

5.11.2. Documentación de campos

5.11.2.1. all

```
uint8_t all
```

Unificación de la vaviable

Definición en la línea 280 del archivo mcp23017_defs.h.

5.11.2.2. BANK

```
uint8_t BANK
```

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

Definición en la línea 289 del archivo mcp23017_defs.h.

5.11.2.3. [struct]

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

5.11.2.4. DISSLW

```
uint8_t DISSLW
```

Bit 4 de IOCON - Configuración de velocidad del MCP23017

Definición en la línea 286 del archivo mcp23017_defs.h.

5.11.2.5. INTPOL

```
uint8_t INTPOL
```

Bit 1 de IOCON - Configuración de polaridad de las entradas

Definición en la línea 283 del archivo mcp23017_defs.h.

5.11.2.6. MIRROR

```
uint8_t MIRROR
```

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

Definición en la línea 288 del archivo mcp23017_defs.h.

5.11.2.7. ODR

```
uint8_t ODR
```

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

Definición en la línea 284 del archivo mcp23017_defs.h.

5.11.2.8. reserved

```
uint8_t reserved
```

Bit 0 de IOCON - Sin uso

Definición en la línea 282 del archivo mcp23017_defs.h.

5.11.2.9. reserved_2

```
uint8_t reserved_2
```

Bit 3 de IOCON - Sin uso

Definición en la línea 285 del archivo mcp23017_defs.h.

5.11.2.10. SEQOP

```
uint8_t SEQOP
```

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

Definición en la línea 287 del archivo mcp23017_defs.h.

La documentación de esta unión está generada del siguiente archivo:

- main/io_control/mcp23017_defs.h

5.12. Referencia de la unión MCP23017_IODIR_t

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

Campos de datos

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

5.12.1. Descripción detallada

Definición en la línea 111 del archivo mcp23017_defs.h.

5.12.2. Documentación de campos

5.12.2.1. all

```
uint8_t all
```

Unificación de la vaviable

Definición en la línea 112 del archivo mcp23017_defs.h.

5.12.2.2. [struct]

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

5.12.2.3. IO0

```
uint8_t IO0
```

Bit 0 de IODIR

Definición en la línea 114 del archivo mcp23017_defs.h.

5.12.2.4. IO1

```
uint8_t IO1
```

Bit 1 de IODIR

Definición en la línea 115 del archivo mcp23017_defs.h.

5.12.2.5. IO2

```
uint8_t IO2
```

Bit 2 de IODIR

Definición en la línea 116 del archivo mcp23017_defs.h.

5.12.2.6. IO3

```
uint8_t IO3
```

Bit 3 de IODIR

Definición en la línea 117 del archivo mcp23017_defs.h.

5.12.2.7. IO4

`uint8_t IO4`

Bit 4 de IODIR

Definición en la línea 118 del archivo mcp23017_defs.h.

5.12.2.8. IO5

`uint8_t IO5`

Bit 5 de IODIR

Definición en la línea 119 del archivo mcp23017_defs.h.

5.12.2.9. IO6

`uint8_t IO6`

Bit 6 de IODIR

Definición en la línea 120 del archivo mcp23017_defs.h.

5.12.2.10. IO7

`uint8_t IO7`

Bit 7 de IODIR

Definición en la línea 121 del archivo mcp23017_defs.h.

La documentación de esta unión está generada del siguiente archivo:

- main/io_control/mcp23017_defs.h

5.13. Referencia de la unión MCP23017_IPOL_t

`#include <mcp23017_defs.h>`

Campos de datos

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

5.13.1. Descripción detallada

Definición en la línea 132 del archivo mcp23017_defs.h.

5.13.2. Documentación de campos

5.13.2.1. all

```
uint8_t all
```

Unificación de la vaviable

Definición en la línea 133 del archivo mcp23017_defs.h.

5.13.2.2. [struct]

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

5.13.2.3. IP0

```
uint8_t IP0
```

Bit 0 de IPOL

Definición en la línea 135 del archivo mcp23017_defs.h.

5.13.2.4. IP1

```
uint8_t IP1
```

Bit 1 de IPOL

Definición en la línea 136 del archivo mcp23017_defs.h.

5.13.2.5. IP2

```
uint8_t IP2
```

Bit 2 de IPOL

Definición en la línea 137 del archivo mcp23017_defs.h.

5.13.2.6. IP3

```
uint8_t IP3
```

Bit 3 de IPOL

Definición en la línea 138 del archivo mcp23017_defs.h.

5.13.2.7. IP4

`uint8_t IP4`

Bit 4 de IPOL

Definición en la línea 139 del archivo mcp23017_defs.h.

5.13.2.8. IP5

`uint8_t IP5`

Bit 5 de IPOL

Definición en la línea 140 del archivo mcp23017_defs.h.

5.13.2.9. IP6

`uint8_t IP6`

Bit 6 de IPOL

Definición en la línea 141 del archivo mcp23017_defs.h.

5.13.2.10. IP7

`uint8_t IP7`

Bit 7 de IPOL

Definición en la línea 142 del archivo mcp23017_defs.h.

La documentación de esta unión está generada del siguiente archivo:

- main/io_control/mcp23017_defs.h

5.14. Referencia de la unión MCP23017_OLAT_t

`#include <mcp23017_defs.h>`

Campos de datos

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

5.14.1. Descripción detallada

Definición en la línea 216 del archivo mcp23017_defs.h.

5.14.2. Documentación de campos

5.14.2.1. all

```
uint8_t all
```

Unificación de la vaviable

Definición en la línea 217 del archivo mcp23017_defs.h.

5.14.2.2. [struct]

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

5.14.2.3. OL0

```
uint8_t OL0
```

Bit 0 de OLAT

Definición en la línea 219 del archivo mcp23017_defs.h.

5.14.2.4. OL1

```
uint8_t OL1
```

Bit 1 de OLAT

Definición en la línea 220 del archivo mcp23017_defs.h.

5.14.2.5. OL2

```
uint8_t OL2
```

Bit 2 de OLAT

Definición en la línea 221 del archivo mcp23017_defs.h.

5.14.2.6. OL3

```
uint8_t OL3
```

Bit 3 de OLAT

Definición en la línea 222 del archivo mcp23017_defs.h.

5.14.2.7. OL4

```
uint8_t OL4
```

Bit 4 de OLAT

Definición en la línea 223 del archivo mcp23017_defs.h.

5.14.2.8. OL5

```
uint8_t OL5
```

Bit 5 de OLAT

Definición en la línea 224 del archivo mcp23017_defs.h.

5.14.2.9. OL6

```
uint8_t OL6
```

Bit 6 de OLAT

Definición en la línea 225 del archivo mcp23017_defs.h.

5.14.2.10. OL7

```
uint8_t OL7
```

Bit 7 de OLAT

Definición en la línea 226 del archivo mcp23017_defs.h.

La documentación de esta unión está generada del siguiente archivo:

- main/io_control/mcp23017_defs.h

5.15. Referencia de la estructura rtc_alarms_t

Campos de datos

- esp_timer_handle_t start_timer
- esp_timer_handle_t stop_timer

5.15.1. Descripción detallada

Definición en la línea 24 del archivo rtc.c.

5.15.2. Documentación de campos

5.15.2.1. start_timer

```
esp_timer_handle_t start_timer
```

Definición en la línea 25 del archivo rtc.c.

5.15.2.2. stop_timer

```
esp_timer_handle_t stop_timer
```

Definición en la línea 26 del archivo rtc.c.

La documentación de esta estructura está generada del siguiente archivo:

- main/rtc/rtc.c

5.16. Referencia de la estructura seccurity_settings_SH1106_t

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

Campos de datos

- seccurity_settings_t seccurity_settings
- sh1106_variable_lines_e * edit
- uint8_t edit_variable
- uint8_t * edit_flag
- uint8_t * multiplier

5.16.1. Descripción detallada

Definición en la línea 143 del archivo LVFV_system.h.

5.16.2. Documentación de campos

5.16.2.1. edit

```
sh1106_variable_lines_e* edit
```

Definición en la línea 145 del archivo LVFV_system.h.

5.16.2.2. edit_flag

```
uint8_t* edit_flag
```

Definición en la línea 147 del archivo LVFV_system.h.

5.16.2.3. edit_variable

```
uint8_t edit_variable
```

Definición en la línea 146 del archivo LVFV_system.h.

5.16.2.4. multiplier

```
uint8_t* multiplier
```

Definición en la línea 148 del archivo LVFV_system.h.

5.16.2.5. seccurity_settings

```
seccurity_settings_t seccurity_settings
```

Definición en la línea 144 del archivo LVFV_system.h.

La documentación de esta estructura está generada del siguiente archivo:

- main/LVFV_system.h

5.17. Referencia de la estructura seccurity_settings_t

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

Campos de datos

- uint16_t vbus_min
- uint16_t ibus_max

5.17.1. Descripción detallada

Definición en la línea 99 del archivo LVFV_system.h.

5.17.2. Documentación de campos

5.17.2.1. ibus_max

```
uint16_t ibus_max
```

Definición en la línea 101 del archivo LVFV_system.h.

5.17.2.2. vbus_min

```
uint16_t vbus_min
```

Definición en la línea 100 del archivo LVFV_system.h.

La documentación de esta estructura está generada del siguiente archivo:

- main/LVFV_system.h

5.18. Referencia de la estructura sh1106_t

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

Campos de datos

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

5.18.1. Descripción detallada

Definición en la línea 13 del archivo sh1106_graphics.h.

5.18.2. Documentación de campos

5.18.2.1. buffer

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

Definición en la línea 17 del archivo sh1106_graphics.h.

5.18.2.2. height

```
uint8_t height
```

Definición en la línea 15 del archivo sh1106_graphics.h.

5.18.2.3. rotation

```
uint8_t rotation
```

Definición en la línea 16 del archivo sh1106_graphics.h.

5.18.2.4. width

```
uint8_t width
```

Definición en la línea 14 del archivo sh1106_graphics.h.

La documentación de esta estructura está generada del siguiente archivo:

- main/display/sh1106_graphics.h

5.19. Referencia de la estructura spi_cmd_item_t

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

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

Campos de datos

- SPI_Request request
- int getValue
- int setValue

5.19.1. Descripción detallada

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

Nota

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

Definición en la línea 59 del archivo sysControl.h.

5.19.2. Documentación de campos

5.19.2.1. getValue

```
int getValue
```

Definición en la línea 61 del archivo sysControl.h.

5.19.2.2. request

```
SPI_Request request
```

Definición en la línea 60 del archivo sysControl.h.

5.19.2.3. setValue

```
int setValue
```

Definición en la línea 62 del archivo sysControl.h.

La documentación de esta estructura está generada del siguiente archivo:

- main/system/sysControl.h

5.20. Referencia de la estructura system_status_t

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

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

Campos de datos

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

5.20.1. Descripción detallada

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

Definición en la línea 109 del archivo LVFV_system.h.

5.20.2. Documentación de campos

5.20.2.1. acceleration

```
uint16_t acceleration
```

Definición en la línea 114 del archivo LVFV_system.h.

5.20.2.2. desacceleration

```
uint16_t desacceleration
```

Definición en la línea 115 del archivo LVFV_system.h.

5.20.2.3. emergency_signals

```
uint8_t emergency_signals
```

Definición en la línea 118 del archivo LVFV_system.h.

5.20.2.4. frequency

```
uint16_t frequency
```

Definición en la línea 110 del archivo LVFV_system.h.

5.20.2.5. frequency_destiny

```
uint16_t frequency_destiny
```

Definición en la línea 111 del archivo LVFV_system.h.

5.20.2.6. ibus_max

```
uint16_t ibus_max
```

Definición en la línea 113 del archivo LVFV_system.h.

5.20.2.7. inputs_status

```
uint8_t inputs_status
```

Definición en la línea 116 del archivo LVFV_system.h.

5.20.2.8. status

```
system_status_e status
```

Definición en la línea 117 del archivo LVFV_system.h.

5.20.2.9. vbus_min

```
uint16_t vbus_min
```

Definición en la línea 112 del archivo LVFV_system.h.

La documentación de esta estructura está generada del siguiente archivo:

- main/LVFV_system.h

5.21. Referencia de la estructura time_settings_SH1106_t

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

Campos de datos

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

5.21.1. Descripción detallada

Definición en la línea 135 del archivo LVFV_system.h.

5.21.2. Documentación de campos

5.21.2.1. edit

```
sh1106_variable_lines_e* edit
```

Definición en la línea 137 del archivo LVFV_system.h.

5.21.2.2. edit_flag

```
uint8_t* edit_flag
```

Definición en la línea 139 del archivo LVFV_system.h.

5.21.2.3. edit_variable

```
uint8_t edit_variable
```

Definición en la línea 138 del archivo LVFV_system.h.

5.21.2.4. multiplier

```
uint8_t* multiplier
```

Definición en la línea 140 del archivo LVFV_system.h.

5.21.2.5. time_settings

```
time_settings_t time_settings
```

Definición en la línea 136 del archivo LVFV_system.h.

La documentación de esta estructura está generada del siguiente archivo:

- main/LVFV_system.h

5.22. Referencia de la estructura time_settings_t

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

Campos de datos

- struct tm * time_system
- struct tm * time_start
- struct tm * time_stop

5.22.1. Descripción detallada

Definición en la línea 93 del archivo LVFV_system.h.

5.22.2. Documentación de campos

5.22.2.1. time_start

```
struct tm* time_start
```

Definición en la línea 95 del archivo LVFV_system.h.

5.22.2.2. time_stop

```
struct tm* time_stop
```

Definición en la línea 96 del archivo LVFV_system.h.

5.22.2.3. time_system

```
struct tm* time_system
```

Definición en la línea 94 del archivo LVFV_system.h.

La documentación de esta estructura está generada del siguiente archivo:

- main/LVFV_system.h

Capítulo 6

Documentación de archivos

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

Funcion de inicialización y tarea lectura del ADC.

```
#include <stdio.h>
#include <math.h>
#include "freertos/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 mili volts.
- 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 contínua, compara con los parámetros seteados y dispara el estado de emergencia si es necesario.

Variables

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

6.1.1. Descripción detallada

Funcion de inicialización y tarea lectura del ADC.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo adc.c.

6.1.2. Documentación de «define»

6.1.2.1. ADC_ATTEN_USED

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

Definición en la línea 36 del archivo adc.c.

6.1.2.2. ADC_CH_GPIO34

```
#define ADC_CH_GPIO34 ADC_CHANNEL_6
```

Definición en la línea 31 del archivo adc.c.

6.1.2.3. ADC_CH_GPIO35

```
#define ADC_CH_GPIO35 ADC_CHANNEL_7
```

Definición en la línea 32 del archivo adc.c.

6.1.2.4. ADC_CH_GPIO36

```
#define ADC_CH_GPIO36 ADC_CHANNEL_0
```

Definición en la línea 33 del archivo adc.c.

6.1.2.5. ADC_CH_GPIO39

```
#define ADC_CH_GPIO39 ADC_CHANNEL_3
```

Definición en la línea 34 del archivo adc.c.

6.1.2.6. ADC_PERIOD_MS

```
#define ADC_PERIOD_MS 100
```

Parámetros

in	*arg*	Sin uso.

Valores devueltos

-	ESP_OK: Si inicialización y calibración de todos los ADC fue satisfactoria • ESP_ERR_INVALID_ARG: Si la iniciaulización de algún ADC falló • ESP_ERR_INVALID_STATE: Si las calibraciones de algún ADC falló

Definición en la línea 28 del archivo adc.c.

6.1.2.7. ADC_UNIT_USED

```
#define ADC_UNIT_USED ADC_UNIT_1
```

Definición en la línea 30 del archivo adc.c.

6.1.3. Documentación de funciones

6.1.3.1. adc_cali_try_init()

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

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

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

Parámetros

in	*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.

6.1.3.2. adc_init()

```
esp_err_t adc_init (
            void )
```

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

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

Definición en la línea 97 del archivo adc.c.

6.1.3.3. adc_task()

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

Función programada como alarma para finalizar el funcionamiento del motor.

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

Definición en la línea 146 del archivo adc.c.

6.1.3.4. readADC()

```
bool readADC (
             void )
```

Desencola mediciones obtenidas desde el ADC para tensión y corriente del bus de 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

-	true: Si el sistema está inicializado y las mediciones pueden ser obtenidas • false: Si el sistema todavía no inición y las mediciones que pudo haber obtenido fueron inválidas

Definición en la línea 239 del archivo adc.c.

6.1.4. Documentación de variables

6.1.4.1. bus_meas_evt_queue

```
QueueHandle_t bus_meas_evt_queue = NULL
```

Definición en la línea 52 del archivo adc.c.

6.1.4.2. calibration_3V3_source

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

Definición en la línea 45 del archivo adc.c.

6.1.4.3. calibration_3V3_source_ok

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

Definición en la línea 50 del archivo adc.c.

6.1.4.4. calibration_5V_source

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

Definición en la línea 44 del archivo adc.c.

6.1.4.5. calibration_5V_source_ok

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

Definición en la línea 49 del archivo adc.c.

6.1.4.6. calibration_bus_voltage

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

Definición en la línea 42 del archivo adc.c.

6.1.4.7. calibration_bus_voltage_ok

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

Definición en la línea 47 del archivo adc.c.

6.1.4.8. calibration_current

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

Definición en la línea 43 del archivo adc.c.

6.1.4.9. calibration_current_ok

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

Definición en la línea 48 del archivo adc.c.

6.1.4.10. s_adc

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

Definición en la línea 40 del archivo adc.c.

6.1.4.11. TAG

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

Definición en la línea 38 del archivo adc.c.

6.2. adc.c

Ir a la documentación de este archivo.

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

```
00031 #define ADC_CH_GPIO34          ADC_CHANNEL_6   // GPIO34
00032 #define ADC_CH_GPIO35          ADC_CHANNEL_7   // GPIO35
00033 #define ADC_CH_GPIO36          ADC_CHANNEL_0   // GPIO36
00034 #define ADC_CH_GPIO39          ADC_CHANNEL_3   // GPIO39
00035
00036 #define ADC_ATTEN_USED         ( (adc_atten_t) 3 ) // ~3.3-3.6 V FS aprox.
00037
00038 static const char *TAG = "ADC";
00039
00040 static adc_oneshot_unit_handle_t s_adc = NULL;
00041
00042 static adc_cali_handle_t calibration_bus_voltage = NULL;
00043 static adc_cali_handle_t calibration_current = NULL;
00044 static adc_cali_handle_t calibration_5V_source = NULL;
00045 static adc_cali_handle_t calibration_3V3_source = NULL;
00046
00047 static bool calibration_bus_voltage_ok = false;
00048 static bool calibration_current_ok = false;
00049 static bool calibration_5V_source_ok = false;
00050 static bool calibration_3V3_source_ok = false;
00051
00052 QueueHandle_t bus_meas_evt_queue = NULL;
00053
00080 static esp_err_t adc_cali_try_init(adc_unit_t unit, adc_channel_t ch, adc_atten_t atten,
      adc_cali_handle_t *out) {
00081     esp_err_t err;
00082
00083     if ( out == NULL ) {
00084         return ESP_ERR_INVALID_ARG;
00085     }
00086
00087     adc_cali_line_fitting_config_t cfg2 = {
00088         .unit_id  = unit,
00089         .atten    = atten,
00090         .bitwidth = ADC_BITWIDTH_DEFAULT,
00091     };
00092     err = adc_cali_create_scheme_line_fitting(&cfg2, out);
00093
00094     return err;
00095 }
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,
      &calibration_5V_source);
00134     calibration_3V3_source_ok = adc_cali_try_init(ADC_UNIT_USED, ADC_CH_GPIO39, ADC_ATTEN_USED,
      &calibration_3V3_source);
00135     calibration_bus_voltage_ok = adc_cali_try_init(ADC_UNIT_USED, ADC_CH_GPIO34, ADC_ATTEN_USED,
      &calibration_bus_voltage);
00136     calibration_current_ok = adc_cali_try_init(ADC_UNIT_USED, ADC_CH_GPIO35, ADC_ATTEN_USED,
      &calibration_current);
00137
00138     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" :
    "OFF");
00141     ESP_LOGI(TAG, "Calibración: DC bus current =%s", calibration_current_ok == ESP_OK ? "ON" : "OFF");
00142
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] =
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
00149     uint16_t vector_index = 0;
00150
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     seccurity_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(seccurity_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,
    &meas_bus_voltage);
00179
00180                 vbus_sum -= vbus_vector[vector_index];
00181                 vbus_vector[vector_index] = (int) truncf(meas_bus_voltage / 9.014);
00182                 vbus_sum += vbus_vector[vector_index];
00183
00184                 if ( vector_filled ) {
00185                     bus_meas.vbus_min = (uint16_t) (vbus_sum / 20);
00186                 }
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             }
00205         } else {
00206             ESP_LOGE(TAG, "Error de lectura corriente");
00207         }
00208
00209         if (adc_oneshot_read(s_adc, ADC_CH_GPIO39, &raw_meas_3V3_source) == ESP_OK) {
00210             if ( calibration_3V3_source_ok == ESP_OK ) {
00211                 adc_cali_raw_to_voltage(calibration_3V3_source, raw_meas_3V3_source,
    &meas_3V3_source);
00212             }
00213         } else {
00214             ESP_LOGE(TAG, "Error de lectura corriente");
00215         }
00216
00217         if (adc_oneshot_read(s_adc, ADC_CH_GPIO36, &raw_meas_5V_source) == ESP_OK) {
00218             if ( calibration_5V_source_ok == ESP_OK ) {
00219                 adc_cali_raw_to_voltage(calibration_5V_source, raw_meas_5V_source, &meas_5V_source);
00220             }
```

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

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

Declraración de funcion 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 contínua, compara con los parámetros seteados y dispara el estado de emergencia si es necesario.

6.3.1. Descripción detallada

Declraración de funcion de inicialización y tarea lectura del ADC.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo adc.h.

6.3.2. Documentación de funciones

6.3.2.1. adc_init()

```
esp_err_t adc_init (
            void )
```

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

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

Definición en la línea 97 del archivo adc.c.

6.3.2.2. adc_task()

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

Función programada como alarma para finalizar el funcionamiento del motor.

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

Definición en la línea 146 del archivo adc.c.

6.3.2.3. readADC()

```
bool readADC (
            void )
```

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

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

Valores devueltos

-	true: Si el sistema está inicializado y las mediciones pueden ser obtenidas • false: Si el sistema todavía no inición y las mediciones que pudo haber obtenido fueron inválidas

Definición en la línea 239 del archivo adc.c.

6.4. adc.h

Ir a la documentación de este archivo.

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

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

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

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

defines

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

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

Enumeraciones

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

  Listado de posibles pantallas que puede ver el usuario.

Funciones

- static void i2c_master_init (void)

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

  Vacía 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 (seccurity_settings_SH1106_t *variables)

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

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

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

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

  Pantalla principal dell 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 ()
- uint16_t get_system_frequency ()

  Entrega el valor de frecuencia de regimen seteado por el usuario.
- uint16_t get_system_acceleration ()

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

  Entrega el valor de desaceleración seteado por el usuario.

- esp_err_t sh1106_init ()

  Función que inicializa el display para comenzar a imprimir.
- esp_err_t system_variables_save (frequency_settings_SH1106_t *frequency_settings, time_settings_SH1106_t *time_settings, seccurity_settings_SH1106_t *seccurity_settings)

  Copia parámetros recibidos desde la UI al estado del sistema, los aplica en tiempo de ejecución y los persiste en NVS.
- void task_display (void *pvParameters)

  Tarea de display que controla las diferentes pantallas de muestra, selección y edición de las variables de sistema.
- esp_err_t DisplayEventPost (systemSignal_e event)

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

Variables

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

6.5.1. Descripción detallada

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

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo display.c.

6.5.2. Documentación de «define»

6.5.2.1. EDIT_ACCELERATION_INDX

```
#define EDIT_ACCELERATION_INDX VARIABLE_SECOND
```

Definición en la línea 49 del archivo display.c.

6.5.2.2. EDIT_DESACCELERATION_INDX

```
#define EDIT_DESACCELERATION_INDX VARIABLE_THIRD
```

Definición en la línea 50 del archivo display.c.

6.5.2.3. EDIT_FREQUENCY_INDX

```
#define EDIT_FREQUENCY_INDX VARIABLE_FIRST
```

Definición en la línea 48 del archivo display.c.

6.5.2.4. EDIT_HFIN_LINE_INDX

```
#define EDIT_HFIN_LINE_INDX VARIABLE_FOURTH
```

Definición en la línea 55 del archivo display.c.

6.5.2.5. EDIT_HINI_LINE_INDX

```
#define EDIT_HINI_LINE_INDX VARIABLE_THIRD
```

Definición en la línea 54 del archivo display.c.

6.5.2.6. EDIT_IBUS_LINE_INDX

```
#define EDIT_IBUS_LINE_INDX VARIABLE_THIRD
```

Definición en la línea 58 del archivo display.c.

6.5.2.7. EDIT_INPUT_VARIATION_INDX

```
#define EDIT_INPUT_VARIATION_INDX VARIABLE_FOURTH
```

Definición en la línea 51 del archivo display.c.

6.5.2.8. EDIT_TIME_LINE_INDX

```
#define EDIT_TIME_LINE_INDX VARIABLE_SECOND
```

Definición en la línea 53 del archivo display.c.

6.5.2.9. EDIT_VBUS_LINE_INDX

```
#define EDIT_VBUS_LINE_INDX VARIABLE_SECOND
```

Definición en la línea 57 del archivo display.c.

6.5.2.10. FREC_LINE_INDX

```
#define FREC_LINE_INDX VARIABLE_FIRST
```

Definición en la línea 42 del archivo display.c.

6.5.2.11. FREQUENCY_CUADRATIC_VARIABLE

```
#define FREQUENCY_CUADRATIC_VARIABLE 1
```

Definición en la línea 40 del archivo display.c.

6.5.2.12. FREQUENCY_LINEAR_VARIABLE

```
#define FREQUENCY_LINEAR_VARIABLE 0
```

Definición en la línea 39 del archivo display.c.

6.5.2.13. HFIN_LINE_INDX

```
#define HFIN_LINE_INDX VARIABLE_FIFTH
```

Definición en la línea 46 del archivo display.c.

6.5.2.14. HINI_LINE_INDX

```
#define HINI_LINE_INDX VARIABLE_FOURTH
```

Definición en la línea 45 del archivo display.c.

6.5.2.15. I2C_DISPLAY_MASTER_NUM

```
#define I2C_DISPLAY_MASTER_NUM I2C_NUM_0
```

Definición en la línea 32 del archivo display.c.

6.5.2.16. I2C_MASTER_FREQ_HZ

```
#define I2C_MASTER_FREQ_HZ 400000
```

Definición en la línea 35 del archivo display.c.

6.5.2.17. I2C_MASTER_RX_BUF_DISABLE

```
#define I2C_MASTER_RX_BUF_DISABLE 0
```

Definición en la línea 37 del archivo display.c.

6.5.2.18. I2C_MASTER_SCL_IO

```
#define I2C_MASTER_SCL_IO 5
```

Definición en la línea 34 del archivo display.c.

6.5.2.19. I2C_MASTER_SDA_IO

```
#define I2C_MASTER_SDA_IO 18
```

Definición en la línea 33 del archivo display.c.

6.5.2.20. I2C_MASTER_TX_BUF_DISABLE

```
#define I2C_MASTER_TX_BUF_DISABLE 0
```

Definición en la línea 36 del archivo display.c.

6.5.2.21. IBUS_LINE_INDX

```
#define IBUS_LINE_INDX VARIABLE_THIRD
```

Definición en la línea 44 del archivo display.c.

6.5.2.22. VBUS_LINE_INDX

```
#define VBUS_LINE_INDX VARIABLE_SECOND
```

Definición en la línea 43 del archivo display.c.

6.5.3. Documentación de enumeraciones

6.5.3.1. screen_selected_e

```
enum screen_selected_e
```

Listado de posibles pantallas que puede ver el usuario.

Valores de enumeraciones

SCREEN_MAIN	
SCREEN_SELECT_VARIABLE	
SCREEN_TIME_EDIT	
SCREEN_SECURITY_EDIT	

SCREEN_FREQUENCY_EDIT	

Definición en la línea 65 del archivo display.c.

6.5.4. Documentación de funciones

6.5.4.1. DisplayEventPost()

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

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

Valores devueltos

pdTRUE	Si se encoló exitosamente errQUEUE_FULL: Cualquier tipo de falla

Definición en la línea 900 del archivo display.c.

6.5.4.2. get_system_acceleration()

```
uint16_t get_system_acceleration ()
```

Entrega el valor de aceleración seteado por el usuario.

Valores devueltos

Valor	de aceleración configurado por el usuario

Definición en la línea 458 del archivo display.c.

6.5.4.3. get_system_desacceleration()

```
uint16_t get_system_desacceleration ()
```

Entrega el valor de desaceleración seteado por el usuario.

Valores devueltos

Valor	de desaceleración configurado por el usuario

Definición en la línea 462 del archivo display.c.

6.5.4.4. get_system_frequency()

```
uint16_t get_system_frequency ()
```

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

Valores devueltos

Valor	de frecuencia de regimen configurado por el usuario

Definición en la línea 454 del archivo display.c.

6.5.4.5. i2c_master_init()

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

Inicializa el bus I2C para comunicarse con el display SH1106.

Definición en la línea 187 del archivo display.c.

6.5.4.6. sh1106_clear_buffer()

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

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

Definición en la línea 200 del archivo display.c.

6.5.4.7. sh1106_frequency_edit_variables()

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

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

Parámetros

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

Definición en la línea 334 del archivo display.c.

6.5.4.8. sh1106_init()

```
esp_err_t sh1106_init ()
```

Función que inicializa el display para comenzar a imprimir.

Valores devueltos

ESP_OK	Si inicializa correctamente ESP_FAIL Si alguno de los comandos de inicialización falla

Definición en la línea 466 del archivo display.c.

6.5.4.9. sh1106_main_screen()

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

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

Definición en la línea 425 del archivo display.c.

6.5.4.10. sh1106_print_emergency()

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

Definición en la línea 235 del archivo display.c.

6.5.4.11. sh1106_print_frame()

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

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

Definición en la línea 227 del archivo display.c.

6.5.4.12. sh1106_refresh()

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

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

Valores devueltos

ESP_OK	si todo funciona bien

Definición en la línea 204 del archivo display.c.

6.5.4.13. sh1106_security_edit_variables()

```
void sh1106_security_edit_variables (
            seccurity_settings_SH1106_t * variables)  [static]
```

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

Parámetros

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

Definición en la línea 298 del archivo display.c.

6.5.4.14. sh1106_select_edit_variables()

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

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

Parámetros

in,out	*variables*	Selector de menúes

Definición en la línea 402 del archivo display.c.

6.5.4.15. sh1106_splash_screen()

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

Pantalla de splash que se imprime al iniciar el sistema.

Definición en la línea 414 del archivo display.c.

6.5.4.16. sh1106_time_edit_variables()

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

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

Parámetros

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

Definición en la línea 244 del archivo display.c.

6.5.4.17. system_variables_save()

```
esp_err_t system_variables_save (
            frequency_settings_SH1106_t * frequency_settings,
            time_settings_SH1106_t * time_settings,
            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.

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_seccurity_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

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

Devuelve

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

Definición en la línea 485 del archivo display.c.

6.5.4.18. task_display()

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

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

Parámetros

in	*pvParameters*	Variable sin uso

Definición en la línea 523 del archivo display.c.

6.5.5. Documentación de variables

6.5.5.1. blink

```
uint8_t blink  [static]
```

Definición en la línea 110 del archivo display.c.

6.5.5.2. button_evt_queue

```
QueueHandle_t button_evt_queue
```

Definición en la línea 73 del archivo display.c.

6.5.5.3. init_seq

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

Valor inicial:

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

Definición en la línea 81 del archivo display.c.

6.5.5.4. oled

```
sh1106_t oled
```

Definición en la línea 107 del archivo display.c.

6.5.5.5. screen_displayed

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

Definición en la línea 79 del archivo display.c.

6.5.5.6. system_frequency_settings

```
frequency_settings_t system_frequency_settings
```

Definición en la línea 75 del archivo display.c.

6.5.5.7. system_seccurity_settings

```
seccurity_settings_t system_seccurity_settings
```

Definición en la línea 77 del archivo display.c.

6.5.5.8. system_time_settings

```
time_settings_t system_time_settings
```

Definición en la línea 76 del archivo display.c.

6.5.5.9. TAG

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

Definición en la línea 109 del archivo display.c.

6.6. display.c

Ir a la documentación de este archivo.

```
00001
00009 #include "esp_log.h"
00010 #include "esp_err.h"
00011
00012 #include "freertos/FreeRTOS.h"
00013 #include "freertos/task.h"
00014 #include "freertos/queue.h"
00015
00016 #include <string.h>
00017 #include <math.h>
00018 #include "esp_system.h"
00019 #include "LVFV_system.h"
00020
00021 #include "driver/i2c.h"
00022 #include "./display/display.h"
00023 #include "./display/sh1106_i2c.h"
00024 #include "./display/sh1106_graphics.h"
00025
00026 #include "./io_control/io_control.h"
00027 #include "../system/SysControl.h"
00028 #include "../system/SysAdmin.h"
00029 #include "../rtc/rtc.h"
00030 #include "../nvs/nvs.h"
00031
00032 #define I2C_DISPLAY_MASTER_NUM                          I2C_NUM_0   // Número de puerto I2C usado
00033 #define I2C_MASTER_SDA_IO                               18          // Pin SDA del puerto I2C ->
      GPIO21
00034 #define I2C_MASTER_SCL_IO                               5           // Pin SCL del puerto I2C ->
      GPIO22
00035 #define I2C_MASTER_FREQ_HZ                              400000      // Frecuencia de clock del puerto
      I2C -> 400kHz
00036 #define I2C_MASTER_TX_BUF_DISABLE                       0           // Largo del buffer de transmisión
      del puerto I2C - Como se usa en modo master, es una variable ignorada
```

```
00037 #define I2C_MASTER_RX_BUF_DISABLE                       0           // Largo del buffer de recepción
      del puerto I2C - Como se usa en modo master, es una variable ignorada
00038
00039 #define FREQUENCY_LINEAR_VARIABLE       0
00040 #define FREQUENCY_CUADRATIC_VARIABLE    1
00041
00042 #define FREC_LINE_INDX                  VARIABLE_FIRST
00043 #define VBUS_LINE_INDX                  VARIABLE_SECOND
00044 #define IBUS_LINE_INDX                  VARIABLE_THIRD
00045 #define HINI_LINE_INDX                  VARIABLE_FOURTH
00046 #define HFIN_LINE_INDX                  VARIABLE_FIFTH
00047
00048 #define EDIT_FREQUENCY_INDX             VARIABLE_FIRST
00049 #define EDIT_ACCELERATION_INDX          VARIABLE_SECOND
00050 #define EDIT_DESACCELERATION_INDX       VARIABLE_THIRD
00051 #define EDIT_INPUT_VARIATION_INDX       VARIABLE_FOURTH
00052
00053 #define EDIT_TIME_LINE_INDX             VARIABLE_SECOND
00054 #define EDIT_HINI_LINE_INDX             VARIABLE_THIRD
00055 #define EDIT_HFIN_LINE_INDX             VARIABLE_FOURTH
00056
00057 #define EDIT_VBUS_LINE_INDX             VARIABLE_SECOND
00058 #define EDIT_IBUS_LINE_INDX             VARIABLE_THIRD
00059
00065 typedef enum {
00066     SCREEN_MAIN,                                                // 0 - Pantalla principal
00067     SCREEN_SELECT_VARIABLE,                                     // 1 - Pantalla de selección de
      pantallas de edición
00068     SCREEN_TIME_EDIT,                                           // 2 - Pantalla de edición de
      variables de tiempo
00069     SCREEN_SECURITY_EDIT,                                       // 3 - Pantalla de edición de
      variables de seguridad
00070     SCREEN_FREQUENCY_EDIT                                       // 4 - Pantalla de edición de
      variables de frecuencia, aceleración y desaceleración
00071 } screen_selected_e;
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,  // SETCHARGEPUMP
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
00117 static void i2c_master_init(void);
00123 static void sh1106_clear_buffer();
00131 static esp_err_t sh1106_refresh();
00137 static void sh1106_print_frame();
00146 static void sh1106_time_edit_variables(time_settings_SH1106_t *variables);
```

```
00155 static void sh1106_security_edit_variables(seccurity_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,
      I2C_MASTER_TX_BUF_DISABLE, 0));
00198 }
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
      página
00207     for ( page = 0; page < 8; page++) {
00208         // Dirección de página
00209         uint8_t command = 0xB0 + page;
00210         ESP_ERROR_CHECK(sh1106_write(&command, 1, SH1106_COMM_CMD));
00211         // Dirección de columna baja y alta (offset 2 que ajusta SH1106)
00212         command = 0x02;
00213         ESP_ERROR_CHECK(sh1106_write(&command, 1, SH1106_COMM_CMD));    // col lo (2)
00214         command = 0x10;
00215         ESP_ERROR_CHECK(sh1106_write(&command, 1, SH1106_COMM_CMD));    // col hi
00216
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_INDX, SH1106_SIZE_1);
00255     sh1106_draw_text( &oled, "Ini:", 15, EDIT_HINI_LINE_INDX, SH1106_SIZE_1);
00256     sh1106_draw_text( &oled, "Fin:", 15, EDIT_HFIN_LINE_INDX, 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_EIGTH ) {
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,
      variables->time_settings.time_system->tm_sec);
00268     } else if( *(variables->edit) == VARIABLE_THIRD && blink < blink_compare) {
00269         sprintf(hora_str, "%02d:  :%02d", variables->time_settings.time_system->tm_hour,
      variables->time_settings.time_system->tm_sec);
00270     } else if( *(variables->edit) == VARIABLE_FOURTH && blink < blink_compare) {
00271         sprintf(hora_str, "%02d:%02d:  ", variables->time_settings.time_system->tm_hour,
      variables->time_settings.time_system->tm_min);
00272     } else {
00273         strftime(hora_str, sizeof(hora_str), "%H:%M:%S", variables->time_settings.time_system);
00274     }
00275     sh1106_draw_text( &oled, hora_str, 64, EDIT_TIME_LINE_INDX, SH1106_SIZE_1);
00276 
00277     if ( *(variables->edit) == VARIABLE_FIFTH && blink < blink_compare ) {
00278         sprintf(hora_str, "  :%02d", variables->time_settings.time_start->tm_min);
00279     } else if( *(variables->edit) == VARIABLE_SIXTH && blink < blink_compare ) {
00280         sprintf(hora_str, "%02d:  ", variables->time_settings.time_start->tm_hour);
00281     } else {
00282         sprintf(hora_str, "%02d:%02d", variables->time_settings.time_start->tm_hour,
      variables->time_settings.time_start->tm_min);
00283     }
00284     sh1106_draw_text( &oled, hora_str, 64, EDIT_HINI_LINE_INDX, SH1106_SIZE_1);
00285 
00286     if ( *(variables->edit) == VARIABLE_SEVENTH && blink < blink_compare ) {
00287         sprintf(hora_str, "  :%02d", variables->time_settings.time_stop->tm_min);
00288     } else if ( *(variables->edit) == VARIABLE_EIGTH && blink < blink_compare ) {
00289         sprintf(hora_str, "%02d:  ", variables->time_settings.time_stop->tm_hour);
00290     } else {
00291         sprintf(hora_str, "%02d:%02d", variables->time_settings.time_stop->tm_hour,
      variables->time_settings.time_stop->tm_min);
00292     }
00293     sh1106_draw_text( &oled, hora_str, 64, EDIT_HFIN_LINE_INDX, SH1106_SIZE_1);
00294 
00295     ESP_ERROR_CHECK(sh1106_refresh());
00296 }
00297 
00298 static void sh1106_security_edit_variables(seccurity_settings_SH1106_t *variables) {
00299     char num_str[12];
00300 
00301     sh1106_clear_buffer( &oled );
00302     sh1106_print_frame();
00303 
00304     sh1106_draw_text( &oled, "V. min:", 15, EDIT_VBUS_LINE_INDX, SH1106_SIZE_1);
00305     sh1106_draw_text( &oled, "V",           117, EDIT_VBUS_LINE_INDX, SH1106_SIZE_1);
00306     sh1106_draw_text( &oled, "I. max:", 15, EDIT_IBUS_LINE_INDX, SH1106_SIZE_1);
00307     sh1106_draw_text( &oled, "A",         117, EDIT_IBUS_LINE_INDX, SH1106_SIZE_1);
00308     sh1106_draw_arrow( &oled, 1, (int) *(variables->edit));
00309 
00310     if ( *(variables->edit) == EDIT_VBUS_LINE_INDX && *(variables->edit_flag) ) {
00311         if ( blink >= 12 ) {
00312             sprintf(num_str, "%03d", variables->seccurity_settings.vbus_min);
00313             sh1106_draw_text( &oled, num_str, 85, EDIT_VBUS_LINE_INDX, SH1106_SIZE_1);
00314         }
00315     } else {
00316         sprintf(num_str, "%03d", variables->seccurity_settings.vbus_min);
00317         sh1106_draw_text( &oled, num_str, 85, EDIT_VBUS_LINE_INDX, SH1106_SIZE_1);
00318     }
00319 
00320     if ( *(variables->edit) == EDIT_IBUS_LINE_INDX && *(variables->edit_flag) ) {
00321         if ( blink >= 12 ) {
00322             sprintf(num_str, "%03d", variables->seccurity_settings.ibus_max);
00323             sh1106_draw_text( &oled, num_str, 85, EDIT_IBUS_LINE_INDX, SH1106_SIZE_1);
00324         }
00325     } else {
00326         sprintf(num_str, "%03d", variables->seccurity_settings.ibus_max);
00327         sh1106_draw_text( &oled, num_str, 85, EDIT_IBUS_LINE_INDX, SH1106_SIZE_1);
00328     }
00329     sh1106_draw_arrow( &oled, 1, (int) variables->edit);
00330 
00331     ESP_ERROR_CHECK(sh1106_refresh());
00332 }
00333 
00334 static void sh1106_frequency_edit_variables(frequency_settings_SH1106_t *variables) {
00335     char num_str[12];
00336 
00337     sh1106_clear_buffer();
00338     sh1106_print_frame();
00339 
00340     sh1106_draw_text( &oled, "Frec:",        15, VARIABLE_FIRST, SH1106_SIZE_1);
00341     sh1106_draw_text( &oled, "Hz",          95, VARIABLE_FIRST, SH1106_SIZE_1);
00342     sh1106_draw_text( &oled, "Acel:",       15, VARIABLE_SECOND, SH1106_SIZE_1);
00343     sh1106_draw_text( &oled, "Hz/s",        95, VARIABLE_SECOND, SH1106_SIZE_1);
00344     sh1106_draw_text( &oled, "Desa:",       15, VARIABLE_THIRD, SH1106_SIZE_1);
00345     sh1106_draw_text( &oled, "Hz/s",        95, VARIABLE_THIRD, SH1106_SIZE_1);
00346     sh1106_draw_text( &oled, "Var entradas", 18, VARIABLE_FOURTH, SH1106_SIZE_1);
00347 
```

```
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 ,
    system_time_settings.time_start->tm_min );
00447     sh1106_draw_text( &oled, hora_str, 86, VARIABLE_FOURTH, SH1106_SIZE_1);
00448     sprintf(hora_str, "%02d:%02d", system_time_settings.time_stop->tm_hour ,
    system_time_settings.time_stop->tm_min );
00449     sh1106_draw_text( &oled, hora_str, 86, VARIABLE_FIFTH, SH1106_SIZE_1);
00450 
00451     ESP_ERROR_CHECK(sh1106_refresh());
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,
    time_settings_SH1106_t *time_settings, seccurity_settings_SH1106_t *seccurity_settings) {
00486     ESP_LOGI(TAG, "Guardando variables del sistema desde el display");
00487     if ( frequency_settings == NULL || time_settings == NULL || seccurity_settings == NULL ) {
00488         return ESP_ERR_INVALID_ARG;
00489     }
00490 
00491     if ( time_settings->time_settings.time_system == NULL || time_settings->time_settings.time_start
    == NULL || time_settings->time_settings.time_stop == NULL ) {
00492         return ESP_ERR_INVALID_ARG;
00493     }
00494 
00495     system_frequency_settings.freq_regime = frequency_settings->frequency_settings.freq_regime;
00496     system_frequency_settings.acceleration = frequency_settings->frequency_settings.acceleration;
00497     system_frequency_settings.desacceleration =
    frequency_settings->frequency_settings.desacceleration;
00498     system_frequency_settings.input_variable = frequency_settings->frequency_settings.input_variable;
00499     system_seccurity_settings.vbus_min = seccurity_settings->seccurity_settings.vbus_min;
00500     system_seccurity_settings.ibus_max = seccurity_settings->seccurity_settings.ibus_max;
00501     system_time_settings.time_system->tm_hour = time_settings->time_settings.time_system->tm_hour;
00502     system_time_settings.time_system->tm_min = time_settings->time_settings.time_system->tm_min;
00503     system_time_settings.time_system->tm_sec = time_settings->time_settings.time_system->tm_sec;
00504     system_time_settings.time_start->tm_hour = time_settings->time_settings.time_start->tm_hour;
00505     system_time_settings.time_start->tm_min = time_settings->time_settings.time_start->tm_min;
00506     system_time_settings.time_start->tm_sec = 0;
00507     system_time_settings.time_stop->tm_hour = time_settings->time_settings.time_stop->tm_hour;
00508     system_time_settings.time_stop->tm_min = time_settings->time_settings.time_stop->tm_min;
00509     system_time_settings.time_stop->tm_sec = 0;
00510 
00511     setTime(  system_time_settings.time_system );
00512     set_frequency_table(system_frequency_settings.input_variable,
    system_frequency_settings.freq_regime);
00513     rtc_schedule_alarms(&system_time_settings);
00514 
00515     if ( save_variables( &system_frequency_settings, &system_time_settings,
```

```
      &system_seccurity_settings) != ESP_OK ) {
00516         ESP_LOGE(TAG, "Algo falló guardando los valores en NVS");
00517     }
00518     set_system_settings( &system_frequency_settings, &system_seccurity_settings);
00519     ESP_LOGI(TAG, "Configuración guardada correctamente");
00520     return ESP_OK;
00521 }
00522
00523 void task_display(void *pvParameters) {
00524
00525     uint8_t new_button;                                                    // Variable Queue
00526
00527     sh1106_variable_lines_e top_variable = first, bottom_variable = fifth;      // Identificador de la
      variable seleccionada
00528     sh1106_variable_lines_e variable_lines = VARIABLE_FIRST;
00529
00530     uint8_t top_multiplier = 100, bottom_multiplier = 1;                         // Incremental de la
      variable seleccionada
00531     uint8_t multiplier = 1;
00532
00533     uint32_t edit_variable_min = 0, edit_variable_max = 0;                       // Variable
      seleccionada
00534     uint16_t *edit_variable = NULL;
00535
00536     uint8_t edit = 0;                                                      // Edición o selección
00537
00538     struct tm timeinfo = {
00539         .tm_hour = 0,
00540         .tm_min = 0,
00541         .tm_sec = 0,
00542         .tm_mday = 0,
00543         .tm_mon = 0,
00544         .tm_year = 0
00545     };
00546     struct tm time_start = {
00547         .tm_hour = 0,
00548         .tm_min = 0,
00549         .tm_sec = 0,
00550         .tm_mday = 0,
00551         .tm_mon = 0,
00552         .tm_year = 0
00553     };
00554     struct tm time_stop = {
00555         .tm_hour = 0,
00556         .tm_min = 0,
00557         .tm_sec = 0,
00558         .tm_mday = 0,
00559         .tm_mon = 0,
00560         .tm_year = 0
00561     };
00562
00563     struct tm timestamp_edit = {
00564         .tm_hour = 0,
00565         .tm_min = 0,
00566         .tm_sec = 0,
00567         .tm_mday = 0,
00568         .tm_mon = 0,
00569         .tm_year = 0
00570     };
00571     struct tm time_start_edit = {
00572         .tm_hour = 0,
00573         .tm_min = 0,
00574         .tm_sec = 0,
00575         .tm_mday = 0,
00576         .tm_mon = 0,
00577         .tm_year = 0
00578     };
00579     struct tm time_stop_edit = {
00580         .tm_hour = 0,
00581         .tm_min = 0,
00582         .tm_sec = 0,
00583         .tm_mday = 0,
00584         .tm_mon = 0,
00585         .tm_year = 0
00586     };
00587
00588     if ( sh1106_init() != ESP_OK ) {
00589         ESP_LOGE(TAG, "Error al inicializar el display SH1106");
00590         ESP_ERROR_CHECK(ESP_FAIL);
00591     }
00592     ESP_LOGI(TAG, "Inicialización del display SH1106 exitosa");
00593
00594
00595     time_settings_SH1106_t time_edit;
00596     seccurity_settings_SH1106_t seccurity_edit;
00597     frequency_settings_SH1106_t frequency_edit;
00598
```

```
00599     frequency_edit.multiplier = &multiplier;
00600     frequency_edit.edit = &variable_lines;
00601     frequency_edit.edit_flag = &edit;
00602 
00603     seccurity_edit.multiplier = &multiplier;
00604     seccurity_edit.edit = &variable_lines;
00605     seccurity_edit.edit_flag = &edit;
00606 
00607     time_edit.multiplier = &multiplier;
00608     time_edit.edit_flag = &edit;
00609     time_edit.time_settings.time_system = &timestamp_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 temportales inicializadas");
00624     set_frequency_table(system_frequency_settings.input_variable,
      system_frequency_settings.freq_regime);
00625     ESP_LOGI(TAG, "Tablas de frecuencia inicializadas");
00626 
00627     if (button_evt_queue == NULL) {
00628         button_evt_queue = xQueueCreate(1, sizeof(uint32_t));
00629         if (button_evt_queue == NULL) {
00630             ESP_LOGE(TAG, "No se pudo crear la cola de interrupciones");
00631             ESP_ERROR_CHECK(ESP_FAIL);
00632         }
00633     }
00634     ESP_LOGI(TAG, "Queue de eventos inicializadas");
00635 
00636     load_variables( &system_frequency_settings, &system_time_settings, &system_seccurity_settings);
00637     set_system_settings( &system_frequency_settings, &system_seccurity_settings);
00638 
00639     vTaskDelay(pdMS_TO_TICKS(2000));
00640 
00641     while (1) {
00642         getTime(&timeinfo);
00643         blink++;
00644         if ( blink >= 24 ) {
00645             blink = 0;
00646         }
00647         if ( xQueueReceive( button_evt_queue, &new_button, pdMS_TO_TICKS(5) ) ) {
00648             switch (new_button) {
00649                 case BUTTON_MENU:
00650                     if ( screen_displayed == SCREEN_MAIN ) {
00651                         screen_displayed = SCREEN_SELECT_VARIABLE;
00652                         variable_lines = VARIABLE_SECOND;
00653                         top_variable = VARIABLE_SECOND;
00654                         bottom_variable = VARIABLE_FOURTH;
00655 
00656                         frequency_edit.frequency_settings.freq_regime =
      system_frequency_settings.freq_regime;
00657                         frequency_edit.frequency_settings.acceleration =
      system_frequency_settings.acceleration;
00658                         frequency_edit.frequency_settings.desacceleration =
      system_frequency_settings.desacceleration;
00659                         frequency_edit.frequency_settings.input_variable =
      system_frequency_settings.input_variable;
00660 
00661                         seccurity_edit.seccurity_settings.vbus_min =
      system_seccurity_settings.vbus_min;
00662                         seccurity_edit.seccurity_settings.ibus_max =
      system_seccurity_settings.ibus_max;
00663 
00664                         time_edit.time_settings.time_system->tm_hour =
      system_time_settings.time_system->tm_hour;
00665                         time_edit.time_settings.time_system->tm_min =
      system_time_settings.time_system->tm_min;
00666                         time_edit.time_settings.time_system->tm_sec =
      system_time_settings.time_system->tm_sec;
00667 
00668                         time_edit.time_settings.time_start->tm_hour =
      system_time_settings.time_start->tm_hour;
00669                         time_edit.time_settings.time_start->tm_min =
      system_time_settings.time_start->tm_min;
00670                         time_edit.time_settings.time_start->tm_sec = 0;
00671 
00672                         time_edit.time_settings.time_stop->tm_hour =
      system_time_settings.time_stop->tm_hour;
```

```
00673                         time_edit.time_settings.time_stop->tm_min =
      system_time_settings.time_stop->tm_min;
00674                         time_edit.time_settings.time_stop->tm_sec = 0;
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_EIGTH;
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*)
      &(frequency_edit.frequency_settings.freq_regime);
00707                                 edit_variable_min = 5;
00708                                 edit_variable_max = 150;
00709                             } else if ( variable_lines == EDIT_ACCELERATION_INDX ) {
00710                                 edit_variable = (uint16_t*)
      &(frequency_edit.frequency_settings.acceleration);
00711                                 edit_variable_min = 1;
00712                                 edit_variable_max = 150;
00713                             } else if ( variable_lines == EDIT_DESACCELERATION_INDX ) {
00714                                 edit_variable = (uint16_t*)
      &(frequency_edit.frequency_settings.desacceleration);
00715                                 edit_variable_min = 1;
00716                                 edit_variable_max = 150;
00717                             } else if ( variable_lines == EDIT_INPUT_VARIATION_INDX ) {
00718                                 edit_variable = (uint16_t*)
      &(frequency_edit.frequency_settings.input_variable);
00719                                 edit_variable_min = 1;
00720                                 edit_variable_max = 2;
00721                                 top_multiplier = 1;
00722                                 bottom_multiplier = 1;
00723                             }
00724                             edit = 1;
00725                             multiplier = 1;
00726                             top_multiplier = 100;
00727                             bottom_multiplier = 1;
00728                         } else {
00729                             edit = 0;
00730                             edit_variable = NULL;
00731                             edit_variable_min = 0;
00732                             edit_variable_max = 0;
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*)
      &(seccurity_edit.seccurity_settings.vbus_min);
00738                                 edit_variable_min = 250;
00739                                 edit_variable_max = 360;
00740                             } else if ( variable_lines == EDIT_IBUS_LINE_INDX ) {
00741                                 edit_variable = (uint16_t*)
      &(seccurity_edit.seccurity_settings.ibus_max);
00742                                 edit_variable_min = 500;
00743                                 edit_variable_max = 2000;
00744                             }
00745                             edit = 1;
00746                             multiplier = 1;
00747                             top_multiplier = 100;
00748                             bottom_multiplier = 1;
00749                         } else {
00750                             edit = 0;
00751                             edit_variable = NULL;
00752                             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*)
      &(time_edit.time_settings.time_system->tm_hour);
00759                               edit_variable_min = 0;
00760                               edit_variable_max = 23;
00761                           } else if ( variable_lines == VARIABLE_THIRD ) {
00762                               edit_variable = (uint16_t*)
      &(time_edit.time_settings.time_system->tm_min);
00763                               edit_variable_min = 0;
00764                               edit_variable_max = 59;
00765                           } else if ( variable_lines == VARIABLE_FOURTH ) {
00766                               edit_variable = (uint16_t*)
      &(time_edit.time_settings.time_system->tm_sec);
00767                               edit_variable_min = 0;
00768                               edit_variable_max = 59;
00769                           } else if ( variable_lines == VARIABLE_FIFTH ) {
00770                               edit_variable = (uint16_t*)
      &(time_edit.time_settings.time_start->tm_hour);
00771                               edit_variable_min = 0;
00772                               edit_variable_max = 23;
00773                           } else if ( variable_lines == VARIABLE_SIXTH ) {
00774                               edit_variable = (uint16_t*)
      &(time_edit.time_settings.time_start->tm_min);
00775                               edit_variable_min = 0;
00776                               edit_variable_max = 59;
00777                           } else if ( variable_lines == VARIABLE_SEVENTH ) {
00778                               edit_variable = (uint16_t*)
      &(time_edit.time_settings.time_stop->tm_hour);
00779                               edit_variable_min = 0;
00780                               edit_variable_max = 23;
00781                           } else if ( variable_lines == VARIABLE_EIGTH ) {
00782                               edit_variable = (uint16_t*)
      &(time_edit.time_settings.time_stop->tm_min);
00783                               edit_variable_min = 0;
00784                               edit_variable_max = 59;
00785                           }
00786                           edit = 1;
00787                           multiplier = 1;
00788                           top_multiplier = 10;
00789                           bottom_multiplier = 1;
00790                       } else {
00791                           edit = 0;
00792                           edit_variable = NULL;
00793                           edit_variable_min = 0;
00794                           edit_variable_max = 0;
00795                       }
00796                   } else if ( screen_displayed == SCREEN_MAIN ) {
00797                       SystemEventPost(START_PRESSED);
00798                   }
00799                   break;
00800               case BUTTON_BACK:
00801                   if ( screen_displayed == SCREEN_SELECT_VARIABLE ) {
00802                       screen_displayed = SCREEN_MAIN;
00803                   } else if ( screen_displayed == SCREEN_SECURITY_EDIT || screen_displayed ==
      SCREEN_FREQUENCY_EDIT || screen_displayed == SCREEN_TIME_EDIT ) {
00804                       screen_displayed = SCREEN_SELECT_VARIABLE;
00805                       variable_lines = VARIABLE_SECOND;
00806                       top_variable = VARIABLE_SECOND;
00807                       bottom_variable = VARIABLE_FOURTH;
00808                       edit_variable = NULL;
00809                       edit_variable_min = 0;
00810                       edit_variable_max = 0;
00811                   } else if ( screen_displayed == SCREEN_MAIN ) {
00812                       SystemEventPost(STOP_PRESSED);
00813                   }
00814                   break;
00815               case BUTTON_UP:
00816                   if ( edit_variable != NULL ) {
00817                       (*edit_variable) += multiplier;
00818                       if ( *edit_variable >= edit_variable_max ) {
00819                           *edit_variable = edit_variable_max;
00820                       }
00821                   } else {
00822                       if ( variable_lines == top_variable ) {
00823                           variable_lines = bottom_variable;
00824                       } else {
00825                           variable_lines -= LINE_INCREMENT;
00826                       }
00827                   }
00828                   break;
00829               case BUTTON_DOWN:
00830                   if ( edit_variable != NULL ) {
00831                       (*edit_variable) -= multiplier;
```

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

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

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

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

Funciones

- uint16_t get_system_frequency ()

  Entrega el valor de frecuencia de regimen seteado por el usuario.
- uint16_t get_system_acceleration ()

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

  Entrega el valor de desaceleración seteado por el usuario.
- esp_err_t sh1106_init ()

  Función que inicializa el display para comenzar a imprimir.
- void task_display (void *pvParameters)

  Tarea de display que controla las diferentes pantallas de muestra, selección y edición de las variables de sistema.
- esp_err_t DisplayEventPost (systemSignal_e event)

  Función que permite encolar acciones para que ejecute la tarea de display.
- esp_err_t system_variables_save (frequency_settings_SH1106_t *frequency_settings, time_settings_SH1106_t *time_settings, seccurity_settings_SH1106_t *seccurity_settings)

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

6.7.1. Descripción detallada

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

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo display.h.

6.7.2. Documentación de funciones

6.7.2.1. DisplayEventPost()

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

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

Valores devueltos

pdTRUE	Si se encoló exitosamente errQUEUE_FULL: Cualquier tipo de falla

Definición en la línea 900 del archivo display.c.

6.7.2.2. get_system_acceleration()

```
uint16_t get_system_acceleration ()
```

Entrega el valor de aceleración seteado por el usuario.

Valores devueltos

Valor	de aceleración configurado por el usuario

Definición en la línea 458 del archivo display.c.

6.7.2.3. get_system_desacceleration()

```
uint16_t get_system_desacceleration ()
```

Entrega el valor de desaceleración seteado por el usuario.

Valores devueltos

Valor	de desaceleración configurado por el usuario

Definición en la línea 462 del archivo display.c.

6.7.2.4. get_system_frequency()

```
uint16_t get_system_frequency ()
```

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

Valores devueltos

Valor	de frecuencia de regimen configurado por el usuario

Definición en la línea 454 del archivo display.c.

6.7.2.5. sh1106_init()

```
esp_err_t sh1106_init ()
```

Función que inicializa el display para comenzar a imprimir.

Valores devueltos

ESP_OK	Si inicializa correctamente ESP_FAIL Si alguno de los comandos de inicialización falla

Definición en la línea 466 del archivo display.c.

6.7.2.6. system_variables_save()

```
esp_err_t system_variables_save (
            frequency_settings_SH1106_t * frequency_settings,
            time_settings_SH1106_t * time_settings,
            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.

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_seccurity_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

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

Devuelve

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

Definición en la línea 485 del archivo display.c.

6.7.2.7. task_display()

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

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

Parámetros

in	*pvParameters*	Variable sin uso

Definición en la línea 523 del archivo display.c.

6.8. display.h

Ir a la documentación de este archivo.

```
00001
00009 #ifndef SH1106_H
00010 #define SH1106_H
00011
00012 #include <stdint.h>
00013 #include <stdbool.h>
00014 #include "../LVFV_system.h"
00015
00024 uint16_t get_system_frequency();
00033 uint16_t get_system_acceleration();
00042 uint16_t get_system_desacceleration();
00052 esp_err_t sh1106_init();
00061 void task_display(void *pvParameters);
00071 esp_err_t DisplayEventPost(systemSignal_e event);
00072
00096 esp_err_t system_variables_save(frequency_settings_SH1106_t *frequency_settings,
      time_settings_SH1106_t *time_settings, seccurity_settings_SH1106_t *seccurity_settings);
00097
00098 #endif
```

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

Funciones que permiten operar sobre el display.

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

Estructuras de datos

- struct bitmap_t

  Estructura que representa un caracter 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,0x↩ F8,0x07,0xE0,0x07,0xE0,0x1F,0xF8,0x3D,0xBC,0x79,0x9E,0x71,0x8E,0x71,0x8E}
- static const unsigned char fail_bmp [ ] = {0x03,0xC0,0x0C,0x30,0x33,0xD8,0x6E,0x6C,0xDC,0x06,0x↩ DE,0x03,0xCF,0xE3,0xC7,0xF3,0xC0,0x7B,0x60,0x76,0x36,0x6C,0x1B,0xD8,0x0C,0x30,0x03,0xC0 }
- static const unsigned char slash [ ] = {0x06,0x0E,0x1C,0x38,0x70,0xE0,0xC0}
- static const unsigned char line_bmp [ ] = {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x↩ FF,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] = {0x00,0x38,0x38,0x38,0x00,0x00,0x00,0x00,0x00,0x38,0x38,0x38,0x00,0x00}
- static const uint8_t font5x7_dot [7] = {0x00,0x00,0x00,0x00,0x00,0x18,0x18}
- static const uint8_t font8x14_dot [14] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x38}
- static const uint8_t font5x7_comma [7] = {0x00,0x00,0x00,0x00,0x18,0x18,0x30}
- static const uint8_t font8x14_comma [14] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x38,0x70}
- static const uint8_t font5x7_close_quest [7] = {0x3C,0x66,0x06,0x0C,0x18,0x00,0x18}
- static const uint8_t font8x14_close_quest [14] = {0x3C,0x3C,0x66,0x66,0x06,0x06,0x0C,0x0C,0x18,0x18,0x00,0x00,0x18,0x18
- static const uint8_t font5x7_close_excl [7] = {0x18,0x18,0x18,0x18,0x18,0x00,0x18}
- static const uint8_t font8x14_close_excl [14] = {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] = {0x00,0x00,0x00,0x00,0x00,0x00,0x7E,0x7E,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]

6.9.1. Descripción detallada

Funciones que permiten operar sobre el display.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo sh1106_graphics.c.

6.9.2. Documentación de «typedef»

6.9.2.1. bitmap_t

```
typedef struct bitmap_t bitmap_t
```

6.9.3. Documentación de funciones

6.9.3.1. sh1106_draw_arrow()

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

Parámetros

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

Definición en la línea 324 del archivo sh1106_graphics.c.

6.9.3.2. sh1106_draw_bitmap()

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

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

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

Parámetros

out	*oled*	Puntero a la estructura que representa todo lo que se enviará al display
in	*bitmap*	Puntero a la estructura del caracter que se desea escribir en el display

Definición en la línea 199 del archivo sh1106_graphics.c.

6.9.3.3. sh1106_draw_fail()

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

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

Parámetros

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

Definición en la línea 335 del archivo sh1106_graphics.c.

6.9.3.4. 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 diujar una línea del ancho del display
in	*y*	Coordenada en y donde se desea diujar una línea del ancho del display

Definición en la línea 346 del archivo sh1106_graphics.c.

6.9.3.5. sh1106_draw_text()

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

Parámetros

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 diujar el texto de caracteres alfanuméricos
in	*y*	Coordenada en y donde se desea diujar 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.

6.9.3.6. sh1106_draw_utn_logo()

```
void sh1106_draw_utn_logo (
            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 diujar el texto de caracteres alfanuméricos
in	*y*	Coordenada en y donde se desea diujar el texto de caracteres alfanuméricos

Definición en la línea 313 del archivo sh1106_graphics.c.

6.9.4. Documentación de variables

6.9.4.1. arrow_bmp

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

Definición en la línea 18 del archivo sh1106_graphics.c.

6.9.4.2. fail_bmp

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

Definición en la línea 15 del archivo sh1106_graphics.c.

6.9.4.3. font5x7_close_excl

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

Definición en la línea 29 del archivo sh1106_graphics.c.

6.9.4.4. font5x7_close_quest

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

Definición en la línea 27 del archivo sh1106_graphics.c.

6.9.4.5. font5x7_comma

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

Definición en la línea 25 del archivo sh1106_graphics.c.

6.9.4.6. font5x7_dot

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

Definición en la línea 23 del archivo sh1106_graphics.c.

6.9.4.7. font5x7_double_dot

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

Definición en la línea 21 del archivo sh1106_graphics.c.

6.9.4.8. font5x7_minus

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

Definición en la línea 31 del archivo sh1106_graphics.c.

6.9.4.9. font5x7_rowmajor

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

Valor inicial:

```
= {

    {0x1E,0x33,0x33,0x3F,0x33,0x33,0x33},

    {0x3E,0x33,0x33,0x3E,0x33,0x33,0x3E},

    {0x1E,0x33,0x30,0x30,0x30,0x33,0x1E},

    {0x3C,0x36,0x33,0x33,0x33,0x36,0x3C},

    {0x3F,0x30,0x30,0x3E,0x30,0x30,0x3F},

    {0x3F,0x30,0x30,0x3E,0x30,0x30,0x30},

    {0x1E,0x33,0x30,0x37,0x33,0x33,0x1F},

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

    {0x1E,0x0C,0x0C,0x0C,0x0C,0x0C,0x1E},

    {0x0F,0x06,0x06,0x06,0x06,0x36,0x1C},

    {0x33,0x36,0x3C,0x38,0x3C,0x36,0x33},

    {0x30,0x30,0x30,0x30,0x30,0x30,0x3F},

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

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

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

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

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

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

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

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

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

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

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

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

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

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

}
```

Definición en la línea 33 del archivo sh1106_graphics.c.

6.9.4.10. font5x7_rowminor

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

Valor inicial:

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

Definición en la línea 62 del archivo sh1106_graphics.c.

6.9.4.11. font5x7_rownumber

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

Valor inicial:

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

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

Definición en la línea 90 del archivo sh1106_graphics.c.

6.9.4.12. font5x7_space

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

Definición en la línea 19 del archivo sh1106_graphics.c.

6.9.4.13. font8x14_close_excl

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

Definición en la línea 30 del archivo sh1106_graphics.c.

6.9.4.14. font8x14_close_quest

```
const uint8_t font8x14_close_quest[14] = {0x3C,0x3C,0x66,0x66,0x06,0x06,0x0C,0x0C,0x18,0x18,0x00,0x00,0x18,0x1
[static]
```

Definición en la línea 28 del archivo sh1106_graphics.c.

6.9.4.15. font8x14_comma

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

Definición en la línea 26 del archivo sh1106_graphics.c.

6.9.4.16. font8x14_dot

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

Definición en la línea 24 del archivo sh1106_graphics.c.

6.9.4.17. font8x14_double_dot

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

Definición en la línea 22 del archivo sh1106_graphics.c.

6.9.4.18. font8x14_minus

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

Definición en la línea 32 del archivo sh1106_graphics.c.

6.9.4.19. font8x14_rowmajor

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

Valor inicial:

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

Definición en la línea 102 del archivo sh1106_graphics.c.

6.9.4.20. font8x14_rowminor

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

Valor inicial:

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

Definición en la línea 130 del archivo sh1106_graphics.c.

6.9.4.21. font8x14_rownumber

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

Valor inicial:

```
= {
    {0x3C,0x66,0xC3,0xC3,0xC3,0xC3,0xC3,0xC3,0xC3,0xC3,0xC3,0x66,0x3C,0x00},
    {0x18,0x38,0x78,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x7E,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,0xC6,0x03,0x03,0x03,0x03,0xC3,0x66,0x3C,0x00},
    {0x3C,0x66,0xC3,0xC0,0xC0,0xFC,0xC6,0xC3,0xC3,0xC3,0xC3,0x66,0x3C,0x00},
    {0xFF,0xC3,0x03,0x06,0x06,0x0C,0x18,0x18,0x30,0x30,0x60,0x60,0x60,0x00},
    {0x3C,0x66,0xC3,0xC3,0xC3,0x66,0x3C,0x66,0xC3,0xC3,0xC3,0x66,0x3C,0x00},
    {0x3C,0x66,0xC3,0xC3,0xC3,0x67,0x3F,0x03,0x03,0x03,0xC3,0x66,0x3C,0x00}
}
```

Definición en la línea 158 del archivo sh1106_graphics.c.

6.9.4.22. font8x14_space

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

Definición en la línea 20 del archivo sh1106_graphics.c.

6.9.4.23. line_bmp

```
const unsigned char line_bmp[] = {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x↩
FF,0xFF,0xFF,0xFF,0xFF}  [static]
```

Definición en la línea 17 del archivo sh1106_graphics.c.

6.9.4.24. logo16_utn_bmp

```
const unsigned char logo16_utn_bmp[] = {0x71,0x8E,0x71,0x8E,0x79,0x9E,0x3D,0xBC,0x1F,0x↩
F8,0x07,0xE0,0x07,0xE0,0x1F,0xF8,0x3D,0xBC,0x79,0x9E,0x71,0x8E,0x71,0x8E}  [static]
```

Definición en la línea 14 del archivo sh1106_graphics.c.

6.9.4.25. slash

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

Definición en la línea 16 del archivo sh1106_graphics.c.

6.10. sh1106_graphics.c

Ir a la documentación de este archivo.

```
00001
00009 #include <string.h>
00010 #include "esp_err.h"
00011 #include "./sh1106_graphics.h"
00012 #include "../LVFV_system.h"
00013
00014 static const unsigned char logo16_utn_bmp[] =
      {0x71,0x8E,0x71,0x8E,0x79,0x9E,0x3D,0xBC,0x1F,0xF8,0x07,0xE0,0x07,0xE0,0x1F,0xF8,0x3D,0xBC,0x79,0x9E,0x71,0x8E,0x71,0x8E};
      // 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,0xD8
      }; // 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};
      // Línea horizontal 16 x 8
00018 static const unsigned char arrow_bmp[] =
      {0x00,0x20,0x00,0x30,0x00,0x38,0x1F,0xFC,0x00,0x38,0x00,0x30,0x00,0x20};
      // Flecha hacia la derecha 15 x 7
00019 static const uint8_t font5x7_space[7] =            {0x00,0x00,0x00,0x00,0x00,0x00,0x00};
      // Espacio 5 x 7
00020 static const uint8_t font8x14_space[14] =
      {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
      // Espacio 8 x 14
00021 static const uint8_t font5x7_double_dot[7] =       {0x18,0x18,0x00,0x00,0x18,0x18,0x00};
      // Dos puntos 5 x 7
00022 static const uint8_t font8x14_double_dot[14] =
      {0x00,0x38,0x38,0x38,0x00,0x00,0x00,0x00,0x00,0x38,0x38,0x38,0x00,0x00};
      // Dos puntos 8 x 14
00023 static const uint8_t font5x7_dot[7] =               {0x00,0x00,0x00,0x00,0x00,0x18,0x18};
      // Punto 5 x 7
00024 static const uint8_t font8x14_dot[14] =
      {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x38};
      // Punto 8 x 14
00025 static const uint8_t font5x7_comma[7] =             {0x00,0x00,0x00,0x00,0x18,0x18,0x30};
      // Coma 5 x 7
00026 static const uint8_t font8x14_comma[14] =
      {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x38,0x70};
      // 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,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] =
      {0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x00,0x00,0x18,0x18};
      // 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] =
      {0x00,0x00,0x00,0x00,0x00,0x00,0x7E,0x7E,0x00,0x00,0x00,0x00,0x00,0x00};
      // Menos 8 x 14
00033 static const uint8_t font5x7_rowmajor[][7] = {
      // Alfabeto mayúsculas 5 x 7
00034     // A-Z
00035     {0x1E,0x33,0x33,0x3F,0x33,0x33,0x33},
      // A
00036     {0x3E,0x33,0x33,0x3E,0x33,0x33,0x3E},
      // B
00037     {0x1E,0x33,0x30,0x30,0x30,0x33,0x1E},
      // C
00038     {0x3C,0x36,0x33,0x33,0x33,0x36,0x3C},
      // D
00039     {0x3F,0x30,0x30,0x3E,0x30,0x30,0x3F},
      // E
00040     {0x3F,0x30,0x30,0x3E,0x30,0x30,0x30},
      // F
00041     {0x1E,0x33,0x30,0x37,0x33,0x33,0x1F},
      // G
00042     {0x33,0x33,0x33,0x3F,0x33,0x33,0x33},
      // H
00043     {0x1E,0x0C,0x0C,0x0C,0x0C,0x0C,0x1E},
      // I
00044     {0x0F,0x06,0x06,0x06,0x06,0x36,0x1C},
      // J
00045     {0x33,0x36,0x3C,0x38,0x3C,0x36,0x33},
      // K
00046     {0x30,0x30,0x30,0x30,0x30,0x30,0x3F},
      // L
```

```
00047     {0x33,0x3F,0x3F,0x33,0x33,0x33,0x33},
      // M
00048     {0x33,0x3B,0x3F,0x37,0x33,0x33,0x33},
      // N
00049     {0x1E,0x33,0x33,0x33,0x33,0x33,0x1E},
      // O
00050     {0x3E,0x33,0x33,0x3E,0x30,0x30,0x30},
      // P
00051     {0x1E,0x33,0x33,0x33,0x37,0x36,0x1D},
      // Q
00052     {0x3E,0x33,0x33,0x3E,0x3C,0x36,0x33},
      // R
00053     {0x1E,0x33,0x30,0x1E,0x03,0x33,0x1E},
      // S
00054     {0x3F,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C},
      // T
00055     {0x33,0x33,0x33,0x33,0x33,0x33,0x1E},
      // U
00056     {0x33,0x33,0x33,0x33,0x33,0x1E,0x0C},
      // V
00057     {0x33,0x33,0x33,0x33,0x3F,0x3F,0x33},
      // W
00058     {0x33,0x33,0x1E,0x0C,0x1E,0x33,0x33},
      // X
00059     {0x33,0x33,0x33,0x1E,0x0C,0x0C,0x0C},
      // Y
00060     {0x3F,0x03,0x06,0x0C,0x18,0x30,0x3F}
      // Z
00061 };
00062 static const uint8_t font5x7_rowminor[][7] = {
      // Alfabeto minúsculas 5 x 7
00063     {0x00,0x00,0x3C,0x06,0x3E,0x66,0x3E},
      // a
00064     {0x60,0x60,0x7C,0x66,0x66,0x66,0x7C},
      // b
00065     {0x00,0x00,0x3C,0x66,0x60,0x66,0x3C},
      // c
00066     {0x06,0x06,0x3E,0x66,0x66,0x66,0x3E},
      // d
00067     {0x00,0x00,0x3C,0x66,0x7E,0x60,0x3C},
      // e
00068     {0x1C,0x30,0x30,0x7C,0x30,0x30,0x30},
      // f
00069     {0x00,0x3E,0x66,0x66,0x3E,0x06,0x7C},
      // g
00070     {0x60,0x60,0x7C,0x66,0x66,0x66,0x66},
      // h
00071     {0x18,0x00,0x38,0x18,0x18,0x18,0x3C},
      // i
00072     {0x06,0x00,0x06,0x06,0x66,0x66,0x3C},
      // j
00073     {0x60,0x60,0x66,0x6C,0x78,0x6C,0x66},
      // k
00074     {0x38,0x18,0x18,0x18,0x18,0x18,0x3C},
      // l
00075     {0x00,0x00,0x6C,0x7E,0x7E,0x6C,0x6C},
      // m
00076     {0x00,0x00,0x7C,0x66,0x66,0x66,0x66},
      // n
00077     {0x00,0x00,0x3C,0x66,0x66,0x66,0x3C},
      // o
00078     {0x00,0x7C,0x66,0x66,0x7C,0x60,0x60},
      // p
00079     {0x00,0x3E,0x66,0x66,0x3E,0x06,0x06},
      // q
00080     {0x00,0x00,0x6C,0x76,0x60,0x60,0x60},
      // r
00081     {0x00,0x00,0x3E,0x60,0x3C,0x06,0x7C},
      // s
00082     {0x30,0x30,0x7C,0x30,0x30,0x30,0x1C},
      // t
00083     {0x00,0x00,0x66,0x66,0x66,0x66,0x3E},
      // u
00084     {0x00,0x00,0x66,0x66,0x66,0x3C,0x18},
      // v
00085     {0x00,0x00,0x66,0x66,0x7E,0x7E,0x6C},
      // w
00086     {0x00,0x00,0x66,0x3C,0x18,0x3C,0x66},
      // x
00087     {0x00,0x66,0x66,0x66,0x3E,0x06,0x7C},
      // y
00088     {0x00,0x7E,0x0C,0x18,0x30,0x7E,0x00}
      // z
00089 };
00090 static const uint8_t font5x7_rownumber[][7] = {
      // Números 5 x 7
00091     {0x1E,0x33,0x37,0x3B,0x33,0x33,0x1E},
```

```
      // 0
00092     {0x0C,0x1C,0x0C,0x0C,0x0C,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,0x33,0x33,0x33,0x33,0x3F,0x3F,0x33,0x33,0x33,0x33,0x33,0x33},
      // A
00104     {0x3E,0x3E,0x33,0x33,0x33,0x33,0x3E,0x3E,0x33,0x33,0x33,0x33,0x3E,0x3E},
      // B
00105     {0x1E,0x1E,0x33,0x33,0x30,0x30,0x30,0x30,0x30,0x30,0x33,0x33,0x1E,0x1E},
      // C
00106     {0x3C,0x3C,0x36,0x36,0x33,0x33,0x33,0x33,0x33,0x33,0x36,0x36,0x3C,0x3C},
      // D
00107     {0x3F,0x3F,0x30,0x30,0x30,0x30,0x3E,0x3E,0x30,0x30,0x30,0x30,0x3F,0x3F},
      // E
00108     {0x3F,0x3F,0x30,0x30,0x30,0x30,0x3E,0x3E,0x30,0x30,0x30,0x30,0x30,0x30},
      // F
00109     {0x1E,0x1E,0x33,0x33,0x30,0x30,0x37,0x37,0x33,0x33,0x33,0x33,0x1F,0x1F},
      // G
00110     {0x33,0x33,0x33,0x33,0x33,0x33,0x3F,0x3F,0x33,0x33,0x33,0x33,0x33,0x33},
      // H
00111     {0x1E,0x1E,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x1E,0x1E},
      // I
00112     {0x0F,0x0F,0x06,0x06,0x06,0x06,0x06,0x06,0x06,0x06,0x36,0x36,0x1C,0x1C},
      // J
00113     {0x33,0x33,0x36,0x36,0x3C,0x3C,0x38,0x38,0x3C,0x3C,0x36,0x36,0x33,0x33},
      // K
00114     {0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x3F,0x3F},
      // L
00115     {0x33,0x33,0x3F,0x3F,0x3F,0x3F,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33},
      // M
00116     {0x33,0x33,0x3B,0x3B,0x3F,0x3F,0x37,0x37,0x33,0x33,0x33,0x33,0x33,0x33},
      // N
00117     {0x1E,0x1E,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x1E,0x1E},
      // O
00118     {0x3E,0x3E,0x33,0x33,0x33,0x33,0x3E,0x3E,0x30,0x30,0x30,0x30,0x30,0x30},
      // P
00119     {0x1E,0x1E,0x33,0x33,0x33,0x33,0x33,0x33,0x37,0x37,0x36,0x36,0x1D,0x1D},
      // Q
00120     {0x3E,0x3E,0x33,0x33,0x33,0x33,0x3E,0x3E,0x3C,0x3C,0x36,0x36,0x33,0x33},
      // R
00121     {0x1E,0x1E,0x33,0x33,0x30,0x30,0x1E,0x1E,0x03,0x03,0x33,0x33,0x1E,0x1E},
      // S
00122     {0x3F,0x3F,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C},
      // T
00123     {0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x1E,0x1E},
      // U
00124     {0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x33,0x1E,0x1E,0x0C,0x0C},
      // V
00125     {0x33,0x33,0x33,0x33,0x33,0x33,0x3F,0x3F,0x3F,0x3F,0x33,0x33,0x33,0x33},
      // W
00126     {0x33,0x33,0x33,0x33,0x1E,0x1E,0x0C,0x0C,0x1E,0x1E,0x33,0x33,0x33,0x33},
      // X
00127     {0x33,0x33,0x33,0x33,0x1E,0x1E,0x0C,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},
      // f
00137     {0x00,0x00,0x3E,0x3E,0x66,0x66,0x66,0x66,0x3E,0x3E,0x06,0x06,0x7C,0x7C},
      // g
00138     {0x60,0x60,0x60,0x60,0x7C,0x7C,0x66,0x66,0x66,0x66,0x66,0x66,0x66,0x66},
      // h
00139     {0x18,0x18,0x00,0x00,0x38,0x38,0x18,0x18,0x18,0x18,0x18,0x18,0x3C,0x3C},
      // i
00140     {0x06,0x06,0x00,0x00,0x06,0x06,0x06,0x06,0x06,0x06,0x66,0x66,0x3C,0x3C},
      // j
00141     {0x60,0x60,0x60,0x60,0x66,0x66,0x6C,0x6C,0x78,0x78,0x6C,0x6C,0x66,0x66},
      // k
00142     {0x38,0x38,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x3C,0x3C},
      // l
00143     {0x00,0x00,0x00,0x00,0x6C,0x6C,0x7E,0x7E,0x7E,0x7E,0x6C,0x6C,0x6C,0x6C},
      // m
00144     {0x00,0x00,0x00,0x00,0x7C,0x7C,0x66,0x66,0x66,0x66,0x66,0x66,0x66,0x66},
      // n
00145     {0x00,0x00,0x00,0x00,0x3C,0x3C,0x66,0x66,0x66,0x66,0x66,0x66,0x3C,0x3C},
      // o
00146     {0x00,0x00,0x7C,0x7C,0x66,0x66,0x66,0x66,0x7C,0x7C,0x60,0x60,0x60,0x60},
      // p
00147     {0x00,0x00,0x3E,0x3E,0x66,0x66,0x66,0x66,0x3E,0x3E,0x06,0x06,0x06,0x06},
      // q
00148     {0x00,0x00,0x00,0x00,0x6C,0x6C,0x76,0x76,0x60,0x60,0x60,0x60,0x60,0x60},
      // r
00149     {0x00,0x00,0x00,0x00,0x3E,0x3E,0x60,0x60,0x3C,0x3C,0x06,0x06,0x7C,0x7C},
      // s
00150     {0x30,0x30,0x30,0x30,0x7C,0x7C,0x30,0x30,0x30,0x30,0x30,0x30,0x1C,0x1C},
      // t
00151     {0x00,0x00,0x00,0x00,0x66,0x66,0x66,0x66,0x66,0x66,0x66,0x66,0x3E,0x3E},
      // u
00152     {0x00,0x00,0x00,0x00,0x66,0x66,0x66,0x66,0x66,0x66,0x3C,0x3C,0x18,0x18},
      // v
00153     {0x00,0x00,0x00,0x00,0x66,0x66,0x66,0x66,0x7E,0x7E,0x7E,0x7E,0x6C,0x6C},
      // w
00154     {0x00,0x00,0x00,0x00,0x66,0x66,0x3C,0x3C,0x18,0x18,0x3C,0x3C,0x66,0x66},
      // x
00155     {0x00,0x00,0x66,0x66,0x66,0x66,0x66,0x66,0x3E,0x3E,0x06,0x06,0x7C,0x7C},
      // y
00156     {0x00,0x00,0x00,0x00,0x7E,0x7E,0x0C,0x0C,0x18,0x18,0x30,0x30,0x7E,0x7E},
      // z
00157 };
00158 static const uint8_t font8x14_rownumber[][14] = {
      // Números 8 x 14
00159     {0x3C,0x66,0xC3,0xC3,0xC3,0xC3,0xC3,0xC3,0xC3,0xC3,0x66,0x3C,0x00},
      // 0
00160     {0x18,0x38,0x78,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x7E,0x00},
      // 1
00161     {0x3C,0x66,0xC3,0x03,0x06,0x0C,0x18,0x30,0x60,0xC0,0xC3,0xFF,0xFE,0x00},
      // 2
00162     {0x7E,0xC3,0x03,0x03,0x06,0x3C,0x06,0x03,0x03,0x03,0xC3,0x66,0x3C,0x00},
      // 3
00163     {0x06,0x0E,0x1E,0x36,0x66,0xC6,0x86,0xFF,0xFF,0x06,0x06,0x06,0x06,0x00},
      // 4
00164     {0xFE,0xC0,0xC0,0xC0,0xFC,0xC6,0x03,0x03,0x03,0x03,0xC3,0x66,0x3C,0x00},
      // 5
00165     {0x3C,0x66,0xC3,0xC0,0xC0,0xFC,0xC6,0xC3,0xC3,0xC3,0xC3,0x66,0x3C,0x00},
      // 6
00166     {0xFF,0xC3,0x03,0x06,0x06,0x0C,0x18,0x18,0x30,0x30,0x60,0x60,0x60,0x00},
      // 7
00167     {0x3C,0x66,0xC3,0xC3,0xC3,0x66,0x3C,0x66,0xC3,0xC3,0xC3,0x66,0x3C,0x00},
      // 8
00168     {0x3C,0x66,0xC3,0xC3,0xC3,0x67,0x3F,0x03,0x03,0x03,0xC3,0x66,0x3C,0x00}
      // 9
00169 };
00170
00176 typedef struct bitmap_t {
00177     const uint8_t *bitmap;
00178     uint8_t w;
00179     uint8_t h;
00180     int x;
00181     int y;
00182 } bitmap_t;
00183
00197 static void sh1106_draw_bitmap( sh1106_t *oled, bitmap_t *bitmap);
00198
00199 static void sh1106_draw_bitmap( sh1106_t *oled, bitmap_t *bitmap) {
00200     // Dibujado sencillo: copia bit por bit en el buffer
00201     // (Implementar según formato del bitmap)
00202     for (uint8_t by = 0; by < bitmap->h; by++) {
00203         for (uint8_t bx = 0; bx < bitmap->w; bx++) {
00204             uint8_t byte = bitmap->bitmap[by * ((bitmap->w + 7) / 8) + (bx / 8)];
00205             uint8_t bit = (byte » (7 - (bx % 8))) & 0x01;
00206             if (bit) {
00207                 int px = bitmap->x + bx;
00208                 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 y_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 caracter no soportado
00303             if ( size == SH1106_SIZE_1 ) {
00304                 bitmap.bitmap = font5x7_space;
00305             } else if ( size == SH1106_SIZE_2 ) {
00306                 bitmap.bitmap = font8x14_space;
00307             }
00308         }
00309         sh1106_draw_bitmap( oled, &bitmap);
00310     }
00311 }
00312
00313 void sh1106_draw_utn_logo( sh1106_t *oled, int x, int y) {
00314     bitmap_t bitmap = {
00315         .bitmap = logo16_utn_bmp,
00316         .w = 16,
00317         .h = 12,
00318         .x = x,
00319         .y = y
00320     };
00321     sh1106_draw_bitmap( oled, &bitmap);
00322 }
00323
00324 void sh1106_draw_arrow( sh1106_t *oled, int x, int y) {
00325     bitmap_t bitmap = {
00326         .bitmap = arrow_bmp,
00327         .w = 16,
00328         .h = 7,
00329         .x = x,
00330         .y = y
00331     };
00332     sh1106_draw_bitmap( oled, &bitmap);
00333 }
00334
00335 void sh1106_draw_fail( sh1106_t *oled ) {
00336     bitmap_t bitmap = {
00337         .bitmap = fail_bmp,
00338         .w = 16,
00339         .h = 14,
00340         .x = 109,
00341         .y = 1
00342     };
00343     sh1106_draw_bitmap( oled, &bitmap);
00344 }
00345
00346 void sh1106_draw_line( sh1106_t *oled, int x, int y) {
00347     bitmap_t bitmap = {
00348         .bitmap = line_bmp,
00349         .w = 128,
00350         .h = 1,
00351         .x = x,
00352         .y = y
00353     };
00354     sh1106_draw_bitmap( oled, &bitmap);
00355 }
```

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

Declaración de funciones que permiten operar sobre el display.

Estructuras de datos

- struct sh1106_t

typedefs

- typedef struct sh1106_t sh1106_t

Funciones

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

  Función que dibuja un signo de exclamación en la esquina superior derecha de la pantalla para expresar una falla en el sistema.
- void sh1106_draw_line (sh1106_t *oled, int x, int y)

6.11.1. Descripción detallada

Declaración de funciones que permiten operar sobre el display.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo sh1106_graphics.h.

6.11.2. Documentación de «typedef»

6.11.2.1. sh1106_t

```
typedef struct sh1106_t sh1106_t
```

6.11.3. Documentación de funciones

6.11.3.1. sh1106_draw_arrow()

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

Parámetros

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

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

Definición en la línea 324 del archivo sh1106_graphics.c.

6.11.3.2. sh1106_draw_fail()

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

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

Parámetros

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

Definición en la línea 335 del archivo sh1106_graphics.c.

6.11.3.3. sh1106_draw_line()

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

Parámetros

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

Definición en la línea 346 del archivo sh1106_graphics.c.

6.11.3.4. sh1106_draw_text()

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

Parámetros

out	*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 diujar el texto de caracteres alfanuméricos
in	*y*	Coordenada en y donde se desea diujar 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.

6.11.3.5. sh1106_draw_utn_logo()

```
void sh1106_draw_utn_logo (
            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 diujar el texto de caracteres alfanuméricos
in	*y*	Coordenada en y donde se desea diujar el texto de caracteres alfanuméricos

Definición en la línea 313 del archivo sh1106_graphics.c.

6.12. sh1106_graphics.h

Ir a la documentación de este archivo.

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

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

Funciones de hardware para el puerto I2C.

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

defines

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

Funciones

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

6.13.1. Descripción detallada

Funciones de hardware para el puerto I2C.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo sh1106_i2c.c.

6.13.2. Documentación de «define»

6.13.2.1. CMD_CONTROL

```
#define CMD_CONTROL 0x00
```

Definición en la línea 16 del archivo sh1106_i2c.c.

6.13.2.2. DATA_CONTROL

```
#define DATA_CONTROL 0x40
```

Definición en la línea 17 del archivo sh1106_i2c.c.

6.13.2.3. I2C_DISPLAY

```
#define I2C_DISPLAY I2C_NUM_0
```

Definición en la línea 14 del archivo sh1106_i2c.c.

6.13.2.4. SH1106_ADDR

```
#define SH1106_ADDR 0x3C
```

Definición en la línea 15 del archivo sh1106_i2c.c.

6.13.3. Documentación de funciones

6.13.3.1. sh1106_write()

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

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

Parámetros

in	*data*	Dato que se desea escribir en el display
in	*len*	Largo del dato que se quiere escribir en el display
in	*comm_type*	Tipo de comunuicación que se desea: Dato o Comando

Valores devueltos

Definición en la línea 19 del archivo sh1106_i2c.c.

6.14. sh1106_i2c.c

Ir a la documentación de este archivo.

```
00001
00009 #include "freertos/FreeRTOS.h"
00010 #include "esp_err.h"
00011 #include "driver/i2c.h"
00012 #include "./sh1106_i2c.h"
00013
00014 #define I2C_DISPLAY     I2C_NUM_0       // Npumero de puerto I2C utilizado para el display
00015 #define SH1106_ADDR     0x3C            // Addess 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     return ret;
00036 }
```

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

Declaración de funciones de hardware para el puerto I2C.

Enumeraciones

- enum sh1106_comm_type_t { SH1106_COMM_CMD = 0 , SH1106_COMM_DATA }
  Tipo de comunicación I2C con el display SH1106: Comando o Dato.

Funciones

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

6.15.1. Descripción detallada

Declaración de funciones de hardware para el puerto I2C.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo sh1106_i2c.h.

6.15.2. Documentación de enumeraciones

6.15.2.1. sh1106_comm_type_t

```
enum sh1106_comm_type_t
```

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

Valores de enumeraciones

SH1106_COMM_CMD	
SH1106_COMM_DATA	

Definición en la línea 18 del archivo sh1106_i2c.h.

6.15.3. Documentación de funciones

6.15.3.1. sh1106_write()

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

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

Parámetros

in	*data*	Dato que se desea escribir en el display
in	*len*	Largo del dato que se quiere escribir en el display
in	*comm_type*	Tipo de comunuicación que se desea: Dato o Comando

Valores devueltos

Definición en la línea 19 del archivo sh1106_i2c.c.

6.16. sh1106_i2c.h

Ir a la documentación de este archivo.

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

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

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

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

defines

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

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

Funciones

- static esp_err_t freq_0_10V_output ()

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

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

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

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

  Tarea que espera comandos a través de la cola relay_evt_queue para activar o desactivar el relay.
- void IRAM_ATTR gpio_isr_handler (void ∗arg)
- esp_err_t set_freq_output (uint16_t freq)

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

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

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

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

Variables

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

6.17.1. Descripción detallada

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

Las tareas de relay y buzzer son tareas aisladas del resto del sistema y son accedidas a través de las funciones RelayEvantPost y BuzzerEvantPost. Esto genera que no sea necesaria que la cola de comandos deba ser utilizada en diversos archivos, pretege 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 contínua, la señal que representa indirectamente la frecuencia de operación del motor.

Versión

2.0

Fecha

2025-11-06

Definición en el archivo io_control.c.

6.17.2. Documentación de «define»

6.17.2.1. BUZZER_PIN

```
#define BUZZER_PIN 0
```

Pin del puerto A que controla el buzzer de la HMI

Definición en la línea 41 del archivo io_control.c.

6.17.2.2. DUTY_MAX_PCT

```
#define DUTY_MAX_PCT 98
```

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

Definición en la línea 84 del archivo io_control.c.

6.17.2.3. DUTY_MIN_PCT

```
#define DUTY_MIN_PCT 54
```

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

Definición en la línea 83 del archivo io_control.c.

6.17.2.4. FREQ_SEL_1_PIN

```
#define FREQ_SEL_1_PIN 2
```

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

Definición en la línea 43 del archivo io_control.c.

6.17.2.5. FREQ_SEL_2_PIN

```
#define FREQ_SEL_2_PIN 3
```

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

Definición en la línea 44 del archivo io_control.c.

6.17.2.6. FREQ_SEL_3_PIN

```
#define FREQ_SEL_3_PIN 4
```

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

Definición en la línea 45 del archivo io_control.c.

6.17.2.7. FREQ_SEL_MASK

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

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

Definición en la línea 49 del archivo io_control.c.

6.17.2.8. INT_A_PIN

```
#define INT_A_PIN GPIO_NUM_22
```

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

Definición en la línea 39 del archivo io_control.c.

6.17.2.9. INT_B_PIN

```
#define INT_B_PIN GPIO_NUM_23
```

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

Definición en la línea 53 del archivo io_control.c.

6.17.2.10. MATRIX_SW1

```
#define MATRIX_SW1 0x17
```

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

Definición en la línea 55 del archivo io_control.c.

6.17.2.11. MATRIX_SW2

```
#define MATRIX_SW2 0x27
```

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

Definición en la línea 56 del archivo io_control.c.

6.17.2.12. MATRIX_SW3

```
#define MATRIX_SW3 0x1B
```

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

Definición en la línea 57 del archivo io_control.c.

6.17.2.13. MATRIX_SW4

```
#define MATRIX_SW4 0x2B
```

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

Definición en la línea 58 del archivo io_control.c.

6.17.2.14. MATRIX_SW5

```
#define MATRIX_SW5 0x1D
```

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

Definición en la línea 59 del archivo io_control.c.

6.17.2.15. MATRIX_SW6

```
#define MATRIX_SW6 0x2D
```

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

Definición en la línea 60 del archivo io_control.c.

6.17.2.16. MATRIX_SW7

```
#define MATRIX_SW7 0x1E
```

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

Definición en la línea 61 del archivo io_control.c.

6.17.2.17. MATRIX_SW8

```
#define MATRIX_SW8 0x2E
```

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

Definición en la línea 62 del archivo io_control.c.

6.17.2.18. PWM_CHANNEL

```
#define PWM_CHANNEL LEDC_CHANNEL_0
```

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

Definición en la línea 80 del archivo io_control.c.

6.17.2.19. PWM_FREQ_HZ

```
#define PWM_FREQ_HZ 1000
```

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

Definición en la línea 77 del archivo io_control.c.

6.17.2.20. PWM_GPIO

```
#define PWM_GPIO 4
```

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

Definición en la línea 76 del archivo io_control.c.

6.17.2.21. PWM_MODE

```
#define PWM_MODE LEDC_LOW_SPEED_MODE
```

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

Definición en la línea 79 del archivo io_control.c.

6.17.2.22. PWM_RES

```
#define PWM_RES LEDC_TIMER_10_BIT
```

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

Definición en la línea 81 del archivo io_control.c.

6.17.2.23. PWM_STEP_MS

```
#define PWM_STEP_MS 250
```

Cambio cada 500 ms

Definición en la línea 82 del archivo io_control.c.

6.17.2.24. PWM_TIMER

```
#define PWM_TIMER LEDC_TIMER_0
```

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

Definición en la línea 78 del archivo io_control.c.

6.17.2.25. RELAY_PIN

```
#define RELAY_PIN 1
```

Pin del puerto A que controla el relé de la HMI

Definición en la línea 42 del archivo io_control.c.

6.17.2.26. START_BUTTON_PIN

```
#define START_BUTTON_PIN GPIO_NUM_17
```

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

Definición en la línea 74 del archivo io_control.c.

6.17.2.27. STOP_BUTTON_PIN

```
#define STOP_BUTTON_PIN GPIO_NUM_16
```

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

Definición en la línea 73 del archivo io_control.c.

6.17.2.28. STOP_PIN

```
#define STOP_PIN 5
```

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

Definición en la línea 46 del archivo io_control.c.

6.17.2.29. STOP_SW_MASK

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

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

Definición en la línea 51 del archivo io_control.c.

6.17.2.30. SWITCH_BUTTON_BACK

```
#define SWITCH_BUTTON_BACK MATRIX_SW1
```

Asignación de boton al SW1 representado por 0x17

Definición en la línea 64 del archivo io_control.c.

6.17.2.31. SWITCH_BUTTON_DOWN

```
#define SWITCH_BUTTON_DOWN MATRIX_SW5
```

Asignación de boton al SW5 representado por 0x1D

Definición en la línea 68 del archivo io_control.c.

6.17.2.32. SWITCH_BUTTON_LEFT

```
#define SWITCH_BUTTON_LEFT MATRIX_SW3
```

Asignación de boton al SW3 representado por 0x1B

Definición en la línea 66 del archivo io_control.c.

6.17.2.33. SWITCH_BUTTON_MENU

```
#define SWITCH_BUTTON_MENU MATRIX_SW8
```

Asignación de boton al SW8 representado por 0x2E

Definición en la línea 71 del archivo io_control.c.

6.17.2.34. SWITCH_BUTTON_OK

```
#define SWITCH_BUTTON_OK MATRIX_SW7
```

Asignación de boton al SW7 representado por 0x1E

Definición en la línea 70 del archivo io_control.c.

6.17.2.35. SWITCH_BUTTON_RIGHT

```
#define SWITCH_BUTTON_RIGHT MATRIX_SW4
```

Asignación de boton al SW4 representado por 0x2B

Definición en la línea 67 del archivo io_control.c.

6.17.2.36. SWITCH_BUTTON_SAVE

```
#define SWITCH_BUTTON_SAVE MATRIX_SW6
```

Asignación de boton al SW6 representado por 0x2D

Definición en la línea 69 del archivo io_control.c.

6.17.2.37. SWITCH_BUTTON_UP

```
#define SWITCH_BUTTON_UP MATRIX_SW2
```

Asignación de boton al SW2 representado por 0x27

Definición en la línea 65 del archivo io_control.c.

6.17.2.38. TERMO_SW_MASK

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

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

Definición en la línea 50 del archivo io_control.c.

6.17.2.39. TERMO_SW_PIN

```
#define TERMO_SW_PIN 6
```

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

Definición en la línea 47 del archivo io_control.c.

6.17.3. Documentación de funciones

6.17.3.1. BuzzerEvantPost()

```
esp_err_t BuzzerEvantPost (
            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 mili segundos durante el que el buzzer sonará.

Devuelve

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

Definición en la línea 601 del archivo io_control.c.

6.17.3.2. freq_0_10V_output()

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

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

Devuelve

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

Definición en la línea 174 del archivo io_control.c.

6.17.3.3. gpio_init_interrupts()

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

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

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

Devuelve

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

<

<

Definición en la línea 211 del archivo io_control.c.

6.17.3.4. GPIO_interrupt_attendance_task()

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

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

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

Definición en la línea 386 del archivo io_control.c.

6.17.3.5. gpio_isr_handler()

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

Definición en la línea 164 del archivo io_control.c.

6.17.3.6. MCP23017_buzzer_control()

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

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

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

Definición en la línea 289 del archivo io_control.c.

6.17.3.7. MCP23017_keyboard_control()

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

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

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

Definición en la línea 313 del archivo io_control.c.

6.17.3.8. MCP23017_relay_control()

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

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

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

Definición en la línea 334 del archivo io_control.c.

6.17.3.9. RelayEvantPost()

```
esp_err_t RelayEvantPost (
            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	*relay_event*	Estado del relay

Devuelve

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

Definición en la línea 593 del archivo io_control.c.

6.17.3.10. set_freq_output()

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

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

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

Parámetros

in	*freq*	Es la frecuencia que está girando el motor.

Devuelve

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

Definición en la línea 359 del archivo io_control.c.

6.17.4. Documentación de variables

6.17.4.1. buzzer_evt_queue

```
QueueHandle_t buzzer_evt_queue = NULL
```

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

Definición en la línea 86 del archivo io_control.c.

6.17.4.2. GPIO_evt_queue

```
QueueHandle_t GPIO_evt_queue = NULL
```

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

Definición en la línea 88 del archivo io_control.c.

6.17.4.3. mcp_porta

```
uint8_t mcp_porta
```

Estado de entradas del puerto A del MCP23017

Definición en la línea 90 del archivo io_control.c.

6.17.4.4. mcp_portb

```
uint8_t mcp_portb
```

Estado de entradas del puerto B del MCP23017

Definición en la línea 91 del archivo io_control.c.

6.17.4.5. relay_evt_queue

```
QueueHandle_t relay_evt_queue = NULL
```

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

Definición en la línea 87 del archivo io_control.c.

6.17.4.6. TAG

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

Varaible que se imprime en los ESP_LOG

Definición en la línea 37 del archivo io_control.c.

6.18. io_control.c

Ir a la documentación de este archivo.

```
00001
00012 #include "sdkconfig.h"
00013
00014 #include "esp_err.h"
00015 #include "esp_log.h"
00016
00017 #include "driver/gpio.h"
00018 #include "driver/ledc.h"
00019 #include "display/display.h"
00020
00021 #include "freertos/FreeRTOS.h"
00022 #include "freertos/semphr.h"
00023 #include "freertos/task.h"
00024 #include "freertos/queue.h"
00025
00026 #include <inttypes.h>
00027 #include <stdio.h>
00028 #include <stdlib.h>
00029
00030 #include "mcp23017_defs.h"
00031
00032 #include "./MCP23017.h"
```

```
00033 #include "./io_control.h"
00034 #include "../system/sysAdmin.h"
00035 #include "../system/sysControl.h"
00036
00037 static const char *TAG = "IO_CTRL";

00038
00039 #define INT_A_PIN                   GPIO_NUM_22

00040
00041 #define BUZZER_PIN                  0

00042 #define RELAY_PIN                   1

00043 #define FREQ_SEL_1_PIN              2

00044 #define FREQ_SEL_2_PIN              3

00045 #define FREQ_SEL_3_PIN              4

00046 #define STOP_PIN                    5

00047 #define TERMO_SW_PIN                6

00048
00049 #define FREQ_SEL_MASK               ( (1 « FREQ_SEL_3_PIN) | (1 « FREQ_SEL_2_PIN) | (1 «
      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     gpio_config_t io_conf;

00215
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,
    INT_B_PIN);
00258 
00259     // Configuración de pines
00260     io_conf_stop_start.intr_type = GPIO_INTR_ANYEDGE;
00261     io_conf_stop_start.mode = GPIO_MODE_INPUT;
00262     io_conf_stop_start.pin_bit_mask = (1ULL<<STOP_BUTTON_PIN) | (1ULL<<START_BUTTON_PIN);
00263     io_conf_stop_start.pull_down_en = GPIO_PULLDOWN_DISABLE;
00264     io_conf_stop_start.pull_up_en = GPIO_PULLUP_DISABLE;
00265 
00266     retval = gpio_config(&io_conf_stop_start);
00267     if ( retval != ESP_OK ) {
00268         ESP_LOGE(TAG, "Error al configurar los GPIO para botones aislados stop y start");
00269         return retval;
00270     }
00271 
00272     // Asocio pines a la rutina ISR
00273     retval = gpio_isr_handler_add(STOP_BUTTON_PIN,  gpio_isr_handler, (void*) (uintptr_t)
    STOP_BUTTON_PIN);
00274     if ( retval != ESP_OK ) {
00275         ESP_LOGE(TAG, "Error al configurar la interrupción de GPIO para botones aislados stop");
00276         return retval;
00277     }
00278 
00279     retval = gpio_isr_handler_add(START_BUTTON_PIN, gpio_isr_handler, (void*) (uintptr_t)
    START_BUTTON_PIN);
00280     if ( retval != ESP_OK ) {
00281         ESP_LOGE(TAG, "Error al configurar la interrupción de GPIO para botones aislados start");
00282         return retval;
00283     }
00284 
00285     ESP_LOGI(TAG, "Interrupciones GPIO configuradas en %d (Stop) y %d (Start)", STOP_BUTTON_PIN,
    START_BUTTON_PIN);
00286     return ESP_OK;
00287 }
00288 
00289 static void MCP23017_buzzer_control(void* arg) {
00290 
00291     uint32_t delay_time;
00292     uint16_t buzzer_time;
00293 
00294     if (buzzer_evt_queue == NULL) {
00295         buzzer_evt_queue = xQueueCreate(10, sizeof(uint32_t));
00296         if (buzzer_evt_queue == NULL) {
00297             ESP_LOGE(TAG, "No se pudo crear la cola de buzzer");
00298             return;
00299         }
00300     }
00301 
00302     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;
    // 00ss ss00
00427                     speed_selector = ( speed_selector » ( FREQ_SEL_1_PIN ) );
    // 0000 ssss
00428                     last_interrupt = SPEED_SELECTOR_0 + speed_selector;
00429                 } else if ( scratch_data & STOP_SW_MASK ) {
00430                     if ( mcp_porta & STOP_SW_MASK ) {
00431                         last_interrupt = STOP_RELEASED;
00432                     } else {
00433                         last_interrupt = STOP_PRESSED;
00434                     }
00435                 }
00436             } else if ( io_num == INT_B_PIN ) {
00437                 ESP_ERROR_CHECK_WITHOUT_ABORT( mcp_read_port(PORTB, &mcp_portb) );
00438                 switch (mcp_portb){
00439                     case SWITCH_BUTTON_MENU:
00440                         last_interrupt = BUTTON_MENU;
00441                         break;
00442                     case SWITCH_BUTTON_DOWN:
00443                         last_interrupt = BUTTON_DOWN;
00444                         break;
00445                     case SWITCH_BUTTON_RIGHT:
00446                         last_interrupt = BUTTON_RIGHT;
00447                         break;
00448                     case SWITCH_BUTTON_LEFT:
00449                         last_interrupt = BUTTON_LEFT;
00450                         break;
00451                     case SWITCH_BUTTON_UP:
00452                         last_interrupt = BUTTON_UP;
00453                         break;
00454                     case SWITCH_BUTTON_OK:
00455                         last_interrupt = BUTTON_OK;
00456                         break;
00457                     case SWITCH_BUTTON_SAVE:
00458                         last_interrupt = BUTTON_SAVE;
00459                         break;
00460                     case SWITCH_BUTTON_BACK:
00461                         last_interrupt = BUTTON_BACK;
00462                         break;
00463                 }
00464             } else if ( io_num == STOP_BUTTON_PIN ) {
00465                 if ( gpio_get_level(io_num) ) {
00466                     last_interrupt = EMERGENCI_STOP_PRESSED;
00467                 } else {
00468                     last_interrupt = EMERGENCI_STOP_RELEASED;
00469                 }
00470             } else if ( io_num == START_BUTTON_PIN ) {
00471                 if ( !gpio_get_level(io_num) ) {
00472                     last_interrupt = START_PRESSED;
00473                 } else {
00474                     last_interrupt = START_RELEASED;
```

```
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                     BuzzerEvantPost( 50 );
00487                 }
00488                 break;
00489             case BUTTON_DOWN:
00490                 if ( mcp_portb == SWITCH_BUTTON_DOWN ) {
00491                     DisplayEventPost(last_interrupt);
00492                     BuzzerEvantPost( 50 );
00493                 }
00494                 break;
00495             case BUTTON_RIGHT:
00496                 if ( mcp_portb == SWITCH_BUTTON_RIGHT ) {
00497                     DisplayEventPost(last_interrupt);
00498                     BuzzerEvantPost( 50 );
00499                 }
00500                 break;
00501             case BUTTON_LEFT:
00502                 if ( mcp_portb == SWITCH_BUTTON_LEFT ) {
00503                     DisplayEventPost(last_interrupt);
00504                     BuzzerEvantPost( 50 );
00505                 }
00506                 break;
00507             case BUTTON_UP:
00508                 if ( mcp_portb == SWITCH_BUTTON_UP ) {
00509                     DisplayEventPost(last_interrupt);
00510                     BuzzerEvantPost( 50 );
00511                 }
00512                 break;
00513             case BUTTON_OK:
00514                 if ( mcp_portb == SWITCH_BUTTON_OK ) {
00515                     DisplayEventPost(last_interrupt);
00516                     BuzzerEvantPost( 50 );
00517                 }
00518                 break;
00519             case BUTTON_SAVE:
00520                 if ( mcp_portb == SWITCH_BUTTON_SAVE ) {
00521                     DisplayEventPost(last_interrupt);
00522                     BuzzerEvantPost( 50 );
00523                 }
00524                 break;
00525             case BUTTON_BACK:
00526                 if ( mcp_portb == SWITCH_BUTTON_BACK ) {
00527                     DisplayEventPost(last_interrupt);
00528                     BuzzerEvantPost( 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;
      // 00ss ss00
00563                 speed_selector = ( speed_selector » ( FREQ_SEL_1_PIN ) );
      // 0000 ssss
00564                 if ( speed_selector + SPEED_SELECTOR_0 == last_interrupt ) {
00565                     SystemEventPost(last_interrupt);
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 RelayEvantPost( 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 BuzzerEvantPost( uint32_t buzzer_time_on ) {
00602     if ( buzzer_time_on > 3000 ) {
00603         buzzer_time_on = 3000;
00604     }
00605     return xQueueSend(buzzer_evt_queue, &buzzer_time_on, 0);
00606 }
```

6.19. Referencia del archivo main/io_control/io_control.h

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

Funciones

- void GPIO_interrupt_attendance_task (void *arg)

  Tarea que en función de las interrupciones recibidas permite obtener cual de los pulsadores del teclado, entradas aisladas, termo switch o entradas del MCP2307 fue presionado.
- esp_err_t set_freq_output (uint16_t freq)

  Convierte la frecuencia en Hz que está girando el motor en el duty necesario para poder operar el PWM de la salida 0-10V.
- esp_err_t RelayEvantPost (uint8_t relay_event)

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

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

6.19.1. Descripción detallada

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

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo io_control.h.

6.19.2. Documentación de funciones

6.19.2.1. BuzzerEvantPost()

```
esp_err_t BuzzerEvantPost (
            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 mili segundos durante el que el buzzer sonará.

Devuelve

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

Definición en la línea 601 del archivo io_control.c.

6.19.2.2. GPIO_interrupt_attendance_task()

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

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

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

Definición en la línea 386 del archivo io_control.c.

6.19.2.3. RelayEvantPost()

```
esp_err_t RelayEvantPost (
            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	*relay_event*	Estado del relay

Devuelve

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

Definición en la línea 593 del archivo io_control.c.

6.19.2.4. set_freq_output()

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

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

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

Parámetros

in	*freq*	Es la frecuencia que está girando el motor.

Devuelve

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

Definición en la línea 359 del archivo io_control.c.

6.20. io_control.h

Ir a la documentación de este archivo.

```
00001
00009 #ifndef __IO_CONTROL_H__
00010
00011 #define __IO_CONTROL_H__
00012
00021 void GPIO_interrupt_attendance_task(void* arg);
00022
00037 esp_err_t set_freq_output(uint16_t freq);
00038
00053 esp_err_t RelayEvantPost( uint8_t relay_event );
00054
00067 esp_err_t BuzzerEvantPost( uint32_t buzzer_time_on );
00068
00069 #endif
```

6.21. Referencia del archivo main/io_control/MCP23017.c

Declaración de funciones de inicialización, lectura y escritura a través del puerto I2C el dispositivo MCP23017.

```
#include "MCP23017.h"
#include "esp_log.h"
#include "esp_err.h"
#include "driver/i2c.h"
#include "freertos/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 escribe el registro OLAT del `port` del MCP23017.
- esp_err_t mcp_read_port (enum mcp_port_e port, uint8_t ∗data)

  Función que lee el registro GPIO del `port` del MCP23017 y lo devuelve en `data`.
- esp_err_t mcp_interrupt_flag (enum mcp_port_e port, uint8_t ∗data)

  Función que lee el registro INTF del MCP23017.

Variables

- static const char ∗ TAG = "MCP23017"
- SemaphoreHandle_t xI2C_Mutex = NULL

6.21.1. Descripción detallada

Declaración de funciones de inicialización, lectura y escritura a través del puerto I2C el dispositivo MCP23017.

Autor

Andrenacci - Carra

Este chip permite usar sus dos puertos de 8 bits por separado o como uno solo de 16 bits. Tiene la función de reportar una interrupción configurable por puerto a través de 2 pines, uno por puerto, que informar si hubo algún cambio en las entradas si así lo deseo; 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 "↩ 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 que 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 contínuamente la/las entradas de interés. Utilizando el modo ICON.BANK = 0, habilita una función especial que permite tooglear entre los registros del puerto A y B secuencialmente en cada acceso al chip.
- Sequenctial mode: El puntero de dirección interno del chip se incrementa a cada byte de datos enviado. Cuando llega al último registro del mapa de registro hace un rollover al registro 0x00.

La secuencia de escritura se define de la siguiente amnera: S 0100AAA0 ADDR DIN P

Versión

2.0

Fecha

2025-11-06

Definición en el archivo MCP23017.c.

6.21.2. Documentación de «define»

6.21.2.1. I2C_IO_MASTER_NUM

```
#define I2C_IO_MASTER_NUM I2C_NUM_1
```

Número de puerto I2C de la comunicación con el MCP23017

Definición en la línea 106 del archivo MCP23017.c.

6.21.2.2. I2C_MASTER_FREQ_HZ

```
#define I2C_MASTER_FREQ_HZ 100000
```

Frecuencia estándar de operción del I2C para comunicación con el MCP23017 en 100000MHz

Definición en la línea 107 del archivo MCP23017.c.

6.21.2.3. I2C_MASTER_RX_BUF_DISABLE

```
#define I2C_MASTER_RX_BUF_DISABLE 0
```

Largo de buffer I2C en 0 porque no es utilizado al ser un dispositivo maestro el ESP32

Definición en la línea 109 del archivo MCP23017.c.

6.21.2.4. I2C_MASTER_SCL_IO

```
#define I2C_MASTER_SCL_IO 15
```

GPIO para SCL de la comunicación con el MCP23017

Definición en la línea 104 del archivo MCP23017.c.

6.21.2.5. I2C_MASTER_SDA_IO

```
#define I2C_MASTER_SDA_IO 2
```

GPIO para SDA de la comunicación con el MCP23017

Definición en la línea 105 del archivo MCP23017.c.

6.21.2.6. I2C_MASTER_TIMEOUT_MS

```
#define I2C_MASTER_TIMEOUT_MS 100
```

Tiempo durante el cual se espera alguna respuesta del esclavo antes de cortar la comunicación

Definición en la línea 103 del archivo MCP23017.c.

6.21.2.7. I2C_MASTER_TX_BUF_DISABLE

```
#define I2C_MASTER_TX_BUF_DISABLE 0
```

Largo de buffer I2C en 0 porque no es utilizado al ser un dispositivo maestro el ESP32

Definición en la línea 108 del archivo MCP23017.c.

6.21.3. Documentación de funciones

6.21.3.1. i2c_is_hardware_free()

```
bool i2c_is_hardware_free () [static]
```

Función que pide recursos de hardware para realizar una operación de lectura o escritura.

En caso que los recursos de hardware estuviesen siendo usados, duerme la app durante 100ms

Atención

Función bloqueante

Definición en la línea 134 del archivo MCP23017.c.

6.21.3.2. i2c_master_init()

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

Función que inicializa el puerto I2C y el mutex para acceder ordenadamente al hardware.

Parámetros

in	*register_address*	Es la dirección de memoria del registro que se quiere acceder en el MCP23017.
in	*data*	Información que quiere escribirse en `register_address`

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 Timueout de la operación porque el bus está ocupado.

Definición en la línea 111 del archivo MCP23017.c.

6.21.3.3. i2c_release_hardware()

```
void i2c_release_hardware ()  [static]
```

Función que devuelve recursos de hardware luego de realizar una operación para que pueda realizarse una nueva lectura o escritura.

Definición en la línea 142 del archivo MCP23017.c.

6.21.3.4. MCP23017_INIT()

```
esp_err_t MCP23017_INIT (
            void )
```

Función dedicada a inicializar y configurar el MCP23017. El chip utiliza i2C.

Inicializa el MCP32017. Configura las entradas IO2, IO3, IO4, IO5 y IO6 del puerto A (Todas ellas sin pull up activo) y las entradas IO0, IO1, IO2 y IO3 del puerto B (Todas ellas con pull up activo); configura como salidas IO0, IO1 y IO7 del puerto A y IO4, IO5, IO6 y IO7 del puerto B. Todas las entradas generan una interrpción 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 Timueout de la operación porque el bus está ocupado.

Definición en la línea 174 del archivo MCP23017.c.

6.21.3.5. mcp_get_on_interrupt_input()

```
esp_err_t mcp_get_on_interrupt_input (
            enum mcp_port_e port,
            uint8_t * data)
```

Función que lee el registro INTCAP del MCP23017.

El registro INTCAP retiene el valor de las entradas al generarse una interrupción.

Parámetros

in	*port*	Puede tomar los valores PORTA o PORTB, dependiendo del puerto que se quiera acceder.
out	*data*	Puntero donde se almaecnará 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 Timueout de la operación porque el bus está ocupado.

Definición en la línea 312 del archivo MCP23017.c.

6.21.3.6. mcp_interrupt_flag()

```
esp_err_t mcp_interrupt_flag (
            enum mcp_port_e port,
            uint8_t * data)
```

Función que lee el registro INTF del MCP23017.

El registro INTF retiene el valor de la entrada que generó la interrupción. Luego de haber leído este registro, el pin INTx vuelve a su estado de reposo.

Parámetros

in	*port*	Puede tomar los valores PORTA o PORTB, dependiendo del puerto que se quiera acceder.
out	*data*	Puntero donde se almaecnará 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 Timueout de la operación porque el bus está ocupado.

Definición en la línea 387 del archivo MCP23017.c.

6.21.3.7. mcp_read_port()

```
esp_err_t mcp_read_port (
            enum mcp_port_e port,
            uint8_t * data)
```

Función que lee el registro GPIO del `port` del MCP23017 y lo devuelve en `data`.

El registro GPIO es el que representa la exteriorización de los estados de los etados lógicos altos o bajos de los pines.

Parámetros

in	*port*	Puede tomar los valores PORTA o PORTB, dependiendo del puerto que se quiera escribir.
out	*data*	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 Timueout de la operación porque el bus está ocupado.

Definición en la línea 370 del archivo MCP23017.c.

6.21.3.8. mcp_register_read()

```
esp_err_t mcp_register_read (
            uint8_t register_address,
            uint8_t * data)  [static]
```

Función que lee register_address del MCP23017. La información obtenida se almacena en data.

Parámetros

in	*register_address*	Es la dirección de memoria del registro que se quiere acceder en el MCP23017.
out	*data*	Información que leida de register_address

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 Timueout de la operación porque el bus está ocupado.

Definición en la línea 146 del archivo MCP23017.c.

6.21.3.9. mcp_register_write()

```
esp_err_t mcp_register_write (
            uint8_t register_address,
            uint8_t data)  [static]
```

Función que escribe en register_address del MCP23017 el dato data.

Parámetros

in	*register_address*	Es la dirección de memoria del registro que se quiere acceder en el MCP23017.
in	*data*	Información que quiere escribirse en `register_address`

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 Timueout de la operación porque el bus está ocupado.

Definición en la línea 161 del archivo MCP23017.c.

6.21.3.10. mcp_write_output_pin()

```
esp_err_t mcp_write_output_pin (
            enum mcp_port_e port,
            uint8_t pin,
            bool state)
```

Función que lee el registro GPIO, escribe `state` en el `pin` del `port` escribe el registro OLAT del MCP23017.

El registro OLAT es el que exterioriza el estado del pin en etados lógicos altos o bajos.

Parámetros

in	*port*	Puede tomar los valores PORTA o PORTB, dependiendo del puerto que se quiera escribir.
in	*pin*	Pin físico que quiere ser modificado su estado
in	*state*	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 Timueout de la operación porque el bus está ocupado.

Definición en la línea 328 del archivo MCP23017.c.

6.21.3.11. mcp_write_output_port()

```
esp_err_t mcp_write_output_port (
            enum mcp_port_e port,
            uint8_t data)
```

Función que escribe `data` en el escribe el registro OLAT del `port` del MCP23017.

Escribe `data` entero, no pin a pin.

Parámetros

in	*port*	Puede tomar los valores PORTA o PORTB, dependiendo del puerto que se quiera escribir.
in	*data*	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 Timueout de la operación porque el bus está ocupado.

Definición en la línea 353 del archivo MCP23017.c.

6.21.4. Documentación de variables

6.21.4.1. TAG

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

Varaible que se imprime en los ESP_LOG

Definición en la línea 100 del archivo MCP23017.c.

6.21.4.2. xI2C_Mutex

```
SemaphoreHandle_t xI2C_Mutex = NULL
```

Semáforo (Mutex) que es utilizado para que dos comandos sean ejecutados al mismo tiempo y evitar que uno superponga al otro, trayendo datos incorrectos del MCP23017

Definición en la línea 101 del archivo MCP23017.c.

6.22. MCP23017.c

Ir a la documentación de este archivo.

```
00001
00015
00016 #include "MCP23017.h"
00017 #include "esp_log.h"
00018 #include "esp_err.h"
00019 #include "driver/i2c.h"
00020 #include "freertos/semphr.h"
00021
00040 static esp_err_t i2c_master_init(void);
00041
00051 static bool i2c_is_hardware_free();
00052
00058 static void i2c_release_hardware();
00059
00078 static esp_err_t mcp_register_read(uint8_t register_address, uint8_t *data);
00079
00098 static esp_err_t mcp_register_write(uint8_t register_address, uint8_t data);
00099
```

```
00100 static const char *TAG = "MCP23017";
00101 SemaphoreHandle_t xI2C_Mutex = NULL;
00102
00103 #define I2C_MASTER_TIMEOUT_MS       100
00104 #define I2C_MASTER_SCL_IO           15
00105 #define I2C_MASTER_SDA_IO           2
00106 #define I2C_IO_MASTER_NUM           I2C_NUM_1
00107 #define I2C_MASTER_FREQ_HZ          100000
00108 #define I2C_MASTER_TX_BUF_DISABLE   0
00109 #define I2C_MASTER_RX_BUF_DISABLE   0
00110
00111 static esp_err_t i2c_master_init(void) {
00112     i2c_config_t conf = {
00113         .mode = I2C_MODE_MASTER,
00114         .sda_io_num = I2C_MASTER_SDA_IO,
00115         .scl_io_num = I2C_MASTER_SCL_IO,
00116         .sda_pullup_en = GPIO_PULLUP_ENABLE,
00117         .scl_pullup_en = GPIO_PULLUP_ENABLE,
00118         .master.clk_speed = I2C_MASTER_FREQ_HZ,
00119     };
00120     ESP_ERROR_CHECK(i2c_param_config(I2C_IO_MASTER_NUM, &conf));
00121     ESP_ERROR_CHECK(i2c_driver_install(I2C_IO_MASTER_NUM, conf.mode,
00122                                        I2C_MASTER_RX_BUF_DISABLE,
00123                                        I2C_MASTER_TX_BUF_DISABLE, 0));
00124     if (xI2C_Mutex == NULL) {
00125         xI2C_Mutex = xSemaphoreCreateMutex();
00126         if (xI2C_Mutex == NULL) {
00127             ESP_LOGE(TAG, "No se pudo crear el mutex de I2C");
00128         }
00129     }
00130
00131     return ESP_OK;
00132 }
00133
00134 static bool i2c_is_hardware_free() {
00135     while ( xSemaphoreTake(xI2C_Mutex, pdMS_TO_TICKS(100)) != pdTRUE ) {
00136         ESP_LOGI(TAG, "Espero recursos de hardware I2C...");
00137         return false;
00138     }
00139     return true;
00140 }
00141
00142 static void i2c_release_hardware() {
00143     xSemaphoreGive(xI2C_Mutex);
00144 }
00145
00146 static esp_err_t mcp_register_read(uint8_t register_address, uint8_t *data) {
00147     esp_err_t ret = ESP_OK;
00148     i2c_cmd_handle_t cmd = i2c_cmd_link_create();
00149     i2c_master_start(cmd);
00150     i2c_master_write_byte(cmd, __MCP23017_WRITE_OPPCODE__,0);
00151     i2c_master_write_byte(cmd, register_address, true);
00152     i2c_master_start(cmd);
00153     i2c_master_write_byte(cmd, __MCP23017_READ_OPPCODE__,0);
00154     i2c_master_read_byte(cmd, data, I2C_MASTER_NACK);
00155     i2c_master_stop(cmd);
00156     ret = i2c_master_cmd_begin(I2C_IO_MASTER_NUM, cmd, pdMS_TO_TICKS(I2C_MASTER_TIMEOUT_MS));
00157     i2c_cmd_link_delete(cmd);
00158     return ret;
00159 }
00160
00161 static esp_err_t mcp_register_write(uint8_t register_address, uint8_t data) {
00162     esp_err_t ret = ESP_OK;
00163     i2c_cmd_handle_t cmd = i2c_cmd_link_create();
00164     i2c_master_start(cmd);
00165     i2c_master_write_byte(cmd, __MCP23017_WRITE_OPPCODE__,0);
00166     i2c_master_write_byte(cmd, register_address, true);
00167     i2c_master_write_byte(cmd, data, true);
00168     i2c_master_stop(cmd);
00169     ret = i2c_master_cmd_begin(I2C_IO_MASTER_NUM, cmd, pdMS_TO_TICKS(I2C_MASTER_TIMEOUT_MS));
00170     i2c_cmd_link_delete(cmd);
00171     return ret;
00172 }
00173
00174 esp_err_t MCP23017_INIT( void ) {
00175     esp_err_t ret = ESP_OK;
00176     ESP_LOGI(TAG, "Iniciando modulo");
00177     if ( i2c_master_init() == ESP_OK ) {
00178         ESP_LOGI(TAG, "I2C Inicializado correctamente");
00179     } else {
00180         ESP_LOGE(TAG, "Error al inicializar I2C");
00181         return ESP_FAIL;
00182     }
00183     MCP23017_IOCON_t iocon;
00184     MCP23017_GPPU_t gppua, gppub;
00185     MCP23017_IODIR_t iodira, iodirb;
00186     MCP23017_GPINTEN_t gpintena, gpintenb;
```

```
00187     MCP23017_DEFVAL_t defvala, defvalb;
00188     MCP23017_INTCON_t intcona, intconb;
00189 
00190     iocon.all = 0x00;
00191     iocon.bits.INTPOL = __MCP23017_INTPOL_POLARITY_HIGH__;          // Los pines de interrupción se
      ponen en 1 cuando hay una interrupción
00192     iocon.bits.ODR = __MCP23017_ODR_ACTIVE_DRIVER_OUTPUT__;         // Los pines de interrupción
      configuran como salidas activas
00193     iocon.bits.DISSLW = __MCP23017_DISSLW_ENABLED__;                // Activo el Slew Rate en el SDA
      para una mejor comunicación
00194     iocon.bits.SEQOP = __MCP23017_SEQOP_DISABLED__;                 // No activo el incremento del
      puntero de direcciones para poder leer constantemente el mismo registro si quiero
00195     iocon.bits.MIRROR = __MCP23017_MIRROR_UNCONNECTED__;            // Los pines de interrupción A y B
      se controlan por separado
00196     iocon.bits.BANK = __MCP23017_FUNC_MODE__;                       // Uso el modo Byte para tratar
      los puertos como si guesen de 8 bits y no de 16
00197 
00198     gppua.all = __MCP23017_GPPU_PULL_UP_DISABLE__;
00199 
00200     gppub.all = __MCP23017_GPPU_PULL_UP_DISABLE__;
00201     gppub.bits.PU0 = __MCP23017_GPPU_PULL_UP_ENABLE__;
00202     gppub.bits.PU1 = __MCP23017_GPPU_PULL_UP_ENABLE__;
00203     gppub.bits.PU2 = __MCP23017_GPPU_PULL_UP_ENABLE__;
00204     gppub.bits.PU3 = __MCP23017_GPPU_PULL_UP_ENABLE__;
00205 
00206     iodira.all = 0x00;
00207     iodira.bits.IO0 = __MCP23017_IODIR_OUTPUT__;
00208     iodira.bits.IO1 = __MCP23017_IODIR_OUTPUT__;
00209     iodira.bits.IO2 = __MCP23017_IODIR_INPUT__;
00210     iodira.bits.IO3 = __MCP23017_IODIR_INPUT__;
00211     iodira.bits.IO4 = __MCP23017_IODIR_INPUT__;
00212     iodira.bits.IO5 = __MCP23017_IODIR_INPUT__;
00213     iodira.bits.IO6 = __MCP23017_IODIR_INPUT__;
00214     iodira.bits.IO7 = __MCP23017_IODIR_OUTPUT__;
00215 
00216     iodirb.all = 0x00;
00217     iodirb.bits.IO0 = __MCP23017_IODIR_INPUT__;
00218     iodirb.bits.IO1 = __MCP23017_IODIR_INPUT__;
00219     iodirb.bits.IO2 = __MCP23017_IODIR_INPUT__;
00220     iodirb.bits.IO3 = __MCP23017_IODIR_INPUT__;
00221     iodirb.bits.IO4 = __MCP23017_IODIR_OUTPUT__;
00222     iodirb.bits.IO5 = __MCP23017_IODIR_OUTPUT__;
00223     iodirb.bits.IO6 = __MCP23017_IODIR_OUTPUT__;
00224     iodirb.bits.IO7 = __MCP23017_IODIR_OUTPUT__;
00225 
00226     gpintena.all = 0x00;
00227     gpintena.bits.GPINT2 = __MCP23017_GPINTEN_ENABLE__;
00228     gpintena.bits.GPINT3 = __MCP23017_GPINTEN_ENABLE__;
00229     gpintena.bits.GPINT4 = __MCP23017_GPINTEN_ENABLE__;
00230     gpintena.bits.GPINT5 = __MCP23017_GPINTEN_ENABLE__;
00231     gpintena.bits.GPINT6 = __MCP23017_GPINTEN_ENABLE__;
00232 
00233     gpintenb.all = 0x00;
00234     gpintenb.bits.GPINT0 = __MCP23017_GPINTEN_ENABLE__;
00235     gpintenb.bits.GPINT1 = __MCP23017_GPINTEN_ENABLE__;
00236     gpintenb.bits.GPINT2 = __MCP23017_GPINTEN_ENABLE__;
00237     gpintenb.bits.GPINT3 = __MCP23017_GPINTEN_ENABLE__;
00238 
00239     defvala.all = 0x00;
00240     defvala.bits.DEF2 = 1;
00241     defvala.bits.DEF3 = 1;
00242     defvala.bits.DEF4 = 1;
00243     defvala.bits.DEF5 = 1;
00244     defvala.bits.DEF6 = 0;
00245 
00246     defvalb.all = 0x0F;
00247 
00248     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
```

```
    reset val
00268     ret = mcp_register_write( (uint8_t) MCP23017_GPINTENA_REGISTER  , gpintena.all  );  //
00269     if ( ret != ESP_OK ) {
00270         ESP_LOGE(TAG, "Error al escribir en el registro GPINTENA");
00271         return ret;
00272     }
00273     ret = mcp_register_write( (uint8_t) MCP23017_GPINTENB_REGISTER  , gpintenb.all  );  //
00274     if ( ret != ESP_OK ) {
00275         ESP_LOGE(TAG, "Error al escribir en el registro GPINTENB");
00276         return ret;
00277     }
00278     ret = mcp_register_write( (uint8_t) MCP23017_DEFVALA_REGISTER   , defvala.all   );  //
00279     if ( ret != ESP_OK ) {
00280         ESP_LOGE(TAG, "Error al escribir en el registro DEFVALA");
00281         return ret;
00282     }
00283     ret = mcp_register_write( (uint8_t) MCP23017_DEFVALB_REGISTER   , defvalb.all   );  //
00284     if ( ret != ESP_OK ) {
00285         ESP_LOGE(TAG, "Error al escribir en el registro DEFVALB");
00286         return ret;
00287     }
00288     ret = mcp_register_write( (uint8_t) MCP23017_INTCONA_REGISTER   , intcona.all   );  //
00289     if ( ret != ESP_OK ) {
00290         ESP_LOGE(TAG, "Error al escribir en el registro DEFVALB");
00291         return ret;
00292     }
00293     ret = mcp_register_write( (uint8_t) MCP23017_INTCONB_REGISTER   , intconb.all   );  //
00294     if ( ret != ESP_OK ) {
00295         ESP_LOGE(TAG, "Error al escribir en el registro DEFVALB");
00296         return ret;
00297     }
00298     ret = mcp_register_write( (uint8_t) MCP23017_GPPUA_REGISTER     , gppua.all     );  //
00299     if ( ret != ESP_OK ) {
00300         ESP_LOGE(TAG, "Error al escribir en el registro DEFVALB");
00301         return ret;
00302     }
00303     ret = mcp_register_write( (uint8_t) MCP23017_GPPUB_REGISTER     , gppub.all     );  //
00304     if ( ret != ESP_OK ) {
00305         ESP_LOGE(TAG, "Error al escribir en el registro DEFVALB");
00306         return ret;
00307     }
00308
00309     return ret;
00310 }
00311
00312 esp_err_t mcp_get_on_interrupt_input(enum mcp_port_e port, uint8_t *data) {
00313     while ( i2c_is_hardware_free() != pdTRUE ) {
00314         ESP_LOGI(TAG, "Espero recursos de hardware I2C...");
00315         vTaskDelay(pdMS_TO_TICKS(10));
00316     }
00317
00318     esp_err_t esp_err;
00319     if ( port == PORTA || port == PORTB ) {
00320         esp_err = mcp_register_read(MCP23017_INTCAPA_REGISTER + port, data);
00321     } else {
00322         esp_err = ESP_ERR_INVALID_ARG;
00323     }
00324     i2c_release_hardware();
00325     return esp_err;
00326 }
00327
00328 esp_err_t mcp_write_output_pin(enum mcp_port_e port, uint8_t pin, bool state) {
00329     esp_err_t esp_err;
00330     uint8_t data;
00331
00332     while ( i2c_is_hardware_free() != pdTRUE ) {
00333         ESP_LOGI(TAG, "Espero recursos de hardware I2C...");
00334         vTaskDelay(pdMS_TO_TICKS(10));
00335     }
00336
00337     if ( port == PORTA || port == PORTB ) {
00338         if ( !(esp_err = mcp_register_read(MCP23017_GPIOA_REGISTER + port, &data) ) ) {
00339             if (state ) {
00340                 data |= (1 << pin);
00341             } else {
00342                 data &= ~(1 << pin);
00343             }
00344             esp_err = mcp_register_write(MCP23017_OLATA_REGISTER + port, data);
00345         }
00346     } else {
00347         esp_err = ESP_ERR_INVALID_ARG;
00348     }
00349     i2c_release_hardware();
00350     return esp_err;
00351 }
00352
00353 esp_err_t mcp_write_output_port(enum mcp_port_e port, uint8_t data) {
```

```
00354     esp_err_t esp_err;
00355
00356     while ( i2c_is_hardware_free() != pdTRUE ) {
00357         ESP_LOGI(TAG, "Espero recursos de hardware I2C...");
00358         vTaskDelay(pdMS_TO_TICKS(10));
00359     }
00360
00361     if ( port == PORTA || port == PORTB ) {
00362         esp_err = mcp_register_write(MCP23017_OLATA_REGISTER + port, data);
00363     } else {
00364         esp_err = ESP_ERR_INVALID_ARG;
00365     }
00366     i2c_release_hardware();
00367     return esp_err;
00368 }
00369
00370 esp_err_t mcp_read_port(enum mcp_port_e port, uint8_t *data) {
00371     esp_err_t esp_err;
00372
00373     while ( i2c_is_hardware_free() != pdTRUE ) {
00374         ESP_LOGI(TAG, "Espero recursos de hardware I2C...");
00375         vTaskDelay(pdMS_TO_TICKS(10));
00376     }
00377
00378     if ( port == PORTA || port == PORTB ) {
00379         esp_err = mcp_register_read(MCP23017_GPIOA_REGISTER + port, data);
00380     } else {
00381         esp_err = ESP_ERR_INVALID_ARG;
00382     }
00383     i2c_release_hardware();
00384     return esp_err;
00385 }
00386
00387 esp_err_t mcp_interrupt_flag(enum mcp_port_e port, uint8_t *data) {
00388     esp_err_t esp_err;
00389
00390     while ( i2c_is_hardware_free() != pdTRUE ) {
00391         ESP_LOGI(TAG, "Espero recursos de hardware I2C...");
00392         vTaskDelay(pdMS_TO_TICKS(10));
00393     }
00394
00395     if ( port == PORTA || port == PORTB ) {
00396         esp_err = mcp_register_read(MCP23017_INTFA_REGISTER + port, data);
00397     } else {
00398         esp_err = ESP_ERR_INVALID_ARG;
00399     }
00400     i2c_release_hardware();
00401     return esp_err;
00402 }
```

6.23. Referencia del archivo main/io_control/MCP23017.h

Declaración de funciones de inicialización, lectura y escritura a través del puerto I2C el dispositivo MCP23017.

```
#include "esp_mac.h"
#include "mcp23017_defs.h"
#include <inttypes.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
```

Funciones

- esp_err_t MCP23017_INIT (void)

  Función dedicada a inicializar y configurar el MCP23017. El chip utiliza i2C.
- esp_err_t mcp_get_on_interrupt_input (enum mcp_port_e port, uint8_t *data)

  Función que lee el registro INTCAP del MCP23017.
- esp_err_t mcp_interrupt_flag (enum mcp_port_e port, uint8_t *data)

  Función que lee el registro INTF del MCP23017.

- esp_err_t mcp_write_output_pin (enum mcp_port_e port, uint8_t pin, bool state)

  Función que lee el registro GPIO, escribe `state` en el `pin` del `port` escribe el registro OLAT del MCP23017.
- esp_err_t mcp_read_port (enum mcp_port_e port, uint8_t *data)

  Función que lee el registro GPIO del `port` del MCP23017 y lo devuelve en `data`.
- esp_err_t mcp_write_output_port (enum mcp_port_e port, uint8_t data)

  Función que escribe `data` en el escribe el registro OLAT del `port` del MCP23017.

6.23.1. Descripción detallada

Declaración de funciones de inicialización, lectura y escritura a través del puerto I2C el dispositivo MCP23017.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo MCP23017.h.

6.23.2. Documentación de funciones

6.23.2.1. MCP23017_INIT()

```
esp_err_t MCP23017_INIT (
            void )
```

Función dedicada a inicializar y configurar el MCP23017. El chip utiliza i2C.

Inicializa el MCP32017. Configura las entradas IO2, IO3, IO4, IO5 y IO6 del puerto A (Todas ellas sin pull up activo) y las entradas IO0, IO1, IO2 y IO3 del puerto B (Todas ellas con pull up activo); configura como salidas IO0, IO1 y IO7 del puerto A y IO4, IO5, IO6 y IO7 del puerto B. Todas las entradas generan una interrpción 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 Timueout de la operación porque el bus está ocupado.

Definición en la línea 174 del archivo MCP23017.c.

6.23.2.2. mcp_get_on_interrupt_input()

```
esp_err_t mcp_get_on_interrupt_input (
            enum mcp_port_e port,
            uint8_t * data)
```

Función que lee el registro INTCAP del MCP23017.

El registro INTCAP retiene el valor de las entradas al generarse una interrupción.

Parámetros

in	*port*	Puede tomar los valores PORTA o PORTB, dependiendo del puerto que se quiera acceder.
out	*data*	Puntero donde se almaecnará 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 Timueout de la operación porque el bus está ocupado.

Definición en la línea 312 del archivo MCP23017.c.

6.23.2.3. mcp_interrupt_flag()

```
esp_err_t mcp_interrupt_flag (
            enum mcp_port_e port,
            uint8_t * data)
```

Función que lee el registro INTF del MCP23017.

El registro INTF retiene el valor de la entrada que generó la interrupción. Luego de haber leído este registro, el pin INTx vuelve a su estado de reposo.

Parámetros

in	*port*	Puede tomar los valores PORTA o PORTB, dependiendo del puerto que se quiera acceder.
out	*data*	Puntero donde se almaecnará 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 Timueout de la operación porque el bus está ocupado.

Definición en la línea 387 del archivo MCP23017.c.

6.23.2.4. mcp_read_port()

```
esp_err_t mcp_read_port (
            enum mcp_port_e port,
            uint8_t * data)
```

Función que lee el registro GPIO del `port` del MCP23017 y lo devuelve en `data`.

El registro GPIO es el que representa la exteriorización de los estados de los etados lógicos altos o bajos de los pines.

Parámetros

in	*port*	Puede tomar los valores PORTA o PORTB, dependiendo del puerto que se quiera escribir.
out	*data*	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 Timueout de la operación porque el bus está ocupado.

Definición en la línea 370 del archivo MCP23017.c.

6.23.2.5. mcp_write_output_pin()

```
esp_err_t mcp_write_output_pin (
            enum mcp_port_e port,
            uint8_t pin,
            bool state)
```

Función que lee el registro GPIO, escribe state en el pin del port escribe el registro OLAT del MCP23017.

El registro OLAT es el que exterioriza el estado del pin en etados lógicos altos o bajos.

Parámetros

in	*port*	Puede tomar los valores PORTA o PORTB, dependiendo del puerto que se quiera escribir.
in	*pin*	Pin físico que quiere ser modificado su estado
in	*state*	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 Timueout de la operación porque el bus está ocupado.

Definición en la línea 328 del archivo MCP23017.c.

6.23.2.6. mcp_write_output_port()

```
esp_err_t mcp_write_output_port (
            enum mcp_port_e port,
            uint8_t data)
```

Función que escribe data en el escribe el registro OLAT del port del MCP23017.

Escribe data entero, no pin a pin.

Parámetros

in	*port*	Puede tomar los valores PORTA o PORTB, dependiendo del puerto que se quiera escribir.
in	*data*	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 Timueout de la operación porque el bus está ocupado.

Definición en la línea 353 del archivo MCP23017.c.

6.24. MCP23017.h

Ir a la documentación de este archivo.

```
00001
00009
00010 #ifndef __MCP23017_H__
00011 #define __MCP23017_H__
00012
00013 #include "esp_mac.h"
00014 #include "mcp23017_defs.h"
00015 #include <inttypes.h>
00016 #include <stdio.h>
00017 #include <stdlib.h>
00018 #include <stdbool.h>
00019 #include <stdbool.h>
00020
00021
00036 esp_err_t MCP23017_INIT( void );
00037
00058 esp_err_t mcp_get_on_interrupt_input(enum mcp_port_e port, uint8_t *data);
00059
00080 esp_err_t mcp_interrupt_flag(enum mcp_port_e port, uint8_t *data);
00081
00105 esp_err_t mcp_write_output_pin(enum mcp_port_e port, uint8_t pin, bool state);
00106
00127 esp_err_t mcp_read_port(enum mcp_port_e port, uint8_t *data);
00128
00149 esp_err_t mcp_write_output_port(enum mcp_port_e port, uint8_t data);
00150
00151 #endif
```

6.25. Referencia del archivo main/io_control/mcp23017_defs.h

```
#include <inttypes.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
```

Estructuras de datos

- union MCP23017_IODIR_t
- union MCP23017_IPOL_t
- union MCP23017_GPINTEN_t
- union MCP23017_GPPU_t
- union MCP23017_GPIO_t
- union MCP23017_OLAT_t
- union MCP23017_DEFVAL_t
- union MCP23017_INTCON_t
- union MCP23017_IOCON_t
- union MCP23017_INTF_t
- union MCP23017_INTCAP_t

defines

- #define __MCP23017_BANK_SEQUENTIAL__ 0
- #define __MCP23017_BANK_SPLIT__ 1
- #define __MCP23017_MIRROR_UNCONNECTED__ 0
- #define __MCP23017_MIRROR_CONNECTED__ 1
- #define __MCP23017_SEQOP_ENABLED__ 0
- #define __MCP23017_SEQOP_DISABLED__ 1
- #define __MCP23017_DISSLW_DISABLED__ 0
- #define __MCP23017_DISSLW_ENABLED__ 1
- #define __MCP23017_ODR_OPEN_DRAIN_OUTPUT__ 1
- #define __MCP23017_ODR_ACTIVE_DRIVER_OUTPUT__ 0
- #define __MCP23017_INTPOL_POLARITY_LOW__ 1
- #define __MCP23017_INTPOL_POLARITY_HIGH__ 0
- #define __MCP23017_GPPU_PULL_UP_ENABLE__ 1
- #define __MCP23017_GPPU_PULL_UP_DISABLE__ 0
- #define __MCP23017_IODIR_INPUT__ 1
- #define __MCP23017_IODIR_OUTPUT__ 0
- #define __MCP23017_GPINTEN_ENABLE__ 1
- #define __MCP23017_GPINTEN_DISABLE__ 0
- #define __MCP23017_INTCON_DEFVAL__ 1
- #define __MCP23017_INTCON_CHANGE__ 0
- #define __MCP23017_WRITE__ 0
- #define __MCP23017_READ__ 1
- #define __MCP23017_POSITIVE_LOGIC__ 0
- #define __MCP23017_OPPOSITE_LOGIC__ 1
- #define __MCP23017_INTERRUPT_ENABLE__ 0
- #define __MCP23017_INTERRUPT_DISABLE__ 1
- #define __MCP23017_ADDRESS__ 0b010
- #define __MCP23017_READ_OPPCODE__ 0b0100 << 4 | __MCP23017_ADDRESS__ << 1 | __MCP23017_READ__
- #define __MCP23017_WRITE_OPPCODE__ 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 abtrae en conocer como se llaman los registros del chip, solo con las funciones específicas de lectura y escritura, datos, pines y esta definición de puerto.

6.25.1. Documentación de «define»

6.25.1.1. __MCP23017_ADDRESS__

```
#define __MCP23017_ADDRESS__ 0b010
```

CONFIGURADO POR HARDWARE: Address del MCP23017 = 0b010 ----> OPCODE: 0100AAA0 -> 01000000 | **MCP23017_ADDRESS** << 1 | R/W

Definición en la línea 50 del archivo mcp23017_defs.h.

6.25.1.2. __MCP23017_BANK_SEQUENTIAL__

```
#define __MCP23017_BANK_SEQUENTIAL__ 0
```

Modo de banqueo de moemora separado por puertos

Definición en la línea 11 del archivo mcp23017_defs.h.

6.25.1.3. __MCP23017_BANK_SPLIT__

```
#define __MCP23017_BANK_SPLIT__ 1
```

Modo de banqueo de moemora entrelazando puertos para manejo con entreos de 16 bits

Definición en la línea 12 del archivo mcp23017_defs.h.

6.25.1.4. __MCP23017_DISSLW_DISABLED__

```
#define __MCP23017_DISSLW_DISABLED__ 0
```

Slew rate deshabilitado

Definición en la línea 20 del archivo mcp23017_defs.h.

6.25.1.5. __MCP23017_DISSLW_ENABLED__

```
#define __MCP23017_DISSLW_ENABLED__ 1
```

Slew rate habilitado

Definición en la línea 21 del archivo mcp23017_defs.h.

6.25.1.6. __MCP23017_FUNC_MODE__

```
#define __MCP23017_FUNC_MODE__ __MCP23017_BANK_SEQUENTIAL__
```

Modo de funcionamieto del MCP23017 que funcionará en el sistema

Definición en la línea 54 del archivo mcp23017_defs.h.

6.25.1.7. __MCP23017_GPINTEN_DISABLE__

```
#define __MCP23017_GPINTEN_DISABLE__ 0
```

Interrupción de GPIO deshabilitada

Definición en la línea 36 del archivo mcp23017_defs.h.

6.25.1.8. __MCP23017_GPINTEN_ENABLE__

```
#define __MCP23017_GPINTEN_ENABLE__ 1
```

Interrupción de GPIO habilitada

Definición en la línea 35 del archivo mcp23017_defs.h.

6.25.1.9. __MCP23017_GPPU_PULL_UP_DISABLE__

```
#define __MCP23017_GPPU_PULL_UP_DISABLE__ 0
```

DESctivación de pull up de GPIO

Definición en la línea 30 del archivo mcp23017_defs.h.

6.25.1.10. __MCP23017_GPPU_PULL_UP_ENABLE__

```
#define __MCP23017_GPPU_PULL_UP_ENABLE__ 1
```

Activación de pull up de GPIO

Definición en la línea 29 del archivo mcp23017_defs.h.

6.25.1.11. __MCP23017_INTCON_CHANGE__

```
#define __MCP23017_INTCON_CHANGE__ 0
```

Interrupción dispara por diferencia con útlimo valor

Definición en la línea 39 del archivo mcp23017_defs.h.

6.25.1.12. __MCP23017_INTCON_DEFVAL__

```
#define __MCP23017_INTCON_DEFVAL__ 1
```

Interrupción dispara por diferencia con valor default

Definición en la línea 38 del archivo mcp23017_defs.h.

6.25.1.13. __MCP23017_INTERRUPT_DISABLE__

```
#define __MCP23017_INTERRUPT_DISABLE__ 1
```

Habilitación de la interrupción del GPIO

Definición en la línea 48 del archivo mcp23017_defs.h.

6.25.1.14. __MCP23017_INTERRUPT_ENABLE__

```
#define __MCP23017_INTERRUPT_ENABLE__ 0
```

Deshabilitación de la interrupción del GPIO

Definición en la línea 47 del archivo mcp23017_defs.h.

6.25.1.15. __MCP23017_INTPOL_POLARITY_HIGH__

```
#define __MCP23017_INTPOL_POLARITY_HIGH__ 0
```

Flags de interrupción de los puerto A y B en 1 significa una nueva interrupción

Definición en la línea 27 del archivo mcp23017_defs.h.

6.25.1.16. __MCP23017_INTPOL_POLARITY_LOW__

```
#define __MCP23017_INTPOL_POLARITY_LOW__ 1
```

Flags de interrupción de los puerto A y B en 0 significa una nueva interrupción

Definición en la línea 26 del archivo mcp23017_defs.h.

6.25.1.17. __MCP23017_IODIR_INPUT__

```
#define __MCP23017_IODIR_INPUT__ 1
```

GPIO configurada como entrada

Definición en la línea 32 del archivo mcp23017_defs.h.

6.25.1.18. __MCP23017_IODIR_OUTPUT__

```
#define __MCP23017_IODIR_OUTPUT__ 0
```

GPIO configurada como salida

Definición en la línea 33 del archivo mcp23017_defs.h.

6.25.1.19. __MCP23017_MIRROR_CONNECTED__

```
#define __MCP23017_MIRROR_CONNECTED__ 1
```

Los pines de interrupción están internamente conectados

Definición en la línea 15 del archivo mcp23017_defs.h.

6.25.1.20. __MCP23017_MIRROR_UNCONNECTED__

```
#define __MCP23017_MIRROR_UNCONNECTED__ 0
```

Los pines de interrupción no están internamente conectados

Definición en la línea 14 del archivo mcp23017_defs.h.

6.25.1.21. __MCP23017_ODR_ACTIVE_DRIVER_OUTPUT__

```
#define __MCP23017_ODR_ACTIVE_DRIVER_OUTPUT__ 0
```

Flags de interrupción de los puerto A y B configuradas como salidas activas

Definición en la línea 24 del archivo mcp23017_defs.h.

6.25.1.22. __MCP23017_ODR_OPEN_DRAIN_OUTPUT__

```
#define __MCP23017_ODR_OPEN_DRAIN_OUTPUT__ 1
```

Flags de interrupción de los puerto A y B configuradas open drain

Definición en la línea 23 del archivo mcp23017_defs.h.

6.25.1.23. __MCP23017_OPPOSITE_LOGIC__

```
#define __MCP23017_OPPOSITE_LOGIC__ 1
```

Configuración de la lógica de los GPIO como negativa

Definición en la línea 45 del archivo mcp23017_defs.h.

6.25.1.24. __MCP23017_POSITIVE_LOGIC__

```
#define __MCP23017_POSITIVE_LOGIC__ 0
```

Configuración de la lógica de los GPIO como positiva

Definición en la línea 44 del archivo mcp23017_defs.h.

6.25.1.25. __MCP23017_READ__

```
#define __MCP23017_READ__ 1
```

Función de lectura del MCP23017 - Se inserta en el opcode

Definición en la línea 42 del archivo mcp23017_defs.h.

6.25.1.26. __MCP23017_READ_OPPCODE__

```
#define __MCP23017_READ_OPPCODE__ 0b0100 << 4 | __MCP23017_ADDRESS__ << 1 | __MCP23017_READ__
```

Código de operación para lectura en el MCP23017

Definición en la línea 52 del archivo mcp23017_defs.h.

6.25.1.27. __MCP23017_SEQOP_DISABLED__

```
#define __MCP23017_SEQOP_DISABLED__ 1
```

El puntero de direccionamiento de memoria no incrementa en cada acceso

Definición en la línea 18 del archivo mcp23017_defs.h.

6.25.1.28. __MCP23017_SEQOP_ENABLED__

```
#define __MCP23017_SEQOP_ENABLED__ 0
```

El puntero de direccionamiento de memoria incrementa en cada acceso

Definición en la línea 17 del archivo mcp23017_defs.h.

6.25.1.29. __MCP23017_WRITE__

```
#define __MCP23017_WRITE__ 0
```

Función de escritura del MCP23017 - Se inserta en el opcode

Definición en la línea 41 del archivo mcp23017_defs.h.

6.25.1.30. __MCP23017_WRITE_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.

6.25.1.31. MCP23017_DEFVALA_REGISTER

```
#define MCP23017_DEFVALA_REGISTER 0x06
```

Dirección de memoria del registro DEFVALA

Definición en la línea 63 del archivo mcp23017_defs.h.

6.25.1.32. MCP23017_DEFVALB_REGISTER

```
#define MCP23017_DEFVALB_REGISTER 0x07
```

Dirección de memoria del registro DEFVALB

Definición en la línea 64 del archivo mcp23017_defs.h.

6.25.1.33. MCP23017_GPINTENA_REGISTER

```
#define MCP23017_GPINTENA_REGISTER 0x04
```

Dirección de memoria del registro GPINTENA

Definición en la línea 61 del archivo mcp23017_defs.h.

6.25.1.34. MCP23017_GPINTENB_REGISTER

```
#define MCP23017_GPINTENB_REGISTER 0x05
```

Dirección de memoria del registro GPINTENB

Definición en la línea 62 del archivo mcp23017_defs.h.

6.25.1.35. MCP23017_GPIOA_REGISTER

```
#define MCP23017_GPIOA_REGISTER 0x12
```

Dirección de memoria del registro GPIOA

Definición en la línea 75 del archivo mcp23017_defs.h.

6.25.1.36. MCP23017_GPIOB_REGISTER

```
#define MCP23017_GPIOB_REGISTER 0x13
```

Dirección de memoria del registro GPIOB

Definición en la línea 76 del archivo mcp23017_defs.h.

6.25.1.37. MCP23017_GPPUA_REGISTER

```
#define MCP23017_GPPUA_REGISTER 0x0C
```

Dirección de memoria del registro GPPUA

Definición en la línea 69 del archivo mcp23017_defs.h.

6.25.1.38. MCP23017_GPPUB_REGISTER

```
#define MCP23017_GPPUB_REGISTER 0x0D
```

Dirección de memoria del registro GPPUB

Definición en la línea 70 del archivo mcp23017_defs.h.

6.25.1.39. MCP23017_INTCAPA_REGISTER

```
#define MCP23017_INTCAPA_REGISTER 0x10
```

Dirección de memoria del registro INTCAPA

Definición en la línea 73 del archivo mcp23017_defs.h.

6.25.1.40. MCP23017_INTCAPB_REGISTER

```
#define MCP23017_INTCAPB_REGISTER 0x11
```

Dirección de memoria del registro INTCAPB

Definición en la línea 74 del archivo mcp23017_defs.h.

6.25.1.41. MCP23017_INTCONA_REGISTER

```
#define MCP23017_INTCONA_REGISTER 0x08
```

Dirección de memoria del registro INTCONA

Definición en la línea 65 del archivo mcp23017_defs.h.

6.25.1.42. MCP23017_INTCONB_REGISTER

```
#define MCP23017_INTCONB_REGISTER 0x09
```

Dirección de memoria del registro INTCONB

Definición en la línea 66 del archivo mcp23017_defs.h.

6.25.1.43. MCP23017_INTFA_REGISTER

```
#define MCP23017_INTFA_REGISTER 0x0E
```

Dirección de memoria del registro INTFA

Definición en la línea 71 del archivo mcp23017_defs.h.

6.25.1.44. MCP23017_INTFB_REGISTER

```
#define MCP23017_INTFB_REGISTER 0x0F
```

Dirección de memoria del registro INTFB

Definición en la línea 72 del archivo mcp23017_defs.h.

6.25.1.45. MCP23017_IOCON_REGISTER

```
#define MCP23017_IOCON_REGISTER 0x0A
```

Dirección de memoria del registro IOCON

Definición en la línea 67 del archivo mcp23017_defs.h.

6.25.1.46. MCP23017_IODIRA_REGISTER

```
#define MCP23017_IODIRA_REGISTER 0x00
```

Dirección de memoria del registro IODIRA

Definición en la línea 57 del archivo mcp23017_defs.h.

6.25.1.47. MCP23017_IODIRB_REGISTER

```
#define MCP23017_IODIRB_REGISTER 0x01
```

Dirección de memoria del registro IODIRB

Definición en la línea 58 del archivo mcp23017_defs.h.

6.25.1.48. MCP23017_IPOLA_REGISTER

```
#define MCP23017_IPOLA_REGISTER 0x02
```

Dirección de memoria del registro IPOLA

Definición en la línea 59 del archivo mcp23017_defs.h.

6.25.1.49. MCP23017_IPOLB_REGISTER

```
#define MCP23017_IPOLB_REGISTER 0x03
```

Dirección de memoria del registro IPOLB

Definición en la línea 60 del archivo mcp23017_defs.h.

6.25.1.50. MCP23017_OLATA_REGISTER

```
#define MCP23017_OLATA_REGISTER 0x14
```

Dirección de memoria del registro OLATA

Definición en la línea 77 del archivo mcp23017_defs.h.

6.25.1.51. MCP23017_OLATB_REGISTER

```
#define MCP23017_OLATB_REGISTER 0x15
```

Dirección de memoria del registro OLATB

Definición en la línea 78 del archivo mcp23017_defs.h.

6.25.2. Documentación de enumeraciones

6.25.2.1. mcp_port_e

```
enum mcp_port_e
```

Selector de puertos del MCP23017. Puede ser PORTA (0) o PORTB (1). De esta manera el usuario se abtrae en conocer como se llaman los registros del chip, solo con las funciones específicas de lectura y escritura, datos, pines y esta definición de puerto.

Valores de enumeraciones

PORTA	Puerto A del MCP23017
PORTB	Puerto B del MCP23017

Definición en la línea 340 del archivo mcp23017_defs.h.

6.26. mcp23017_defs.h

Ir a la documentación de este archivo.

```
00001 #ifndef __MCP23017_DEFS_H__
00002
00003 #define __MCP23017_DEFS_H__
00004
00005 #include <inttypes.h>
00006 #include <stdio.h>
00007 #include <stdlib.h>
00008 #include <stdbool.h>
00009 #include <stdbool.h>
00010
00011 #define __MCP23017_BANK_SEQUENTIAL__            0
00012 #define __MCP23017_BANK_SPLIT__                 1
00013
00014 #define __MCP23017_MIRROR_UNCONNECTED__         0
00015 #define __MCP23017_MIRROR_CONNECTED__           1
00016
00017 #define __MCP23017_SEQOP_ENABLED__              0
00018 #define __MCP23017_SEQOP_DISABLED__             1
00019
00020 #define __MCP23017_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_OPPCODE__           0b0100 << 4 | __MCP23017_ADDRESS__ << 1 | __MCP23017_READ__
00053 #define __MCP23017_WRITE_OPPCODE__          0b0100 << 4 | __MCP23017_ADDRESS__ << 1 | __MCP23017_WRITE__
00054 #define __MCP23017_FUNC_MODE__              __MCP23017_BANK_SEQUENTIAL__
00055
00056 #if __MCP23017_FUNC_MODE__ == __MCP23017_SEQUENTIAL_MODE__
00057     #define MCP23017_IODIRA_REGISTER            0x00
00058     #define MCP23017_IODIRB_REGISTER            0x01
00059     #define MCP23017_IPOLA_REGISTER             0x02
00060     #define MCP23017_IPOLB_REGISTER             0x03
00061     #define MCP23017_GPINTENA_REGISTER          0x04
00062     #define MCP23017_GPINTENB_REGISTER          0x05
00063     #define MCP23017_DEFVALA_REGISTER           0x06
00064     #define MCP23017_DEFVALB_REGISTER           0x07
00065     #define MCP23017_INTCONA_REGISTER           0x08
```

```
00066     #define MCP23017_INTCONB_REGISTER           0x09
00067     #define MCP23017_IOCON_REGISTER             0x0A
00068     // #define MCP23017_IOCON_REGISTER             0x0B                    /*!< Dirección de
      memoria del registro IOCON */
00069     #define MCP23017_GPPUA_REGISTER             0x0C
00070     #define MCP23017_GPPUB_REGISTER             0x0D
00071     #define MCP23017_INTFA_REGISTER             0x0E
00072     #define MCP23017_INTFB_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                     /*!< Dirección de
      memoria del registro IOCON */
00092     #define MCP23017_GPPUA_REGISTER              0x06
00093     #define MCP23017_GPPUB_REGISTER              0x16
00094     #define MCP23017_INTFA_REGISTER              0x07
00095     #define MCP23017_INTFB_REGISTER              0x17
00096     #define MCP23017_INTCAPA_REGISTER            0x08
00097     #define MCP23017_INTCAPB_REGISTER            0x18
00098     #define MCP23017_GPIOA_REGISTER              0x09
00099     #define MCP23017_GPIOB_REGISTER              0x19
00100     #define MCP23017_OLATA_REGISTER              0x0A
00101     #define MCP23017_OLATB_REGISTER              0x1A
00102 #endif
00103
00111 typedef union {
00112     uint8_t all;
00113     struct {
00114         uint8_t IO0 : 1;
00115         uint8_t IO1 : 1;
00116         uint8_t IO2 : 1;
00117         uint8_t IO3 : 1;
00118         uint8_t IO4 : 1;
00119         uint8_t IO5 : 1;
00120         uint8_t IO6 : 1;
00121         uint8_t IO7 : 1;
00122     } bits;
00123 } MCP23017_IODIR_t;
00124
00132 typedef union {
00133     uint8_t all;
00134     struct {
00135         uint8_t IP0 : 1;
00136         uint8_t IP1 : 1;
00137         uint8_t IP2 : 1;
00138         uint8_t IP3 : 1;
00139         uint8_t IP4 : 1;
00140         uint8_t IP5 : 1;
00141         uint8_t IP6 : 1;
00142         uint8_t IP7 : 1;
00143     } bits;
00144 } MCP23017_IPOL_t;
00145
00153 typedef union {
00154     uint8_t all;
00155     struct {
00156         uint8_t GPINT0 : 1;
00157         uint8_t GPINT1 : 1;
00158         uint8_t GPINT2 : 1;
00159         uint8_t GPINT3 : 1;
00160         uint8_t GPINT4 : 1;
00161         uint8_t GPINT5 : 1;
00162         uint8_t GPINT6 : 1;
00163         uint8_t GPINT7 : 1;
00164     } bits;
00165 } MCP23017_GPINTEN_t;
00166
00174 typedef union {
00175     uint8_t all;
00176     struct {
00177         uint8_t PU0 : 1;
00178         uint8_t PU1 : 1;
```

```
00179         uint8_t PU2 : 1;
00180         uint8_t PU3 : 1;
00181         uint8_t PU4 : 1;
00182         uint8_t PU5 : 1;
00183         uint8_t PU6 : 1;
00184         uint8_t PU7 : 1;
00185     } bits;
00186 } MCP23017_GPPU_t;
00187
00195 typedef union {
00196     uint8_t all;
00197     struct {
00198         uint8_t GP0 : 1;
00199         uint8_t GP1 : 1;
00200         uint8_t GP2 : 1;
00201         uint8_t GP3 : 1;
00202         uint8_t GP4 : 1;
00203         uint8_t GP5 : 1;
00204         uint8_t GP6 : 1;
00205         uint8_t GP7 : 1;
00206     } bits;
00207 } MCP23017_GPIO_t;
00208
00216 typedef union {
00217     uint8_t all;
00218     struct {
00219         uint8_t OL0 : 1;
00220         uint8_t OL1 : 1;
00221         uint8_t OL2 : 1;
00222         uint8_t OL3 : 1;
00223         uint8_t OL4 : 1;
00224         uint8_t OL5 : 1;
00225         uint8_t OL6 : 1;
00226         uint8_t OL7 : 1;
00227     } bits;
00228 } MCP23017_OLAT_t;
00229
00237 typedef union {
00238     uint8_t all;
00239     struct {
00240         uint8_t DEF0 : 1;
00241         uint8_t DEF1 : 1;
00242         uint8_t DEF2 : 1;
00243         uint8_t DEF3 : 1;
00244         uint8_t DEF4 : 1;
00245         uint8_t DEF5 : 1;
00246         uint8_t DEF6 : 1;
00247         uint8_t DEF7 : 1;
00248     } bits;
00249 } MCP23017_DEFVAL_t;
00250
00258 typedef union {
00259     uint8_t all;
00260     struct {
00261         uint8_t IOC0 : 1;
00262         uint8_t IOC1 : 1;
00263         uint8_t IOC2 : 1;
00264         uint8_t IOC3 : 1;
00265         uint8_t IOC4 : 1;
00266         uint8_t IOC5 : 1;
00267         uint8_t IOC6 : 1;
00268         uint8_t IOC7 : 1;
00269     } bits;
00270 } MCP23017_INTCON_t;
00271
00279 typedef union {
00280     uint8_t all;
00281     struct {
00282         uint8_t reserved : 1;
00283         uint8_t INTPOL : 1;
00284         uint8_t ODR : 1;
00285         uint8_t reserved_2 : 1;
00286         uint8_t DISSLW : 1;
00287         uint8_t SEQOP : 1;
00288         uint8_t MIRROR : 1;
00289         uint8_t BANK : 1;
00290     } bits;
00291 } MCP23017_IOCON_t;
00292
00300 typedef union {
00301     uint8_t all;
00302     struct {
00303         uint8_t INT0 : 1;
00304         uint8_t INT1 : 1;
00305         uint8_t INT2 : 1;
00306         uint8_t INT3 : 1;
00307         uint8_t INT4 : 1;
```

```
00308         uint8_t INT5 : 1;
00309         uint8_t INT6 : 1;
00310         uint8_t INT7 : 1;
00311     } bits;
00312 } MCP23017_INTF_t;
00313
00321 typedef union {
00322     uint8_t all;
00323     struct {
00324         uint8_t ICP0 : 1;
00325         uint8_t ICP1 : 1;
00326         uint8_t ICP2 : 1;
00327         uint8_t ICP3 : 1;
00328         uint8_t ICP4 : 1;
00329         uint8_t ICP5 : 1;
00330         uint8_t ICP6 : 1;
00331         uint8_t ICP7 : 1;
00332     } bits;
00333 } MCP23017_INTCAP_t;
00334
00340 enum mcp_port_e {
00341     PORTA = 0,
00342     PORTB = 1
00343 };
00344
00345 #endif
```

6.27. Referencia del archivo main/LVFV_system.h

Declaraciones de estructuras, variables, definiciones y estados generales del sistema utilizados en varios de los archivos.

```
#include <inttypes.h>
#include <time.h>
```

Estructuras de datos

- struct frequency_settings_t
- struct time_settings_t
- struct seccurity_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 seccurity_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 seccurity_settings_t seccurity_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 seccurity_settings_SH1106_t seccurity_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_EXCEDED , 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 desición a nivel de sistema.
- enum system_status_e {
  SYSTEM_IDLE , SYSTEM_ACCLE_DESACCEL , SYSTEM_REGIME , SYSTEM_BREAKING ,
  SYSTEM_EMERGENCY , SYSTEM_EMERGENCY_OK }

  Posibles estados general que puede tomar el sistema.
- enum sh1106_variable_lines_e {
  first = VARIABLE_FIRST , second = VARIABLE_SECOND , third = VARIABLE_THIRD , fourth = VARIABLE↩
  _FOURTH ,
  fifth = VARIABLE_FIFTH , sixth = VARIABLE_SIXTH , seventh = VARIABLE_SEVENTH }

6.27.1. Descripción detallada

Declaraciones de estructuras, variables, definiciones y estados generales del sistema utilizados en varios de los archivos.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo LVFV_system.h.

6.27.2. Documentación de «define»

6.27.2.1. LINE_INCREMENT

```
#define LINE_INCREMENT 9
```

Definición en la línea 15 del archivo LVFV_system.h.

6.27.2.2. SH1106_SIZE_1

```
#define SH1106_SIZE_1 0
```

Definición en la línea 124 del archivo LVFV_system.h.

6.27.2.3. SH1106_SIZE_2

```
#define SH1106_SIZE_2 1
```

Definición en la línea 125 del archivo LVFV_system.h.

6.27.2.4. VARIABLE_EIGTH

```
#define VARIABLE_EIGTH 83
```

Definición en la línea 24 del archivo LVFV_system.h.

6.27.2.5. VARIABLE_FIFTH

```
#define VARIABLE_FIFTH 56
```

Definición en la línea 21 del archivo LVFV_system.h.

6.27.2.6. VARIABLE_FIRST

```
#define VARIABLE_FIRST 20
```

Definición en la línea 17 del archivo LVFV_system.h.

6.27.2.7. VARIABLE_FOURTH

```
#define VARIABLE_FOURTH 47
```

Definición en la línea 20 del archivo LVFV_system.h.

6.27.2.8. VARIABLE_SECOND

```
#define VARIABLE_SECOND 29
```

Definición en la línea 18 del archivo LVFV_system.h.

6.27.2.9. VARIABLE_SEVENTH

```
#define VARIABLE_SEVENTH 74
```

Definición en la línea 23 del archivo LVFV_system.h.

6.27.2.10. VARIABLE_SIXTH

```
#define VARIABLE_SIXTH 65
```

Definición en la línea 22 del archivo LVFV_system.h.

6.27.2.11. VARIABLE_THIRD

```
#define VARIABLE_THIRD 38
```

Definición en la línea 19 del archivo LVFV_system.h.

6.27.3. Documentación de «typedef»

6.27.3.1. frequency_settings_SH1106_t

```
typedef struct frequency_settings_SH1106_t frequency_settings_SH1106_t
```

6.27.3.2. frequency_settings_t

```
typedef struct frequency_settings_t frequency_settings_t
```

6.27.3.3. seccurity_settings_SH1106_t

```
typedef struct seccurity_settings_SH1106_t seccurity_settings_SH1106_t
```

6.27.3.4. seccurity_settings_t

```
typedef struct seccurity_settings_t seccurity_settings_t
```

6.27.3.5. sh1106_variable_lines_e

```
typedef enum sh1106_variable_lines_e sh1106_variable_lines_e
```

6.27.3.6. system_status_e

```
typedef enum system_status_e system_status_e
```

6.27.3.7. system_status_t

```
typedef struct system_status_t system_status_t
```

6.27.3.8. time_settings_SH1106_t

```
typedef struct time_settings_SH1106_t time_settings_SH1106_t
```

6.27.3.9. time_settings_t

```
typedef struct time_settings_t time_settings_t
```

6.27.4. Documentación de enumeraciones

6.27.4.1. sh1106_variable_lines_e

```
enum sh1106_variable_lines_e
```

Valores de enumeraciones

first	
second	
third	
fourth	
fifth	
sixth	
seventh	

Definición en la línea 76 del archivo LVFV_system.h.

6.27.4.2. system_status_e

```
enum system_status_e
```

Posibles estados general que puede tomar el sistema.

Valores de enumeraciones

SYSTEM_IDLE	
SYSTEM_ACCLE_DESACCEL	
SYSTEM_REGIME	
SYSTEM_BREAKING	
SYSTEM_EMERGENCY	
SYSTEM_EMERGENCY_OK	

Definición en la línea 67 del archivo LVFV_system.h.

6.27.4.3. systemSignal_e

```
enum systemSignal_e
```

Variable dedicada a la identificación del tipo de señal que es recibida por las entradas digitales y analógicas del sistema para que se tome una desición 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_EXCEDED	
SECURITY_OK	

Definición en la línea 31 del archivo LVFV_system.h.

6.28. LVFV_system.h

Ir a la documentación de este archivo.

```
00001
00009 #ifndef LVFV_SYSTEM_H
00010 #define LVFV_SYSTEM_H
00011
00012 #include <inttypes.h>
00013 #include <time.h>
00014
00015 #define LINE_INCREMENT                  9
00016
00017 #define VARIABLE_FIRST                  20
00018 #define VARIABLE_SECOND                 29
00019 #define VARIABLE_THIRD                  38
00020 #define VARIABLE_FOURTH                 47
00021 #define VARIABLE_FIFTH                  56
00022 #define VARIABLE_SIXTH                  65
00023 #define VARIABLE_SEVENTH                74
00024 #define VARIABLE_EIGTH                  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     EMERGENCI_STOP_PRESSED,
00041     EMERGENCI_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_EXCEDED,
00059     SECURITY_OK
00060 } systemSignal_e;
00061
00067 typedef enum system_status_e {
```

```
00068     SYSTEM_IDLE,
00069     SYSTEM_ACCLE_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 seccurity_settings_t {
00100     uint16_t vbus_min;
00101     uint16_t ibus_max;
00102 } seccurity_settings_t;
00103
00109 typedef struct system_status_t {
00110     uint16_t frequency;                                    // Frecuencia actual de giro del motor
00111     uint16_t frequency_destiny;                            // Frecuencia a la que terminará
      funcionando el motor
00112     uint16_t vbus_min;                                     // Tensión mínima del bus de contínua en
      la que será una condición buena
00113     uint16_t ibus_max;                                     // Corriente máxima del bus de contínua en
      la que será una condición buena
00114     uint16_t acceleration;                                 // Velocidad de aceleración del motor
      configurada por el usuario
00115     uint16_t desacceleration;                              // Velocidad de desaceleración del motor
      configurada por el usuario
00116     uint8_t inputs_status;                                 // Índice de entrada aislada seleccionado
      para la variación de velocidad
00117     system_status_e status;                                // Estado general del sistema
00118     uint8_t emergency_signals;                             // Estado actual de las señales de
      emergencia b0: Entrada emergencia; b1: Security; b3: Termo-switch
00119 } system_status_t;
00120
00121 /*
00122  * Definiciones para display
00123  */
00124 #define SH1106_SIZE_1           0
00125 #define SH1106_SIZE_2           1
00126
00127 typedef struct frequency_settings_SH1106_t {
00128     frequency_settings_t frequency_settings;
00129     sh1106_variable_lines_e *edit;
00130     uint8_t edit_variable;
00131     uint8_t *edit_flag;
00132     uint8_t *multiplier;
00133 } frequency_settings_SH1106_t;
00134
00135 typedef struct time_settings_SH1106_t {
00136     time_settings_t time_settings;
00137     sh1106_variable_lines_e *edit;
00138     uint8_t edit_variable;
00139     uint8_t *edit_flag;
00140     uint8_t *multiplier;
00141 } time_settings_SH1106_t;
00142
00143 typedef struct seccurity_settings_SH1106_t {
00144     seccurity_settings_t seccurity_settings;
00145     sh1106_variable_lines_e *edit;
00146     uint8_t edit_variable;
00147     uint8_t *edit_flag;
00148     uint8_t *multiplier;
00149 } seccurity_settings_SH1106_t;
00150
00151 #endif
```

6.29. Referencia del archivo main/main.c

Inicialización de tareas y periféricos que no tienen una tarea específica. El resto de los periféricos se inicializan en las tareas que son utilizados.

```
#include <inttypes.h>
#include <time.h>
#include <sys/time.h>
#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_chip_info.h"
#include "esp_flash.h"
#include "esp_system.h"
#include "esp_log.h"
#include "esp_err.h"
#include "esp_sleep.h"
#include "esp_clk_tree.h"
#include "driver/gpio.h"
#include "./display/display.h"
#include "./io_control/io_control.h"
#include "./System/SysControl.h"
#include "./adc/adc.h"
#include "./nvs/nvs.h"
#include "./rtc/rtc.h"
#include "./wifi/wifi.h"
#include "LVFV_system.h"
```

Funciones

- void app_main (void)

  Tag de logs del sistema.

Variables

- static const char * TAG = "UTN-CA-PF2025"

6.29.1. Descripción detallada

Inicialización de tareas y periféricos que no tienen una tarea específica. El resto de los periféricos se inicializan en las tareas que son utilizados.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo main.c.

6.29.2. Documentación de funciones

6.29.2.1. app_main()

```
void app_main (
             void )
```

Tag de logs del sistema.

Función principal de la aplicación

Definición en la línea 43 del archivo main.c.

6.29.3. Documentación de variables

6.29.3.1. TAG

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

Definición en la línea 37 del archivo main.c.

6.30. main.c

Ir a la documentación de este archivo.

```
00001
00009 #include <inttypes.h>
00010 #include <time.h>
00011 #include <sys/time.h>
00012
00013 #include "sdkconfig.h"
00014 #include "freertos/FreeRTOS.h"
00015 #include "freertos/task.h"
00016
00017 #include "esp_chip_info.h"
00018 #include "esp_flash.h"
00019 #include "esp_system.h"
00020 #include "esp_log.h"
00021 #include "esp_err.h"
00022 #include "esp_sleep.h"
00023 #include "esp_clk_tree.h"
00024
00025 #include "driver/gpio.h"
00026
00027 #include "./display/display.h"
00028 #include "./io_control/io_control.h"
00029 #include "./System/SysControl.h"
00030 #include "./adc/adc.h"
00031 #include "./nvs/nvs.h"
00032 #include "./rtc/rtc.h"
00033 #include "./wifi/wifi.h"
00034
00035 #include "LVFV_system.h"
00036
00037 static const char *TAG = "UTN-CA-PF2025";
00038
00043 void app_main(void) {
00044
00045     nvs_init_once();
00046     if ( initRTC() == ESP_OK ) {
00047         ESP_LOGI(TAG, "RTC inicializado correctamente");
00048     } else {
00049         ESP_LOGE(TAG, "Error al inicializar el RTC");
00050         ESP_ERROR_CHECK(ESP_FAIL);
00051     }
00052     xTaskCreatePinnedToCore( task_display,                  "Tarea Display",                  4096,
      NULL, 10, NULL, 1);
```

```
00053
00054     xTaskCreatePinnedToCore(adc_task,                       "adc_task",                       4096,
      NULL, 9,  NULL, 0);
00055     xTaskCreatePinnedToCore(SPI_communication,              "SPI_communication",              4096,
      NULL, 9,  NULL, 0);
00056     xTaskCreatePinnedToCore(GPIO_interrupt_attendance_task, "GPIO_interrupt_attendance_task",  4096,
      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 }
```

6.31. Referencia del archivo main/nvs/nvs.c

Funciones de lectura y escritura de variables no volátiles. Carga en sistema además las variables de seguridad al momento de ser guardadas en la memoria.

```
#include "nvs_flash.h"
#include "nvs.h"
#include "esp_log.h"
#include "../LVFV_system.h"
```

defines

- #define save_frequency(freq_op)
- #define save_acceleration(freq_acceleration)
- #define save_desacceleration(freq_desacceleration)
- #define save_input_variable(freq_input_variable)
- #define save_vbus_min(vbus_min)
- #define save_ibus_max(ibus_max)
- #define save_hour_ini(hour_ini)
- #define save_min_ini(min_ini)
- #define save_hour_fin(hour_fin)
- #define save_min_fin(min_fin)
- #define load_frequency(freq_op)
- #define load_acceleration(freq_acceleration)
- #define load_desacceleration(freq_desacceleration)
- #define load_input_variable(freq_input_variable)
- #define load_vbus_min(vbus_min)
- #define load_ibus_max(ibus_max)
- #define load_hour_ini(hour_ini)
- #define load_min_ini(min_ini)
- #define load_hour_fin(hour_fin)
- #define load_min_fin(min_fin)

Funciones

- static esp_err_t save_16 (const char *var_tag, int16_t value)

  Accede a la memoria NVS para guardar un dato de 16 bits en memoria no volatil.
- static esp_err_t load_16 (const char *var_tag, int16_t *value, int16_t defval)

  Accede a la memoria NVS para obtener un dato de 16 bits guardado en memoria no volatil.
- esp_err_t nvs_init_once (void)

  Inicializa la memoria NVS una única vez.
- esp_err_t load_variables (frequency_settings_t *frequency_settings, time_settings_t *time_settings, seccurity_settings_t *seccurity_settings)

  Carga desde NVS los parámetros de frecuencia, seguridad y ventanas horarias.
- esp_err_t save_variables (frequency_settings_t *frequency_settings, time_settings_t *time_settings, seccurity_settings_t *seccurity_settings)

  Guarda en NVS los parámetros de frecuencia, seguridad y ventanas horarias.

Variables

- static const char * TAG = "NVS"

6.31.1. Descripción detallada

Funciones de lectura y escritura de variables no volátiles. Carga en sistema además las variables de seguridad al momento de ser guardadas en la memoria.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo nvs.c.

6.31.2. Documentación de «define»

6.31.2.1. load_acceleration

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

Valor:

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

Definición en la línea 28 del archivo nvs.c.

6.31.2.2. load_desacceleration

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

Valor:

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

Definición en la línea 29 del archivo nvs.c.

6.31.2.3. load_frequency

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

Valor:

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

Definición en la línea 27 del archivo nvs.c.

6.31.2.4. load_hour_fin

```
#define load_hour_fin(
              hour_fin)
```

Valor:

```
load_16("hour_fin", (hour_fin), 20 )
```

Definición en la línea 35 del archivo nvs.c.

6.31.2.5. load_hour_ini

```
#define load_hour_ini(
              hour_ini)
```

Valor:

```
load_16("hour_ini", (hour_ini), 20 )
```

Definición en la línea 33 del archivo nvs.c.

6.31.2.6. load_ibus_max

```
#define load_ibus_max(
              ibus_max)
```

Valor:

```
load_16("ibus_max", (ibus_max), 2000 )
```

Definición en la línea 32 del archivo nvs.c.

6.31.2.7. load_input_variable

```
#define load_input_variable(
              freq_input_variable)
```

Valor:

```
load_16("freq_input", (freq_input_variable), 1 )
```

Definición en la línea 30 del archivo nvs.c.

6.31.2.8. load_min_fin

```
#define load_min_fin(
              min_fin)
```

Valor:

```
load_16("min_fin", (min_fin), 35 )
```

Definición en la línea 36 del archivo nvs.c.

6.31.2.9. load_min_ini

```
#define load_min_ini(
              min_ini)
```

Valor:

```
load_16("min_ini", (min_ini), 30 )
```

Definición en la línea 34 del archivo nvs.c.

6.31.2.10. load_vbus_min

```
#define load_vbus_min(
              vbus_min)
```

Valor:

```
load_16("vbus_min", (vbus_min), 310 )
```

Definición en la línea 31 del archivo nvs.c.

6.31.2.11. save_acceleration

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

Valor:

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

Definición en la línea 17 del archivo nvs.c.

6.31.2.12. save_desacceleration

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

Valor:

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

Definición en la línea 18 del archivo nvs.c.

6.31.2.13. save_frequency

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

Valor:

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

Definición en la línea 16 del archivo nvs.c.

6.31.2.14. save_hour_fin

```
#define save_hour_fin(
              hour_fin)
```

Valor:

```
save_16("hour_fin", (hour_fin) )
```

Definición en la línea 24 del archivo nvs.c.

6.31.2.15. save_hour_ini

```
#define save_hour_ini(
              hour_ini)
```

Valor:

```
save_16("hour_ini", (hour_ini) )
```

Definición en la línea 22 del archivo nvs.c.

6.31.2.16. save_ibus_max

```
#define save_ibus_max(
              ibus_max)
```

Valor:

```
save_16("ibus_max", (ibus_max) )
```

Definición en la línea 21 del archivo nvs.c.

6.31.2.17. save_input_variable

```
#define save_input_variable(
              freq_input_variable)
```

Valor:

```
save_16("freq_input", (freq_input_variable) )
```

Definición en la línea 19 del archivo nvs.c.

6.31.2.18. save_min_fin

```
#define save_min_fin(
              min_fin)
```

Valor:

```
save_16("min_fin", (min_fin) )
```

Definición en la línea 25 del archivo nvs.c.

6.31.2.19. save_min_ini

```
#define save_min_ini(
              min_ini)
```

Valor:

```
save_16("min_ini", (min_ini) )
```

Definición en la línea 23 del archivo nvs.c.

6.31.2.20. save_vbus_min

```
#define save_vbus_min(
              vbus_min)
```

Valor:

```
save_16("vbus_min", (vbus_min) )
```

Definición en la línea 20 del archivo nvs.c.

6.31.3. Documentación de funciones

6.31.3.1. load_16()

```
esp_err_t load_16 (
            const char * var_tag,
            int16_t * value,
            int16_t defval)  [static]
```

Accede a la memoria NVS para obtener un dato de 16 bits guardado en memoria no volatil.

Abre la memoria NVS con el namespace "storage" y intenta leer la variable cuyo namespace es `var_tag` para guardarla en `value`. En caso de error, guarda `defval` como valor default en `value`.

Parámetros

in	*var_tag*	Nombre del namespace donde se almacena la variable que se está intentando leer. Es un string.
out	*value*	Puntero donde se guardará el dato que se desea leer de la memoria.
in	*defval*	Valor default que se utiliza en caso que nunca se haya creado la variable o si existe algún problema al abrir la NVS.

Valores devueltos

-	ESP_OK: Si la lectura de la variable fue exitosa • ESP_FAIL: si hay un interno error; generalmente ocurre cuando la memoria tiene una oartición 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 espacia para une nueva entrad o si hay demasiados namespaces diferentes (max 254). • ESP_ERR_NOT_ALLOWED: Si la partición es solo lectura y se abrió en modo NVS_READWRITE. • ESP_ERR_INVALID_ARG: Si el handler de la NVS es NULL.

Definición en la línea 203 del archivo nvs.c.

6.31.3.2. load_variables()

```
esp_err_t load_variables (
            frequency_settings_t * frequency_settings,
            time_settings_t * time_settings,
            seccurity_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	*frequency_settings*	Estructura de parámetros de frecuencia a cargar.
out	*time_settings*	Estructura de parámetros de tiempo (ventanas start/stop) a cargar.
out	*seccurity_settings*	Estructura de parámetros de seguridad (vbus/ibus) a cargar.

Valores devueltos

-	ESP_OK: Si todas las lecturas fueron exitosas (o claves ausentes con default aplicado). • ESP_FAIL: si hay un error interno. • ESP_ERR_NVS_NOT_INITIALIZED: si la NVS no está inicializada. • ESP_ERR_NVS_PART_NOT_FOUND: si la partición "nvs" no se encuentra. • ESP_ERR_NVS_INVALID_NAME: si alguna clave no cumple restricciones. • ESP_ERR_NO_MEM / ESP_ERR_NVS_NOT_ENOUGH_SPACE: errores internos de la NVS. • ESP_ERR_NOT_ALLOWED: si la partición es solo lectura. • ESP_ERR_INVALID_ARG: si el handler interno de NVS es NULL.

Definición en la línea 245 del archivo nvs.c.

6.31.3.3. nvs_init_once()

```
esp_err_t nvs_init_once (
            void )
```

Inicializa la memoria NVS una única vez.

Intenta inicializar la NVS mediante nvs_flash_init(). Si la memoria está llena (ESP_ERR_NVS_NO_FREE_PAGES) o se detecta una nueva versión de NVS (ESP_ERR_NVS_NEW_VERSION_FOUND), se borra la partición NVS y se vuelve a inicializar. No realiza inicialización repetida si ya fue hecha por el sistema.

Valores devueltos

-	ESP_OK: Si la inicialización fue exitosa. • ESP_ERR_NVS_NO_FREE_PAGES: Sin páginas libres; requiere borrado y reinicialización. • ESP_ERR_NVS_NEW_VERSION_FOUND: Versión de NVS incompatible; requiere borrado. • ESP_ERR_NVS_NOT_INITIALIZED: si la memoria no pudo inicializarse. • ESP_ERR_NVS_PART_NOT_FOUND: si la partición con la etiqueta "nvs" no se encuentra. • ESP_ERR_NOT_ALLOWED: Si la partición es solo lectura. • Otros códigos de error propagados por nvs_flash_init() o nvs_flash_erase().

Definición en la línea 236 del archivo nvs.c.

6.31.3.4. save_16()

```
esp_err_t save_16 (
            const char * var_tag,
            int16_t value)  [static]
```

Accede a la memoria NVS para guardar un dato de 16 bits en memoria no volatil.

Abre la memoria NVS con el namespace "storage" y guarda la variable cuyo namespace es `var_tag` para guardarla en `value`.

Parámetros

in	*var_tag*	Nombre del namespace donde se almacenará la variable que se está intentando guardar. Es un string.
out	*value*	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 oartición 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 espacia para une nueva entrad o si hay demasiados namespaces diferentes (max 254). • ESP_ERR_NOT_ALLOWED: Si la partición es solo lectura y se abrió en modo NVS_READWRITE. • ESP_ERR_INVALID_ARG: Si el handler de la NVS es NULL.

Definición en la línea 104 del archivo nvs.c.

6.31.3.5. save_variables()

```
esp_err_t save_variables (
            frequency_settings_t * frequency_settings,
            time_settings_t * time_settings,
            seccurity_settings_t * seccurity_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	*frequency_settings*	Estructura con parámetros de frecuencia a guardar.
in	*time_settings*	Estructura con parámetros de tiempo (ventanas start/stop) a guardar.
in	*seccurity_settings*	Estructura con parámetros de seguridad (vbus/ibus) a guardar.

Valores devueltos

-	ESP_OK: Si todas las escrituras fueron exitosas. • ESP_FAIL: si hay un error interno. • ESP_ERR_NVS_NOT_INITIALIZED: si la NVS no está inicializada. • ESP_ERR_NVS_PART_NOT_FOUND: si la partición "nvs" no se encuentra. • ESP_ERR_NVS_INVALID_NAME: si alguna clave no cumple restricciones. • ESP_ERR_NO_MEM / ESP_ERR_NVS_NOT_ENOUGH_SPACE: errores internos de la NVS o espacio insuficiente. • ESP_ERR_NOT_ALLOWED: si la partición es solo lectura. • ESP_ERR_INVALID_ARG: si el handler interno de NVS es NULL.

Definición en la línea 309 del archivo nvs.c.

6.31.4. Documentación de variables

6.31.4.1. TAG

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

Definición en la línea 38 del archivo nvs.c.

6.32. nvs.c

Ir a la documentación de este archivo.

```
00001
00009 #include "nvs_flash.h"
00010 #include "nvs.h"
00011 #include "esp_log.h"
00012
00013 #include "../LVFV_system.h"
00014 #include "./nvs.h"
00015
00016 #define save_frequency(freq_op)                          save_16("freq_freq", (freq_op) )
00017 #define save_acceleration(freq_acceleration)             save_16("freq_acce",
      (freq_acceleration) )
00018 #define save_desacceleration(freq_desacceleration)       save_16("freq_desa",
      (freq_desacceleration) )
00019 #define save_input_variable(freq_input_variable)         save_16("freq_input",
      (freq_input_variable) )
00020 #define save_vbus_min(vbus_min)                          save_16("vbus_min", (vbus_min) )
00021 #define save_ibus_max(ibus_max)                          save_16("ibus_max", (ibus_max) )
00022 #define save_hour_ini(hour_ini)                          save_16("hour_ini", (hour_ini) )
00023 #define save_min_ini(min_ini)                            save_16("min_ini", (min_ini) )
00024 #define save_hour_fin(hour_fin)                          save_16("hour_fin", (hour_fin) )
00025 #define save_min_fin(min_fin)                            save_16("min_fin", (min_fin) )
00026
00027 #define load_frequency(freq_op)                          load_16("freq_freq", (freq_op), 50 )
00028 #define load_acceleration(freq_acceleration)             load_16("freq_acce",
      (freq_acceleration), 5 )
00029 #define load_desacceleration(freq_desacceleration)       load_16("freq_desa",
      (freq_desacceleration), 3 )
00030 #define load_input_variable(freq_input_variable)         load_16("freq_input",
      (freq_input_variable), 1 )
00031 #define load_vbus_min(vbus_min)                          load_16("vbus_min", (vbus_min), 310 )
00032 #define load_ibus_max(ibus_max)                          load_16("ibus_max", (ibus_max), 2000 )
00033 #define load_hour_ini(hour_ini)                          load_16("hour_ini", (hour_ini), 20 )
00034 #define load_min_ini(min_ini)                            load_16("min_ini", (min_ini), 30 )
00035 #define load_hour_fin(hour_fin)                          load_16("hour_fin", (hour_fin), 20 )
```

```
00036 #define load_min_fin(min_fin)                                   load_16("min_fin", (min_fin), 35 )
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
      @p var_tag para guardarla en @p value.
00046 //  *
00047 //  * @param[in] var_tag
00048 //  *      Nombre del namespace donde se almacenará la variable que se está intentando guardar. Es un
      string.
00049 //  *
00050 //  * @param[out] value
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
      oartición corrupta.
00056 //  *      - ESP_ERR_NVS_NOT_INITIALIZED: si la memoria no fue inicializada.
00057 //  *      - ESP_ERR_NVS_PART_NOT_FOUND: si la partición con la etiqueta "nvs" no se encuentra.
00058 //  *      - ESP_ERR_NVS_NOT_FOUND: id namespace doesn't exist yet and mode is NVS_READONLY.
00059 //  *      - ESP_ERR_NVS_INVALID_NAME: si el nombre del namespace no cumple con las restricciones.
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 espacia para une nueva entrad o si hay demasiados
      namespaces diferentes (max 254).
00062 //  *      - ESP_ERR_NVS_NOT_ALLOWED: Si la partición es solo lectura y se abrió en modo NVS_READWRITE.
00063 //  *      - ESP_ERR_INVALID_ARG: Si el handler de la NVS es NULL.
00064 //  */
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
      namespace es @p var_tag para guardarla en @p value. En caso de error, guarda @p defval como valor
      default en @p value.
00127 //  *
00128 //  * @param[in] var_tag
00129 //  *      Nombre del namespace donde se almacena la variable que se está intentando leer. Es un
      string.
00130 //  *
00131 //  * @param[out] value
00132 //  *      Puntero donde se guardará el dato que se desea leer de la memoria.
00133 //  *
00134 //  * @param[in] defval
00135 //  *      Valor default que se utiliza en caso que nunca se haya creado la variable o si existe algún
      problema al abrir la NVS.
00136 //  *
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 espacia para une nueva entrad 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,
      seccurity_settings_t *seccurity_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,
      seccurity_settings_t *seccurity_settings) {
00310     esp_err_t retval;
00311     retval = save_frequency( (int16_t) frequency_settings->freq_regime);
00312     if ( retval != ESP_OK ){
00313         return retval;
00314     }
00315
00316     retval = save_acceleration( (int16_t) frequency_settings->acceleration);
00317     if ( retval != ESP_OK ){
00318         return retval;
00319     }
00320
00321     retval = save_desacceleration( (int16_t) frequency_settings->desacceleration);
00322     if ( retval != ESP_OK ){
00323         return retval;
00324     }
00325
00326     retval = save_input_variable( (int16_t) frequency_settings->input_variable);
00327     if ( retval != ESP_OK ){
00328         return retval;
00329     }
00330
00331     retval = save_vbus_min( (int16_t) seccurity_settings->vbus_min);
00332     if ( retval != ESP_OK ){
00333         return retval;
00334     }
00335
00336     retval = save_ibus_max( (int16_t) seccurity_settings->ibus_max);
00337     if ( retval != ESP_OK ){
```

```
00338         return retval;
00339     }
00340
00341     retval = save_hour_ini( (int16_t) time_settings->time_start->tm_hour);
00342     if ( retval != ESP_OK ){
00343         return retval;
00344     }
00345
00346     retval = save_min_ini( (int16_t) time_settings->time_start->tm_min);
00347     if ( retval != ESP_OK ){
00348         return retval;
00349     }
00350
00351     retval = save_hour_fin( (int16_t) time_settings->time_stop->tm_hour);
00352     if ( retval != ESP_OK ){
00353         return retval;
00354     }
00355
00356     retval = save_min_fin( (int16_t) time_settings->time_stop->tm_min);
00357     if ( retval != ESP_OK ){
00358         return retval;
00359     }
00360
00361     return retval;
00362 }
```

6.33. Referencia del archivo main/nvs/nvs.h

Declaración de funciones de lectura y escritura de memoria no volatil. Útiles para los reset y no tener que reconfigurar siempre las variables básicas del sistema.

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

Funciones

- esp_err_t nvs_init_once (void)

  Inicializa la memoria NVS una única vez.
- esp_err_t load_variables (frequency_settings_t *frequency_settings, time_settings_t *time_settings, seccurity_settings_t *seccurity_settings)

  Carga desde NVS los parámetros de frecuencia, seguridad y ventanas horarias.
- esp_err_t save_variables (frequency_settings_t *frequency_settings, time_settings_t *time_settings, seccurity_settings_t *seccurity_settings)

  Guarda en NVS los parámetros de frecuencia, seguridad y ventanas horarias.

6.33.1. Descripción detallada

Declaración de funciones de lectura y escritura de memoria no volatil. Útiles para los reset y no tener que reconfigurar siempre las variables básicas del sistema.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo nvs.h.

6.33.2. Documentación de funciones

6.33.2.1. load_variables()

```
esp_err_t load_variables (
            frequency_settings_t * frequency_settings,
            time_settings_t * time_settings,
            seccurity_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	*frequency_settings*	Estructura de parámetros de frecuencia a cargar.
out	*time_settings*	Estructura de parámetros de tiempo (ventanas start/stop) a cargar.
out	*seccurity_settings*	Estructura de parámetros de seguridad (vbus/ibus) a cargar.

Valores devueltos

-	ESP_OK: Si todas las lecturas fueron exitosas (o claves ausentes con default aplicado). • ESP_FAIL: si hay un error interno. • ESP_ERR_NVS_NOT_INITIALIZED: si la NVS no está inicializada. • ESP_ERR_NVS_PART_NOT_FOUND: si la partición "nvs" no se encuentra. • ESP_ERR_NVS_INVALID_NAME: si alguna clave no cumple restricciones. • ESP_ERR_NO_MEM / ESP_ERR_NVS_NOT_ENOUGH_SPACE: errores internos de la NVS. • ESP_ERR_NOT_ALLOWED: si la partición es solo lectura. • ESP_ERR_INVALID_ARG: si el handler interno de NVS es NULL.

Definición en la línea 245 del archivo nvs.c.

6.33.2.2. nvs_init_once()

```
esp_err_t nvs_init_once (
            void )
```

Inicializa la memoria NVS una única vez.

Intenta inicializar la NVS mediante nvs_flash_init(). Si la memoria está llena (ESP_ERR_NVS_NO_FREE_PAGES) o se detecta una nueva versión de NVS (ESP_ERR_NVS_NEW_VERSION_FOUND), se borra la partición NVS y se vuelve a inicializar. No realiza inicialización repetida si ya fue hecha por el sistema.

Valores devueltos

-	ESP_OK: Si la inicialización fue exitosa. • ESP_ERR_NVS_NO_FREE_PAGES: Sin páginas libres; requiere borrado y reinicialización. • ESP_ERR_NVS_NEW_VERSION_FOUND: Versión de NVS incompatible; requiere borrado. • ESP_ERR_NVS_NOT_INITIALIZED: si la memoria no pudo inicializarse. • ESP_ERR_NVS_PART_NOT_FOUND: si la partición con la etiqueta "nvs" no se encuentra. • ESP_ERR_NOT_ALLOWED: Si la partición es solo lectura. • Otros códigos de error propagados por nvs_flash_init() o nvs_flash_erase().

Definición en la línea 236 del archivo nvs.c.

6.33.2.3. save_variables()

```
esp_err_t save_variables (
            frequency_settings_t * frequency_settings,
            time_settings_t * time_settings,
            seccurity_settings_t * seccurity_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	*frequency_settings*	Estructura con parámetros de frecuencia a guardar.
in	*time_settings*	Estructura con parámetros de tiempo (ventanas start/stop) a guardar.
in	*seccurity_settings*	Estructura con parámetros de seguridad (vbus/ibus) a guardar.

Valores devueltos

-	ESP_OK: Si todas las escrituras fueron exitosas. • ESP_FAIL: si hay un error interno. • ESP_ERR_NVS_NOT_INITIALIZED: si la NVS no está inicializada. • ESP_ERR_NVS_PART_NOT_FOUND: si la partición "nvs" no se encuentra. • ESP_ERR_NVS_INVALID_NAME: si alguna clave no cumple restricciones. • ESP_ERR_NO_MEM / ESP_ERR_NVS_NOT_ENOUGH_SPACE: errores internos de la NVS o espacio insuficiente. • ESP_ERR_NOT_ALLOWED: si la partición es solo lectura. • ESP_ERR_INVALID_ARG: si el handler interno de NVS es NULL.

Definición en la línea 309 del archivo nvs.c.

6.34. nvs.h

Ir a la documentación de este archivo.

```
00001
00009 #ifndef VARIABLE_ADMIN_H
00010 #define VARIABLE_ADMIN_H
00011
00012 #include <stdint.h>
00013 #include "../LVFV_system.h"
00033 esp_err_t nvs_init_once(void);
00034
00064 esp_err_t load_variables(frequency_settings_t *frequency_settings, time_settings_t *time_settings,
      seccurity_settings_t *seccurity_settings);
00065
00093 esp_err_t save_variables(frequency_settings_t *frequency_settings, time_settings_t *time_settings,
      seccurity_settings_t *seccurity_settings);
00094
00095 #endif
```

6.35. Referencia del archivo main/rtc/rtc.c

Funciones de lectura y escritura del RTC y alarmas.

```
#include <sys/time.h>
#include "sdkconfig.h"
#include "esp_err.h"
#include "esp_timer.h"
#include "esp_log.h"
#include "../system/sysControl.h"
#include "RTC.h"
```

Estructuras de datos

- struct rtc_alarms_t

Funciones

- static esp_err_t program_alarm (esp_timer_handle_t *timer_handle, void(*callback)(void *), int hour, int minute, const char *name)

  Función dedicada a programar la alarma según hour y minute.
- static void alarm_start_cb (void *arg)

  Función programada como alarma para iniciar el funcionamiento del motor.
- static void alarm_stop_cb (void *arg)

  Función programada como alarma para finalizar el funcionamiento del motor.
- esp_err_t initRTC ()

  Inicializa el RTC en la hora 00:00:00.
- esp_err_t setTime (struct tm *setting_time)

  Inicializa el RTC en la hora cargada en setting_time.
- esp_err_t getTime (struct tm *current_timeinfo)

  Carga la hora actuak desde el RTC del ESP32.
- esp_err_t rtc_schedule_alarms (time_settings_t *time_settings)

  Inicializa las alarmas de inicio y fin de funcionamiento del motor.

Variables

- static const char * TAG = "RTC"
- static rtc_alarms_t rtc_alarms
- static time_settings_t alarm_settings
- static struct tm time_start
- static struct tm time_stop

6.35.1. Descripción detallada

Funciones de lectura y escritura del RTC y alarmas.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo rtc.c.

6.35.2. Documentación de funciones

6.35.2.1. alarm_start_cb()

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

Función programada como alarma para iniciar el funcionamiento del motor.

Envía la señal al sistema para poder iniciar el funcionamiento del motor según la frecuencia de régimen, aceleración configuradas.

Parámetros

in	*arg*	Sin uso.

Definición en la línea 212 del archivo rtc.c.

6.35.2.2. alarm_stop_cb()

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

Función programada como alarma para finalizar el funcionamiento del motor.

Envía la señal al sistema para poder finalizar el funcionamiento del motor según la desaceleración configurada.

Parámetros

in	*arg*	Sin uso.

Definición en la línea 233 del archivo rtc.c.

6.35.2.3. getTime()

```
esp_err_t getTime (
            struct tm * current_timeinfo)
```

Carga la hora actuak desde el RTC del ESP32.

Lee la hora del RTC del ESP32 expresada en unix time y la convierte en un formato entendible por el usuario en la estructura `current_timeinfo`

Parámetros

in	*current_timeinfo*	Esturctura tm en donde se cargará la fecha y hora del sistema.

Devuelve

- ESP_OK si carga la hora exitosamente.
- ESP_ERR_INVALID_ARG si `current_timeinfo` es NULL

Definición en la línea 124 del archivo rtc.c.

6.35.2.4. initRTC()

```
esp_err_t initRTC ()
```

Inicializa el RTC en la hora 00:00:00.

No es importante el día configurado para el sistema, pero carga la hora 0 de UNIX_TIME.

Devuelve

- ESP_OK siempre.

Definición en la línea 92 del archivo rtc.c.

6.35.2.5. program_alarm()

```
esp_err_t program_alarm (
            esp_timer_handle_t * timer_handle,
            void(* callback )(void *),
            int hour,
            int minute,
            const char * name)  [static]
```

Función dedicada a programar la alarma según hour y minute.

No distingue de inicio o fin de marcha, para ello se le pasa el puntero a función 'callback' para la función de inicio o fin de marcha, el handler 'timer_handeler' y un nombre descriptivo. Los handlers de los timers son creados como tareas y no como interrupciones para poder utilizar las funciones .

Parámetros

out	*timer_handle*	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 (esp_timer↩ _stop / esp_timer_delete).
in	*callback*	Puntero a función de tipo `void (*)(void *)` que se invocará cuando la alarma dispare. Se ejecuta en el contexto de la tarea interna de esp_timer (no en ISR). Puede usar APIs no-ISR (colas, logs, etc.).
in	*hour*	Hora en formato 24 h. Rango válido: 0–23.
in	*minute*	Minutos. Rango válido: 0–59.
in	*name*	Nombre descriptivo del timer (aparece en logs/diagnóstico). Debe apuntar a memoria de duración estática (p. ej., literal o `static const`).

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 la allocación de memoria para los handlers falla

Definición en la línea 171 del archivo rtc.c.

6.35.2.6. rtc_schedule_alarms()

```
esp_err_t rtc_schedule_alarms (
            time_settings_t * time_settings)
```

Inicializa las alarmas de inicio y fin de funcionamiento del motor.

La hora y minuto cargada en time_start es la que dará inicio al funcionamiento del motor, mientas que la cargada en time_stop será la que finalizará el mismo.

Parámetros

in	*time_settings*	Esturctura time_settings_t que contiene los horarios de alarma de inicio y fin. Además contiene la hora actual, una variable que no es utilizada por la función.

Devuelve

- ESP_OK si Configura las alarmas exitosamente.
- ESP_FAIL si falla en la creación de las alarmas.

Definición en la línea 137 del archivo rtc.c.

6.35.2.7. setTime()

```
esp_err_t setTime (
            struct tm * setting_time)
```

Inicializa el RTC en la hora cargada en setting_time.

No es importante el día configurado para el sistema, pero carga la hora 0 de UNIX_TIME.

Parámetros

in	*setting_time*	Esturctura tm que es cargada en el RTC del ESP32 según su año, mes, día, hora, minuto y segundo.

Devuelve

- ESP_OK si carga el horario correctamente.
- ESP_ERR_INVALID_ARG en caso de recibir un puntero NULL

Definición en la línea 107 del archivo rtc.c.

6.35.3. Documentación de variables

6.35.3.1. alarm_settings

```
time_settings_t alarm_settings  [static]
```

Definición en la línea 31 del archivo rtc.c.

6.35.3.2. rtc_alarms

```
rtc_alarms_t rtc_alarms  [static]
```

Definición en la línea 30 del archivo rtc.c.

6.35.3.3. TAG

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

Definición en la línea 29 del archivo rtc.c.

6.35.3.4. time_start

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

Definición en la línea 33 del archivo rtc.c.

6.35.3.5. time_stop

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

Definición en la línea 34 del archivo rtc.c.

6.36. rtc.c

Ir a la documentación de este archivo.

```
00001
00009 #include <sys/time.h>
00010
00011 #include "sdkconfig.h"
00012 #include "esp_err.h"
00013 #include "esp_timer.h"
00014 #include "esp_log.h"
00015
00016 #include "../system/sysControl.h"
00017 #include "RTC.h"
00018
00024 typedef struct {
00025     esp_timer_handle_t start_timer;
00026     esp_timer_handle_t stop_timer;
00027 } rtc_alarms_t;
00028
00029 static const char *TAG = "RTC";                                  // Variable dedicada a la impresión de
      mensajes de logs
00030 static rtc_alarms_t rtc_alarms;                                  // Declaración de contenedor de handlers
      de alarma para inicio y fin de marcha
00031 static time_settings_t alarm_settings;                           // Settings de horarios de inicio y fin de
      marcha de motor
00032
00033 static struct tm time_start;
00034 static struct tm time_stop;
00035
00066 static esp_err_t program_alarm(esp_timer_handle_t *timer_handle, void (*callback)(void *), int hour,
      int minute, const char *name);
00067
00078 static void alarm_start_cb(void *arg);
00079
00090 static void alarm_stop_cb(void *arg);
00091
00092 esp_err_t initRTC() {
00093
00094     struct timeval tv = {
00095         .tv_sec = 0,
00096         .tv_usec = 0
00097     };
00098
00099     settimeofday(&tv, NULL); // Cargar en RTC interno
00100
00101     alarm_settings.time_start = &time_start;
00102     alarm_settings.time_stop = &time_stop;
00103
00104     return ESP_OK;
00105 }
00106
00107 esp_err_t setTime(struct tm *setting_time) {
00108
00109     if ( setting_time == NULL ) {
00110         return ESP_ERR_INVALID_ARG;
00111     }
00112
00113     time_t now = mktime(setting_time);  // Convierte a timestamp UNIX
00114
00115     struct timeval tv = {
00116         .tv_sec = now,
00117         .tv_usec = 0
00118     };
00119
00120     settimeofday(&tv, NULL); // Cargar en RTC interno
00121     return ESP_OK;
00122 }
00123
00124 esp_err_t getTime(struct tm *current_timeinfo) {
00125     time_t now;
00126
00127     if ( current_timeinfo == NULL ) {
00128         return ESP_ERR_INVALID_ARG;
00129     }
00130
00131     time(&now);
00132     localtime_r(&now, current_timeinfo);
00133
00134     return ESP_OK;
00135 }
00136
00137 esp_err_t rtc_schedule_alarms(time_settings_t *time_settings) {
00138
00139     if ( time_settings != NULL ) {
00140         alarm_settings.time_start->tm_hour = time_settings->time_start->tm_hour;
```

```
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
00161     if (err1 == ESP_OK && err2 == ESP_OK){
00162         ESP_LOGI(TAG, "Configurando alarmas: INICIO %02d:%02d", alarm_settings.time_start->tm_hour,
      alarm_settings.time_start->tm_min);
00163         ESP_LOGI(TAG, "                    :  FIN  %02d:%02d", alarm_settings.time_stop->tm_hour,
      alarm_settings.time_stop->tm_min);
00164         return ESP_OK;
00165     } else {
00166         ESP_LOGE(TAG, "ERROR AL CONFIGURAR LAS ALARMAS");
00167         return ESP_FAIL;
00168     }
00169 }
00170
00171 static esp_err_t program_alarm(esp_timer_handle_t *timer_handle, void (*callback)(void *), int hour,
      int minute, const char *name) {
00172     time_t now;
00173     struct tm timeinfo_alarm;
00174     time(&now);
00175     localtime_r(&now, &timeinfo_alarm);
00176
00177     struct tm alarm_tm = timeinfo_alarm;
00178     alarm_tm.tm_hour = hour;
00179     alarm_tm.tm_min  = minute;
00180     alarm_tm.tm_sec  = 0;
00181
00182     time_t alarm_epoch = mktime(&alarm_tm);
00183     if (alarm_epoch <= now) {
00184         alarm_epoch += 24 * 3600;
00185     }
00186
00187     int64_t usec_until_alarm = (alarm_epoch - now) * 1000000LL;
00188
00189     esp_timer_create_args_t timer_args = {
00190         .callback = callback,
00191         .arg = NULL,
00192         .dispatch_method = ESP_TIMER_TASK,
00193         .name = name
00194     };
00195
00196     esp_err_t err = esp_timer_create(&timer_args, timer_handle);
00197     if (err != ESP_OK) {
00198         return err;
00199     }
00200
00201     err = esp_timer_start_once(*timer_handle, usec_until_alarm);
00202     if (err != ESP_OK) {
00203         return err;
00204     }
00205
00206     ESP_LOGI(TAG, "Programada %s para %02d:%02d (en %lld segundos)", name, hour, minute, (alarm_epoch -
      now));
00207
00208     return ESP_OK;
00209 }
00210
00211 // Callback para inicio de tarea
00212 static void alarm_start_cb(void *arg) {
00213     ESP_LOGI(TAG, "Alarma de INICIO disparada");
00214
00215     // Cancelo las anteriores si existían
00216     if (rtc_alarms.start_timer) {
00217         esp_timer_stop(rtc_alarms.start_timer);
00218     }
00219     // Programo las nuevas
00220     ESP_ERROR_CHECK(program_alarm(&rtc_alarms.start_timer, alarm_start_cb,
      alarm_settings.time_start->tm_hour, alarm_settings.time_start->tm_min, "start_alarm"));
```

```
00221
00222     // Cancelo las anteriores si existían
00223     if (rtc_alarms.stop_timer) {
00224         esp_timer_stop(rtc_alarms.stop_timer);
00225     }
00226     // Programo las nuevas
00227     ESP_ERROR_CHECK(program_alarm(&rtc_alarms.stop_timer, alarm_stop_cb,
      alarm_settings.time_stop->tm_hour, alarm_settings.time_stop->tm_min, "stop_alarm"));
00228
00229     SystemEventPost(START_PRESSED);
00230 }
00231
00232 // Callback para fin de tarea
00233 static void alarm_stop_cb(void *arg) {
00234     ESP_LOGI(TAG, "Alarma de FIN disparada");
00235
00236     // Cancelo las anteriores si existían
00237     if (rtc_alarms.start_timer) {
00238         esp_timer_stop(rtc_alarms.start_timer);
00239     }
00240     // Programo las nuevas
00241     ESP_ERROR_CHECK(program_alarm(&rtc_alarms.start_timer, alarm_start_cb,
      alarm_settings.time_start->tm_hour, alarm_settings.time_start->tm_min, "start_alarm"));
00242
00243     // Cancelo las anteriores si existían
00244     if (rtc_alarms.stop_timer) {
00245         esp_timer_stop(rtc_alarms.stop_timer);
00246     }
00247     // Programo las nuevas
00248     ESP_ERROR_CHECK(program_alarm(&rtc_alarms.stop_timer, alarm_stop_cb,
      alarm_settings.time_stop->tm_hour, alarm_settings.time_stop->tm_min, "stop_alarm"));
00249
00250     SystemEventPost(STOP_PRESSED);
00251 }
```

6.37. Referencia del archivo main/rtc/rtc.h

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

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

Funciones

- esp_err_t initRTC ()

  Inicializa el RTC en la hora 00:00:00.
- esp_err_t setTime (struct tm *setting_time)

  Inicializa el RTC en la hora cargada en setting_time.
- esp_err_t rtc_schedule_alarms (time_settings_t *time_settings)

  Inicializa las alarmas de inicio y fin de funcionamiento del motor.
- esp_err_t getTime (struct tm *current_timeinfo)

  Carga la hora actuak desde el RTC del ESP32.

6.37.1. Descripción detallada

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

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo rtc.h.

6.37.2. Documentación de funciones

6.37.2.1. getTime()

```
esp_err_t getTime (
            struct tm * current_timeinfo)
```

Carga la hora actuak desde el RTC del ESP32.

Lee la hora del RTC del ESP32 expresada en unix time y la convierte en un formato entendible por el usuario en la estructura `current_timeinfo`

Parámetros

`in`	*current_timeinfo*	Esturctura tm en donde se cargará la fecha y hora del sistema.

Devuelve

- ESP_OK si carga la hora exitosamente.
- ESP_ERR_INVALID_ARG si `current_timeinfo` es NULL

Definición en la línea 124 del archivo rtc.c.

6.37.2.2. initRTC()

```
esp_err_t initRTC ()
```

Inicializa el RTC en la hora 00:00:00.

No es importante el día configurado para el sistema, pero carga la hora 0 de UNIX_TIME.

Devuelve

- ESP_OK siempre.

Definición en la línea 92 del archivo rtc.c.

6.37.2.3. rtc_schedule_alarms()

```
esp_err_t rtc_schedule_alarms (
            time_settings_t * time_settings)
```

Inicializa las alarmas de inicio y fin de funcionamiento del motor.

La hora y minuto cargada en time_start es la que dará inicio al funcionamiento del motor, mientas que la cargada en time_stop será la que finalizará el mismo.

Parámetros

in	*time_settings*	Esturctura time_settings_t que contiene los horarios de alarma de inicio y fin. Además contiene la hora actual, una variable que no es utilizada por la función.

Devuelve

- ESP_OK si Configura las alarmas exitosamente.
- ESP_FAIL si falla en la creación de las alarmas.

Definición en la línea 137 del archivo rtc.c.

6.37.2.4. setTime()

```
esp_err_t setTime (
            struct tm * setting_time)
```

Inicializa el RTC en la hora cargada en setting_time.

No es importante el día configurado para el sistema, pero carga la hora 0 de UNIX_TIME.

Parámetros

in	*setting_time*	Esturctura tm que es cargada en el RTC del ESP32 según su año, mes, día, hora, minuto y segundo.

Devuelve

- ESP_OK si carga el horario correctamente.
- ESP_ERR_INVALID_ARG en caso de recibir un puntero NULL

Definición en la línea 107 del archivo rtc.c.

6.38. rtc.h

Ir a la documentación de este archivo.

```
00001
00009 #ifndef __RTC_H__
00010
00011 #define __RTC_H__
00012
00013 #include "../LVFV_system.h"
00014 #include "esp_err.h"
00015
00026 esp_err_t initRTC();
00027
00042 esp_err_t setTime(struct tm *setting_time);
00043
00058 esp_err_t rtc_schedule_alarms(time_settings_t *time_settings);
00059
00074 esp_err_t getTime(struct tm *current_timeinfo);
00075
00076 #endif
```

6.39. Referencia del archivo main/system/sysAdmin.c

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

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

  Tareas dedicada a decrementar la frecuencia que es impresa en el display de acuerdo a la desaceleración y hasta la frecuencia de regimen o 0Hz.
- uint16_t engine_start ()

  Función que setea la frecuencia de régimen de acuerdo a lo configurado por el usuario y ejecuta la tarea accelerating.
- uint16_t engine_stop ()

  Función que setea la frecuencia de régimen de acuerdo a lo configurado por el usuario y ejecuta la tarea desaccelerating.
- 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.
- esp_err_t get_status (system_status_t *s_e)

  Obtiene una réplica del status del sistema.
- system_status_e update_meas (uint16_t vbus_meas, uint16_t ibus_meas)

  Actualiza las mediciones de tensión y corriente del bus de contínua.
- 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, seccurity_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

6.39.1. Descripción detallada

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

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo sysAdmin.c.

6.39.2. Documentación de funciones

6.39.2.1. accelerating()

```
void accelerating (
            void * pvParameters)  [static]
```

Tareas dedicada a incrementar la frecuencia que es impresa en el display de acuerdo a la aceleración y hasta la frecuencia de regimen.

Es una tarea que debe ser creada cada vez que se necesita incrementar la frecuencia impresa ya que, al terminar de llegar a régimen, termina su propia ejecución. Al terminar su ejecución pasa de SYSTEM_ACCLE_DESACCEL a SYSTEM_REGIME

Parámetros

in	*pvParameters*	Sin uso

Definición en la línea 53 del archivo sysAdmin.c.

6.39.2.2. change_frequency()

```
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.

De acuerdo a la frecuencia de régimen elegida y el tipo de variación, la frecuencia de destión tendrá diferentes valores, siempre menores a las de régimen.

Parámetros

in	*speed_slector*	Índice dentro del array con frecuencias de trabajo

Devuelve

0Hz si `speed_selector` fue mal pasada como argumento, sino la frecuencia de destino

Definición en la línea 151 del archivo sysAdmin.c.

6.39.2.3. desaccelerating()

```
void desaccelerating (
            void * pvParameters)  [static]
```

Tareas dedicada a decrementar la frecuencia que es impresa en el display de acuerdo a la desaceleración y hasta la frecuencia de regimen o 0Hz.

Es una tarea que debe ser creada cada vez que se necesita decrementar la frecuencia impresa ya que, al terminar de llegar a régimen, termina su propia ejecución. Al terminar su ejecución pasa de SYSTEM_BREAKING a SYSTEM_REGIME o de SYSTEM_ACCLE_DESACCEL a SYSTEM_REGIME. En caso de estar en SYSTEM_↩ EMERGENCY no cambia el status

Parámetros

in	*pvParameters*	Sin uso

Definición en la línea 69 del archivo sysAdmin.c.

6.39.2.4. engine_emergency_stop()

```
void engine_emergency_stop (
            uint8_t signal)
```

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

Termina las tareas accelerating y desaccelerating y guarda la señal que dispara la emergencia para contabilizar todos las señales vigentes

Parámetros

in	*signal*	Señal que genera el estado de emergencia

Definición en la línea 136 del archivo sysAdmin.c.

6.39.2.5. engine_emergency_stop_release()

```
bool engine_emergency_stop_release (
            uint8_t signal)
```

Pasa a estado SYSTEM_EMERGENCY_OK si todas las señales de emergencia son 0 para que el usuario pueda enviar la señal de stop al STM32 necesaria para poder arrancar nuevamente el motor.

Parámetros

in	*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.

6.39.2.6. engine_start()

```
uint16_t engine_start ()
```

Función que setea la frecuencia de régimen de acuerdo a lo configurado por el usuario y ejecuta la tarea accelerating.

En caso de que desaccelerating esté corriendo, antes de ejecutar accelerating, la cierra para no generar un efecto de subida y bajada de frecuencia. El status pasará a SYSTEM_ACCLE_DESACCEL

Devuelve

Frecuencia de destino configurada

Definición en la línea 93 del archivo sysAdmin.c.

6.39.2.7. engine_stop()

```
uint16_t engine_stop ()
```

Función que setea la frecuencia de régimen de acuerdo a lo configurado por el usuario y ejecuta la tarea desaccelerating.

En caso de que accelerating esté corriendo, antes de ejecutar desaccelerating, la cierra para no generar un efecto de subida y bajada de frecuencia. El status pasará a SYSTEM_ACCLE_DESACCEL o SYSTEM_BREAKING de a cuerdo a la acción que haya ocurrido

Devuelve

Frecuencia de destino configurada

Definición en la línea 106 del archivo sysAdmin.c.

6.39.2.8. get_status()

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

Obtiene una réplica del status del sistema.

Parámetros

out	*s_e*	Puntero a la variable donde se devolverá la información de status

Devuelve

- ESP_ERR_NOT_ALLOWED: En caso de que s_e sea un puntero a NULL
- ESP_OK: Si la copia fue exitosa

Definición en la línea 185 del archivo sysAdmin.c.

6.39.2.9. set_frequency_table()

```
void set_frequency_table (
            uint16_t input_variable,
            uint16_t freq_regime)
```

Carga en el vector de velocidades configurables por entradas, las diferentes frecuencias de trabajo.

La posición 0 del buffer es la frecuencia de regimen, la posición 7 es la frecuencia más baja distinta de cero

Parámetros

in	*input_variable*	Tipo de variación entre las diferentes entradas. 1 para variación lineal; 2 para variación cuadrática.
in	*freq_regime*	Frecuencia de regimen ingresada por el usuario

Definición en la línea 229 del archivo sysAdmin.c.

6.39.2.10. set_system_settings()

```
void set_system_settings (
            frequency_settings_t * f_s,
            seccurity_settings_t * s_s)
```

Actualiza las variables de seguridad del sistema.

Parámetros

in	*f_s*	Puntero a la estructura con las variables de frecuencia a actualizar
in	*s_s*	Puntero a la estructura con las variables de seguridad a actualizar

Definición en la línea 251 del archivo sysAdmin.c.

6.39.2.11. update_meas()

```
system_status_e update_meas (
            uint16_t vbus_meas,
            uint16_t ibus_meas)
```

Actualiza las mediciones de tensión y corriente del bus de contínua.

En caso que superar los límites configurados, entra en SYSTEM_EMERGENCY

Parámetros

in	*vbus_meas*	• Tensión de bus de contínua leído por el ADC
in	*ibus_meas*	• Corriente de bus de contínua leído por el ADC

Devuelve

- SYSTEM_IDLE: El sistema se encuentra en estado de reposo
- SYSTEM_ACCLE_DESACCEL: El sistema se encuentra acelerando o desacelerando
- SYSTEM_REGIME: El sistema está con el motor girando a régimen
- SYSTEM_BREAKING: El sistema está frenando
- SYSTEM_EMERGENCY: El sistema entra por primera vez en estado de emergencia

Definición en la línea 197 del archivo sysAdmin.c.

6.39.3. Documentación de variables

6.39.3.1. accelerating_handle

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

Definición en la línea 26 del archivo sysAdmin.c.

6.39.3.2. desaccelerating_handle

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

Definición en la línea 27 del archivo sysAdmin.c.

6.39.3.3. frequency_table

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

Definición en la línea 25 del archivo sysAdmin.c.

6.39.3.4. system_seccurity_settings

```
seccurity_settings_t system_seccurity_settings  [static]
```

Definición en la línea 24 del archivo sysAdmin.c.

6.39.3.5. system_status

```
system_status_t system_status  [static]
```

Definición en la línea 23 del archivo sysAdmin.c.

6.39.3.6. TAG

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

Definición en la línea 22 del archivo sysAdmin.c.

6.40. sysAdmin.c

Ir a la documentación de este archivo.

```
00001
00009 #include "esp_log.h"
00010 #include "esp_err.h"
00011
00012 #include "freertos/FreeRTOS.h"
00013 #include "freertos/task.h"
00014 #include "freertos/queue.h"
00015
00016 #include "esp_system.h"
00017
00018 #include "SysAdmin.h"
00019 #include "SysControl.h"
00020 #include "../display/display.h"
00021
00022 static const char *TAG = "sysAdmin";
00023 static system_status_t system_status;
00024 static seccurity_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
00039 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             system_status.frequency = system_status.frequency_destiny;
00059         } else {
00060             system_status.frequency += system_status.acceleration;
00061         }
00062         vTaskDelay(pdMS_TO_TICKS(1000));
00063     }
00064     system_status.status = SYSTEM_REGIME;
00065     accelerating_handle = NULL;
00066     vTaskDelete(NULL);
00067 }
00068
00069 static void desaccelerating( void *pvParameters ) {
00070     // uint16_t desacceleration = *((uint16_t*)pvParameters);
00071     vTaskDelay(pdMS_TO_TICKS(200));
00072     while( system_status.frequency != system_status.frequency_destiny ) {
00073         if ( system_status.frequency - system_status.desacceleration < system_status.frequency_destiny
    ) {
00074             system_status.frequency = system_status.frequency_destiny;
00075         } else if ( system_status.frequency > system_status.desacceleration ) {
00076             system_status.frequency -= system_status.desacceleration;
00077         } else {
00078             system_status.frequency = 0;
00079         }
00080         vTaskDelay(pdMS_TO_TICKS(1000));
00081     }
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_ACCLE_DESACCEL;
00097         if ( desaccelerating_handle != NULL ) {
00098             vTaskDelete( desaccelerating_handle );
00099             desaccelerating_handle = NULL;
00100         }
00101         xTaskCreatePinnedToCore(accelerating, "accelerating", 1024, (void*)
      &system_status.acceleration, 9, &accelerating_handle, 1);
00102     }
00103     return system_status.frequency_destiny;
00104 }
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*)
      &system_status.desacceleration, 9, &desaccelerating_handle, 1);
00120     }
00121     return system_status.frequency_destiny;
00122 }
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 grnade %d",
      speed_slector);
00154         return 0;
00155     }
00156     system_status.inputs_status = speed_slector;
00157     if ( system_status.frequency > frequency_table[system_status.inputs_status] ) {
00158
00159         if ( accelerating_handle != NULL ) {
00160             vTaskDelete( accelerating_handle );
00161             accelerating_handle = NULL;
00162         }
00163         uint16_t desacceleration = get_system_acceleration();
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",
    system_status.frequency_destiny);
00169     } else if ( system_status.frequency < frequency_table[system_status.inputs_status] ) {
00170         if ( desaccelerating_handle != NULL ) {
00171             vTaskDelete( desaccelerating_handle );
00172             desaccelerating_handle = NULL;
00173         }
00174         uint16_t acceleration = get_system_acceleration();
00175         system_status.frequency_destiny = frequency_table[system_status.inputs_status];
00176         if ( accelerating_handle == NULL ) {
00177             xTaskCreatePinnedToCore(accelerating, "accelerating", 1024, (void*) &(acceleration), 9,
    &accelerating_handle, 1);
00178         }
00179         ESP_LOGI( TAG, "Cambiando la frecuencia de regimen a %d. Acelerando",
    system_status.frequency_destiny);
00180     }
00181     system_status.status = SYSTEM_ACCLE_DESACCEL;
00182     return 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_e 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_seccurity_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_seccurity_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_EXCEDED);
00219             ESP_LOGI( TAG, "Mando senal.");
00220         }
00221     } else if ( system_status.status == SYSTEM_EMERGENCY && ( system_status.emergency_signals & 0b010
    ) ) {
00222         SystemEventPost(SECURITY_OK);
00223         ESP_LOGI( TAG, "Salgo de emergencia por seguridad.");
00224     }
00225
00226     return system_status.status;
00227 }
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, seccurity_settings_t *s_s ) {
00252     system_status.acceleration = f_s->acceleration;
00253     system_status.desacceleration = f_s->desacceleration;
00254     system_seccurity_settings.vbus_min = s_s->vbus_min;
00255     system_seccurity_settings.ibus_max = s_s->ibus_max;
00256     set_frequency_table( f_s->input_variable, f_s->freq_regime );
00257 }
```

6.41. Referencia del archivo main/system/sysAdmin.h

Header con las funciones de funcionamiento del sistema.

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

Funciones

- esp_err_t get_status (system_status_t *s_e)

  Obtiene una réplica del status del sistema.
- uint16_t engine_start ()

  Función que setea la frecuencia de régimen de acuerdo a lo configurado por el usuario y ejecuta la tarea accelerating.
- uint16_t engine_stop ()

  Función que setea la frecuencia de régimen de acuerdo a lo configurado por el usuario y ejecuta la tarea desacceleratting.
- 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 contínua.
- 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, seccurity_settings_t *s_s)

  Actualiza las variables de seguridad del sistema.

6.41.1. Descripción detallada

Header con las funciones de funcionamiento del sistema.

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo sysAdmin.h.

6.41.2. Documentación de funciones

6.41.2.1. change_frequency()

```
uint16_t change_frequency (
            uint8_t speed_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.

De acuerdo a la frecuencia de régimen elegida y el tipo de variación, la frecuencia de destión tendrá diferentes valores, siempre menores a las de régimen.

Parámetros

in	*speed_slector*	Índice dentro del array con frecuencias de trabajo

Devuelve

0Hz si `speed_selector` fue mal pasada como argumento, sino la frecuencia de destino

Definición en la línea 151 del archivo sysAdmin.c.

6.41.2.2. engine_emergency_stop()

```
void engine_emergency_stop (
            uint8_t signal)
```

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

Termina las tareas accelerating y desaccelerating y guarda la señal que dispara la emergencia para contabilizar todos las señales vigentes

Parámetros

in	*signal*	Señal que genera el estado de emergencia

Definición en la línea 136 del archivo sysAdmin.c.

6.41.2.3. engine_emergency_stop_release()

```
bool engine_emergency_stop_release (
            uint8_t signal)
```

Pasa a estado SYSTEM_EMERGENCY_OK si todas las señales de emergencia son 0 para que el usuario pueda enviar la señal de stop al STM32 necesaria para poder arrancar nuevamente el motor.

Parámetros

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.

6.41.2.4. engine_start()

```
uint16_t engine_start ()
```

Función que setea la frecuencia de régimen de acuerdo a lo configurado por el usuario y ejecuta la tarea accelerating.

En caso de que desaccelerating esté corriendo, antes de ejecutar accelerating, la cierra para no generar un efecto de subida y bajada de frecuencia. El status pasará a SYSTEM_ACCLE_DESACCEL

Devuelve

Frecuencia de destino configurada

Definición en la línea 93 del archivo sysAdmin.c.

6.41.2.5. engine_stop()

```
uint16_t engine_stop ()
```

Función que setea la frecuencia de régimen de acuerdo a lo configurado por el usuario y ejecuta la tarea desaccelerating.

En caso de que accelerating esté corriendo, antes de ejecutar desaccelerating, la cierra para no generar un efecto de subida y bajada de frecuencia. El status pasará a SYSTEM_ACCLE_DESACCEL o SYSTEM_BREAKING de a cuerdo a la acción que haya ocurrido

Devuelve

Frecuencia de destino configurada

Definición en la línea 106 del archivo sysAdmin.c.

6.41.2.6. get_status()

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

Obtiene una réplica del status del sistema.

Parámetros

out	*s_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.

6.41.2.7. set_frequency_table()

```
void set_frequency_table (
            uint16_t input_variable,
            uint16_t freq_regime)
```

Carga en el vector de velocidades configurables por entradas, las diferentes frecuencias de trabajo.

La posición 0 del buffer es la frecuencia de regimen, la posición 7 es la frecuencia más baja distinta de cero

Parámetros

in	*input_variable*	Tipo de variación entre las diferentes entradas. 1 para variación lineal; 2 para variación cuadrática.
in	*freq_regime*	Frecuencia de regimen ingresada por el usuario

Definición en la línea 229 del archivo sysAdmin.c.

6.41.2.8. set_system_settings()

```
void set_system_settings (
            frequency_settings_t * f_s,
            seccurity_settings_t * s_s)
```

Actualiza las variables de seguridad del sistema.

Parámetros

in	*f_s*	Puntero a la estructura con las variables de frecuencia a actualizar
in	*s_s*	Puntero a la estructura con las variables de seguridad a actualizar

Definición en la línea 251 del archivo sysAdmin.c.

6.41.2.9. update_meas()

```
system_status_e update_meas (
            uint16_t vbus_meas,
            uint16_t ibus_meas)
```

Actualiza las mediciones de tensión y corriente del bus de contínua.

En caso que superar los límites configurados, entra en SYSTEM_EMERGENCY

Parámetros

in	*vbus_meas*	• Tensión de bus de contínua leído por el ADC
in	*ibus_meas*	• Corriente de bus de contínua leído por el ADC

Devuelve

- SYSTEM_IDLE: El sistema se encuentra en estado de reposo
- SYSTEM_ACCLE_DESACCEL: El sistema se encuentra acelerando o desacelerando
- SYSTEM_REGIME: El sistema está con el motor girando a régimen
- SYSTEM_BREAKING: El sistema está frenando
- SYSTEM_EMERGENCY: El sistema entra por primera vez en estado de emergencia

Definición en la línea 197 del archivo sysAdmin.c.

6.42. sysAdmin.h

Ir a la documentación de este archivo.

```
00001
00009 #ifndef __SYS_ADMIN_H__
00010
00011 #define __SYS_ADMIN_H__
00012
00013 #include "../LVFV_system.h"
00014
00027 esp_err_t get_status(system_status_t *s_e);
00028
00038 uint16_t engine_start();
00039
00049 uint16_t engine_stop();
00050
00060 bool engine_emergency_stop_release(uint8_t signal);
00061
00071 void engine_emergency_stop(uint8_t signal);
00072
00085 uint16_t change_frequency(uint8_t speed_slector);
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, seccurity_settings_t *s_s );
00136
00137 #endif
```

6.43. Referencia del archivo main/system/sysControl.c

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

```
#include <stdio.h>
#include <string.h>
#include "esp_log.h"
#include "esp_err.h"
```

```
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "driver/spi_master.h"
#include "driver/gpio.h"
#include "./io_control/io_control.h"
#include "../LVFV_system.h"
#include "./display/display.h"
#include "./adc/adc.h"
#include "./SysAdmin.h"
#include "./SysControl.h"
```

defines

- #define PIN_NUM_CS 12 /** @def PIN_NUM_CS @brief Chip select del SPI para comunicación con el STM32. IO12 */
- #define PIN_NUM_CLK 14 /** @def PIN_NUM_CLK @brief Clock del SPI para comunicación con el STM32. IO14 */
- #define PIN_NUM_MISO 26 /** @def PIN_NUM_MISO @brief master input slave output del SPI para comunicación con el STM32. IO26 */
- #define PIN_NUM_MOSI 25 /** @def PIN_NUM_MOSI @brief master output slave input del SPI para comunicación con el STM32. IO25 */
- #define SPI_HOST_USED SPI2_HOST /** @def SPI_HOST_USED @brief don't know */
- #define SPI_CLOCK_HZ 1*1000*1000 /** @def SPI_CLOCK_HZ @brief Velocidad de clock: 1 MHz */
- #define SPI_QUEUE_TX_DEPTH 1 /** @def SPI_QUEUE_TX_DEPTH @brief Profundidad de comandos para el puerto SPI */

Funciones

- static SPI_Response SPI_SendRequest (spi_cmd_item_t *spi_cmd_item)

  Envía el comando configurado en `spi_cmd_item->request` con el argumento `spi_cmd_item->setValue` en caso de ser necesario y obtiene los datos que responde el STM32 en `spi_cmd_item->getValue` si corresponde.
- esp_err_t SPI_Init (void)

  Inicializa el módulo SPI del ESP32.
- void SPI_communication (void *arg)

  Tarea que controla en arranque, parada y emergencia del sistema.
- esp_err_t SystemEventPost (systemSignal_e event)

  Encola comandos en la cola system_event_queue.

Variables

- static const char * TAG = "sysControl"
- QueueHandle_t system_event_queue = NULL
- static spi_device_handle_t spi_handle = NULL

6.43.1. Descripción detallada

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

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo sysControl.c.

6.43.2. Documentación de «define»

6.43.2.1. PIN_NUM_CLK

```
#define PIN_NUM_CLK 14 /** @def PIN_NUM_CLK @brief Clock del SPI para comunicación con el
STM32. IO14 */
```

Definición en la línea 31 del archivo sysControl.c.

6.43.2.2. PIN_NUM_CS

```
#define PIN_NUM_CS 12 /** @def PIN_NUM_CS @brief Chip select del SPI para comunicación con el
STM32. IO12 */
```

Definición en la línea 30 del archivo sysControl.c.

6.43.2.3. PIN_NUM_MISO

```
#define PIN_NUM_MISO 26 /** @def PIN_NUM_MISO @brief master input slave output del SPI para
comunicación con el STM32. IO26 */
```

Definición en la línea 32 del archivo sysControl.c.

6.43.2.4. PIN_NUM_MOSI

```
#define PIN_NUM_MOSI 25 /** @def PIN_NUM_MOSI @brief master output slave input del SPI para
comunicación con el STM32. IO25 */
```

Definición en la línea 33 del archivo sysControl.c.

6.43.2.5. SPI_CLOCK_HZ

```
#define SPI_CLOCK_HZ 1*1000*1000 /** @def SPI_CLOCK_HZ @brief Velocidad de clock: 1 MHz */
```

Definición en la línea 37 del archivo sysControl.c.

6.43.2.6. SPI_HOST_USED

```
#define SPI_HOST_USED SPI2_HOST /** @def SPI_HOST_USED @brief don't know */
```

Definición en la línea 36 del archivo sysControl.c.

6.43.2.7. SPI_QUEUE_TX_DEPTH

```
#define SPI_QUEUE_TX_DEPTH 1 /** @def SPI_QUEUE_TX_DEPTH @brief Profundidad de comandos para el
puerto SPI */
```

Definición en la línea 38 del archivo sysControl.c.

6.43.3. Documentación de funciones

6.43.3.1. SPI_communication()

```
void SPI_communication (
            void * arg)
```

Tarea que controla en arranque, parada y emergencia del sistema.

Espera comandos a través de la cola system_event_queue para evaluar si enviar comandos de start, stop, emergencia, cambios de velocidad, etc. Hace indirectamente el polling de las mediciones del ADC para evaluar si está en estado de emergencia o no.

Parámetros

in,out	*arg*	Sin uso

Definición en la línea 186 del archivo sysControl.c.

6.43.3.2. SPI_Init()

```
esp_err_t SPI_Init (
            void )
```

Inicializa el módulo SPI del ESP32.

Con figura los pines PIN_NUM_CS en el pin IO12, PIN_NUM_CLK en el pin IO14, PIN_NUM_MISO en el pin IO26 y PIN_NUM_MOSI en el pin IO25; además configura el clock del SPI en 1MHz y tendrá solo un comando para encolar

Devuelve

- ESP_ERR_INVALID_ARG Si la configuración es inválida. La combinación de los parámetros puede resultar inválida.
- ESP_ERR_INVALID_STATE Si el host ya se encontraba en uso, si la fuente de clock es inviable o si el SPI no fue inicializado correctamente.
- ESP_ERR_NOT_FOUND if there is no available DMA channel
- ESP_ERR_NO_MEM Si no hay memoria suficiente para inicializar el módulo
- ESP_OK si la configuración fue exitosa

Definición en la línea 155 del archivo sysControl.c.

6.43.3.3. SPI_SendRequest()

```
SPI_Response SPI_SendRequest (
            spi_cmd_item_t * spi_cmd_item)  [static]
```

Envía el comando configurado en `spi_cmd_item->request` con el argumento `spi_cmd_item->setValue` en caso de ser necesario y obtiene los datos que responde el STM32 en `spi_cmd_item->getValue` si corresponde.

El ESP32 debe enviar comandos en dos instancias. La primera envía el comando deseado por el usuario, se detiene la ejecución durante 200 mili segundos y luego se envía un nuevo comando para consultarle al STM32 si el comando enviado por el usuario fue recibido correctamente o no.

Parámetros

`in,out`	*spi_cdm_item*	Estrutura de datos con los parámetros necesarios para llevar a cabo la comunicación. Allí se almacena el comando, valores a enviar y valores recibidos.

Devuelve

- SPI_RESPONSE_ERR: Error en la transmisión del comando por problemas de inicialización del handler. El comando no sale del ESP32
- SPI_RESPONSE_ERR_CMD_UNKNOWN: El comando enviado no está dentro los posibles
- SPI_RESPONSE_ERR_MOVING: Se está intentando enviar un comando de start con el motor en movimiento
- SPI_RESPONSE_ERR_NOT_MOVING: Se está intentando enviar un comando de stop con el motor detenido
- SPI_RESPONSE_ERR_DATA_MISSING: Faltó enviar un dato dentro de la trama
- SPI_RESPONSE_ERR_DATA_INVALID: Los datos enviados como argumento dentro del comando están fuera de rango
- SPI_RESPONSE_OK: La comunicación entre ESP32 y STM32 fue exitosa

Definición en la línea 67 del archivo sysControl.c.

6.43.3.4. SystemEventPost()

```
esp_err_t SystemEventPost (
            systemSignal_e event)
```

Encola comandos en la cola system_event_queue.

Parámetros

`in`	*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.

6.43.4. Documentación de variables

6.43.4.1. spi_handle

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

Definición en la línea 44 del archivo sysControl.c.

6.43.4.2. system_event_queue

```
QueueHandle_t system_event_queue = NULL
```

Definición en la línea 42 del archivo sysControl.c.

6.43.4.3. TAG

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

Definición en la línea 40 del archivo sysControl.c.

6.44. sysControl.c

Ir a la documentación de este archivo.

```
00001
00009 #include <stdio.h>
00010 #include <string.h>
00011
00012 #include "esp_log.h"
00013 #include "esp_err.h"
00014
00015 #include "freertos/FreeRTOS.h"
00016 #include "freertos/task.h"
00017 #include "freertos/queue.h"
00018
00019 #include "driver/spi_master.h"
00020 #include "driver/gpio.h"
00021
00022 #include "./io_control/io_control.h"
00023 #include "../LVFV_system.h"
00024 #include "./display/display.h"
00025 #include "./adc/adc.h"
00026 #include "./SysAdmin.h"
00027 #include "./SysControl.h"
00028
00029 // ===== Pines =====
00030 #define PIN_NUM_CS                  12
00031 #define PIN_NUM_CLK                 14
00032 #define PIN_NUM_MISO                26
00033 #define PIN_NUM_MOSI                25
00034
00035 // ===== SPI =====
00036 #define SPI_HOST_USED               SPI2_HOST
00037 #define SPI_CLOCK_HZ                1*1000*1000
00038 #define SPI_QUEUE_TX_DEPTH          1
00039
00040 static const char *TAG = "sysControl";
00041
00042 QueueHandle_t system_event_queue = NULL;
00043
00044 static spi_device_handle_t spi_handle = NULL;                  // Handler del puerto SPI para
      comunicación con el STM32. Inicializado en esp_err_t SPI_Init(void)
00045
00065 static SPI_Response SPI_SendRequest(spi_cmd_item_t *spi_cmd_item);
00066
00067 static SPI_Response SPI_SendRequest(spi_cmd_item_t *spi_cmd_item) {
```

```
00068
00069     uint8_t tx_buffer[4];
00070     uint8_t rx_buffer[4];
00071     esp_err_t ret;
00072     int flagDevolverValor;      // Se necesita devolver el valor por parametro
00073
00074     ESP_LOGI( TAG, "[SPI Module] Request %d", spi_cmd_item->request);
00075
00076     // Chequeo de errores fuera de rango y comando desconocido
00077     if(spi_cmd_item->setValue > 512 || spi_cmd_item->setValue < 0) {
00078         return SPI_RESPONSE_ERR_DATA_INVALID;
00079     }
00080
00081     if(spi_cmd_item->request < SPI_REQUEST_START || spi_cmd_item->request > SPI_REQUEST_EMERGENCY) {
00082         ESP_LOGI( TAG, "[SPI Module] Comando desconocido\n");
00083         return SPI_RESPONSE_ERR_CMD_UNKNOWN;
00084     }
00085
00086     // Es de los comandos que deben devolver un valor por parametro?
00087     if(spi_cmd_item->request >= SPI_REQUEST_GET_FREC && spi_cmd_item->request <= SPI_REQUEST_IS_STOP)
      {
00088         flagDevolverValor = 1;
00089     }else {
00090         flagDevolverValor = 0;
00091     }
00092
00093     // Se arma la cadena a enviar
00094     tx_buffer[0] = spi_cmd_item->request;
00095
00096     if(spi_cmd_item->setValue > 255) {
00097         tx_buffer[1] = 255;
00098         tx_buffer[2] = spi_cmd_item->setValue - 255;
00099         tx_buffer[3] = ';';
00100     }else {
00101         tx_buffer[1] = spi_cmd_item->setValue;
00102         tx_buffer[2] = 0;
00103         tx_buffer[3] = ';';
00104     }
00105
00106
00107     // Limpiamos el buffer de recepcion
00108     memset(rx_buffer, 0, sizeof(rx_buffer));
00109
00110     // Siempre envio y recibo 4 bytes
00111     spi_transaction_t t = {
00112         .length = 8 * 4,
00113         .tx_buffer = tx_buffer,
00114         .rx_buffer = rx_buffer,
00115         .rxlength = 8 * 4,
00116     };
00117
00118     ret = spi_device_transmit(spi_handle, &t);
00119     if (ret != ESP_OK) {
00120         ESP_LOGI( TAG, "[ESP32] Error en SPI transmit: %d", ret);
00121         return SPI_RESPONSE_ERR;
00122     }
00123
00124     tx_buffer[0] = SPI_REQUEST_RESPONSE;
00125     tx_buffer[1] = ';';
00126
00127     t.length = 8 * 4;
00128     t.tx_buffer = tx_buffer;
00129     t.rx_buffer = rx_buffer;
00130     t.rxlength = 8 * 4;
00131
00132     vTaskDelay(pdMS_TO_TICKS(400));
00133
00134     ret = spi_device_transmit(spi_handle, &t);
00135     if (ret != ESP_OK) {
00136         ESP_LOGI( TAG, "[ESP32] Error en SPI transmit: %d", ret);
00137         return SPI_RESPONSE_ERR;
00138     }
00139
00140
00141
00142     ESP_LOGI( TAG, "rx_buffer[0]: %d", rx_buffer[0]);
00143     if(flagDevolverValor && rx_buffer[0] == SPI_RESPONSE_OK) {
00144
00145         spi_cmd_item->getValue = rx_buffer[1] + rx_buffer[2];
00146         ESP_LOGI( TAG, "RxValores: %d - %d", rx_buffer[1], rx_buffer[2]);
00147
00148     }else {
00149         spi_cmd_item->getValue = 0;
00150     }
00151
00152     return rx_buffer[0];
00153 }
```

```
00154
00155 esp_err_t SPI_Init(void) {
00156     spi_bus_config_t buscfg = {
00157         .miso_io_num = PIN_NUM_MISO,
00158         .mosi_io_num = PIN_NUM_MOSI,
00159         .sclk_io_num = PIN_NUM_CLK,
00160         .quadwp_io_num = -1,
00161         .quadhd_io_num = -1,
00162         .max_transfer_sz = 32
00163     };
00164
00165     spi_device_interface_config_t devcfg = {
00166         .clock_speed_hz = SPI_CLOCK_HZ,
00167         .mode = 0,                                // SPI mode 0
00168         .spics_io_num = PIN_NUM_CS,               // CS pin
00169         .queue_size = 1,
00170         .flags = SPI_DEVICE_NO_DUMMY              // Sin dummy bits
00171     };
00172
00173     // Inicializar bus SPI
00174     esp_err_t ret = spi_bus_initialize(VSPI_HOST, &buscfg, 1);
00175     if ( ret != ESP_OK ) {
00176         return ret;
00177     }
00178     ESP_ERROR_CHECK(ret);
00179
00180     // Añadir dispositivo
00181     ret = spi_bus_add_device(VSPI_HOST, &devcfg, &spi_handle);
00182     ESP_ERROR_CHECK(ret);
00183     return ESP_OK;
00184 }
00185
00186 void SPI_communication(void *arg) {
00187     spi_cmd_item_t item;
00188
00189     systemSignal_e new_button;
00190
00191     if ( SPI_Init() != ESP_OK) {
00192         ESP_LOGE(TAG, "SPI_Init falló; no creo SPI_communication");
00193         ESP_ERROR_CHECK(ESP_FAIL);
00194     }
00195
00196     if (system_event_queue == NULL) {
00197         system_event_queue = xQueueCreate(1, sizeof(systemSignal_e));
00198         if (system_event_queue == NULL) {
00199             ESP_LOGE(TAG, "No se pudo crear la cola de interrupciones");
00200             ESP_ERROR_CHECK(ESP_FAIL);
00201         }
00202     }
00203
00204     vTaskDelay(pdMS_TO_TICKS(4000));
00205
00206     do {
00207         item.request = SPI_REQUEST_STOP;
00208         item.setValue = 0;
00209         item.getValue = 0;
00210         vTaskDelay(pdMS_TO_TICKS(100));
00211     } while ( SPI_SendRequest(&item) != SPI_RESPONSE_ERR_NOT_MOVING );
00212
00213     for ( uint32_t i = 0;; i++ ) {
00214         if ( readADC() ) {
00215             while( xQueueReceive( system_event_queue, &new_button, pdMS_TO_TICKS(0) ) );
00216             SystemEventPost(STOP_PRESSED);
00217             break;
00218         }
00219         vTaskDelay(pdMS_TO_TICKS(100));
00220     }
00221
00222     SystemEventPost(STOP_PRESSED);
00223
00224     ESP_LOGI(TAG, "SPI_communication lista. Esperando comandos...");
00225
00226     while (1) {
00227         system_status_t s_e;
00228         get_status( &s_e );
00229         readADC();
00230         if ( xQueueReceive( system_event_queue, &new_button, pdMS_TO_TICKS(20) ) ) {
00231             switch ( new_button ) {
00232                 case 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                         RelayEvantPost( 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_EXCEDED:
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                         RelayEvantPost( 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                         RelayEvantPost( 1 );
00266                     } else {
00267                         ESP_LOGI(TAG, "El sistema ya se encontraba en emergencia");
00268                     }
00269                     engine_emergency_stop(0b100);
00270                     break;
00271                 case EMERGENCI_STOP_RELEASED:
00272                     ESP_LOGI(TAG, "Botón de EMERGENCIA liberado");
00273                     engine_emergency_stop_release(0b001);
00274                     break;
00275                 case SECURITY_OK:
00276                     ESP_LOGI(TAG, "Tensión y corriente normalizadas");
00277                     engine_emergency_stop_release(0b010);
00278                     break;
00279                 case TERMO_SW_RELEASED:
00280                     ESP_LOGI(TAG, "Termoswitch desactivado");
00281                     engine_emergency_stop_release(0b100);
00282                     break;
00283                 case STOP_PRESSED:
00284                     if ( s_e.status == SYSTEM_EMERGENCY ) {
00285                         ESP_LOGI(TAG,"Primero salir de estado de emergencia");
00286                     } else if ( s_e.status == SYSTEM_ACCLE_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                         RelayEvantPost( 0 );
00301                     }
00302                     break;
00303                 case START_PRESSED:
00304                     if ( s_e.status == SYSTEM_IDLE ) {
00305                         ESP_LOGI(TAG, "Botón de Inicio presionado");
00306                         SPI_Response SPI_commando_response;
00307 
00308                         item.request = SPI_REQUEST_SET_FREC;
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_ACEL;
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_ACCLE_DESACCEL ) {
00364                         ESP_LOGI(TAG, "Cambio de velocidad:%d", new_button - SPEED_SELECTOR_0);
00365                         uint8_t old_input_status = s_e.inputs_status;
00366                         item.request = SPI_REQUEST_SET_FREC;
00367                         item.setValue = change_frequency( new_button - SPEED_SELECTOR_0 );
00368                         item.getValue = 0;
00369                         if ( SPI_SendRequest(&item) != SPI_RESPONSE_OK ) {
00370                             change_frequency( old_input_status );
00371                             ESP_LOGE(TAG, "El STM32 no respondio correctamente");
00372                             xQueueSend(system_event_queue, &new_button, pdMS_TO_TICKS(1000));
00373                         }
00374                     } else {
00375                         ESP_LOGI(TAG, "No puede cambiar la velocidad, status =%d", s_e.status);
00376                     }
00377                     break;
00378                 default:
00379                     break;
00380             }
00381         }
00382     }
00383 }
00384 
00385 esp_err_t SystemEventPost(systemSignal_e event) {
00386     if (system_event_queue == NULL) {
00387         return ESP_FAIL;
00388     }
00389     return xQueueSend(system_event_queue, &event, 0);
00390 }
```

6.45. Referencia del archivo main/system/sysControl.h

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

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

Estructuras de datos

- struct spi_cmd_item_t

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

Enumeraciones

- enum SPI_Request {
  SPI_REQUEST_START = 10 , SPI_REQUEST_STOP , SPI_REQUEST_SET_FREC , SPI_REQUEST_SET_ACEL
  ,
  SPI_REQUEST_SET_DESACEL , SPI_REQUEST_SET_DIR , SPI_REQUEST_GET_FREC , SPI_REQUEST_GET_ACEL
  ,
  SPI_REQUEST_GET_DESACEL , SPI_REQUEST_GET_DIR , SPI_REQUEST_IS_STOP , SPI_REQUEST_EMERGENCY
  ,
  SPI_REQUEST_RESPONSE = 0x50 }

  Posibles comandos que puede enviar el STM32.
- enum SPI_Response {
  SPI_RESPONSE_ERR = 0xA0 , SPI_RESPONSE_ERR_CMD_UNKNOWN , SPI_RESPONSE_ERR_NO_COMMAND
  , SPI_RESPONSE_ERR_MOVING ,
  SPI_RESPONSE_ERR_NOT_MOVING , SPI_RESPONSE_ERR_DATA_MISSING , SPI_RESPONSE_ERR_DATA_INVALID
  , SPI_RESPONSE_ERR_DATA_OUT_RANGE ,
  SPI_RESPONSE_OK = 0xFF }

  Posibles respuesta que puede enviar el STM32 en consecuencia por los request enviados.

Funciones

- esp_err_t SPI_Init (void)

  Inicializa el módulo SPI del ESP32.
- void SPI_communication (void ∗arg)

  Tarea que controla en arranque, parada y emergencia del sistema.
- esp_err_t SystemEventPost (systemSignal_e event)

  Encola comandos en la cola system_event_queue.

6.45.1. Descripción detallada

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

Autor

Andrenacci - Carra

Versión

2.0

Fecha

2025-11-06

Definición en el archivo sysControl.h.

6.45.2. Documentación de enumeraciones

6.45.2.1. SPI_Request

```
enum SPI_Request
```

Posibles comandos que puede enviar el STM32.

Valores de enumeraciones

SPI_REQUEST_START	
SPI_REQUEST_STOP	
SPI_REQUEST_SET_FREC	
SPI_REQUEST_SET_ACEL	
SPI_REQUEST_SET_DESACEL	
SPI_REQUEST_SET_DIR	
SPI_REQUEST_GET_FREC	
SPI_REQUEST_GET_ACEL	
SPI_REQUEST_GET_DESACEL	
SPI_REQUEST_GET_DIR	
SPI_REQUEST_IS_STOP	
SPI_REQUEST_EMERGENCY	
SPI_REQUEST_RESPONSE	

Definición en la línea 19 del archivo sysControl.h.

6.45.2.2. SPI_Response

```
enum SPI_Response
```

Posibles respuesta que puede enviar el STM32 en consecuencia por los request enviados.

Valores de enumeraciones

SPI_RESPONSE_ERR	
SPI_RESPONSE_ERR_CMD_UNKNOWN	
SPI_RESPONSE_ERR_NO_COMMAND	
SPI_RESPONSE_ERR_MOVING	
SPI_RESPONSE_ERR_NOT_MOVING	
SPI_RESPONSE_ERR_DATA_MISSING	
SPI_RESPONSE_ERR_DATA_INVALID	
SPI_RESPONSE_ERR_DATA_OUT_RANGE	
SPI_RESPONSE_OK	

Definición en la línea 40 del archivo sysControl.h.

6.45.3. Documentación de funciones

6.45.3.1. SPI_communication()

```
void SPI_communication (
            void * arg)
```

Tarea que controla en arranque, parada y emergencia del sistema.

Espera comandos a través de la cola system_event_queue para evaluar si enviar comandos de start, stop, emergencia, cambios de velocidad, etc. Hace indirectamente el polling de las mediciones del ADC para evaluar si está en estado de emergencia o no.

Parámetros

in,out	*arg*	Sin uso

Definición en la línea 186 del archivo sysControl.c.

6.45.3.2. SPI_Init()

```
esp_err_t SPI_Init (
            void )
```

Inicializa el módulo SPI del ESP32.

Con figura los pines PIN_NUM_CS en el pin IO12, PIN_NUM_CLK en el pin IO14, PIN_NUM_MISO en el pin IO26 y PIN_NUM_MOSI en el pin IO25; además configura el clock del SPI en 1MHz y tendrá solo un comando para encolar

Devuelve

- ESP_ERR_INVALID_ARG Si la configuración es inválida. La combinación de los parámetros puede resultar inválida.
- ESP_ERR_INVALID_STATE Si el host ya se encontraba en uso, si la fuente de clock es inviable o si el SPI no fue inicializado correctamente.
- ESP_ERR_NOT_FOUND if there is no available DMA channel
- ESP_ERR_NO_MEM Si no hay memoria suficiente para inicializar el módulo
- ESP_OK si la configuración fue exitosa

Definición en la línea 155 del archivo sysControl.c.

6.45.3.3. SystemEventPost()

```
esp_err_t SystemEventPost (
            systemSignal_e event)
```

Encola comandos en la cola system_event_queue.

Parámetros

in	*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.

6.46. sysControl.h

Ir a la documentación de este archivo.

```
00001
00009 #ifndef SPI_MODULE_H
00010 #define SPI_MODULE_H
00011
00012 #include "../LVFV_system.h"
00013
00019 typedef enum {
00020     SPI_REQUEST_START = 10,                    // 10 - Comando de arranque de motor
00021     SPI_REQUEST_STOP,                          // 11 - Comando de parada de motor
00022     SPI_REQUEST_SET_FREC,                      // 12 - Comando para configurar la frecuencia de
      régimen
00023     SPI_REQUEST_SET_ACEL,                      // 13 - Comando para setear la aceleración hasta la
      frecuencia de régimen
00024     SPI_REQUEST_SET_DESACEL,                   // 14 - Comando para setear la desaceleración a 0Hz o
      frecuencia de régimen
00025     SPI_REQUEST_SET_DIR,                       // 15 - Comando para setear la dirección - TODO
00026     SPI_REQUEST_GET_FREC,                      // 16 - Comando de consulta de frecuencia de régimen
00027     SPI_REQUEST_GET_ACEL,                      // 17 - Comando de consulta de aceleración
00028     SPI_REQUEST_GET_DESACEL,                   // 18 - Comando de consulta de desaceleración
00029     SPI_REQUEST_GET_DIR,                       // 19 - Comando de consulta de dirección de giro
00030     SPI_REQUEST_IS_STOP,                       // 20 - Comando de consulta para saber si el motor
      está parado o en movimiento
00031     SPI_REQUEST_EMERGENCY,                     // 21 - Comando para entrar en estado de emergencia -
      deja de conmutar la salida
00032     SPI_REQUEST_RESPONSE = 0x50                // 80 - Comando para pedirle al STM32 la respuesta al
      comando enviado
00033 } SPI_Request;
00034
00040 typedef enum {
00041     SPI_RESPONSE_ERR = 0xA0,                   // 160 - Error en la transmisión del comando por
      problemas de inicialización del handler. El comando no sale del ESP32
00042     SPI_RESPONSE_ERR_CMD_UNKNOWN,              // 161 - El comando enviado no está dentro los
      posibles
00043     SPI_RESPONSE_ERR_NO_COMMAND,               // 162 - Llego mensaje pero sin comando
00044     SPI_RESPONSE_ERR_MOVING,                   // 163 - Se está intentando enviar un comando de start
      con el motor en movimiento
00045     SPI_RESPONSE_ERR_NOT_MOVING,               // 164 - Se está intentando enviar un comando de stop
      con el motor detenido
00046     SPI_RESPONSE_ERR_DATA_MISSING,             // 165 - Faltó enviar un dato dentro de la trama
00047     SPI_RESPONSE_ERR_DATA_INVALID,             // 166 - Los datos enviados como argumento dentro del
      comando están fuera de rango
00048     SPI_RESPONSE_ERR_DATA_OUT_RANGE,           // 167 - Comando con datos fuera de rango permitido
00049     SPI_RESPONSE_OK = 0xFF,                    // 255 - La comunicación entre ESP32 y STM32 fue
      exitosa
00050 } SPI_Response;
00051
00059 typedef struct {
00060     SPI_Request request;                       // Comando a enviar al STM32
00061     int getValue;                              // Variable devuelta por el STM32 en caso de ser un
      comando get
00062     int setValue;                              // Dato enviado con los comandos set
00063 } spi_cmd_item_t;
00064
00079 esp_err_t SPI_Init(void);
00080
00091 void SPI_communication(void *arg);
00092
00105 esp_err_t SystemEventPost(systemSignal_e event);
00106
00107 #endif // SPI_MODULE_H
```

6.47. Referencia del archivo main/wifi/form.c

Variables

- const char * HTML_FORM

6.47.1. Documentación de variables

6.47.1.1. HTML_FORM

```
const char* HTML_FORM
```

Definición en la línea 1 del archivo form.c.

6.48. form.c

Ir a la documentación de este archivo.

```
00001 const char *HTML_FORM =
00002 "<!DOCTYPE html>"
00003 "<html lang=\"es\">"
00004 "<head>"
00005 "  <meta charset=\"UTF-8\">"
00006 "  <meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">"
00007 "  <title>Configuración del Variador</title>"
00008 "  <style>"
00009 "    body{font-family:system-ui,-apple-system,Segoe
      UI,Roboto,sans-serif;background:#f8f9fa;margin:0;}"
00010 "    .container{max-width:960px;margin:0 auto;padding:1rem;}"
00011 "    h1{font-size:1.8rem;margin-bottom:1.5rem;}"
00012 "    .row{display:flex;flex-wrap:wrap;margin:-0.5rem;}"
00013 "    .col-md-4{flex:0 0 100%;max-width:100%;padding:0.5rem;}"
00014 "    @media (min-width:768px){.col-md-4{flex:0 0 33.3333%;max-width:33.3333%;}}"
00015 "    .form-label{display:block;font-weight:600;margin-bottom:0.25rem;font-size:0.95rem;}"
00016 "    .form-control,.form-select{display:block;width:100%;padding:0.375rem 0.75rem;"
00017 "      font-size:1rem;line-height:1.5;color:#212529;background-color:#fff;"
00018 "      border:1px solid #ced4da;border-radius:0.25rem;box-sizing:border-box;}"
00019 "    .form-control:focus,.form-select:focus{outline:2px solid #0d6efd33;outline-offset:1px;}"
00020 "    .btn{display:inline-block;font-weight:400;line-height:1.5;text-align:center;"
00021 "      text-decoration:none;border:1px solid transparent;padding:0.375rem 0.75rem;"
00022 "      font-size:1rem;border-radius:0.25rem;cursor:pointer;margin:1.5rem;}"
00023 "    .btn-primary{color:#fff;background-color:#0d6efd;border-color:#0d6efd;width:100%;}"
00024 "    .btn-primary:hover{background-color:#0b5ed7;border-color:#0a58ca;}"
00025 "    .btn-success{color:#fff;background-color:#198754;border-color:#198754;}"
00026 "    .btn-danger{color:#fff;background-color:#dc3545;border-color:#dc3545;}"
00027 "    .btn-success:hover{background-color:#157347;border-color:#146c43;}"
00028 "    .btn-danger:hover{background-color:#bb2d3b;border-color:#b02a37;}"
00029 "    .mb-4{margin-bottom:1.5rem;}"
00030 "    .mt-4{margin-top:1.5rem;}"
00031 "    .py-4{padding-top:1.5rem;padding-bottom:1.5rem;}"
00032 "    .me-2{margin-right:0.5rem;}"
00033 "  </style>"
00034 "</head>"
00035 "<body class=\"bg-light\">"
00036 "<div class=\"container py-4\">"
00037 "  <h1 class=\"mb-4\">Configuración del Variador</h1>"
00038 "  <form id=\"config-form\" method=\"POST\" action=\"/save\" class=\"row g-3\">"
00039
00040 "    <div class=\"col-md-4\">"
00041 "      <label class=\"form-label\">Frecuencia de operación (Hz)</label>"
00042 "      <input type=\"number\" class=\"form-control\" name=\"frequency\" min=\"1\" max=\"150\"
      required>"
00043 "    </div>"
00044
00045 "    <div class=\"col-md-4\">"
00046 "      <label class=\"form-label\">Aceleración (Hz/s)</label>"
00047 "      <input type=\"number\" class=\"form-control\" name=\"accel\" min=\"1\" max=\"50\" required>"
00048 "    </div>"
00049
00050 "    <div class=\"col-md-4\">"
00051 "      <label class=\"form-label\">Desaceleración (Hz/s)</label>"
00052 "      <input type=\"number\" class=\"form-control\" name=\"decel\" min=\"1\" max=\"50\" required>"
00053 "    </div>"
00054
00055 "    <div class=\"col-md-4\">"
00056 "      <label class=\"form-label\">Variación de las entradas</label>"
00057 "      <select class=\"form-select\" name=\"variation\" required>"
00058 "        <option value=\"1\">Lineal</option>"
00059 "        <option value=\"2\">Cuadrática</option>"
00060 "      </select>"
00061 "    </div>"
00062
00063 "    <div class=\"col-md-4\">"
```

```
00064 "      <label class=\"form-label\">Corriente máx. bus DC (mA)</label>"
00065 "      <input type=\"number\" class=\"form-control\" name=\"imax\" min=\"500\" max=\"2000\" required>"
00066 "    </div>"
00067
00068 "    <div class=\"col-md-4\">"
00069 "      <label class=\"form-label\">Tensión máx. bus DC (V)</label>"
00070 "      <input type=\"number\" class=\"form-control\" name=\"vmin\" min=\"250\" max=\"350\" required>"
00071 "    </div>"
00072
00073 "    <div class=\"col-md-4\">"
00074 "      <label class=\"form-label\">Hora del sistema (HH:mm:ss)</label>"
00075 "      <input type=\"text\" class=\"form-control\" name=\"sys_time\" placeholder=\"12:34:56\"
      required>"
00076 "    </div>"
00077
00078 "    <div class=\"col-md-4\">"
00079 "      <label class=\"form-label\">Horario de arranque (HH:mm)</label>"
00080 "      <input type=\"text\" class=\"form-control\" name=\"start_time\" placeholder=\"08:00\"
      required>"
00081 "    </div>"
00082
00083 "    <div class=\"col-md-4\">"
00084 "      <label class=\"form-label\">Horario de parada (HH:mm)</label>"
00085 "      <input type=\"text\" class=\"form-control\" name=\"stop_time\" placeholder=\"18:00\" required>"
00086 "    </div>"
00087
00088 "    <div class=\"col-12 mt-4\">"
00089 "      <button type=\"submit\" class=\"btn btn-primary me-2\" name=\"cmd\" value=\"save\">Guardar
      configuración</button>"
00090 "      <button type=\"submit\" class=\"btn btn-success me-2\" name=\"cmd\" value=\"start\">Arrancar
      motor</button>"
00091 "      <button type=\"submit\" class=\"btn btn-danger\" name=\"cmd\" value=\"stop\">Parar
      motor</button>"
00092 "    </div>"
00093 "  </form>"
00094 "</div>"
00095
00096 "<script>"
00097 "  const formId = 'config-form';"
00098 "  const storageKey = 'variador_form_state';"
00099 "  function saveFormState() {"
00100 "    const form = document.getElementById(formId);"
00101 "    if (!form) return;"
00102 "    const data = {};"
00103 "    Array.from(form.elements).forEach(el => {"
00104 "      if (!el.name) return;"
00105 "      if (el.type === 'checkbox' || el.type === 'radio') {"
00106 "        data[el.name] = el.checked;"
00107 "      } else {"
00108 "        data[el.name] = el.value;"
00109 "      }"
00110 "    });"
00111 "    try {"
00112 "      localStorage.setItem(storageKey, JSON.stringify(data));"
00113 "    } catch (e) {"
00114 "      console.log('No se pudo guardar estado:', e);"
00115 "    }"
00116 "  }"
00117 "  function loadFormState() {"
00118 "    const form = document.getElementById(formId);"
00119 "    if (!form) return;"
00120 "    const raw = localStorage.getItem(storageKey);"
00121 "    if (!raw) return;"
00122 "    try {"
00123 "      const data = JSON.parse(raw);"
00124 "      Array.from(form.elements).forEach(el => {"
00125 "        if (!el.name || !(el.name in data)) return;"
00126 "        if (el.name === 'cmd') return;"
00127 "        if (el.type === 'checkbox' || el.type === 'radio') {"
00128 "          el.checked = !!data[el.name];"
00129 "        } else {"
00130 "          el.value = data[el.name];"
00131 "        }"
00132 "      });"
00133 "    } catch (e) {"
00134 "      console.log('No se pudo leer estado:', e);"
00135 "    }"
00136 "  }"
00137 "  window.addEventListener('load', () => {"
00138 "    loadFormState();"
00139 "    const form = document.getElementById(formId);"
00140 "    if (!form) return;"
00141 "    form.addEventListener('input', saveFormState);"
00142 "  });"
00143 "</script>"
00144
00145 "</body>"
```

```
00146 "</html>";
```

6.49. Referencia del archivo main/wifi/form.h

Variables

- const char * HTML_FORM

6.49.1. Documentación de variables

6.49.1.1. HTML_FORM

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

Definición en la línea 1 del archivo form.c.

6.50. form.h

Ir a la documentación de este archivo.

```
00001
00002 #ifndef __FORM_H__
00003 #define __FORM_H__
00004
00005 extern const char *HTML_FORM; // definido antes
00006
00007 #endif
```

6.51. Referencia del archivo main/wifi/wifi.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_event.h"
#include "esp_log.h"
#include "nvs_flash.h"
#include "esp_netif.h"
#include "../LVFV_system.h"
#include "../system/sysControl.h"
#include "../display/display.h"
#include "./wifi.h"
#include "form.h"
```

Funciones

- static int hex2int (char c)

  Convierte un dígito hex a entero (0..15).
- static void url_decode (char ∗dst, const char ∗src)

  Decodifica URL (reemplaza '+' por espacio y HH por byte).
- static bool get_param_value (const char ∗body, const char ∗key, char ∗dest, size_t dest_len)

  Obtiene el valor de un parámetro key=val del body x-www-form-urlencoded.
- static esp_err_t root_get_handler (httpd_req_t ∗req)

  Handler GET "/" – devuelve el formulario HTML.
- static esp_err_t save_post_handler (httpd_req_t ∗req)

  Handler POST "/save" – parsea el formulario y actúa.
- httpd_handle_t start_webserver (void)

  Arranca el servidor HTTP y registra endpoints.
- void wifi_init_softap (void)

  Configura ESP-NETIF + Wi-Fi en modo AP y arranca SSID/clave. (Canal 1, máx 1 cliente, WPA/WPA2).

Variables

- static const char ∗ TAG = "WEB_CFG"

6.51.1. Documentación de funciones

6.51.1.1. get_param_value()

```
bool get_param_value (
            const char * body,
            const char * key,
            char * dest,
            size_t dest_len)  [static]
```

Obtiene el valor de un parámetro key=val del body x-www-form-urlencoded.

Parámetros

body	cuerpo completo del POST (NUL-terminated).
key	nombre del parámetro (sin '=').
dest	buffer destino para el valor decodificado.
dest_len	tamaño de dest, incluye NUL.

Devuelve

true si se encontró la clave; false si no.

Nota

Devuelve el valor decodificado (URL-decoding).

Definición en la línea 115 del archivo wifi.c.

6.51.1.2. hex2int()

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

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

Definición en la línea 91 del archivo wifi.c.

6.51.1.3. root_get_handler()

```
esp_err_t root_get_handler (
            httpd_req_t * req)  [static]
```

Handler GET "/" – devuelve el formulario HTML.

Definición en la línea 152 del archivo wifi.c.

6.51.1.4. save_post_handler()

```
esp_err_t save_post_handler (
            httpd_req_t * req)  [static]
```

Handler POST "/save" – parsea el formulario y actúa.

Flujo: 1) Lee el body (x-www-form-urlencoded) a 'body'. 2) Extrae parámetros: frequency, accel, decel, imax, vmin, variation_str (linear/cuadratica), sys_time, start_time, stop_time. 3) Llena estructuras frequency_edit, seccurity_edit, time_edit. 4) Si cmd=save → system_variables_save(...). Si cmd=start → SystemEventPost(START_PRESSED). Si cmd=stop → SystemEventPost(STOP_PRESSED). 5) Responde nuevamente con el formulario.

Atención

> Los campos time_* se cargan como punteros a variables locales (stack). Si system_variables_save() no copia por valor de inmediato y almacena los punteros, quedarían colgando. Ver “Posibles issues” más abajo.

Definición en la línea 158 del archivo wifi.c.

6.51.1.5. start_webserver()

```
httpd_handle_t start_webserver (
            void )
```

Arranca el servidor HTTP y registra endpoints.

- GET "/" → formulario
- POST "/save" → procesa y responde

Definición en la línea 282 del archivo wifi.c.

6.51.1.6. url_decode()

```
void url_decode (
            char * dst,
            const char * src)  [static]
```

Decodifica URL (reemplaza '+' por espacio y HH por byte).

Parámetros

dst	buffer destino (puede ser mismo que src si hay espacio).
src	cadena codificada.

Definición en la línea 98 del archivo wifi.c.

6.51.1.7. wifi_init_softap()

```
void wifi_init_softap (
            void )
```

Configura ESP-NETIF + Wi-Fi en modo AP y arranca SSID/clave. (Canal 1, máx 1 cliente, WPA/WPA2).

Definición en la línea 307 del archivo wifi.c.

6.51.2. Documentación de variables

6.51.2.1. TAG

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

Definición en la línea 31 del archivo wifi.c.

6.52. wifi.c

Ir a la documentación de este archivo.

```
00001
00009 #include <stdio.h>
00010 #include <string.h>
00011 #include <stdlib.h>
00012 #include <ctype.h>
00013
00014 #include "freertos/FreeRTOS.h"
00015 #include "freertos/task.h"
00016
00017 #include "esp_event.h"
00018 #include "esp_log.h"
00019 #include "nvs_flash.h"
00020 // #include "mdns.h"
00021
00022 #include "esp_netif.h"
00023
00024 // Tus headers con las structs y funciones:
00025 #include "../LVFV_system.h"
00026 #include "../system/sysControl.h"
00027 #include "../display/display.h"
00028 #include "./wifi.h"
00029 #include "form.h"
00030
00031 static const char *TAG = "WEB_CFG";
00032
00033 /* ---------- Helpers para parsear POST x-www-form-urlencoded ---------- */
00034
00040 static int hex2int(char c);
00041
00049 static void url_decode(char *dst, const char *src);
00050
00063 static bool get_param_value(const char *body, const char *key, char *dest, size_t dest_len);
00064
00070 static esp_err_t root_get_handler(httpd_req_t *req);
00071
00089 static esp_err_t save_post_handler(httpd_req_t *req);
```

```
00090
00091 static int hex2int(char c) {
00092     if (c >= '0' && c <= '9') return c - '0';
00093     if (c >= 'a' && c <= 'f') return 10 + (c - 'a');
00094     if (c >= 'A' && c <= 'F') return 10 + (c - 'A');
00095     return 0;
00096 }
00097
00098 static void url_decode(char *dst, const char *src) {
00099     while (*src) {
00100         if (*src == '+') {
00101             *dst++ = ' ';
00102             src++;
00103         } else if (*src == '%' && isxdigit((unsigned char)src[1]) && isxdigit((unsigned char)src[2]))
      {
00104             int hi = hex2int(src[1]);
00105             int lo = hex2int(src[2]);
00106             *dst++ = (char)((hi << 4) | lo);
00107             src += 3;
00108         } else {
00109             *dst++ = *src++;
00110         }
00111     }
00112     *dst = '\0';
00113 }
00114
00115 static bool get_param_value(const char *body, const char *key, char *dest, size_t dest_len) {
00116     size_t key_len = strlen(key);
00117     const char *p = body;
00118
00119     while (p && *p) {
00120         const char *eq = strchr(p, '=');
00121         if (!eq) break;
00122
00123         // Nombre de parámetro
00124         size_t this_key_len = (size_t)(eq - p);
00125
00126         if (this_key_len == key_len && strncmp(p, key, key_len) == 0) {
00127             // Encontramos la clave. Buscar fin del valor (& o final de cadena)
00128             const char *val_start = eq + 1;
00129             const char *amp = strchr(val_start, '&');
00130             size_t val_len = amp ? (size_t)(amp - val_start) : strlen(val_start);
00131             if (val_len >= dest_len) val_len = dest_len - 1;
00132
00133             char encoded[256];
00134             if (val_len >= sizeof(encoded)) val_len = sizeof(encoded) - 1;
00135
00136             memcpy(encoded, val_start, val_len);
00137             encoded[val_len] = '\0';
00138
00139             url_decode(dest, encoded);
00140             return true;
00141         }
00142
00143         // Ir al siguiente par key=value
00144         const char *amp = strchr(eq + 1, '&');
00145         if (!amp) break;
00146         p = amp + 1;
00147     }
00148
00149     return false;
00150 }
00151
00152 static esp_err_t root_get_handler(httpd_req_t *req) {
00153     httpd_resp_set_type(req, "text/html");
00154     httpd_resp_send(req, HTML_FORM, HTTPD_RESP_USE_STRLEN);
00155     return ESP_OK;
00156 }
00157
00158 static esp_err_t save_post_handler(httpd_req_t *req) {
00159     char body[512];
00160     char cmd[16] = {0};
00161
00162     time_settings_SH1106_t time_edit;
00163     seccurity_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 
00240     struct tm time_system;
00241     time_system.tm_hour = sys_h;
00242     time_system.tm_min = sys_m;
00243     time_system.tm_sec = sys_s;
00244     struct tm time_start;
00245     time_start.tm_hour = start_h;
00246     time_start.tm_min = start_m;
00247     struct tm time_stop;
00248     time_stop.tm_hour = stop_h;
00249     time_stop.tm_min = stop_m;
00250 
00251     frequency_edit.frequency_settings.freq_regime = frequency;
00252     frequency_edit.frequency_settings.acceleration = accel;
00253     frequency_edit.frequency_settings.desacceleration = decel;
00254     frequency_edit.frequency_settings.input_variable = variation;
00255 
00256     seccurity_edit.seccurity_settings.ibus_max = imax;
00257     seccurity_edit.seccurity_settings.vbus_min = vmin;
00258 
00259     time_edit.time_settings.time_system = &time_system;
00260     time_edit.time_settings.time_start = &time_start;
00261     time_edit.time_settings.time_stop = &time_stop;
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, &seccurity_edit);
00267         } else if (strcmp(cmd, "start") == 0) {
00268             SystemEventPost(START_PRESSED);
00269         } else if (strcmp(cmd, "stop") == 0) {
00270             SystemEventPost(STOP_PRESSED);
00271         }
00272     }
00273
00274     /* -------- Respuesta al navegador -------- */
00275
00276     httpd_resp_set_type(req, "text/html");
00277     httpd_resp_send(req, HTML_FORM, HTTPD_RESP_USE_STRLEN);
00278
00279     return ESP_OK;
00280 }
00281
00282 httpd_handle_t start_webserver(void) {
00283     httpd_config_t config = HTTPD_DEFAULT_CONFIG();
00284     config.server_port = 80;
00285
00286     httpd_handle_t server = NULL;
00287     if (httpd_start(&server, &config) == ESP_OK) {
00288         httpd_uri_t root_uri = {
00289             .uri      = "/",
00290             .method   = HTTP_GET,
00291             .handler  = root_get_handler,
00292             .user_ctx = NULL
00293         };
00294         httpd_register_uri_handler(server, &root_uri);
00295
00296         httpd_uri_t save_uri = {
00297             .uri      = "/save",
00298             .method   = HTTP_POST,
00299             .handler  = save_post_handler,
00300             .user_ctx = NULL
00301         };
00302         httpd_register_uri_handler(server, &save_uri);
00303     }
00304     return server;
00305 }
00306
00307 void wifi_init_softap(void) {
00308     ESP_ERROR_CHECK(esp_netif_init());
00309     ESP_ERROR_CHECK(esp_event_loop_create_default());
00310     esp_netif_create_default_wifi_ap();
00311
00312     wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
00313     ESP_ERROR_CHECK(esp_wifi_init(&cfg));
00314
00315     wifi_config_t wifi_config = {
00316         .ap = {
00317             .ssid = "LVFV_2025",
00318             .ssid_len = 0,
00319             .channel = 1,
00320             .password = "LVFV_2025",
00321             .max_connection = 1,
00322             .authmode = WIFI_AUTH_WPA_WPA2_PSK
00323         },
00324     };
00325     if (strlen((char *)wifi_config.ap.password) == 0) {
00326         wifi_config.ap.authmode = WIFI_AUTH_OPEN;
00327     }
00328
00329     ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_AP));
00330     ESP_ERROR_CHECK(esp_wifi_set_config(WIFI_IF_AP, &wifi_config));
00331     ESP_ERROR_CHECK(esp_wifi_start());
00332
00333     ESP_LOGI(TAG, "AP iniciado. SSID: %s, pass: %s", wifi_config.ap.ssid, wifi_config.ap.password);
00334 }
```

6.53. Referencia del archivo main/wifi/wifi.h

```
#include "esp_wifi.h"
#include "esp_http_server.h"
```

Funciones

- httpd_handle_t start_webserver (void)

  Arranca el servidor HTTP y registra endpoints.
- void wifi_init_softap (void)

  Configura ESP-NETIF + Wi-Fi en modo AP y arranca SSID/clave. (Canal 1, máx 1 cliente, WPA/WPA2).

6.53.1. Documentación de funciones

6.53.1.1. start_webserver()

```
httpd_handle_t start_webserver (
            void )
```

Arranca el servidor HTTP y registra endpoints.

- GET "/" → formulario
- POST "/save" → procesa y responde

Definición en la línea 282 del archivo wifi.c.

6.53.1.2. wifi_init_softap()

```
void wifi_init_softap (
            void )
```

Configura ESP-NETIF + Wi-Fi en modo AP y arranca SSID/clave. (Canal 1, máx 1 cliente, WPA/WPA2).

Definición en la línea 307 del archivo wifi.c.

6.54. wifi.h

Ir a la documentación de este archivo.

```
00001
00002 #ifndef __MAIN_WIFI_WIFI_H__
00003 #define __MAIN_WIFI_WIFI_H__
00004
00005 #include "esp_wifi.h"
00006 #include "esp_http_server.h"
00007
00015 httpd_handle_t start_webserver(void);
00016
00022 void wifi_init_softap(void);
00023 // void start_mdns(void);
00024
00025 #endif
```

Índice alfabético