Indoor Module Datasheet

Author: Łukasz Wolf

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

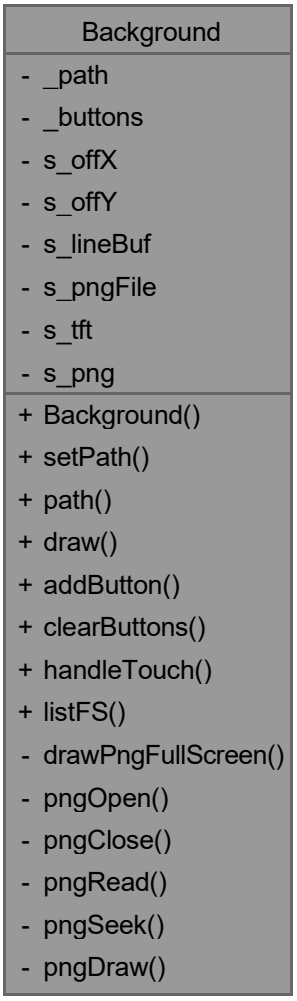
Class Documentation

3.1 Background Class Reference

Handles loading background PNGs and managing associated touch buttons.

`#include <Background.h>`

Collaboration diagram for Background:



Public Member Functions

- Background (const String &path="")

  Constructs a new Background object.
- void setPath (const String &path)

  Sets the file path for the background image.
- String path () const

  Gets the current file path.
- bool draw (TFT_eSPI &tft, PNG &png, bool center=true)

  Draws the background image to the TFT screen.
- void addButton (const Button &btn)

  Adds an interactive button region to this background.
- void clearButtons ()

  Removes all registered buttons from this background.

- bool handleTouch (int16_t touchX, int16_t touchY, bool isPressedNow)

  Processes touch input for all registered buttons.

Static Public Member Functions

- static void listFS (const char *dir)

  Lists files in a directory to Serial for debugging purposes.

Private Member Functions

- bool drawPngFullScreen (const char *path, bool center)

  Internal helper to setup and execute the PNG drawing process.

Static Private Member Functions

- static void * pngOpen (const char *filename, int32_t *pFileSize)
- static void pngClose (void *handle)
- static int32_t pngRead (PNGFILE *pFile, uint8_t *pBuff, int32_t iLen)
- static int32_t pngSeek (PNGFILE *pFile, int32_t iPosition)
- static int pngDraw (PNGDRAW *p)

Private Attributes

- String _path

  Path to the background image file.
- std::vector< Button > _buttons

  List of interactive buttons on this background.

Static Private Attributes

- static int s_offX = 0

  X offset for centering the image.
- static int s_offY = 0

  Y offset for centering the image.
- static uint16_t s_lineBuf [480]

  Buffer for one line of decoded pixels.
- static File s_pngFile

  Handle to the open PNG file.
- static TFT_eSPI * s_tft = nullptr

  Pointer to the display driver used during draw.
- static PNG * s_png = nullptr

  Pointer to the PNG decoder instance.

3.1.1 Detailed Description

Handles loading background PNGs and managing associated touch buttons.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Background()

```
Background::Background (
            const String & path = "")
```

Constructs a new Background object.

Parameters

	path	Path to the background PNG file.

3.1.3 Member Function Documentation

3.1.3.1 addButton()

```
void Background::addButton (
            const Button & btn)
```

Adds an interactive button region to this background.

Parameters

	btn	The Button object defining the hitbox and callback.

Here is the caller graph for this function:

3.1.3.2 clearButtons()

```
void Background::clearButtons ()
```

Removes all registered buttons from this background.

3.1.3.3 draw()

```
bool Background::draw (
            TFT_eSPI & tft,
            PNG & png,
            bool center = true)
```

Draws the background image to the TFT screen.

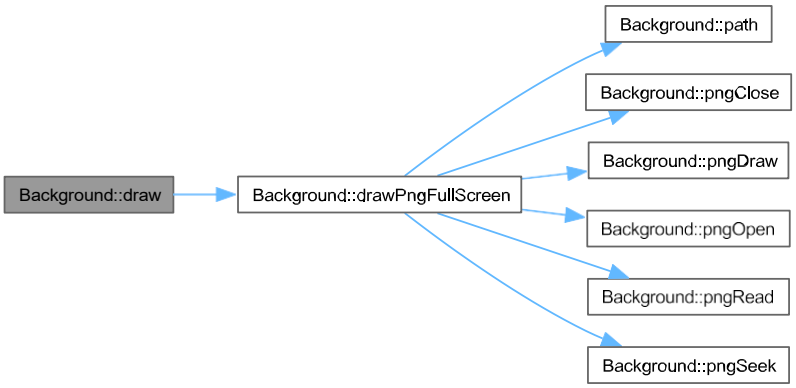
Parameters

	tft	Reference to the TFT object.
	png	Reference to the PNG decoder object.
	center	Whether to center the image on the screen (default: true).

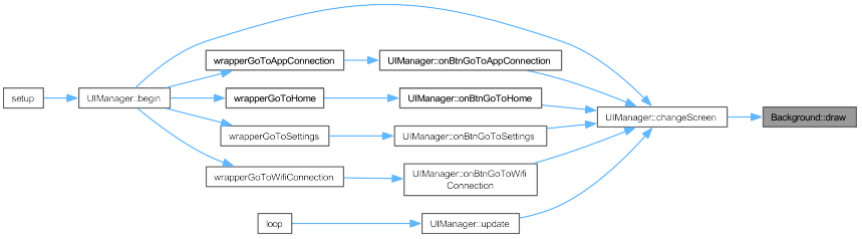
Returns

true if drawing was successful, false otherwise.

Here is the call graph for this function:



Here is the caller graph for this function:



3.1.3.4 drawPngFullScreen()

```
bool Background::drawPngFullScreen (
            const char * path,
            bool center) [private]
```

Internal helper to setup and execute the PNG drawing process.

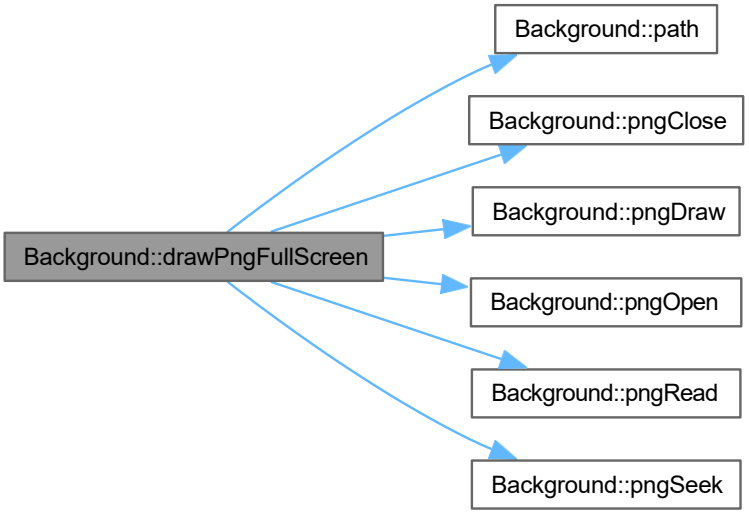
Parameters

	path	File path.
	center	Centering flag.

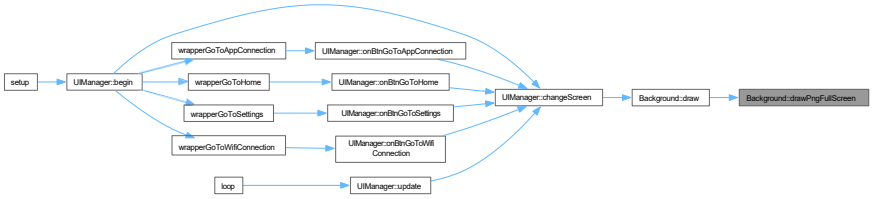
Returns

true on success.

Here is the call graph for this function:



Here is the caller graph for this function:



3.1.3.5 handleTouch()

```
bool Background::handleTouch (
            int16_t touchX,
            int16_t touchY,
            bool isPressedNow)
```

Processes touch input for all registered buttons.

Iterates through all buttons added to this background and updates their state.

Parameters

	touchX	X-coordinate of the touch event.
	touchY	Y-coordinate of the touch event.
isPressedNow	Current state of the touch (true=pressed, false=released).	

Returns

true if any button was clicked (action triggered), false otherwise.

Here is the caller graph for this function:

3.1.3.6 listFS()

```
void Background::listFS (
            const char * dir)  [static]
```

Lists files in a directory to Serial for debugging purposes.

Parameters

	dir	Directory path to list (e.g., "/").

3.1.3.7 path()

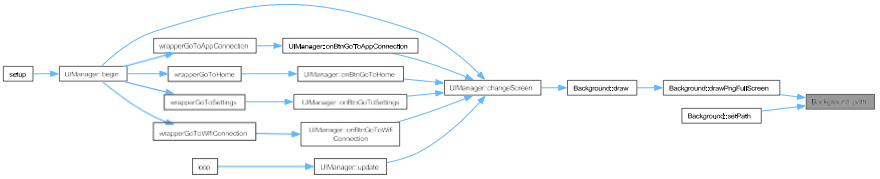
```
String Background::path () const
```

Gets the current file path.

Returns

The file path as a String.

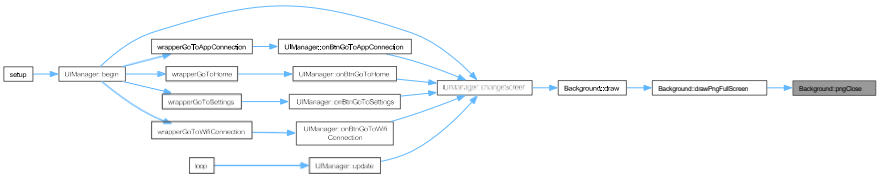
Here is the caller graph for this function:



3.1.3.8 pngClose()

```
void Background::pngClose (
            void * handle)  [static], [private]
```

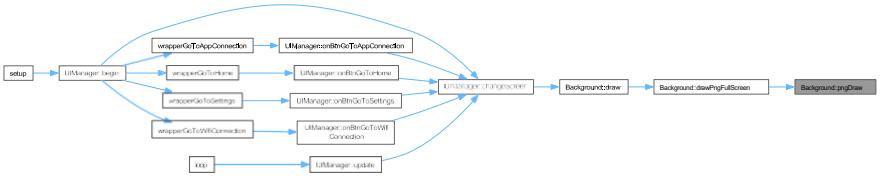
Here is the caller graph for this function:



3.1.3.9 pngDraw()

```
int Background::pngDraw (
            PNGDRAW * p)  [static], [private]
```

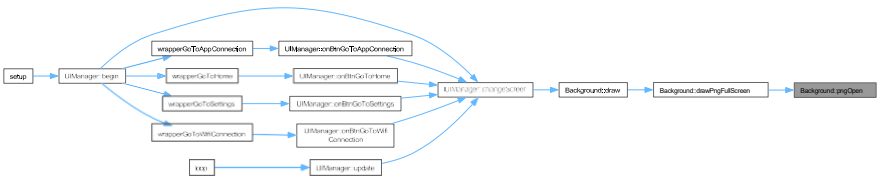
Here is the caller graph for this function:



3.1.3.10 pngOpen()

```
void * Background::pngOpen (
            const char * filename,
            int32_t * pFileSize)  [static], [private]
```

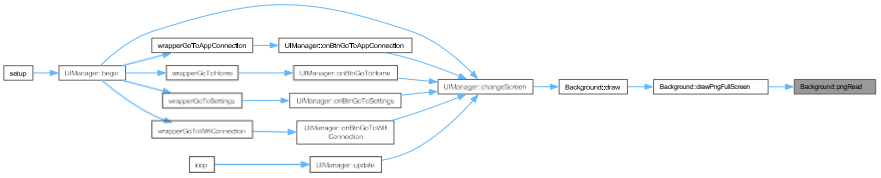
Here is the caller graph for this function:



3.1.3.11 pngRead()

```
int32_t Background::pngRead (
            PNGFILE * pFile,
            uint8_t * pBuff,
            int32_t iLen)  [static], [private]
```

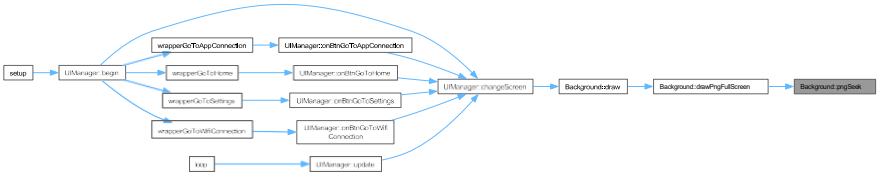
Here is the caller graph for this function:



3.1.3.12 pngSeek()

```
int32_t Background::pngSeek (
            PNGFILE * pFile,
            int32_t iPosition)  [static], [private]
```

Here is the caller graph for this function:



3.1.3.13 setPath()

```
void Background::setPath (
            const String & path)
```

Sets the file path for the background image.

Parameters

	path	The new file path (e.g., "/images/bg.png").

Here is the call graph for this function:

3.1.4 Member Data Documentation

3.1.4.1 _buttons

```
std::vector<Button> Background::_buttons  [private]
```

List of interactive buttons on this background.

3.1.4.2 _path

```
String Background::_path  [private]
```

Path to the background image file.

3.1.4.3 s_lineBuf

```
uint16_t Background::s_lineBuf  [static], [private]
```

Buffer for one line of decoded pixels.

3.1.4.4 s_offX

```
int Background::s_offX = 0  [static], [private]
```

X offset for centering the image.

3.1.4.5 s_offY

```
int Background::s_offY = 0  [static], [private]
```

Y offset for centering the image.

3.1.4.6 s_png

```
PNG * Background::s_png = nullptr  [static], [private]
```

Pointer to the PNG decoder instance.

3.1.4.7 s_pngFile

```
File Background::s_pngFile  [static], [private]
```

Handle to the open PNG file.

3.1.4.8 s_tft

```
TFT_eSPI * Background::s_tft = nullptr  [static], [private]
```

Pointer to the display driver used during draw.

The documentation for this class was generated from the following files:

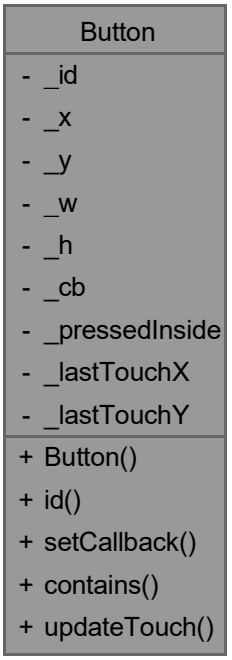
- src/Background.h
- src/Background.cpp

3.2 Button Class Reference

Represents a rectangular touch-sensitive area with a callback.

```
#include <Button.h>
```

Collaboration diagram for Button:



Public Member Functions

- Button (uint8_t id, int16_t x, int16_t y, int16_t w, int16_t h, ButtonCallback cb=nullptr)

  Constructs a new Button object.
- uint8_t id () const
- void setCallback (ButtonCallback cb)
- bool contains (int16_t x, int16_t y) const

  Checks if a point is within the button's bounds.
- bool updateTouch (int16_t touchX, int16_t touchY, bool isPressedNow)

  Updates the button state based on current touch input.

Private Attributes

- uint8_t _id

  Unique ID of the button.
- int16_t _x
- int16_t _y
- int16_t _w
- int16_t _h

  Dimensions and position of the button.
- ButtonCallback _cb

  Function to call when button is clicked.
- bool _pressedInside

  Tracks if the button is currently being pressed.
- int16_t _lastTouchX

  X-coordinate of the last touch within bounds.
- int16_t _lastTouchY

  Y-coordinate of the last touch within bounds.

3.2.1 Detailed Description

Represents a rectangular touch-sensitive area with a callback.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Button()

```
Button::Button (
            uint8_t id,
            int16_t x,
            int16_t y,
            int16_t w,
            int16_t h,
            ButtonCallback cb = nullptr)
```

Constructs a new Button object.

Parameters

	id	Unique identifier for the button.
	x	X-coordinate of top-left corner.
	y	Y-coordinate of top-left corner.
	w	Width of the button.
	h	Height of the button.
	cb	Callback function to execute on click.

3.2.3 Member Function Documentation

3.2.3.1 contains()

```
bool Button::contains (
            int16_t x,
            int16_t y) const
```

Checks if a point is within the button's bounds.

Here is the caller graph for this function:

3.2.3.2 id()

```
uint8_t Button::id () const  [inline]
```

3.2.3.3 setCallback()

```
void Button::setCallback (
            ButtonCallback cb)  [inline]
```

3.2.3.4 updateTouch()

```
bool Button::updateTouch (
            int16_t touchX,
            int16_t touchY,
            bool isPressedNow)
```

Updates the button state based on current touch input.

Parameters

	touchX	Current touch X coordinate.
	touchY	Current touch Y coordinate.
isPressedNow		Whether the screen is currently touched.

Returns

true if the button was clicked (pressed and released inside), false otherwise.

Here is the call graph for this function:

3.2.4 Member Data Documentation

3.2.4.1 _cb

```
ButtonCallback Button::_cb  [private]
```

Function to call when button is clicked.

3.2.4.2 _h

```
int16_t Button::_h  [private]
```

Dimensions and position of the button.

3.2.4.3 _id

```
uint8_t Button::_id  [private]
```

Unique ID of the button.

3.2.4.4 _lastTouchX

```
int16_t Button::_lastTouchX  [private]
```

X-coordinate of the last touch within bounds.

3.2.4.5 _lastTouchY

```
int16_t Button::_lastTouchY  [private]
```

Y-coordinate of the last touch within bounds.

3.2.4.6 _pressedInside

```
bool Button::_pressedInside  [private]
```

Tracks if the button is currently being pressed.

3.2.4.7 _w

```
int16_t Button::_w  [private]
```

3.2.4.8 _x

```
int16_t Button::_x  [private]
```

3.2.4.9 _y

```
int16_t Button::_y  [private]
```

The documentation for this class was generated from the following files:

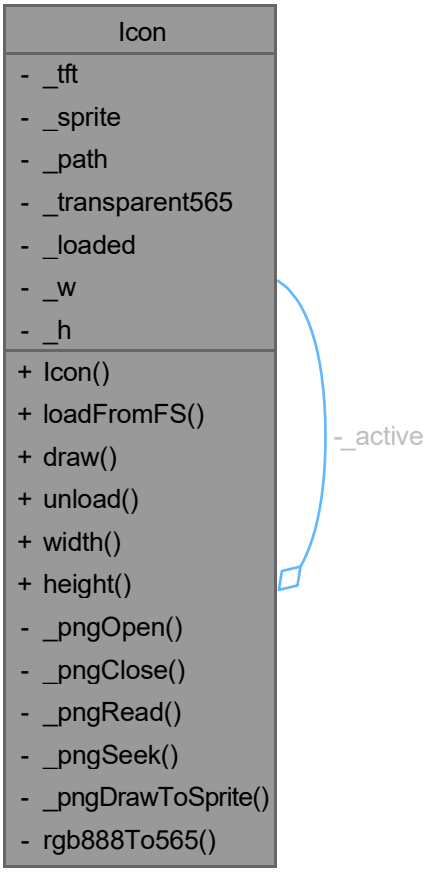
- src/Button.h
- src/Button.cpp

3.3 Icon Class Reference

Represents a graphical icon loaded from the filesystem.

```
#include <Icon.h>
```

Collaboration diagram for Icon:



Public Member Functions

- Icon (TFT_eSPI ∗tft, const char ∗path, uint8_t trR, uint8_t trG, uint8_t trB)

  Constructs a new Icon object.
- bool loadFromFS ()

  Loads the PNG image from LittleFS into a sprite.
- void draw (int16_t x, int16_t y, uint16_t bgColor=TFT_BLACK)

  Draws the icon at the specified coordinates.
- void unload ()

  Unloads the icon and frees memory (deletes sprite).
- int16_t width () const

  Gets the width of the loaded icon.
- int16_t height () const

  Gets the height of the loaded icon.

Static Private Member Functions

- static void * _pngOpen (const char *filename, int32_t *size)
- static void _pngClose (void *handle)
- static int32_t _pngRead (PNGFILE *file, uint8_t *buf, int32_t len)
- static int32_t _pngSeek (PNGFILE *file, int32_t pos)
- static int _pngDrawToSprite (PNGDRAW *pDraw)
- static uint16_t rgb888To565 (uint8_t r, uint8_t g, uint8_t b)

Private Attributes

- TFT_eSPI * _tft

  Pointer to the display driver.
- TFT_eSprite _sprite

  Sprite buffer for the icon.
- String _path

  File path of the icon image.
- uint16_t _transparent565

  Transparent color key in RGB565 format.
- bool _loaded

  Flag indicating if the icon is currently loaded.
- int16_t _w

  Icon width.
- int16_t _h

  Icon height.

Static Private Attributes

- static Icon * _active = nullptr

  Pointer to the active Icon instance for callbacks.

3.3.1 Detailed Description

Represents a graphical icon loaded from the filesystem.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Icon()

```
Icon::Icon (
            TFT_eSPI * tft,
            const char * path,
            uint8_t trR,
            uint8_t trG,
            uint8_t trB)
```

Constructs a new Icon object.

Parameters

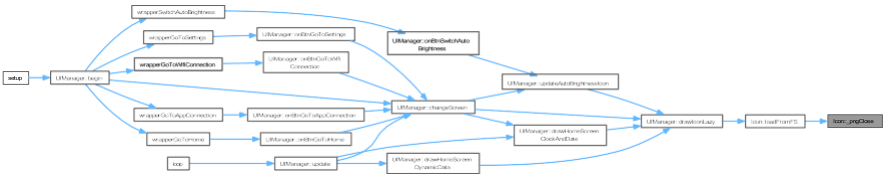
tft		Pointer to the TFT_eSPI object.
path		File path to the PNG image in LittleFS.
trR		Red component of the transparent color (0-255).
trG		Green component of the transparent color (0-255).
trB		Blue component of the transparent color (0-255).

3.3.3 Member Function Documentation

3.3.3.1 _pngClose()

```
void Icon::_pngClose (
            void * handle)  [static], [private]
```

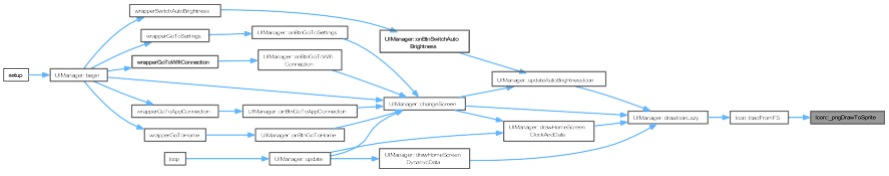
Here is the caller graph for this function:



3.3.3.2 _pngDrawToSprite()

```
int Icon::_pngDrawToSprite (
            PNGDRAW * pDraw)  [static], [private]
```

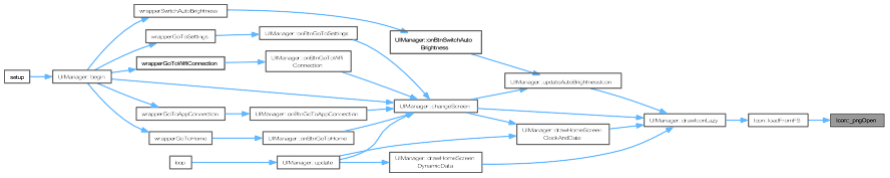
Here is the caller graph for this function:



3.3.3.3 _pngOpen()

```
void * Icon::_pngOpen (
            const char * filename,
            int32_t * size)  [static], [private]
```

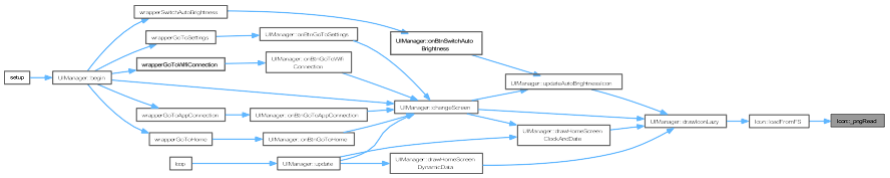
Here is the caller graph for this function:



3.3.3.4 _pngRead()

```
int32_t Icon::_pngRead (
            PNGFILE * file,
            uint8_t * buf,
            int32_t len)  [static], [private]
```

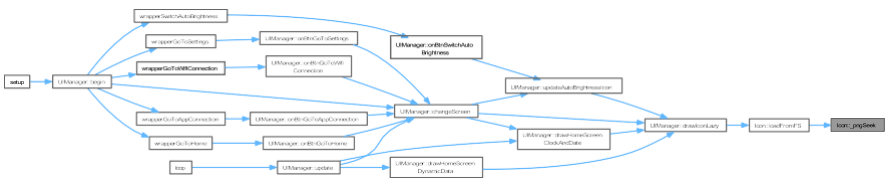
Here is the caller graph for this function:



3.3.3.5 _pngSeek()

```
int32_t Icon::_pngSeek (
            PNGFILE * file,
            int32_t pos)  [static], [private]
```

Here is the caller graph for this function:



3.3.3.6 draw()

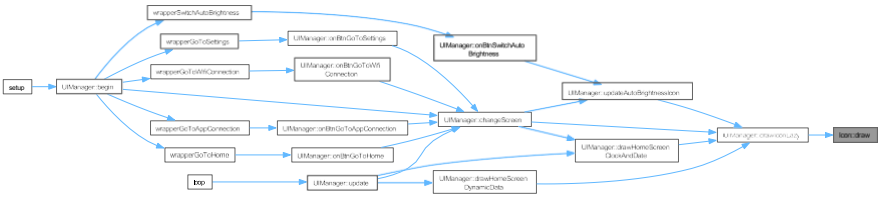
```
void Icon::draw (
            int16_t x,
            int16_t y,
            uint16_t bgColor = TFT_BLACK)
```

Draws the icon at the specified coordinates.

Parameters

	x	X-coordinate.
	y	Y-coordinate.
bgColor	Background color for the sprite push (default TFT_BLACK).	

Here is the caller graph for this function:



3.3.3.7 height()

```
int16_t Icon::height () const  [inline]
```

Gets the height of the loaded icon.

Returns

Height in pixels.

3.3.3.8 loadFromFS()

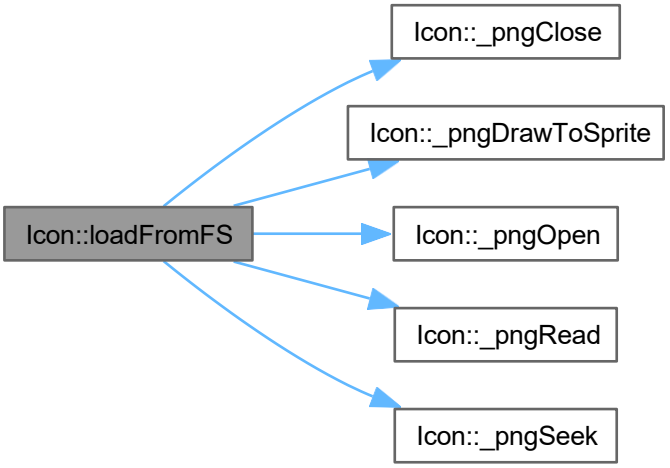
```
bool Icon::loadFromFS ()
```

Loads the PNG image from LittleFS into a sprite.

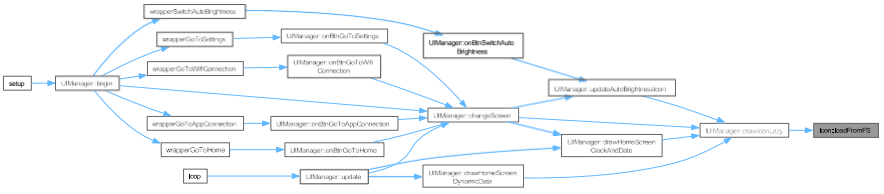
Returns

true if loading was successful, false otherwise.

Here is the call graph for this function:



Here is the caller graph for this function:



3.3.3.9 rgb888To565()

```
uint16_t Icon::rgb888To565 (
            uint8_t r,
            uint8_t g,
            uint8_t b)  [static], [private]
```

3.3.3.10 unload()

```
void Icon::unload ()
```

Unloads the icon and frees memory (deletes sprite).

3.3.3.11 width()

```
int16_t Icon::width () const  [inline]
```

Gets the width of the loaded icon.

Returns

Width in pixels.

3.3.4 Member Data Documentation

3.3.4.1 _active

```
Icon * Icon::_active = nullptr  [static], [private]
```

Pointer to the active Icon instance for callbacks.

3.3.4.2 _h

```
int16_t Icon::_h  [private]
```

Icon height.

3.3.4.3 _loaded

```
bool Icon::_loaded  [private]
```

Flag indicating if the icon is currently loaded.

3.3.4.4 _path

```
String Icon::_path  [private]
```

File path of the icon image.

3.3.4.5 _sprite

```
TFT_eSprite Icon::_sprite  [private]
```

Sprite buffer for the icon.

3.3.4.6 _tft

```
TFT_eSPI* Icon::_tft  [private]
```

Pointer to the display driver.

3.3.4.7 _transparent565

```
uint16_t Icon::_transparent565 [private]
```

Transparent color key in RGB565 format.

3.3.4.8 _w

```
int16_t Icon::_w [private]
```

Icon width.

The documentation for this class was generated from the following files:

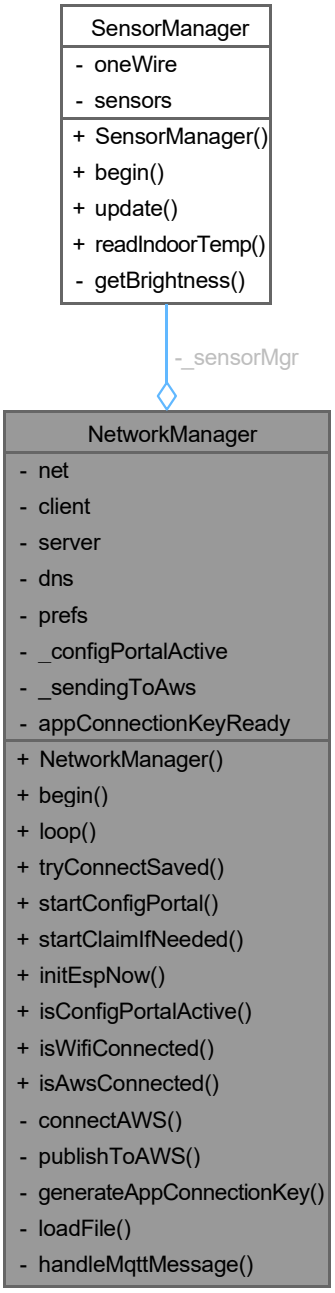
- src/Icon.h
- src/Icon.cpp

3.4 NetworkManager Class Reference

Handles all network-related operations.

```
#include <NetworkManager.h>
```

Collaboration diagram for NetworkManager:



Public Member Functions

- NetworkManager (SensorManager *sensorMgr)

  Constructs a new NetworkManager.
- void begin ()

  Initializes network services.
- void loop ()

Main network loop handling MQTT and data transmission.

- bool tryConnectSaved (unsigned timeoutMs=3000)

  Attempts to connect to saved WiFi credentials.
- void startConfigPortal ()

  Starts the configuration AP and captive portal.
- void startClaimIfNeeded ()

  Initiates the device claiming process with AWS.
- bool initEspNow ()

  Initializes ESP-NOW protocol.
- bool isConfigPortalActive ()
- bool isWifiConnected ()
- bool isAwsConnected ()

Private Member Functions

- bool connectAWS ()
- void publishToAWS ()
- void generateAppConnectionKey ()
- String loadFile (const char *path)
- void handleMqttMessage (char *topic, byte *payload, unsigned int len)

Private Attributes

- SensorManager * _sensorMgr

  Pointer to access sensor data.
- WiFiClientSecure net

  Secure WiFi client for TLS.
- PubSubClient client

  MQTT client.
- AsyncWebServer server

  Web server for provisioning.
- AsyncDNSServer dns

  DNS server for captive portal.
- Preferences prefs

  Storage for WiFi/Claim credentials.
- bool _configPortalActive = false

  Flag indicating active portal.
- bool _sendingToAws = false

  Flag for transmission state.
- bool appConnectionKeyReady = false

  Flag for nonce generation state.

Friends

- void mqttCallbackWrapper (char *topic, byte *payload, unsigned int len)

3.4.1 Detailed Description

Handles all network-related operations.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 NetworkManager()

```
NetworkManager::NetworkManager (
            SensorManager * sensorMgr)
```

Constructs a new NetworkManager.

Parameters

	sensorMgr	Pointer to the SensorManager instance.

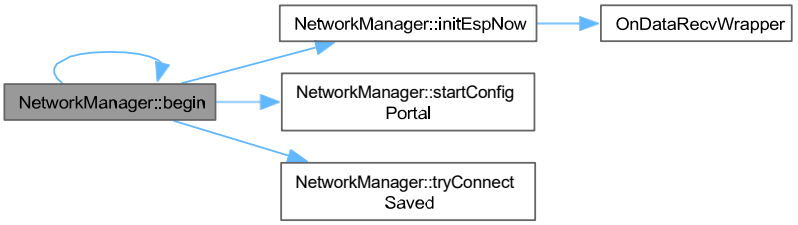
3.4.3 Member Function Documentation

3.4.3.1 begin()

```
void NetworkManager::begin ()
```

Initializes network services.

Here is the call graph for this function:



Here is the caller graph for this function:

3.4.3.2 connectAWS()

```
bool NetworkManager::connectAWS ()  [private]
```

Here is the call graph for this function:

Here is the caller graph for this function:

3.4.3.3 generateAppConnectionKey()

```
void NetworkManager::generateAppConnectionKey ()  [private]
```

Here is the caller graph for this function:

3.4.3.4 handleMqttMessage()

```
void NetworkManager::handleMqttMessage (
            char * topic,
            byte * payload,
            unsigned int len)  [private]
```

3.4.3.5 initEspNow()

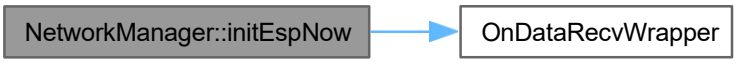
```
bool NetworkManager::initEspNow ()
```

Initializes ESP-NOW protocol.

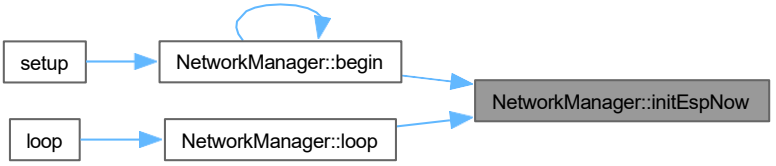
Returns

true if successful.

Here is the call graph for this function:



Here is the caller graph for this function:



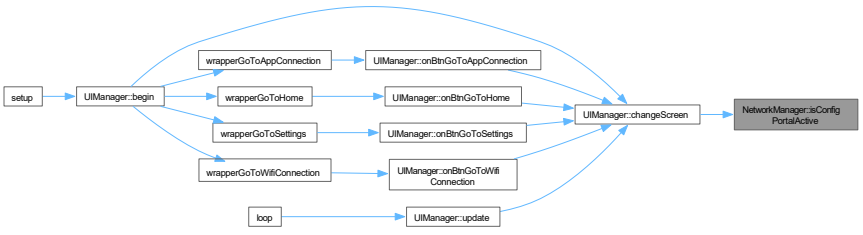
3.4.3.6 isAwsConnected()

```
bool NetworkManager::isAwsConnected ()
```

3.4.3.7 isConfigPortalActive()

```
bool NetworkManager::isConfigPortalActive ()
```

Here is the caller graph for this function:



3.4.3.8 isWifiConnected()

```
bool NetworkManager::isWifiConnected ()
```

Here is the caller graph for this function:

3.4.3.9 loadFile()

```
String NetworkManager::loadFile (
            const char * path)  [private]
```

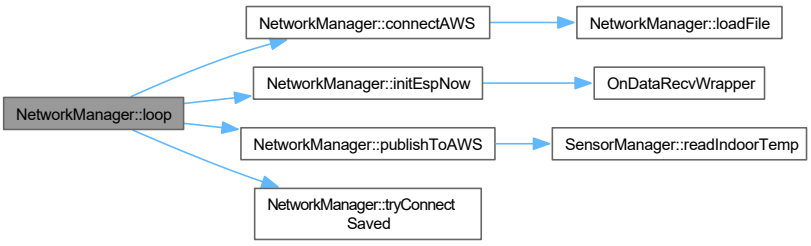
Here is the caller graph for this function:

3.4.3.10 loop()

```
void NetworkManager::loop ()
```

Main network loop handling MQTT and data transmission.

Here is the call graph for this function:



Here is the caller graph for this function:

3.4.3.11 publishToAWS()

```
void NetworkManager::publishToAWS () [private]
```

Here is the call graph for this function:

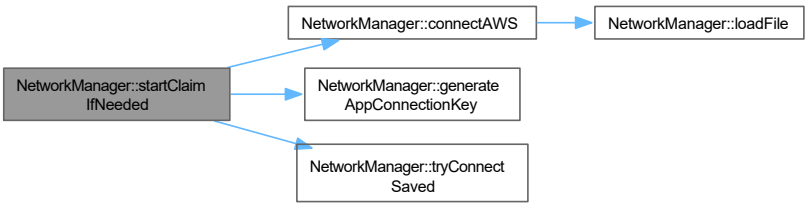
Here is the caller graph for this function:

3.4.3.12 startClaimIfNeeded()

```
void NetworkManager::startClaimIfNeeded ()
```

Initiates the device claiming process with AWS.

Here is the call graph for this function:



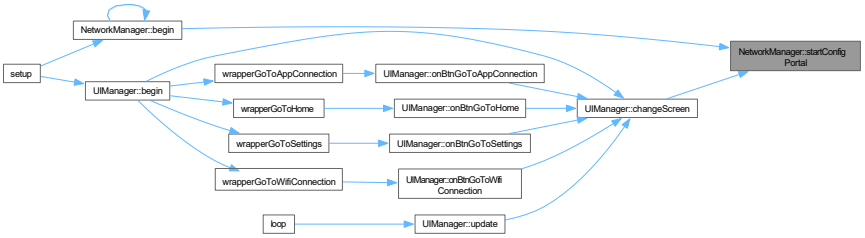
Here is the caller graph for this function:

3.4.3.13 startConfigPortal()

```
void NetworkManager::startConfigPortal ()
```

Starts the configuration AP and captive portal.

Here is the caller graph for this function:



3.4.3.14 tryConnectSaved()

```
bool NetworkManager::tryConnectSaved (
            unsigned timeoutMs = 3000)
```

Attempts to connect to saved WiFi credentials.

Parameters

	timeoutMs	Connection timeout in milliseconds.

Returns

true if connected, false otherwise.

Here is the caller graph for this function:

3.4.4 Friends And Related Symbol Documentation

3.4.4.1 mqttCallbackWrapper

```
void mqttCallbackWrapper (
            char * topic,
            byte * payload,
            unsigned int len)  [friend]
```

3.4.5 Member Data Documentation

3.4.5.1 _configPortalActive

```
bool NetworkManager::_configPortalActive = false  [private]
```

Flag indicating active portal.

3.4.5.2 _sendingToAws

```
bool NetworkManager::_sendingToAws = false  [private]
```

Flag for transmission state.

3.4.5.3 _sensorMgr

```
SensorManager* NetworkManager::_sensorMgr  [private]
```

Pointer to access sensor data.

3.4.5.4 appConnectionKeyReady

```
bool NetworkManager::appConnectionKeyReady = false  [private]
```

Flag for nonce generation state.

3.4.5.5 client

```
PubSubClient NetworkManager::client  [private]
```

MQTT client.

3.4.5.6 dns

```
AsyncDNSServer NetworkManager::dns  [private]
```

DNS server for captive portal.

3.4.5.7 net

```
WiFiClientSecure NetworkManager::net [private]
```

Secure WiFi client for TLS.

3.4.5.8 prefs

```
Preferences NetworkManager::prefs [private]
```

Storage for WiFi/Claim credentials.

3.4.5.9 server

```
AsyncWebServer NetworkManager::server [private]
```

Web server for provisioning.

The documentation for this class was generated from the following files:

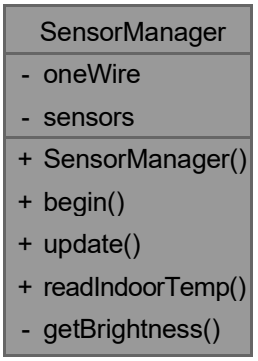
- src/NetworkManager.h
- src/NetworkManager.cpp

3.5 SensorManager Class Reference

Handles reading sensors and managing hardware state.

```
#include <SensorManager.h>
```

Collaboration diagram for SensorManager:



Public Member Functions

- SensorManager ()
- void begin ()

  Initializes sensor hardware.
- void update ()

  Updates sensor readings and hardware state.
- float readIndoorTemp ()

  Reads the indoor temperature from DS18B20.

Private Member Functions

- int getBrightness ()

Private Attributes

- OneWire oneWire

  OneWire interface for DS18B20.
- DallasTemperature sensors

  DallasTemp wrapper.

3.5.1 Detailed Description

Handles reading sensors and managing hardware state.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 SensorManager()

```
SensorManager::SensorManager ()
```

3.5.3 Member Function Documentation

3.5.3.1 begin()

```
void SensorManager::begin ()
```

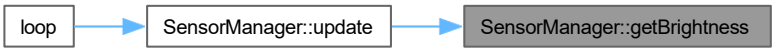
Initializes sensor hardware.

Here is the caller graph for this function:

3.5.3.2 getBrightness()

```
int SensorManager::getBrightness ()  [private]
```

Here is the caller graph for this function:



3.5.3.3 readIndoorTemp()

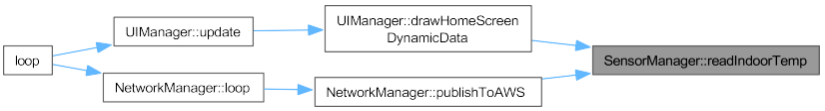
```
float SensorManager::readIndoorTemp ()
```

Reads the indoor temperature from DS18B20.

Returns

Temperature in Celsius or NAN on error.

Here is the caller graph for this function:

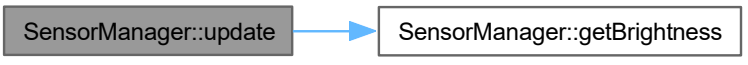


3.5.3.4 update()

```
void SensorManager::update ()
```

Updates sensor readings and hardware state.

Here is the call graph for this function:



Here is the caller graph for this function:

3.5.4 Member Data Documentation

3.5.4.1 oneWire

```
OneWire SensorManager::oneWire [private]
```

OneWire interface for DS18B20.

3.5.4.2 sensors

```
DallasTemperature SensorManager::sensors [private]
```

DallasTemp wrapper.

The documentation for this class was generated from the following files:

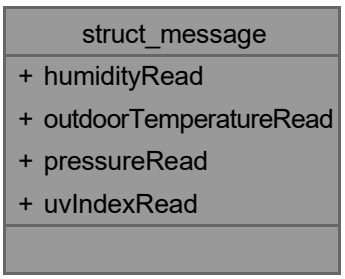
- src/SensorManager.h
- src/SensorManager.cpp

3.6 struct_message Struct Reference

Data structure for ESP-NOW sensor data.

```
#include <Globals.h>
```

Collaboration diagram for struct_message:



Public Attributes

- uint8_t humidityRead

  Humidity reading.
- int16_t outdoorTemperatureRead

  Outdoor temp value ∗ 10 (e.g., 255 = 25.5C).
- uint16_t pressureRead

  Atmospheric pressure reading.
- uint8_t uvIndexRead

  UV Index value ∗ 10.

3.6.1 Detailed Description

Data structure for ESP-NOW sensor data.

3.6.2 Member Data Documentation

3.6.2.1 humidityRead

```
uint8_t struct_message::humidityRead
```

Humidity reading.

3.6.2.2 outdoorTemperatureRead

```
int16_t struct_message::outdoorTemperatureRead
```

Outdoor temp value ∗ 10 (e.g., 255 = 25.5C).

3.6.2.3 pressureRead

```
uint16_t struct_message::pressureRead
```

Atmospheric pressure reading.

3.6.2.4 uvIndexRead

```
uint8_t struct_message::uvIndexRead
```

UV Index value ∗ 10.

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

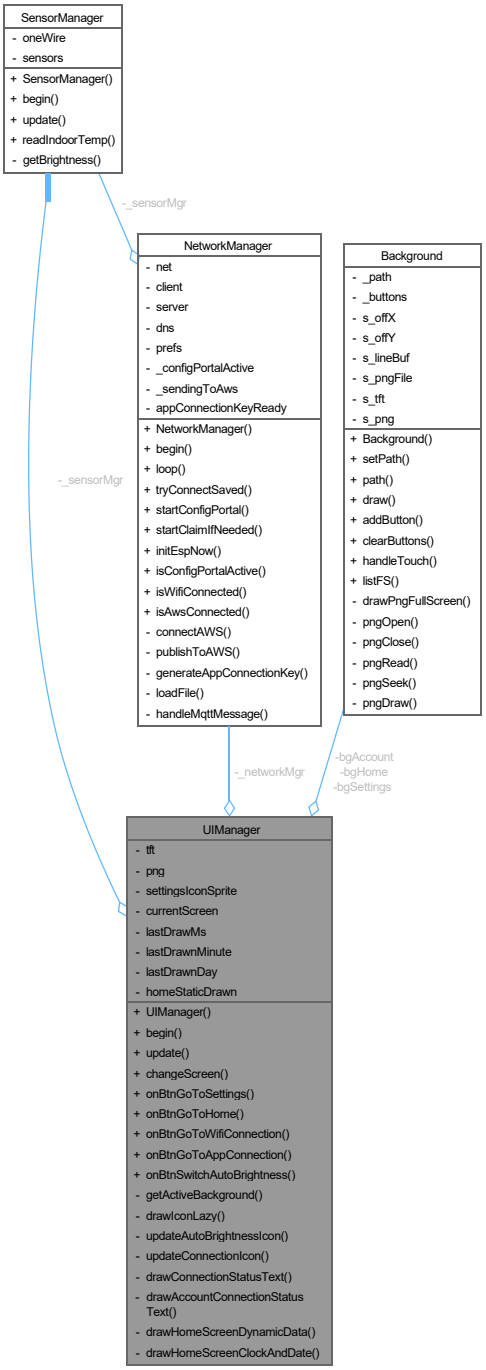
- src/Globals.h

3.7 UIManager Class Reference

Controls the display, rendering logic, and user input.

`#include <UIManager.h>`

Collaboration diagram for UIManager:



Public Member Functions

- UIManager (SensorManager ∗sensorMgr, NetworkManager ∗networkMgr)

  Constructs a new UIManager object.
- void begin ()

  Initializes the display and UI resources.
- void update ()

  Main UI loop (updates display and handles touch).
- void changeScreen (SCREEN s)

  Switches the active screen.
- void onBtnGoToSettings ()
- void onBtnGoToHome ()
- void onBtnGoToWifiConnection ()
- void onBtnGoToAppConnection ()
- void onBtnSwitchAutoBrightness ()

Private Member Functions

- Background ∗ getActiveBackground ()
- void drawIconLazy (const char ∗path, int x, int y, uint16_t bg=TFT_BLACK)
- void updateAutoBrightnessIcon (bool status)
- void updateConnectionIcon (bool status)
- void drawConnectionStatusText ()
- void drawAccountConnectionStatusText ()
- void drawHomeScreenDynamicData ()
- void drawHomeScreenClockAndDate ()

Private Attributes

- TFT_eSPI tft

  TFT driver.
- PNG png

  PNG decoder.
- SensorManager ∗ _sensorMgr

  Pointer to SensorManager.
- NetworkManager ∗ _networkMgr

  Pointer to NetworkManager.
- Background ∗ bgHome

  Home screen background.
- Background ∗ bgSettings

  Settings screen background.
- Background ∗ bgAccount

  Account/WiFi screen background.
- TFT_eSprite settingsIconSprite

  Helper sprite for settings icon.
- SCREEN currentScreen

  Currently active screen.
- uint32_t lastDrawMs = 0

  Timestamp of last UI redraw.
- int lastDrawnMinute = -1

  Last drawn minute value (to avoid redraws).
- int lastDrawnDay = -1

  Last drawn day value.
- bool homeStaticDrawn = false

  Flag if static elements are drawn.

3.7.1 Detailed Description

Controls the display, rendering logic, and user input.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 UIManager()

```
UIManager::UIManager (
            SensorManager * sensorMgr,
            NetworkManager * networkMgr)
```

Constructs a new UIManager object.

Parameters

	sensorMgr	Pointer to SensorManager.
	networkMgr	Pointer to NetworkManager.

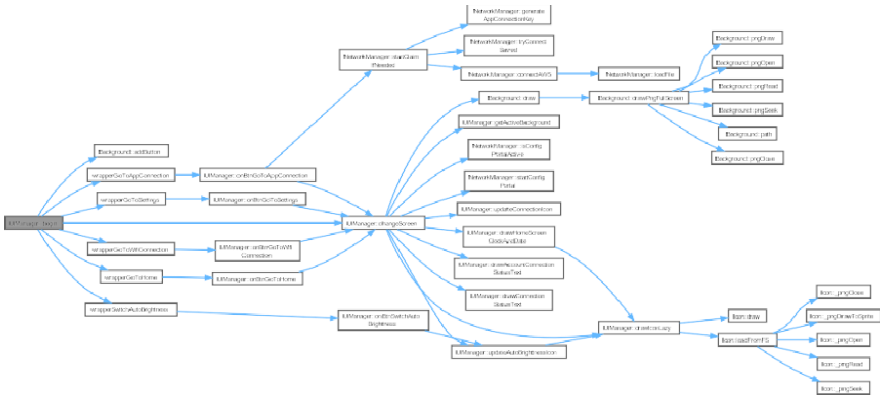
3.7.3 Member Function Documentation

3.7.3.1 begin()

```
void UIManager::begin ()
```

Initializes the display and UI resources.

Here is the call graph for this function:



Here is the caller graph for this function:

3.7.3.2 changeScreen()

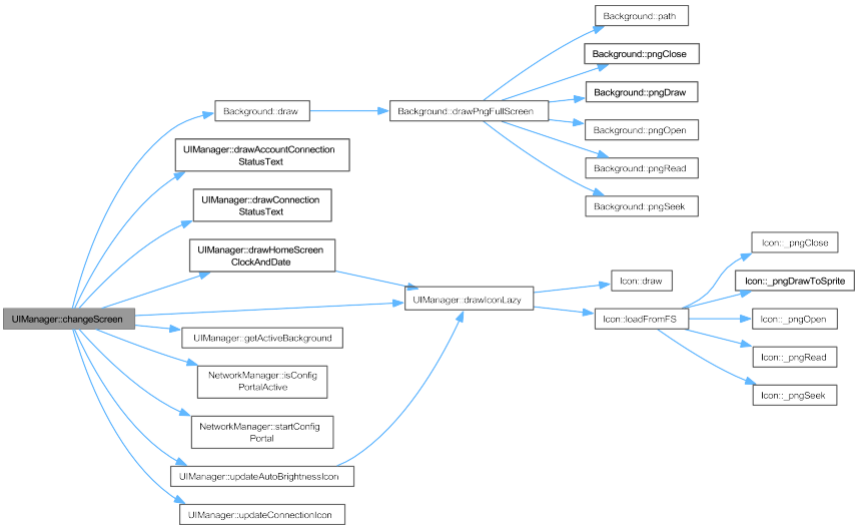
```
void UIManager::changeScreen (
            SCREEN s)
```

Switches the active screen.

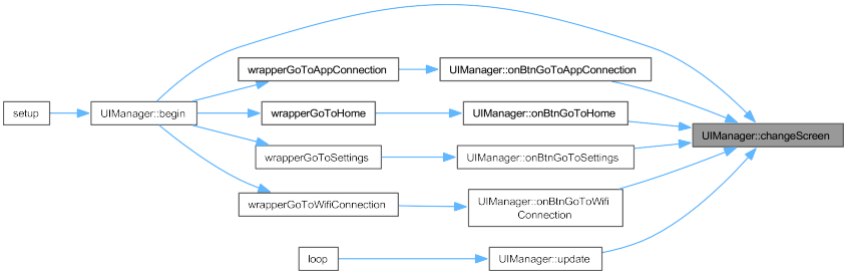
Parameters

	s	The target screen enum.

Here is the call graph for this function:



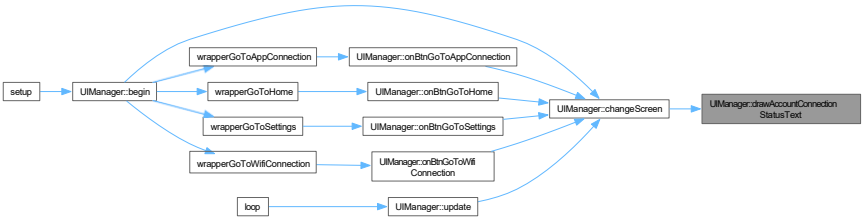
Here is the caller graph for this function:



3.7.3.3 drawAccountConnectionStatusText()

```
void UIManager::drawAccountConnectionStatusText () [private]
```

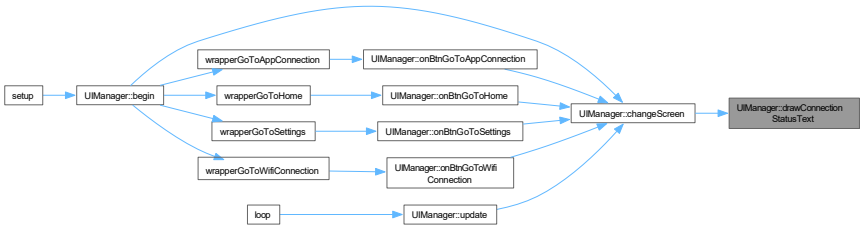
Here is the caller graph for this function:



3.7.3.4 drawConnectionStatusText()

```
void UIManager::drawConnectionStatusText () [private]
```

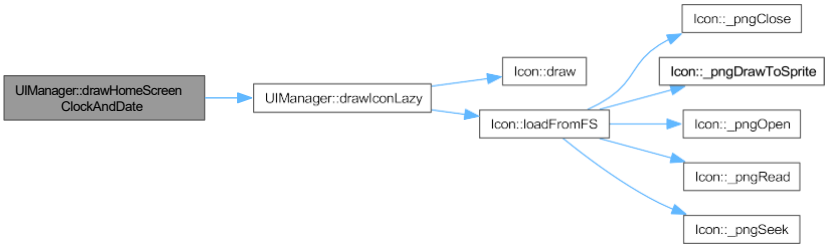
Here is the caller graph for this function:



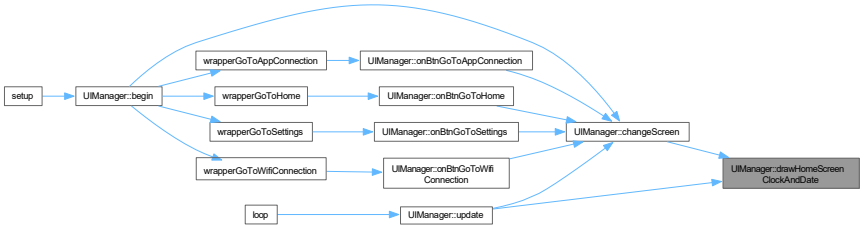
3.7.3.5 drawHomeScreenClockAndDate()

```
void UIManager::drawHomeScreenClockAndDate () [private]
```

Here is the call graph for this function:



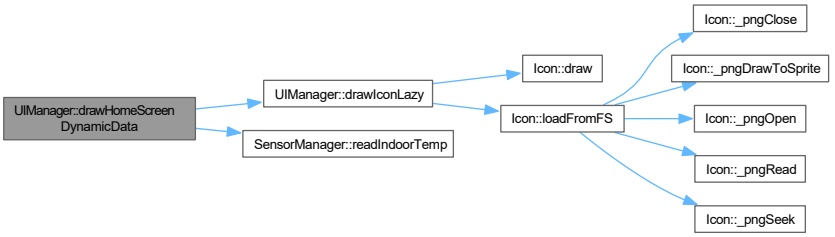
Here is the caller graph for this function:



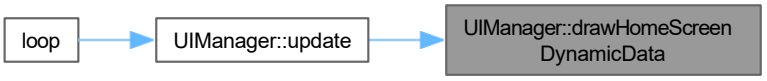
3.7.3.6 drawHomeScreenDynamicData()

```
void UIManager::drawHomeScreenDynamicData ()  [private]
```

Here is the call graph for this function:



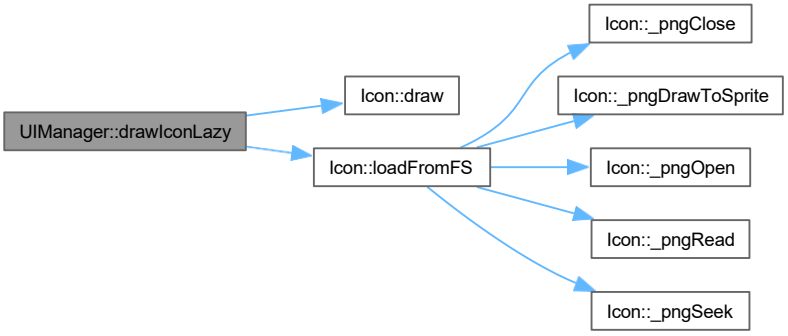
Here is the caller graph for this function:



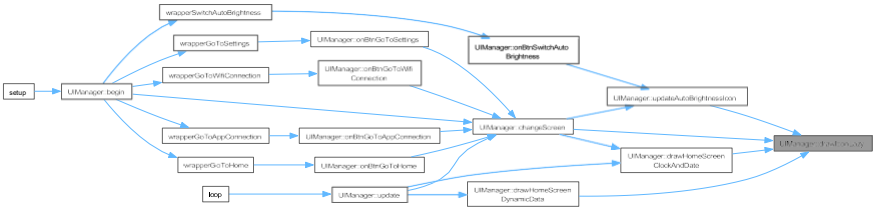
3.7.3.7 drawIconLazy()

```
void UIManager::drawIconLazy (
            const char * path,
            int x,
            int y,
            uint16_t bg = TFT_BLACK)  [private]
```

Here is the call graph for this function:



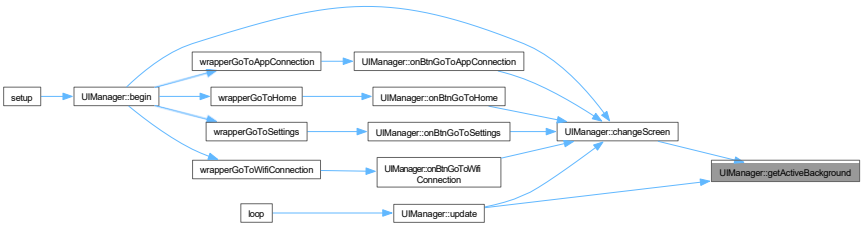
Here is the caller graph for this function:



3.7.3.8 getActiveBackground()

```
Background * UIManager::getActiveBackground () [private]
```

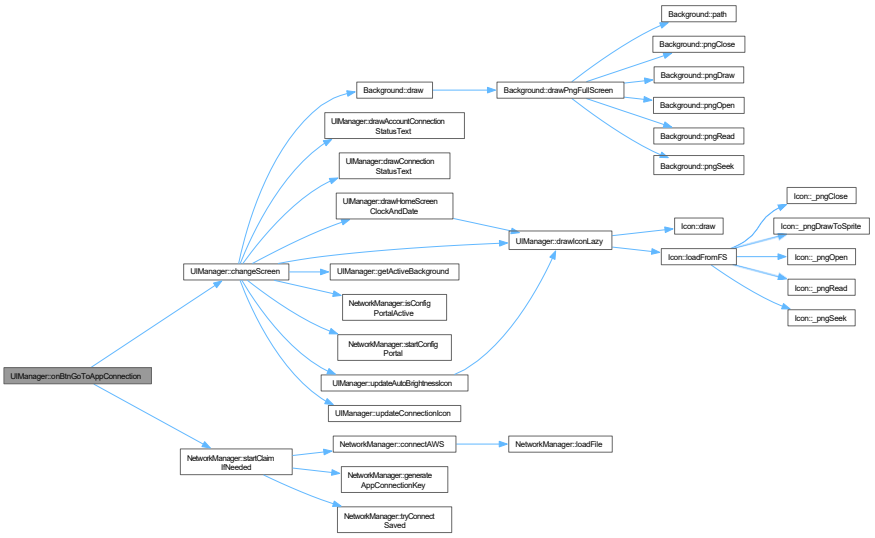
Here is the caller graph for this function:



3.7.3.9 onBtnGoToAppConnection()

```
void UIManager::onBtnGoToAppConnection ()
```

Here is the call graph for this function:

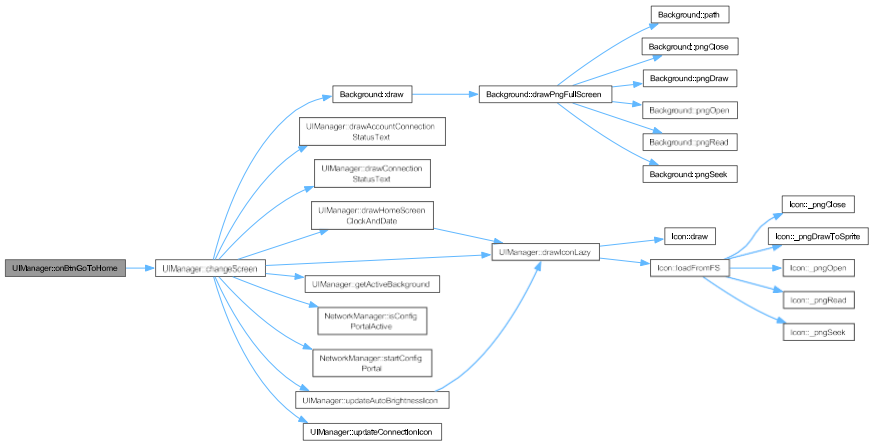


Here is the caller graph for this function:

3.7.3.10 onBtnGoToHome()

```
void UIManager::onBtnGoToHome ()
```

Here is the call graph for this function:

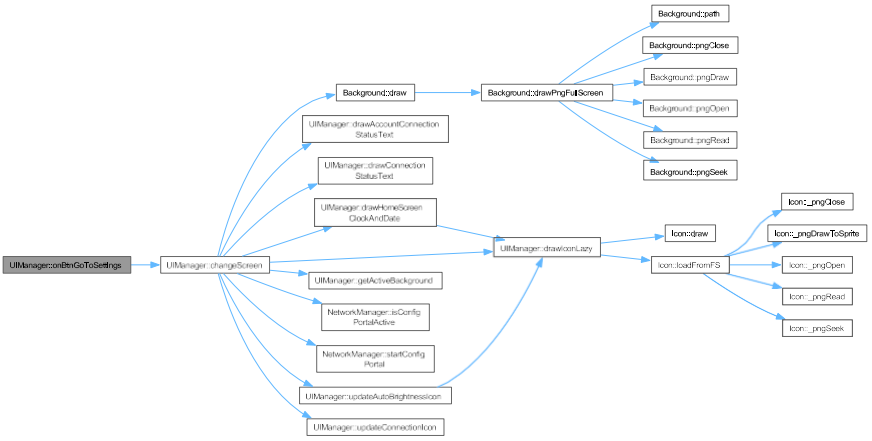


Here is the caller graph for this function:

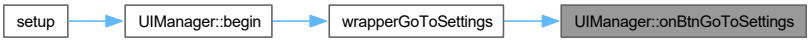
3.7.3.11 onBtnGoToSettings()

```
void UIManager::onBtnGoToSettings ()
```

Here is the call graph for this function:



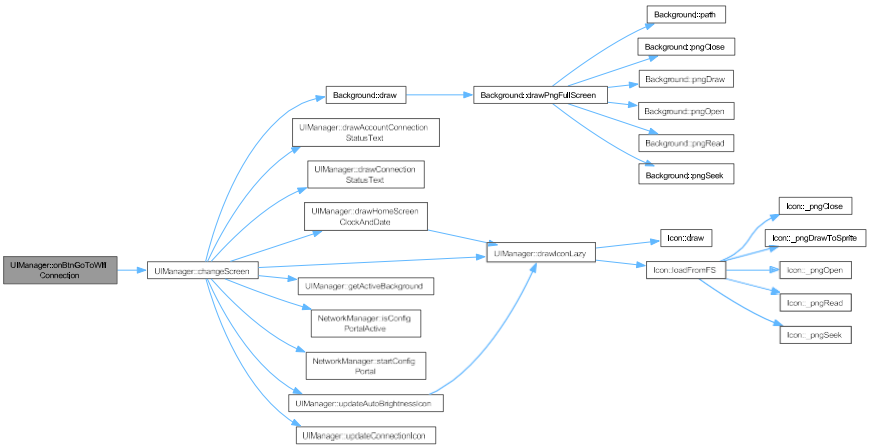
Here is the caller graph for this function:



3.7.3.12 onBtnGoToWifiConnection()

```
void UIManager::onBtnGoToWifiConnection ()
```

Here is the call graph for this function:

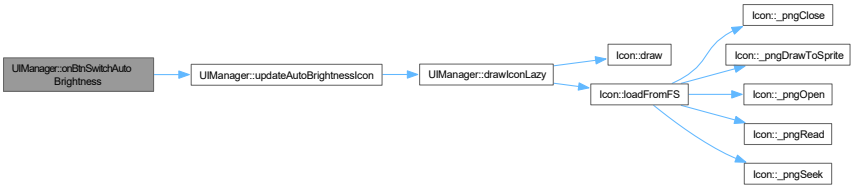


Here is the caller graph for this function:

3.7.3.13 onBtnSwitchAutoBrightness()

```
void UIManager::onBtnSwitchAutoBrightness ()
```

Here is the call graph for this function:



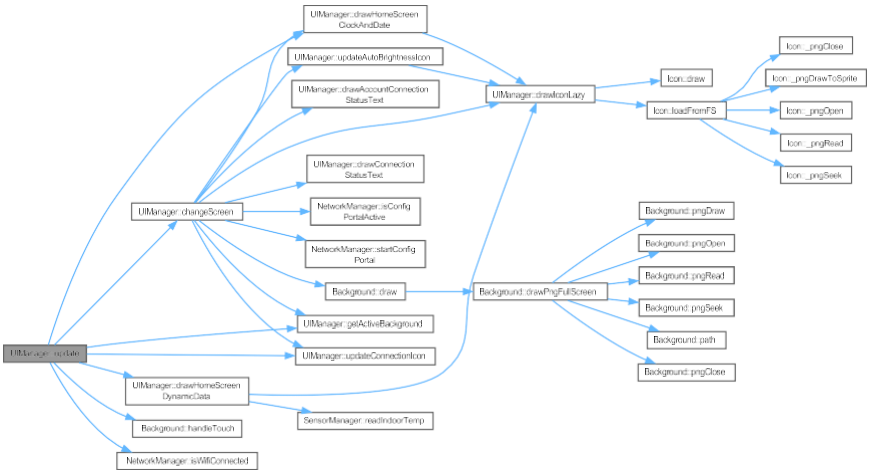
Here is the caller graph for this function:

3.7.3.14 update()

```
void UIManager::update ()
```

Main UI loop (updates display and handles touch).

Here is the call graph for this function:

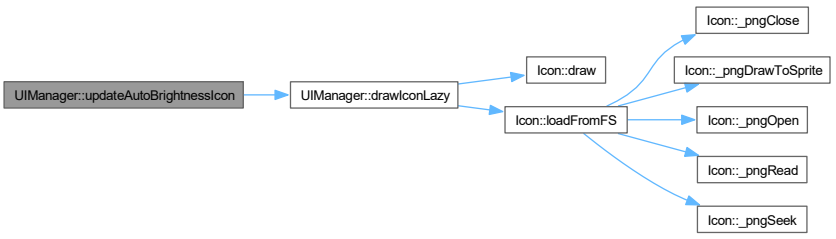


Here is the caller graph for this function:

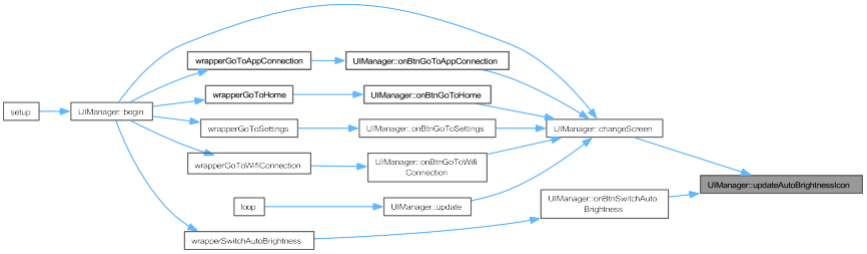
3.7.3.15 updateAutoBrightnessIcon()

```
void UIManager::updateAutoBrightnessIcon (
            bool status)  [private]
```

Here is the call graph for this function:



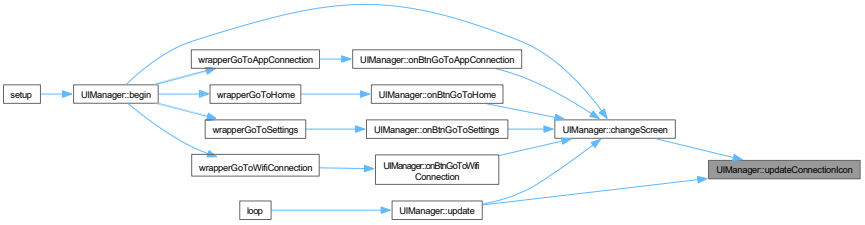
Here is the caller graph for this function:



3.7.3.16 updateConnectionIcon()

```
void UIManager::updateConnectionIcon (
            bool status)  [private]
```

Here is the caller graph for this function:



3.7.4 Member Data Documentation

3.7.4.1 _networkMgr

```
NetworkManager* UIManager::_networkMgr  [private]
```

Pointer to NetworkManager.

3.7.4.2 _sensorMgr

```
SensorManager* UIManager::_sensorMgr  [private]
```

Pointer to SensorManager.

3.7.4.3 bgAccount

```
Background* UIManager::bgAccount  [private]
```

Account/WiFi screen background.

3.7.4.4 bgHome

```
Background* UIManager::bgHome  [private]
```

Home screen background.

3.7.4.5 bgSettings

```
Background* UIManager::bgSettings  [private]
```

Settings screen background.

3.7.4.6 currentScreen

```
SCREEN UIManager::currentScreen  [private]
```

Currently active screen.

3.7.4.7 homeStaticDrawn

```
bool UIManager::homeStaticDrawn = false  [private]
```

Flag if static elements are drawn.

3.7.4.8 lastDrawMs

```
uint32_t UIManager::lastDrawMs = 0  [private]
```

Timestamp of last UI redraw.

3.7.4.9 lastDrawnDay

```
int UIManager::lastDrawnDay = -1  [private]
```

Last drawn day value.

3.7.4.10 lastDrawnMinute

```
int UIManager::lastDrawnMinute = -1  [private]
```

Last drawn minute value (to avoid redraws).

3.7.4.11 png

```
PNG UIManager::png  [private]
```

PNG decoder.

3.7.4.12 settingsIconSprite

```
TFT_eSprite UIManager::settingsIconSprite [private]
```

Helper sprite for settings icon.

3.7.4.13 tft

```
TFT_eSPI UIManager::tft [private]
```

TFT driver.

The documentation for this class was generated from the following files:

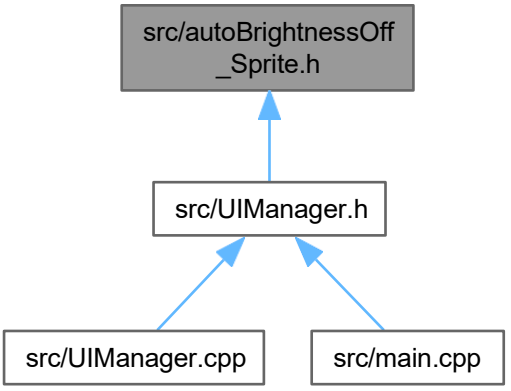
- src/UIManager.h
- src/UIManager.cpp

Chapter 4

File Documentation

4.1 src/autoBrightnessOff_Sprite.h File Reference

This graph shows which files directly or indirectly include this file:



Variables

- const unsigned short autoBrightnessOff_Sprite[576] PROGMEM

4.1.1 Variable Documentation

4.1.1.1 PROGMEM

```
const unsigned short autoBrightnessOff_Sprite [576] PROGMEM
```

4.2 autoBrightnessOff_Sprite.h

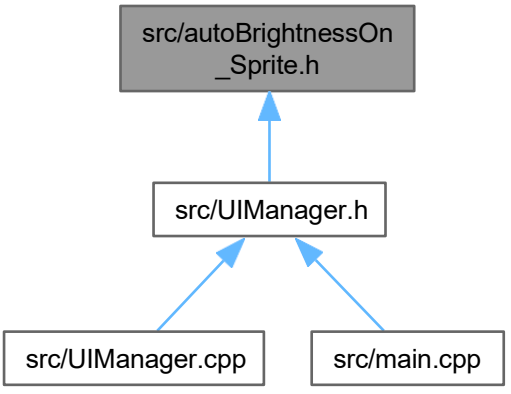
Go to the documentation of this file.

```
00001 // Generated by   : ImageConverter 565 Online
00002 // Generated from : autoBrightnessOff_Sprite.png
00003 // Time generated : Fri, 05 Dec 25 21:39:53 +0100  (Server timezone: CET)
00004 // Image Size     : 24x24 pixels
00005 // Memory usage   : 1152 bytes
00006
00007 #if defined(__AVR__)
00008 #include <avr/pgmspace.h>
00009 #elif defined(__PIC32MX__)
00010 #define PROGMEM
00011 #elif defined(__arm__)
00012 #define PROGMEM
00013 #endif
00014
00015 const unsigned short
00016     autoBrightnessOff_Sprite[576] PROGMEM =
00017         {
00018              0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00019              0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00020              0x5B92, 0x5B92, 0x5B92, 0x5B72, // 0x0010 (16) pixels
00021              0x5B92, 0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B92,
00022              0x63B3, 0x6C14, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00023              0x5B72, 0x5B72, 0x5B72, 0x5B72, // 0x0020 (32) pixels
00024              0x5B72, 0x5B72, 0x5B72, 0x5B71, 0x5B71, 0x5B71,
00025              0x5B71, 0x5B71, 0x5B71, 0x5B71, 0x5B51, 0x5B51,
00026              0x5B51, 0x5B51, 0x5B51, 0x5B92, // 0x0030 (48) pixels
00027              0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B72,
00028              0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72,
00029              0x5B71, 0x5B71, 0x5B71, 0x5B72, // 0x0040 (64) pixels
00030              0x5B71, 0x5B71, 0x5B51, 0x5B51, 0x5B51, 0x5B51,
00031              0x5B51, 0x5B72, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00032              0x5B92, 0x5B72, 0x5B72, 0x5B72, // 0x0050 (80) pixels
00033              0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B71, 0x5B71,
00034              0x5B71, 0x5B71, 0x5B72, 0x5B71, 0x5B71, 0x5B51,
00035              0x5B51, 0x5B51, 0x5B51, 0x5B71, // 0x0060 (96) pixels
00036              0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00037              0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72,
00038              0x5B72, 0x5B71, 0x5B71, 0x5B71, // 0x0070 (112) pixels
00039              0x5B71, 0x5B71, 0x5B71, 0x5B51, 0x5B51, 0x5B51,
00040              0x5B51, 0x5B71, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00041              0x5B92, 0x5B92, 0x5B92, 0x5B72, // 0x0080 (128) pixels
00042              0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72,
00043              0x5B71, 0x5B71, 0x5B71, 0x5B71, 0x5B71, 0x5B51,
00044              0x5B51, 0x5B51, 0x5B51, 0x5B51, // 0x0090 (144) pixels
00045              0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00046              0x5B92, 0x5B92, 0x5B72, 0x5B72, 0x5B72, 0x5B72,
00047              0x5B72, 0x5B72, 0x5B72, 0x5B71, // 0x00A0 (160) pixels
00048              0x5B71, 0x5B71, 0x5B71, 0x5B71, 0x5B51, 0x5B51,
00049              0x5B51, 0x5B51, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00050              0x5B92, 0x5B92, 0x5B92, 0x5B92, // 0x00B0 (176) pixels
00051              0x5B92, 0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72,
00052              0x5B72, 0x5B72, 0x5B71, 0x5B71, 0x5B71, 0x5B71,
00053              0x5B71, 0x5B51, 0x5B51, 0x5B71, // 0x00C0 (192) pixels
00054              0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00055              0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B72, 0x5B72,
00056              0x5B72, 0x5B72, 0x5B72, 0x5B72, // 0x00D0 (208) pixels
00057              0x5B72, 0x5B71, 0x5B71, 0x5B71, 0x5B71, 0x5B51,
00058              0x5B51, 0x5B71, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00059              0x5B92, 0x5B92, 0x5B92, 0x5B92, // 0x00E0 (224) pixels
00060              0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72,
00061              0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B71, 0x5B71,
00062              0x5B71, 0x5B71, 0x5B51, 0x5B71, // 0x00F0 (240) pixels
00063              0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00064              0x5B92, 0x5B92, 0x5B72, 0x5B72, 0x5B72, 0x5B72,
00065              0x5B72, 0x5B72, 0x5B72, 0x5B72, // 0x0100 (256) pixels
00066              0x5B72, 0x5B72, 0x5B72, 0x5B71, 0x5B71, 0x5B71,
00067              0x5B71, 0x5B71, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00068              0x5B92, 0x5B92, 0x5B92, 0x5B92, // 0x0110 (272) pixels
00069              0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72,
00070              0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72,
00071              0x5B71, 0x5B71, 0x5B71, 0x5B71, // 0x0120 (288) pixels
00072              0x5BB2, 0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00073              0x5B92, 0x5B92, 0x5B92, 0x5B72, 0x5B72, 0x5B72,
00074              0x5B72, 0x5B72, 0x5B72, 0x5B72, // 0x0130 (304) pixels
00075              0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B71, 0x5B71,
00076              0x5B71, 0x5B71, 0x5BB2, 0x5B92, 0x5B92, 0x5B92,
00077              0x5B92, 0x5B92, 0x5B92, 0x5B92, // 0x0140 (320) pixels
00078              0x5B92, 0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72,
00079              0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72,
00080              0x5B72, 0x5B71, 0x5B71, 0x5B71, // 0x0150 (336) pixels
00081              0x5BB2, 0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00082              0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B72, 0x5B72,
```

```
00083             0x5B72, 0x5B72, 0x5B72, 0x5B72, // 0x0160 (352) pixels
00084             0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B71,
00085             0x5B71, 0x5B71, 0x5BB2, 0x5B92, 0x5B92, 0x5B92,
00086             0x5B92, 0x5B92, 0x5B92, 0x5B92, // 0x0170 (368) pixels
00087             0x5B92, 0x5B92, 0x5B92, 0x5B72, 0x5B72, 0x5B72,
00088             0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72,
00089             0x5B72, 0x5B71, 0x5B71, 0x5B71, // 0x0180 (384) pixels
00090             0x5BB2, 0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00091             0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00092             0x5B92, 0x5B72, 0x5B72, 0x5B72, // 0x0190 (400) pixels
00093             0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B71,
00094             0x5B71, 0x5B71, 0x5BB2, 0x5B92, 0x5B92, 0x5B92,
00095             0x5B92, 0x5B92, 0x5B92, 0x5B92, // 0x01A0 (416) pixels
00096             0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B72,
00097             0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72,
00098             0x5B72, 0x5B72, 0x5B71, 0x5B71, // 0x01B0 (432) pixels
00099             0x5BB2, 0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00100             0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00101             0x5B92, 0x5B72, 0x5B72, 0x5B72, // 0x01C0 (448) pixels
00102             0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72,
00103             0x5B71, 0x5B71, 0x5BB2, 0x5B92, 0x5B92, 0x5B92,
00104             0x5B92, 0x5B92, 0x5B92, 0x5B92, // 0x01D0 (464) pixels
00105             0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00106             0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72,
00107             0x5B72, 0x5B72, 0x5B72, 0x5B71, // 0x01E0 (480) pixels
00108             0x63B2, 0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00109             0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00110             0x5B92, 0x5B92, 0x5B92, 0x5B72, // 0x01F0 (496) pixels
00111             0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72,
00112             0x5B72, 0x5B72, 0x63B2, 0x5B92, 0x5B92, 0x5B92,
00113             0x5B92, 0x5B92, 0x5B92, 0x5B92, // 0x0200 (512) pixels
00114             0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00115             0x5B92, 0x5B92, 0x5B72, 0x5B72, 0x5B72, 0x5B72,
00116             0x5B72, 0x5B72, 0x5B72, 0x5B72, // 0x0210 (528) pixels
00117             0x63D3, 0x5BB2, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00118             0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00119             0x5B92, 0x5B92, 0x5B92, 0x5B92, // 0x0220 (544) pixels
00120             0x5B92, 0x5B72, 0x5B72, 0x5B72, 0x5B72, 0x5B72,
00121             0x5B72, 0x5B72, 0x7454, 0x63D3, 0x63B2, 0x5BB2,
00122             0x5B92, 0x5B92, 0x5B92, 0x5B92, // 0x0230 (560) pixels
00123             0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00124             0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92, 0x5B92,
00125             0x5B92, 0x5B92, 0x5B92, 0x5B72, // 0x0240 (576) pixels
00126 };
```

4.3 src/autoBrightnessOn_Sprite.h File Reference

This graph shows which files directly or indirectly include this file:



Variables

- const unsigned short autoBrightnessOn_Sprite[576] PROGMEM

4.3.1 Variable Documentation

4.3.1.1 PROGMEM

```
const unsigned short autoBrightnessOn_Sprite [576] PROGMEM
```

4.4 autoBrightnessOn_Sprite.h

Go to the documentation of this file.

```
00001 // Generated by   : ImageConverter 565 Online
00002 // Generated from : autoBrightnessOn_Sprite.png
00003 // Time generated : Fri, 05 Dec 25 21:39:30 +0100  (Server timezone: CET)
00004 // Image Size     : 24x24 pixels
00005 // Memory usage   : 1152 bytes
00006
00007 #if defined(__AVR__)
00008 #include <avr/pgmspace.h>
00009 #elif defined(__PIC32MX__)
00010 #define PROGMEM
00011 #elif defined(__arm__)
00012 #define PROGMEM
00013 #endif
00014
00015 const unsigned short
00016     autoBrightnessOn_Sprite[576] PROGMEM =
00017         {
00018             0x86BF, 0x8EDF, 0x96FF, 0x96FF, 0x9F1F, 0xA71F,
00019             0xAF3F, 0xB73F, 0xB75F, 0xBF7F, 0xC77F, 0xCF9F,
00020             0xC77F, 0xB73F, 0x9EFF, 0x8EDF, // 0x0010 (16) pixels
00021             0x769F, 0x665F, 0x665F, 0x665F, 0x665F, 0x665F,
00022             0x665F, 0x665F, 0x7EBF, 0x86BF, 0x8EDF, 0x96FF,
00023             0x96FF, 0x9F1F, 0xA71F, 0xAF3F, // 0x0020 (32) pixels
00024             0xB73F, 0xB75F, 0xBF7F, 0xC77F, 0xCF9F, 0xC77F,
00025             0xB73F, 0x9EFF, 0x8EDF, 0x769F, 0x665F, 0x665F,
00026             0x5D9D, 0x004E, 0x3BD8, 0x665F, // 0x0030 (48) pixels
00027             0x769F, 0x7EBF, 0x86BF, 0x8EDF, 0x96FF, 0x96FF,
00028             0x9F1F, 0xA71F, 0xAF3F, 0xB73F, 0xB75F, 0xBF7F,
00029             0xC77F, 0xCF9F, 0xC77F, 0xB73F, // 0x0040 (64) pixels
00030             0x9EFF, 0x8EDF, 0x769F, 0x3BD8, 0x000D, 0x000D,
00031             0x553C, 0x665F, 0x769F, 0x769F, 0x7EBF, 0x86BF,
00032             0x8EDF, 0x96FF, 0x96FF, 0x9F1F, // 0x0050 (80) pixels
00033             0xA71F, 0xAF3F, 0xB73F, 0xB75F, 0xBF7F, 0xC77F,
00034             0xCF9F, 0xC77F, 0xB73F, 0x9EFF, 0x4376, 0x004E,
00035             0x002D, 0x2AD5, 0x665F, 0x665F, // 0x0060 (96) pixels
00036             0x6E7F, 0x769F, 0x769F, 0x7EBF, 0x86BF, 0x8EDF,
00037             0x96FF, 0x96FF, 0x9F1F, 0xA71F, 0xAF3F, 0xB73F,
00038             0xB75F, 0xBF7F, 0xC77F, 0xCF9F, // 0x0070 (112) pixels
00039             0xC77F, 0x5BD7, 0x004E, 0x000D, 0x088E, 0x5E1F,
00040             0x665F, 0x665F, 0x665F, 0x6E7F, 0x769F, 0x769F,
00041             0x7EBF, 0x86BF, 0x8EDF, 0x96FF, // 0x0080 (128) pixels
00042             0x96FF, 0x9F1F, 0xA71F, 0xAF3F, 0xB73F, 0xB75F,
00043             0xBF7F, 0xC77F, 0x959B, 0x004E, 0x000D, 0x006E,
00044             0x5C79, 0x769F, 0x665F, 0x665F, // 0x0090 (144) pixels
00045             0x5E5F, 0x665F, 0x6E7F, 0x769F, 0x769F, 0x7EBF,
00046             0x86BF, 0x8EDF, 0x96FF, 0x96FF, 0x9F1F, 0xA71F,
00047             0xAF3F, 0xB73F, 0xB75F, 0xAE9D, // 0x00A0 (160) pixels
00048             0x086E, 0x000D, 0x000D, 0x2170, 0x9EFF, 0x8EDF,
00049             0x769F, 0x665F, 0x563F, 0x5E5F, 0x665F, 0x6E7F,
00050             0x769F, 0x769F, 0x7EBF, 0x86BF, // 0x00B0 (176) pixels
00051             0x8EDF, 0x96FF, 0x96FF, 0x9F1F, 0xA71F, 0xAF3F,
00052             0xAF1F, 0x1950, 0x002D, 0x000D, 0x004E, 0x9E1C,
00053             0xB73F, 0x9EFF, 0x8EDF, 0x769F, // 0x00C0 (192) pixels
00054             0x563F, 0x563F, 0x5E5F, 0x665F, 0x6E7F, 0x769F,
00055             0x769F, 0x7EBF, 0x86BF, 0x8EDF, 0x96FF, 0x96FF,
00056             0x9F1F, 0xA71F, 0x3AB4, 0x004E, // 0x00D0 (208) pixels
00057             0x000D, 0x004D, 0x4AD4, 0xCF9F, 0xC77F, 0xB73F,
00058             0x9EFF, 0x8EDF, 0x4E1F, 0x563F, 0x563F, 0x5E5F,
00059             0x665F, 0x6E7F, 0x769F, 0x769F, // 0x00E0 (224) pixels
00060             0x7EBF, 0x86BF, 0x8EDF, 0x96FF, 0x96FF, 0x751A,
```

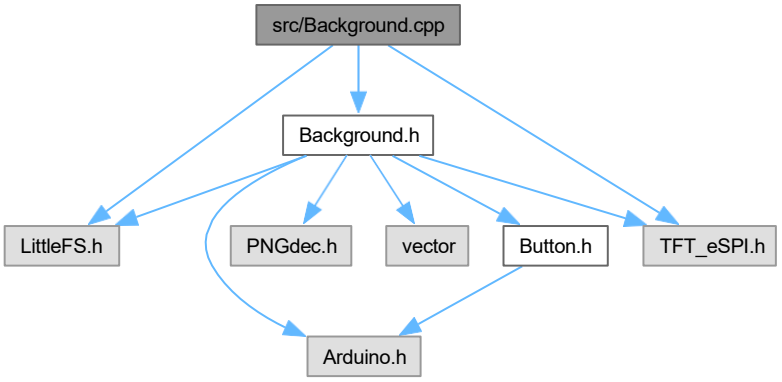
```
00061             0x004E, 0x000D, 0x000D, 0x086E, 0xAEDE, 0xC77F,
00062             0xCF9F, 0xC77F, 0xB73F, 0x9EFF, // 0x00F0 (240) pixels
00063             0x45FF, 0x4E1F, 0x563F, 0x563F, 0x5E5F, 0x665F,
00064             0x6E7F, 0x769F, 0x769F, 0x7EBF, 0x86BF, 0x8EDF,
00065             0x8EBF, 0x08AF, 0x000D, 0x000D, // 0x0100 (256) pixels
00066             0x006E, 0x6417, 0xB75F, 0xBF7F, 0xC77F, 0xCF9F,
00067             0xC77F, 0xB73F, 0x3DFF, 0x45FF, 0x4E1F, 0x563F,
00068             0x563F, 0x5E5F, 0x665F, 0x6E7F, // 0x0110 (272) pixels
00069             0x769F, 0x769F, 0x7EBF, 0x86BF, 0x4356, 0x004E,
00070             0x000D, 0x000D, 0x088E, 0xA6FF, 0xB73F, 0xB75F,
00071             0xBF7F, 0xC77F, 0xCF9F, 0xC77F, // 0x0120 (288) pixels
00072             0x35DF, 0x3DFF, 0x3D5D, 0x2BD9, 0x4DBE, 0x563F,
00073             0x5E5F, 0x665F, 0x6E7F, 0x769F, 0x769F, 0x6DDD,
00074             0x004E, 0x000D, 0x000D, 0x004E, // 0x0130 (304) pixels
00075             0x53D7, 0xA71F, 0xAF3F, 0xB73F, 0xB75F, 0xBF7F,
00076             0xC77F, 0xCF9F, 0x35DF, 0x35DF, 0x351D, 0x004E,
00077             0x000D, 0x1151, 0x33B8, 0x55BE, // 0x0140 (320) pixels
00078             0x665F, 0x6E7F, 0x767F, 0x19B2, 0x002D, 0x000D,
00079             0x000D, 0x006E, 0x8E9E, 0x9F1F, 0xA71F, 0xAF3F,
00080             0xB73F, 0xB75F, 0xBF7F, 0xC77F, // 0x0150 (336) pixels
00081             0x2DBF, 0x35DF, 0x35DF, 0x2C9B, 0x002D, 0x000D,
00082             0x004E, 0x002D, 0x1130, 0x3397, 0x43D8, 0x004E,
00083             0x000D, 0x000D, 0x004E, 0x4356, // 0x0160 (352) pixels
00084             0x96FF, 0x96FF, 0x9F1F, 0xA71F, 0xAF3F, 0xB73F,
00085             0xB75F, 0xBF7F, 0x25BF, 0x2DBF, 0x35DF, 0x35DF,
00086             0x2C3A, 0x004E, 0x000D, 0x000D, // 0x0170 (368) pixels
00087             0x000D, 0x004E, 0x004E, 0x000D, 0x000D, 0x000D,
00088             0x004E, 0x7E3E, 0x8EDF, 0x96FF, 0x96FF, 0x9F1F,
00089             0xA71F, 0xAF3F, 0xB73F, 0xB75F, // 0x0180 (384) pixels
00090             0x1D9F, 0x25BF, 0x2DBF, 0x35DF, 0x35DF, 0x23D9,
00091             0x004E, 0x000D, 0x000D, 0x000D, 0x000D, 0x000D,
00092             0x000D, 0x002D, 0x32D5, 0x7EBF, // 0x0190 (400) pixels
00093             0x86BF, 0x8EDF, 0x96FF, 0x96FF, 0x9F1F, 0xA71F,
00094             0xAF3F, 0xB73F, 0x157F, 0x1D9F, 0x25BF, 0x2DBF,
00095             0x35DF, 0x35DF, 0x2357, 0x004E, // 0x01A0 (416) pixels
00096             0x000D, 0x000D, 0x000D, 0x000D, 0x000D, 0x004D,
00097             0x65BD, 0x769F, 0x7EBF, 0x86BF, 0x8EDF, 0x96FF,
00098             0x96FF, 0x9F1F, 0xA71F, 0xAF3F, // 0x01B0 (432) pixels
00099             0x157F, 0x157F, 0x1D9F, 0x25BF, 0x2DBF, 0x35DF,
00100             0x35DF, 0x1AD6, 0x004E, 0x000D, 0x000D, 0x000D,
00101             0x000D, 0x08EF, 0x6E7F, 0x769F, // 0x01C0 (448) pixels
00102             0x769F, 0x7EBF, 0x86BF, 0x8EDF, 0x96FF, 0x96FF,
00103             0x9F1F, 0xA71F, 0x157F, 0x157F, 0x157F, 0x1D9F,
00104             0x25BF, 0x2DBF, 0x35DF, 0x35DF, // 0x01D0 (464) pixels
00105             0x1A95, 0x004E, 0x000D, 0x000D, 0x002D, 0x2B16,
00106             0x665F, 0x6E7F, 0x769F, 0x769F, 0x7EBF, 0x86BF,
00107             0x8EDF, 0x96FF, 0x96FF, 0x9F1F, // 0x01E0 (480) pixels
00108             0x157F, 0x157F, 0x157F, 0x157F, 0x1D9F, 0x25BF,
00109             0x2DBF, 0x35DF, 0x35DF, 0x1AF6, 0x004E, 0x000D,
00110             0x006E, 0x3C19, 0x5E5F, 0x665F, // 0x01F0 (496) pixels
00111             0x6E7F, 0x769F, 0x769F, 0x7EBF, 0x86BF, 0x8EDF,
00112             0x96FF, 0x96FF, 0x157F, 0x157F, 0x157F, 0x157F,
00113             0x157F, 0x1D9F, 0x25BF, 0x2DBF, // 0x0200 (512) pixels
00114             0x35DF, 0x35DF, 0x2357, 0x004E, 0x002D, 0x3419,
00115             0x563F, 0x5E5F, 0x665F, 0x6E7F, 0x769F, 0x769F,
00116             0x7EBF, 0x86BF, 0x8EDF, 0x96FF, // 0x0210 (528) pixels
00117             0x157F, 0x157F, 0x157F, 0x157F, 0x157F, 0x157F,
00118             0x1D9F, 0x25BF, 0x2DBF, 0x35DF, 0x35DF, 0x23D9,
00119             0x000D, 0x347A, 0x563F, 0x563F, // 0x0220 (544) pixels
00120             0x5E5F, 0x665F, 0x6E7F, 0x769F, 0x769F, 0x7EBF,
00121             0x86BF, 0x8EDF, 0x157F, 0x157F, 0x157F, 0x157F,
00122             0x157F, 0x157F, 0x157F, 0x1D9F, // 0x0230 (560) pixels
00123             0x25BF, 0x2DBF, 0x35DF, 0x35DF, 0x3DFF, 0x45FF,
00124             0x4E1F, 0x563F, 0x563F, 0x5E5F, 0x665F, 0x6E7F,
00125             0x769F, 0x769F, 0x7EBF, 0x86BF, // 0x0240 (576) pixels
00126 };
```

4.5 src/Background.cpp File Reference

Implementation of the Background class.

```
#include "Background.h"
#include <LittleFS.h>
#include <TFT_eSPI.h>
```

Include dependency graph for Background.cpp:



4.5.1 Detailed Description

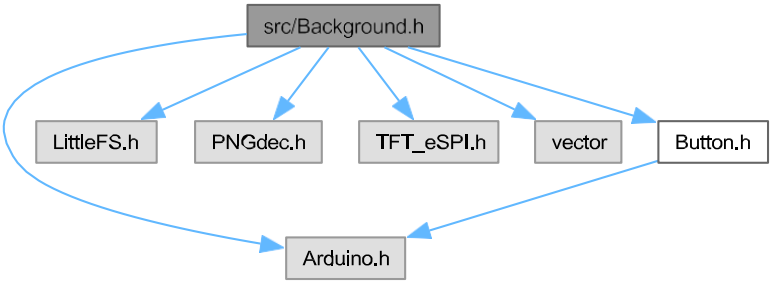
Implementation of the Background class.

4.6 src/Background.h File Reference

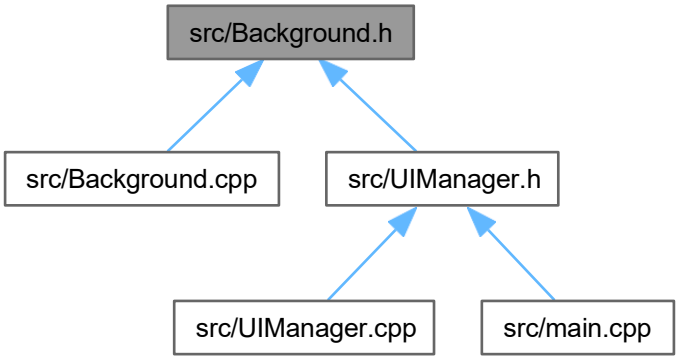
Manages full-screen background images and interactive buttons.

```
#include <Arduino.h>
#include <LittleFS.h>
#include <PNGdec.h>
#include <TFT_eSPI.h>
#include <vector>
#include "Button.h"
```

Include dependency graph for Background.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Background

  Handles loading background PNGs and managing associated touch buttons.

4.6.1 Detailed Description

Manages full-screen background images and interactive buttons.

4.7 Background.h

Go to the documentation of this file.

```
00001
00006 #pragma once
00007
00008 #include <Arduino.h>
00009 #include <LittleFS.h>
00010 #include <PNGdec.h>
00011 #include <TFT_eSPI.h>
00012 #include <vector>
00013
00014 #include "Button.h"
00015
00020 class Background {
00021 public:
00026   Background(const String &path = "");
00027
00032   void setPath(const String &path);
00033
00038   String path() const;
00039
00048   bool draw(TFT_eSPI &tft, PNG &png, bool center = true);
00049
00054   static void listFS(const char *dir);
00055
00060   void addButton(const Button &btn);
00061
00065   void clearButtons();
00066
00079   bool handleTouch(int16_t touchX, int16_t touchY, bool isPressedNow);
```

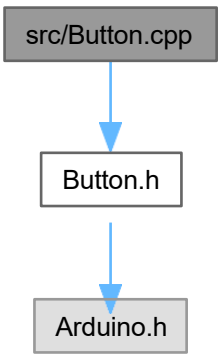
```
00080
00081 private:
00082   String _path;
00083   std::vector<Button>
00084       _buttons;
00085
00086   // --- Static members for PNGdec callbacks ---
00087   // These must be static because PNGdec requires C-style function pointers.
00088
00089   static int s_offX;
00090   static int s_offY;
00091   static uint16_t s_lineBuf[480];
00092   static File s_pngFile;
00093   static TFT_eSPI *s_tft;
00094   static PNG *s_png;
00095
00096   // --- PNGdec Callback Wrappers ---
00097   static void *pngOpen(const char *filename, int32_t *pFileSize);
00098   static void pngClose(void *handle);
00099   static int32_t pngRead(PNGFILE *pFile, uint8_t *pBuff, int32_t iLen);
00100   static int32_t pngSeek(PNGFILE *pFile, int32_t iPosition);
00101   static int pngDraw(PNGDRAW *p);
00102
00109   bool drawPngFullScreen(const char *path, bool center);
00110 };
```

4.8 src/Button.cpp File Reference

Implementation of the Button class.

```
#include "Button.h"
```

Include dependency graph for Button.cpp:



4.8.1 Detailed Description

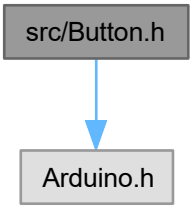
Implementation of the Button class.

4.9 src/Button.h File Reference

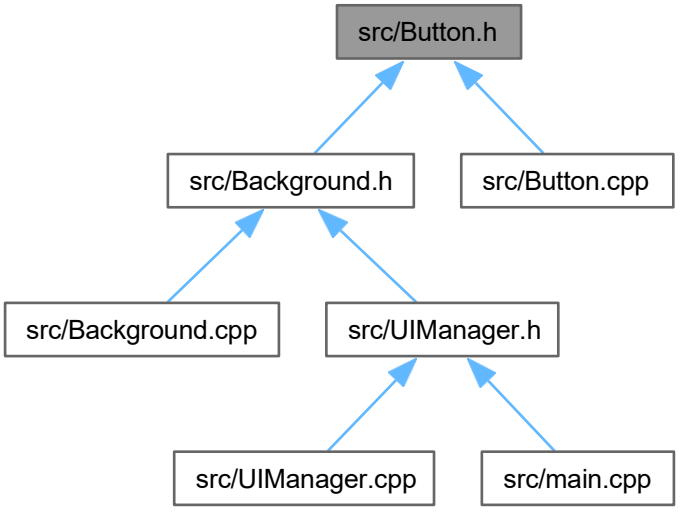
Defines a touch-interactive button area.

```
#include <Arduino.h>
```

Include dependency graph for Button.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Button

  Represents a rectangular touch-sensitive area with a callback.

Typedefs

- using ButtonCallback = void (∗)(uint8_t id, int16_t x, int16_t y)

  Callback function type for button events.

4.9.1 Detailed Description

Defines a touch-interactive button area.

4.9.2 Typedef Documentation

4.9.2.1 ButtonCallback

```
using ButtonCallback = void (*)(uint8_t id, int16_t x, int16_t y)
```

Callback function type for button events.

Parameters

		Button ID.
	x	X-coordinate of the touch.
	y	Y-coordinate of the touch.

4.10 Button.h

Go to the documentation of this file.

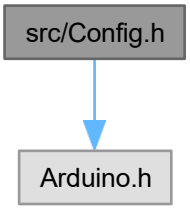
```
00001
00006 #pragma once
00007 #include <Arduino.h>
00008
00015 using ButtonCallback = void (*)(uint8_t id, int16_t x, int16_t y);
00016
00021 class Button {
00022 public:
00032   Button(uint8_t id, int16_t x, int16_t y, int16_t w, int16_t h,
00033          ButtonCallback cb = nullptr);
00034
00035   uint8_t id() const { return _id; }
00036   void setCallback(ButtonCallback cb) { _cb = cb; }
00037
00041   bool contains(int16_t x, int16_t y) const;
00042
00052   bool updateTouch(int16_t touchX, int16_t touchY, bool isPressedNow);
00053
00054 private:
00055   uint8_t _id;
00056   int16_t _x, _y, _w, _h;
00057   ButtonCallback _cb;
00058   bool _pressedInside;
00059   int16_t _lastTouchX;
00060   int16_t _lastTouchY;
00061 };
```

4.11 src/Config.h File Reference

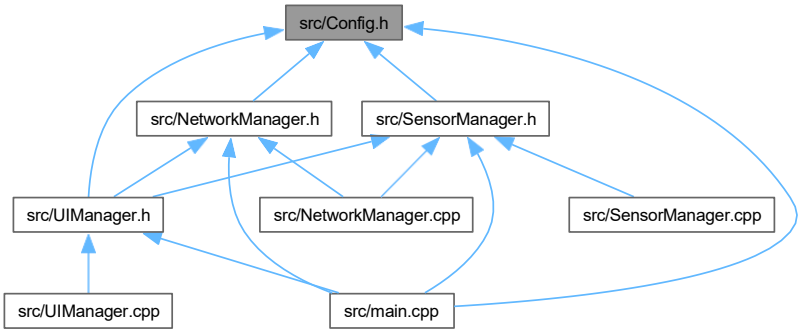
System-wide configuration constants and settings.

```
#include <Arduino.h>
```

Include dependency graph for Config.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define FOTORESISTOR_PIN 32

  Analog input pin for photoresistor.
- #define ONE_WIRE_BUS 33

  Pin for OneWire (DS18B20)
- #define I2C_SDA 25

  I2C SDA Pin.
- #define I2C_SCL 26

  I2C SCL Pin.
- #define TFT_LED_PIN 2

  TFT backlight control pin.
- #define EXTRA_SMALL_FONT_NAME "fonts/Lato-Regular-18"

- #define SMALL_FONT_NAME "fonts/Lato-Regular-24"
- #define TIME_FONT_NAME "fonts/Lato-Regular-92"
- #define MEDIUM_BOLD_FONT_NAME "fonts/Lato-Semibold-32"
- #define BG_HOME_PATH "/images/main_screen-min.png"
- #define BG_SETTINGS_PATH "/images/settings_screen-min.png"
- #define BG_ACCOUNT_PATH "/images/app-connecting-screen-min.png"

Enumerations

- enum SCREEN { HOME_SCREEN , SETTINGS_SCREEN , APP_CONNECTION_SCREEN , WIFI_CONNECTION_SCREEN }

  Enumeration of available UI screens.

Variables

- const char *const AWS_ENDPOINT
- const int AWS_PORT = 8883
- const char *const CLIENT_ID = "station-001"
- const char *const THING_NAME = "station-001"

4.11.1 Detailed Description

System-wide configuration constants and settings.

4.11.2 Macro Definition Documentation

4.11.2.1 BG_ACCOUNT_PATH

```
#define BG_ACCOUNT_PATH "/images/app-connecting-screen-min.png"
```

4.11.2.2 BG_HOME_PATH

```
#define BG_HOME_PATH "/images/main_screen-min.png"
```

4.11.2.3 BG_SETTINGS_PATH

```
#define BG_SETTINGS_PATH "/images/settings_screen-min.png"
```

4.11.2.4 EXTRA_SMALL_FONT_NAME

```
#define EXTRA_SMALL_FONT_NAME "fonts/Lato-Regular-18"
```

4.11.2.5 FOTORESISTOR_PIN

```
#define FOTORESISTOR_PIN 32
```

Analog input pin for photoresistor.

4.11.2.6 I2C_SCL

```
#define I2C_SCL 26
```

I2C SCL Pin.

4.11.2.7 I2C_SDA

```
#define I2C_SDA 25
```

I2C SDA Pin.

4.11.2.8 MEDIUM_BOLD_FONT_NAME

```
#define MEDIUM_BOLD_FONT_NAME "fonts/Lato-Semibold-32"
```

4.11.2.9 ONE_WIRE_BUS

```
#define ONE_WIRE_BUS 33
```

Pin for OneWire (DS18B20)

4.11.2.10 SMALL_FONT_NAME

```
#define SMALL_FONT_NAME "fonts/Lato-Regular-24"
```

4.11.2.11 TFT_LED_PIN

```
#define TFT_LED_PIN 2
```

TFT backlight control pin.

4.11.2.12 TIME_FONT_NAME

```
#define TIME_FONT_NAME "fonts/Lato-Regular-92"
```

4.11.3 Enumeration Type Documentation

4.11.3.1 SCREEN

```
enum SCREEN
```

Enumeration of available UI screens.

Enumerator

HOME_SCREEN	
SETTINGS_SCREEN	
APP_CONNECTION_SCREEN	
WIFI_CONNECTION_SCREEN	

4.11.4 Variable Documentation

4.11.4.1 AWS_ENDPOINT

```
const char* const AWS_ENDPOINT
```

Initial value:

```
=
    "an7hi8lzvqru3-ats.iot.eu-north-1.amazonaws.com"
```

4.11.4.2 AWS_PORT

```
const int AWS_PORT = 8883
```

4.11.4.3 CLIENT_ID

```
const char* const CLIENT_ID = "station-001"
```

4.11.4.4 THING_NAME

```
const char* const THING_NAME = "station-001"
```

4.12 Config.h

Go to the documentation of this file.

```
00001
00006 #pragma once
00007 #include <Arduino.h>
00008
00009 // --- Hardware Pins ---
00010 #define FOTORESISTOR_PIN 32
00011 #define ONE_WIRE_BUS 33
00012 #define I2C_SDA 25
00013 #define I2C_SCL 26
00014 #define TFT_LED_PIN 2
00015
00016 // --- Fonts ---
00017 #define EXTRA_SMALL_FONT_NAME "fonts/Lato-Regular-18"
00018 #define SMALL_FONT_NAME "fonts/Lato-Regular-24"
00019 #define TIME_FONT_NAME "fonts/Lato-Regular-92"
00020 #define MEDIUM_BOLD_FONT_NAME "fonts/Lato-Semibold-32"
00021
00022 // --- Background Images ---
00023 #define BG_HOME_PATH "/images/main_screen-min.png"
00024 #define BG_SETTINGS_PATH "/images/settings_screen-min.png"
00025 #define BG_ACCOUNT_PATH "/images/app-connecting-screen-min.png"
00026
00027 // --- AWS IoT Config ---
```

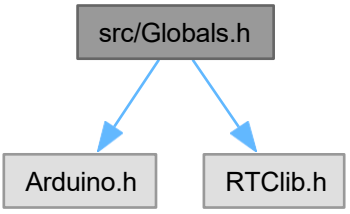
```
00028 const char *const AWS_ENDPOINT =
00029     "an7hi8lzvqru3-ats.iot.eu-north-1.amazonaws.com";
00030 const int AWS_PORT = 8883;
00031 const char *const CLIENT_ID = "station-001";
00032 const char *const THING_NAME = "station-001";
00033
00034 // --- Application States ---
00039 typedef enum {
00040   HOME_SCREEN,
00041   SETTINGS_SCREEN,
00042   APP_CONNECTION_SCREEN,
00043   WIFI_CONNECTION_SCREEN
00044 } SCREEN;
```

4.13 src/Globals.h File Reference

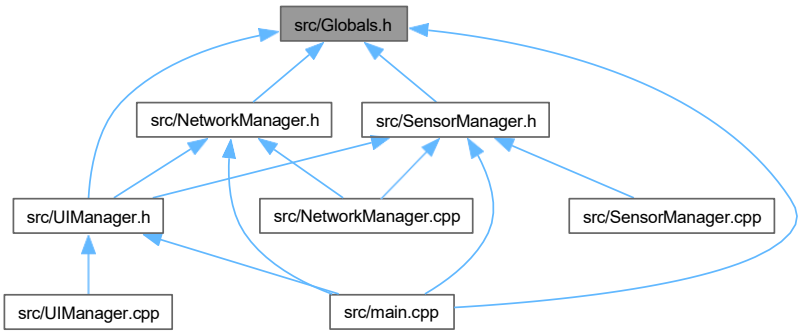
Declaration of global variables and shared data structures.

```
#include <Arduino.h>
#include <RTClib.h>
```

Include dependency graph for Globals.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct struct_message

  Data structure for ESP-NOW sensor data.

Typedefs

- typedef struct struct_message struct_message

Variables

- struct_message telemetryData
  Latest sensor data from ESP-NOW.
- float homeTemperatureRead
  Latest local indoor temperature.
- volatile bool newDataReceived
  Flag indicating new ESP-NOW data arrived.
- volatile bool screenDataDirty
  Flag indicating UI needs an update.
- volatile uint32_t lastDataReceivedMs
  Timestamp of last data reception.
- bool connectionGood
  WiFi/AWS connection status flag.
- bool autoBrightness
  Auto-brightness mode status.
- String ownerIdentityId
  Cloud user identity ID for ownership.
- String AppConnectionKey
  Generated claiming nonce for app pairing.
- RTC_DS3231 rtc
  RTC instance.
- DateTime now
  Current system time (updated in loop).

4.13.1 Detailed Description

Declaration of global variables and shared data structures.

4.13.2 Typedef Documentation

4.13.2.1 struct_message

```
typedef struct struct_message struct_message
```

4.13.3 Variable Documentation

4.13.3.1 AppConnectionKey

```
String AppConnectionKey  [extern]
```

Generated claiming nonce for app pairing.

4.13.3.2 autoBrightness

```
bool autoBrightness [extern]
```

Auto-brightness mode status.

4.13.3.3 connectionGood

```
bool connectionGood [extern]
```

WiFi/AWS connection status flag.

4.13.3.4 telemetryData

```
struct_message telemetryData
```

[extern] Latest sensor data from

ESP-NOW. Latest sensor data from

ESP-NOW.

4.13.3.5 homeTemperatureRead

```
float homeTemperatureRead [extern]
```

Latest local indoor temperature.

Latest local indoor temperature.

4.13.3.6 lastDataReceivedMs

```
volatile uint32_t lastDataReceivedMs [extern]
```

Timestamp of last data reception.

Timestamp of last data reception.

4.13.3.7 newDataReceived

```
volatile bool newDataReceived [extern]
```

Flag indicating new ESP-NOW data arrived.

4.13.3.8 now

```
DateTime now [extern]
```

Current system time (updated in loop).

4.13.3.9 ownerIdentityId

```
String ownerIdentityId  [extern]
```

Cloud user identity ID for ownership.

4.13.3.10 rtc

```
RTC_DS3231 rtc  [extern]
```

RTC instance.

4.13.3.11 screenDataDirty

```
volatile bool screenDataDirty  [extern]
```

Flag indicating UI needs an update.

4.14 Globals.h

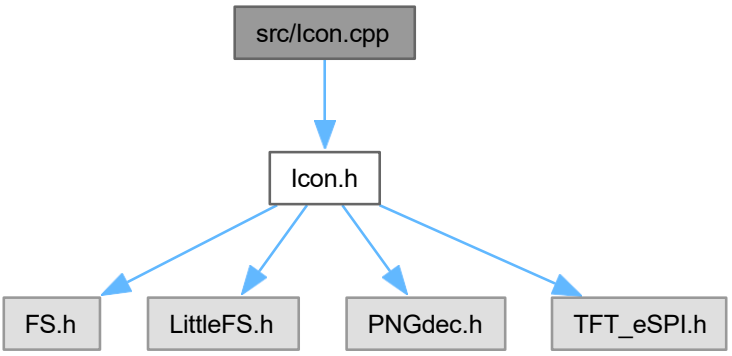
Go to the documentation of this file.

```
00001
00006 #pragma once
00007 #include <Arduino.h>
00008 #include <RTClib.h>
00009
00014 typedef struct struct_message {
00015   uint8_t humidityRead;
00016   int16_t
00017       outdoorTemperatureRead;
00018   uint16_t pressureRead;
00019   uint8_t uvIndexRead;
00020 } struct_message;
00021
00022 // --- Global Variables ---
00023 extern struct_message telemetry
Data;
 00024 extern float
homeTemperatureRead; 00025 extern
volatile bool
00026     newDataReceived;
00027 extern volatile bool screenDataDirty;
00028 extern volatile uint32_t
00029     lastDataReceivedMs;
00030
00031 extern bool connectionGood;
00032 extern bool autoBrightness;
00033 extern String ownerIdentityId;
00034 extern String AppConnectionKey;
00035
00036 extern RTC_DS3231 rtc;
00037 extern DateTime now;
```

4.15 src/Icon.cpp File Reference

```
#include "Icon.h"
```

Include dependency graph for Icon.cpp:



Variables

- PNG png

  Global PNG decoder instance.

4.15.1 Variable Documentation

4.15.1.1 png

```
PNG png  [extern]
```

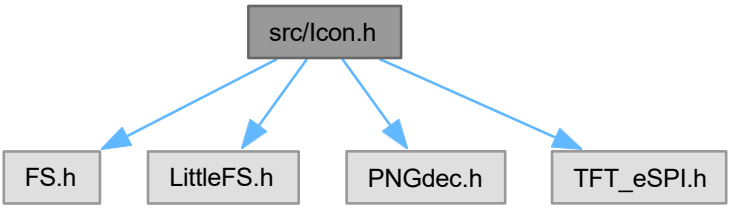
Global PNG decoder instance.

4.16 src/Icon.h File Reference

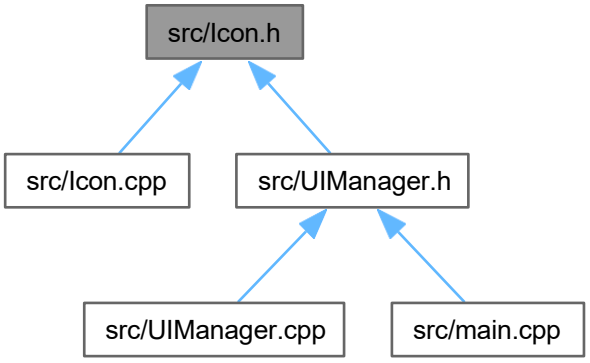
Handles loading and drawing PNG icons from LittleFS to TFT sprites.

```
#include <FS.h>
#include <LittleFS.h>
#include <PNGdec.h>
```

`#include <TFT_eSPI.h>`
Include dependency graph for Icon.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Icon

  Represents a graphical icon loaded from the filesystem.

Variables

- PNG png

  Global PNG decoder instance.

4.16.1 Detailed Description

Handles loading and drawing PNG icons from LittleFS to TFT sprites.

This class uses the PNGdec library to decode PNG images and render them onto a TFT_eSprite for fast, flicker-free drawing with transparency support.

4.16.2 Variable Documentation

4.16.2.1 png

```
PNG png  [extern]
```

Global PNG decoder instance.

4.17 Icon.h

Go to the documentation of this file.

```
00001
00009 #pragma once
00010 #include <FS.h>
00011 #include <LittleFS.h>
00012 #include <PNGdec.h>
00013 #include <TFT_eSPI.h>
00014
00015 extern PNG png;
00016
00021 class Icon {
00022 public:
00032   Icon(TFT_eSPI *tft, const char *path, uint8_t trR, uint8_t trG, uint8_t trB);
00033
00038   bool loadFromFS();
00039
00047   void draw(int16_t x, int16_t y, uint16_t bgColor = TFT_BLACK);
00048
00052   void unload();
00053
00058   int16_t width() const { return _w; }
00059
00064   int16_t height() const { return _h; }
00065
00066 private:
00067   static Icon *_active;
00068
00069   // PNGdec callbacks
00070   static void *_pngOpen(const char *filename, int32_t *size);
00071   static void _pngClose(void *handle);
00072   static int32_t _pngRead(PNGFILE *file, uint8_t *buf, int32_t len);
00073   static int32_t _pngSeek(PNGFILE *file, int32_t pos);
00074   static int _pngDrawToSprite(PNGDRAW *pDraw);
00075
00076   static uint16_t rgb888To565(uint8_t r, uint8_t g, uint8_t b);
00077
00078   TFT_eSPI *_tft;
00079   TFT_eSprite _sprite;
00080   String _path;
00081   uint16_t _transparent565;
00082   bool _loaded;
00083   int16_t _w;
00084   int16_t _h;
00085 };
```

4.18 src/main.cpp File Reference

Entry point of the application. Handles initialization and the main loop.

```
#include <Arduino.h>
#include <LittleFS.h>
#include <PNGdec.h>
#include "Config.h"
#include "Globals.h"
#include "NetworkManager.h"
#include "SensorManager.h"
```

```
#include "UIManager.h"
```

Include dependency graph for main.cpp:

Functions

- void setup ()

  Setup function.
- void loop ()

  Main Loop.

Variables

- struct_message telemetryData

  ESP-NOW data buffer.
- float homeTemperatureRead = 0.0

  Local temperature.
- volatile bool newDataReceived = false

  Flag indicating new ESP-NOW data arrived.
- volatile bool screenDataDirty = false

  Flag indicating UI needs an update.
- bool connectionGood = false

  WiFi/AWS connection status flag.
- bool autoBrightness = false

  Auto-brightness mode status.
- String ownerIdentityId = ""

  Cloud user identity ID for ownership.
- String AppConnectionKey = ""

  Generated claiming nonce for app pairing.
- RTC_DS3231 rtc

  RTC instance.
- DateTime now

  Current system time (updated in loop).
- volatile uint32_t lastDataReceivedMs = 0

  Timeout tracker.
- PNG png

  Global PNG decoder instance.
- SensorManager * sensorMgr = nullptr
- NetworkManager * netMgr = nullptr
- UIManager * uiMgr = nullptr

4.18.1 Detailed Description

Entry point of the application. Handles initialization and the main loop.

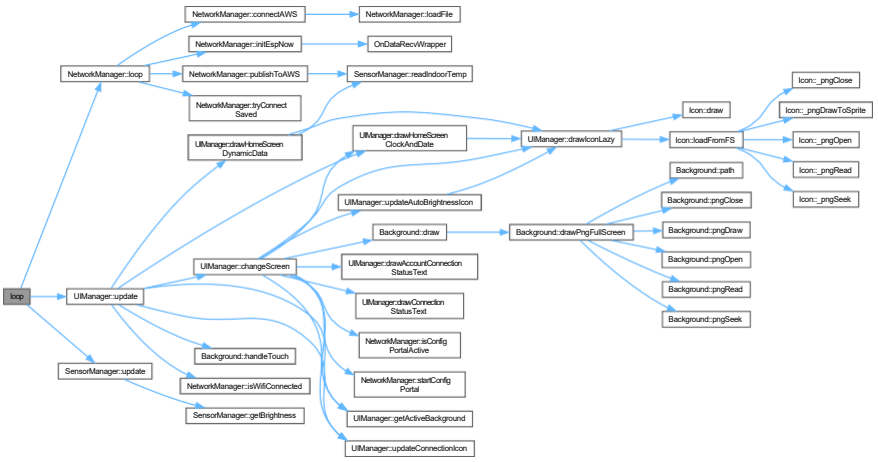
4.18.2 Function Documentation

4.18.2.1 loop()

```
void loop ()
```

Main Loop.

Here is the call graph for this function:

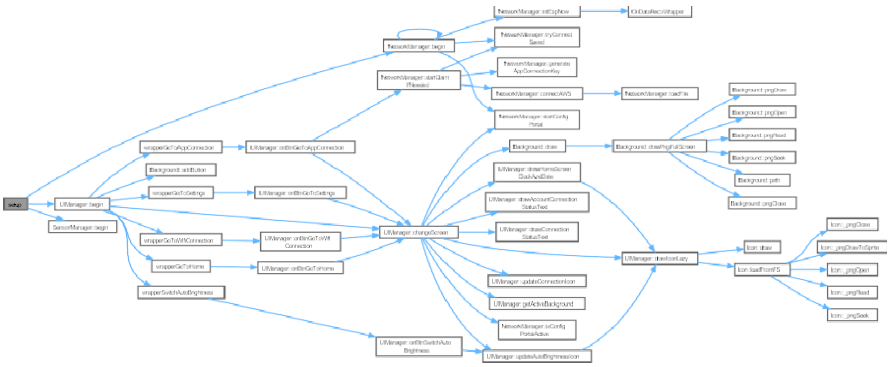


4.18.2.2 setup()

```
void setup ()
```

Setup function.

Here is the call graph for this function:



4.18.3 Variable Documentation

4.18.3.1 AppConnectionKey

```
String AppConnectionKey = ""
```

Generated claiming nonce for app pairing.

4.18.3.2 autoBrightness

```
bool autoBrightness = false
```

Auto-brightness mode status.

4.18.3.3 connectionGood

```
bool connectionGood = false
```

WiFi/AWS connection status flag.

4.18.3.4 telemetryData

```
struct_message Data
```

ESP-NOW data buffer.

Latest sensor data from ESP-NOW.

4.18.3.5 homeTemperatureRead

```
float homeTemperatureRead = 0.0
```

Local temperature.

Latest local indoor temperature.

4.18.3.6 lastDataReceivedMs

```
volatile uint32_t lastDataReceivedMs = 0
```

Timeout tracker.

Timestamp of last data reception.

4.18.3.7 netMgr

```
NetworkManager* netMgr = nullptr
```

4.18.3.8 newDataReceived

```
volatile bool newDataReceived = false
```

Flag indicating new ESP-NOW data arrived.

4.18.3.9 now

```
DateTime now
```

Current system time (updated in loop).

4.18.3.10 ownerIdentityId

```
String ownerIdentityId = ""
```

Cloud user identity ID for ownership.

4.18.3.11 png

```
PNG png
```

Global PNG decoder instance.

4.18.3.12 rtc

```
RTC_DS3231 rtc
```

RTC instance.

4.18.3.13 screenDataDirty

```
volatile bool screenDataDirty = false
```

Flag indicating UI needs an update.

4.18.3.14 sensorMgr

```
SensorManager* sensorMgr = nullptr
```

4.18.3.15 uiMgr

```
UIManager* uiMgr = nullptr
```

4.19 src/NetworkManager.cpp File Reference

Implementation of the NetworkManager class.

```
#include "NetworkManager.h"
#include "SensorManager.h"
```

Include dependency graph for NetworkManager.cpp:

Functions

- void OnDataRecvWrapper (const uint8_t *mac, const uint8_t *incomingData, int len)
- void mqttCallbackWrapper (char *topic, byte *payload, unsigned int len)

4.19.1 Detailed Description

Implementation of the NetworkManager class.

4.19.2 Function Documentation

4.19.2.1 mqttCallbackWrapper()

```
void mqttCallbackWrapper (
            char * topic,
            byte * payload,
            unsigned int len)
```

4.19.2.2 OnDataRecvWrapper()

```
void OnDataRecvWrapper (
            const uint8_t * mac,
            const uint8_t * incomingData,
            int len)
```

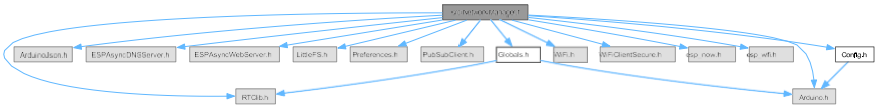
Here is the caller graph for this function:

4.20 src/NetworkManager.h File Reference

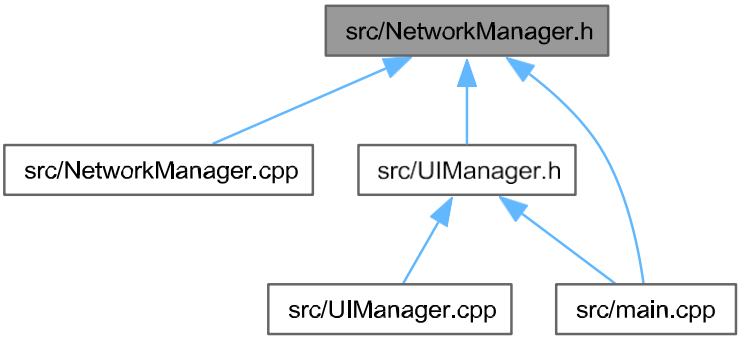
Manages WiFi, ESP-NOW, AWS IoT, and provisioning.

```
#include <Arduino.h>
#include <ArduinoJson.h>
#include <ESPAsyncDNSServer.h>
#include <ESPAsyncWebServer.h>
#include <LittleFS.h>
#include <Preferences.h>
#include <PubSubClient.h>
#include <RTClib.h>
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <esp_now.h>
#include <esp_wifi.h>
#include "Config.h"
#include "Globals.h"
```

Include dependency graph for NetworkManager.h:



This graph shows which files directly or indirectly include this file:



Classes

- class NetworkManager

  Handles all network-related operations.

4.20.1 Detailed Description

Manages WiFi, ESP-NOW, AWS IoT, and provisioning.

4.21 NetworkManager.h

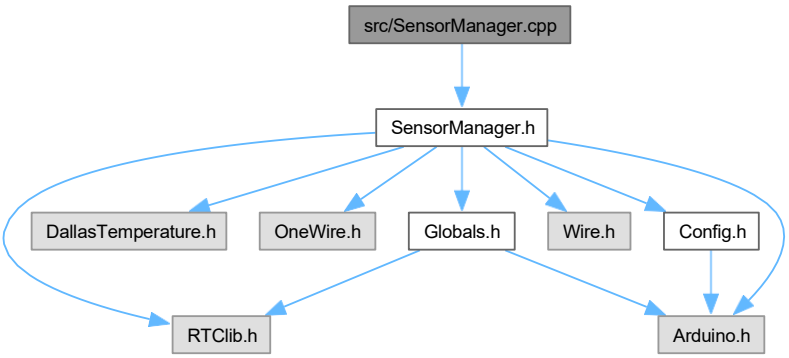
Go to the documentation of this file.

```
00001
00006 #pragma once
00007 #include <Arduino.h>
00008 #include <ArduinoJson.h>
00009 #include <ESPAsyncDNSServer.h>
00010 #include <ESPAsyncWebServer.h>
00011 #include <LittleFS.h>
00012 #include <Preferences.h>
00013 #include <PubSubClient.h>
00014 #include <RTClib.h>
00015 #include <WiFi.h>
00016 #include <WiFiClientSecure.h>
00017 #include <esp_now.h>
00018 #include <esp_wifi.h>
00019
00020 #include "Config.h"
00021 #include "Globals.h"
00022
00023 class SensorManager;
00024
00029 class NetworkManager {
00030 public:
00035   NetworkManager(SensorManager *sensorMgr);
00036
00040   void begin();
00041
00045   void loop();
00046
00052   bool tryConnectSaved(unsigned timeoutMs = 3000);
00053
00057   void startConfigPortal();
00058
00062   void startClaimIfNeeded();
00063
00068   bool initEspNow();
00069
00070   bool isConfigPortalActive();
00071   bool isWifiConnected();
00072   bool isAwsConnected();
00073
00074 private:
00075   SensorManager *_sensorMgr;
00076   WiFiClientSecure net;
00077   PubSubClient client;
00078   AsyncWebServer server;
00079   AsyncDNSServer dns;
00080   Preferences prefs;
00081
00082   bool _configPortalActive = false;
00083   bool _sendingToAws = false;
00084   bool appConnectionKeyReady = false;
00085
00086   bool connectAWS();
00087   void publishToAWS();
00088   void generateAppConnectionKey();
00089   String loadFile(const char *path);
00090   void handleMqttMessage(char *topic, byte *payload, unsigned int len);
00091
00092   friend void mqttCallbackWrapper(char *topic, byte *payload, unsigned int len);
00093 };
```

4.22 src/SensorManager.cpp File Reference

Implementation of the SensorManager class.

```
#include "SensorManager.h"
```
Include dependency graph for SensorManager.cpp:



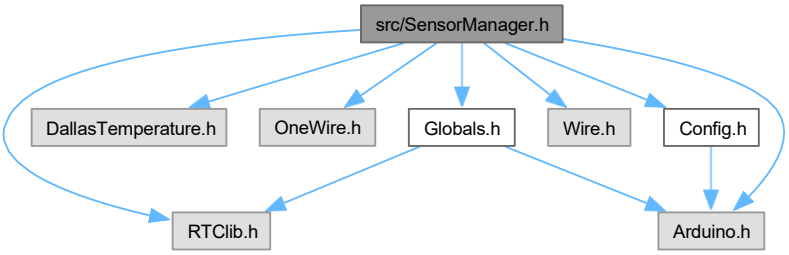
4.22.1 Detailed Description

Implementation of the SensorManager class.

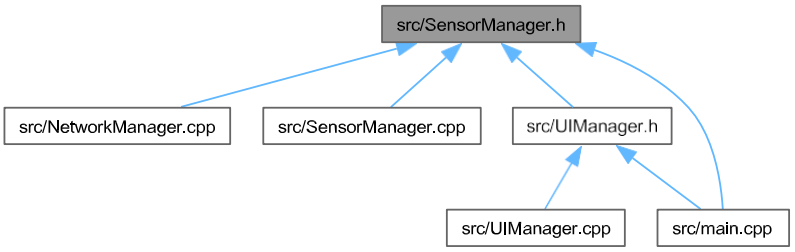
4.23 src/SensorManager.h File Reference

Manages sensors (DS18B20, RTC) and actuators (Display Brightness).

```
#include <Arduino.h>
#include <DallasTemperature.h>
#include <OneWire.h>
#include <RTClib.h>
#include <Wire.h>
#include "Config.h"
#include "Globals.h"
```
Include dependency graph for SensorManager.h:



This graph shows which files directly or indirectly include this file:



Classes

- class SensorManager

  Handles reading sensors and managing hardware state.

4.23.1 Detailed Description

Manages sensors (DS18B20, RTC) and actuators (Display Brightness).

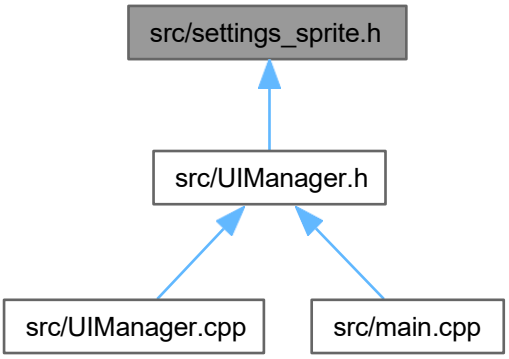
4.24 SensorManager.h

Go to the documentation of this file.

```
00001
00006 #pragma once
00007
00008 #include <Arduino.h>
00009 #include <DallasTemperature.h>
00010 #include <OneWire.h>
00011 #include <RTClib.h>
00012 #include <Wire.h>
00013
00014 #include "Config.h"
00015 #include "Globals.h"
00016
00021 class SensorManager {
00022 public:
00023   SensorManager();
00024
00028   void begin();
00029
00033   void update();
00034
00039   float readIndoorTemp();
00040
00041 private:
00042   OneWire oneWire;
00043   DallasTemperature sensors;
00044
00045   int getBrightness();
00046 };
```

4.25 src/settings_sprite.h File Reference

This graph shows which files directly or indirectly include this file:



Variables

- const unsigned short settings_sprite[900] PROGMEM

4.25.1 Variable Documentation

4.25.1.1 PROGMEM

```
const unsigned short settings_sprite [900] PROGMEM
```

4.26 settings_sprite.h

Go to the documentation of this file.

```
00001 // Generated by   : ImageConverter 565 Online
00002 // Generated from : zebatka.png
00003 // Time generated : Fri, 05 Dec 25 14:51:27 +0100  (Server timezone: CET)
00004 // Image Size     : 30x30 pixels
00005 // Memory usage   : 1800 bytes
00006
00007 #if defined(__AVR__)
00008 #include <avr/pgmspace.h>
00009 #elif defined(__PIC32MX__)
00010 #define PROGMEM
00011 #elif defined(__arm__)
00012 #define PROGMEM
00013 #endif
00014
00015 const unsigned short
00016     settings_sprite[900] PROGMEM =
00017         {
00018             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00019             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00020             0x0000, 0x95B7, 0x9597, 0x9597, // 0x0010 (16) pixels
00021             0x95B7, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00022             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
```

```
00023             0x0000, 0x0000, 0x0000, 0x0000, // 0x0020 (32) pixels
00024             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x9DF8,
00025             0x95D8, 0x95D8, 0x0000, 0x0000, 0x9DF8, 0x95D7,
00026             0x5B4F, 0x5B4F, 0x95D7, 0x9DF8, // 0x0030 (48) pixels
00027             0x0000, 0x0000, 0x95D8, 0x95D8, 0x9DF8, 0x0000,
00028             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00029             0x0000, 0x0000, 0x0000, 0x0000, // 0x0040 (64) pixels
00030             0x0000, 0x0000, 0x9DF8, 0x95B7, 0x8D76, 0x95B7,
00031             0x9DF8, 0x9DF8, 0x9DF8, 0x8D35, 0x428C, 0x428C,
00032             0x8D36, 0x9DD8, 0x9DF8, 0x95D8, // 0x0050 (80) pixels
00033             0x95B7, 0x8D76, 0x95B7, 0x9DF8, 0x0000, 0x0000,
00034             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00035             0x0000, 0x0000, 0x0000, 0x0000, // 0x0060 (96) pixels
00036             0x95B7, 0x7473, 0x428C, 0x5B6F, 0x95D7, 0x95B7,
00037             0x7CD4, 0x5B4F, 0x428C, 0x428C, 0x5B4F, 0x7CD4,
00038             0x95B7, 0x95D7, 0x5B6F, 0x428C, // 0x0070 (112) pixels
00039             0x7473, 0x95B7, 0x0000, 0x0000, 0x0000, 0x0000,
00040             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00041             0x0000, 0x0000, 0x95D7, 0x7493, // 0x0080 (128) pixels
00042             0x428C, 0x428C, 0x4AAD, 0x428C, 0x428C, 0x428C,
00043             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x4AAD,
00044             0x428C, 0x428C, 0x7493, 0x95D7, // 0x0090 (144) pixels
00045             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00046             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00047             0x95D8, 0x8D56, 0x428C, 0x428C, // 0x00A0 (160) pixels
00048             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00049             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00050             0x8D56, 0x95D8, 0x0000, 0x0000, // 0x00B0 (176) pixels
00051             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00052             0x9DF8, 0x95B7, 0x95D7, 0x95D8, 0x95B7, 0x530E,
00053             0x428C, 0x428C, 0x428C, 0x428C, // 0x00C0 (192) pixels
00054             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00055             0x428C, 0x428C, 0x428C, 0x428C, 0x530E, 0x95B7,
00056             0x95D8, 0x95D7, 0x95B7, 0x9DF8, // 0x00D0 (208) pixels
00057             0x0000, 0x0000, 0x0000, 0x9DF8, 0x95B7, 0x7473,
00058             0x7493, 0x8D56, 0x530E, 0x428C, 0x428C, 0x428C,
00059             0x428C, 0x428C, 0x428C, 0x428C, // 0x00E0 (224) pixels
00060             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00061             0x428C, 0x428C, 0x428C, 0x530E, 0x8D56, 0x7493,
00062             0x7473, 0x95B7, 0x9DF8, 0x0000, // 0x00F0 (240) pixels
00063             0x0000, 0x95D8, 0x8D76, 0x428C, 0x428C, 0x428C,
00064             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00065             0x428C, 0x428C, 0x428C, 0x428C, // 0x0100 (256) pixels
00066             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00067             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x8D76,
00068             0x95D8, 0x0000, 0x0000, 0x95D8, // 0x0110 (272) pixels
00069             0x95B7, 0x5B6F, 0x428C, 0x428C, 0x428C, 0x428C,
00070             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00071             0x428C, 0x428C, 0x428C, 0x428C, // 0x0120 (288) pixels
00072             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00073             0x428C, 0x428C, 0x5B6F, 0x95B7, 0x95D8, 0x0000,
00074             0x0000, 0x0000, 0x9DF8, 0x95D7, // 0x0130 (304) pixels
00075             0x4AAD, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00076             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00077             0x428C, 0x428C, 0x428C, 0x428C, // 0x0140 (320) pixels
00078             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x4AAD,
00079             0x95D7, 0x9DF8, 0x0000, 0x0000, 0x0000, 0x0000,
00080             0x9DF8, 0x95B7, 0x428C, 0x428C, // 0x0150 (336) pixels
00081             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00082             0x428C, 0x428C, 0x532E, 0x532E, 0x428C, 0x428C,
00083             0x428C, 0x428C, 0x428C, 0x428C, // 0x0160 (352) pixels
00084             0x428C, 0x428C, 0x428C, 0x428C, 0x95B7, 0x9DF8,
00085             0x0000, 0x0000, 0x0000, 0x9DF8, 0x9DF8, 0x7CD4,
00086             0x428C, 0x428C, 0x428C, 0x428C, // 0x0170 (368) pixels
00087             0x428C, 0x428C, 0x428C, 0x428C, 0x5B4F, 0x8D56,
00088             0x95B7, 0x95B7, 0x8D56, 0x5B4F, 0x428C, 0x428C,
00089             0x428C, 0x428C, 0x428C, 0x428C, // 0x0180 (384) pixels
00090             0x428C, 0x428C, 0x7CD4, 0x9DF8, 0x9DF8, 0x0000,
00091             0x95B7, 0x95D7, 0x8D35, 0x5B4F, 0x428C, 0x428C,
00092             0x428C, 0x428C, 0x428C, 0x428C, // 0x0190 (400) pixels
00093             0x428C, 0x428C, 0x8D56, 0x9DF8, 0x9DF8, 0x9DF8,
00094             0x9DF8, 0x8D56, 0x428C, 0x428C, 0x428C, 0x428C,
00095             0x428C, 0x428C, 0x428C, 0x428C, // 0x01A0 (416) pixels
00096             0x5B4F, 0x8D35, 0x95D7, 0x95B7, 0x9597, 0x5B4F,
00097             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00098             0x428C, 0x428C, 0x428C, 0x532E, // 0x01B0 (432) pixels
00099             0x95B7, 0x9DF8, 0x0000, 0x0000, 0x9DF8, 0x95B7,
00100            0x532E, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00101            0x428C, 0x428C, 0x428C, 0x428C, // 0x01C0 (448) pixels
00102            0x5B4F, 0x9597, 0x9597, 0x5B4F, 0x428C, 0x428C,
00103            0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00104            0x428C, 0x532E, 0x95B7, 0x9DF8, // 0x01D0 (464) pixels
00105            0x0000, 0x0000, 0x9DF8, 0x95B7, 0x532E, 0x428C,
00106            0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00107            0x428C, 0x428C, 0x5B4F, 0x9597, // 0x01E0 (480) pixels
00108            0x95B7, 0x95D7, 0x8D36, 0x5B4F, 0x428C, 0x428C,
00109            0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
```

```
00110             0x8D56, 0x9DF8, 0x9DF8, 0x9DF8, // 0x01F0 (496) pixels
00111             0x9DF8, 0x8D56, 0x428C, 0x428C, 0x428C, 0x428C,
00112             0x428C, 0x428C, 0x428C, 0x428C, 0x5B4F, 0x8D36,
00113             0x95D7, 0x95B7, 0x0000, 0x9DF8, // 0x0200 (512) pixels
00114             0x9DF8, 0x7CD4, 0x428C, 0x428C, 0x428C, 0x428C,
00115             0x428C, 0x428C, 0x428C, 0x428C, 0x5B4F, 0x8D56,
00116             0x95B7, 0x95B7, 0x8D56, 0x5B4F, // 0x0210 (528) pixels
00117             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00118             0x428C, 0x428C, 0x7CD4, 0x9DD8, 0x9DF8, 0x0000,
00119             0x0000, 0x0000, 0x9DF8, 0x95B7, // 0x0220 (544) pixels
00120             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00121             0x428C, 0x428C, 0x428C, 0x428C, 0x532E, 0x532E,
00122             0x428C, 0x428C, 0x428C, 0x428C, // 0x0230 (560) pixels
00123             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00124             0x95B7, 0x9DF8, 0x0000, 0x0000, 0x0000, 0x0000,
00125             0x95D8, 0x95D7, 0x4AAD, 0x428C, // 0x0240 (576) pixels
00126             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00127             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00128             0x428C, 0x428C, 0x428C, 0x428C, // 0x0250 (592) pixels
00129             0x428C, 0x428C, 0x428C, 0x4AAD, 0x95D7, 0x9DF8,
00130             0x0000, 0x0000, 0x0000, 0x95D8, 0x95B7, 0x5B6F,
00131             0x428C, 0x428C, 0x428C, 0x428C, // 0x0260 (608) pixels
00132             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00133             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00134             0x428C, 0x428C, 0x428C, 0x428C, // 0x0270 (624) pixels
00135             0x428C, 0x428C, 0x5B6F, 0x95B7, 0x95D8, 0x0000,
00136             0x0000, 0x95D8, 0x8D76, 0x428C, 0x428C, 0x428C,
00137             0x428C, 0x428C, 0x428C, 0x428C, // 0x0280 (640) pixels
00138             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00139             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00140             0x428C, 0x428C, 0x428C, 0x428C, // 0x0290 (656) pixels
00141             0x428C, 0x8D76, 0x95D8, 0x0000, 0x0000, 0x9DF8,
00142             0x95B7, 0x7473, 0x7493, 0x8D56, 0x530E, 0x428C,
00143             0x428C, 0x428C, 0x428C, 0x428C, // 0x02A0 (672) pixels
00144             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00145             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x530E,
00146             0x8D56, 0x7493, 0x7473, 0x95B7, // 0x02B0 (688) pixels
00147             0x9DF8, 0x0000, 0x0000, 0x0000, 0x9DF8, 0x95B7,
00148             0x95D7, 0x95D8, 0x95B7, 0x530E, 0x428C, 0x428C,
00149             0x428C, 0x428C, 0x428C, 0x428C, // 0x02C0 (704) pixels
00150             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00151             0x428C, 0x428C, 0x530E, 0x95B7, 0x95D8, 0x95D7,
00152             0x95B7, 0x9DF8, 0x0000, 0x0000, // 0x02D0 (720) pixels
00153             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00154             0x95D8, 0x8D56, 0x428C, 0x428C, 0x428C, 0x428C,
00155             0x428C, 0x428C, 0x428C, 0x428C, // 0x02E0 (736) pixels
00156             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00157             0x8D56, 0x95D8, 0x0000, 0x0000, 0x0000, 0x0000,
00158             0x0000, 0x0000, 0x0000, 0x0000, // 0x02F0 (752) pixels
00159             0x0000, 0x0000, 0x0000, 0x0000, 0x95D7, 0x7493,
00160             0x428C, 0x428C, 0x4AAD, 0x428C, 0x428C, 0x428C,
00161             0x428C, 0x428C, 0x428C, 0x428C, // 0x0300 (768) pixels
00162             0x428C, 0x4AAD, 0x428C, 0x428C, 0x7493, 0x95D7,
00163             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00164             0x0000, 0x0000, 0x0000, 0x0000, // 0x0310 (784) pixels
00165             0x0000, 0x0000, 0x95B7, 0x7473, 0x428C, 0x5B6F,
00166             0x95D7, 0x95B7, 0x7CD4, 0x5B4F, 0x428C, 0x428C,
00167             0x5B4F, 0x7CD4, 0x95B7, 0x95D7, // 0x0320 (800) pixels
00168             0x5B6F, 0x428C, 0x7473, 0x95B7, 0x0000, 0x0000,
00169             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00170             0x0000, 0x0000, 0x0000, 0x0000, // 0x0330 (816) pixels
00171             0x9DF8, 0x95B7, 0x8D76, 0x95B7, 0x9DF8, 0x9DF8,
00172             0x9DF8, 0x8D35, 0x428C, 0x428C, 0x8D36, 0x9DD8,
00173             0x9DF8, 0x95D8, 0x95B7, 0x8D76, // 0x0340 (832) pixels
00174             0x95B7, 0x9DF8, 0x0000, 0x0000, 0x0000, 0x0000,
00175             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00176             0x0000, 0x0000, 0x0000, 0x9DF8, // 0x0350 (848) pixels
00177             0x95D8, 0x95D8, 0x0000, 0x0000, 0x9DF8, 0x95D7,
00178             0x5B4F, 0x5B4F, 0x95D7, 0x9DF8, 0x0000, 0x0000,
00179             0x95D8, 0x95D8, 0x9DF8, 0x0000, // 0x0360 (864) pixels
00180             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00181             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00182             0x0000, 0x0000, 0x0000, 0x0000, // 0x0370 (880) pixels
00183             0x0000, 0x0000, 0x0000, 0x95B7, 0x9597, 0x9597,
00184             0x95B7, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00185             0x0000, 0x0000, 0x0000, 0x0000, // 0x0380 (896) pixels
00186             0x0000, 0x0000, 0x0000, 0x0000};
```

4.27 src/UIManager.cpp File Reference

Implementation of the UIManager class.

```
#include "UIManager.h"
```

Include dependency graph for UIManager.cpp:

Functions

- void wrapperGoToSettings (uint8_t, int16_t, int16_t)
- void wrapperGoToHome (uint8_t, int16_t, int16_t)
- void wrapperSwitchAutoBrightness (uint8_t, int16_t, int16_t)
- void wrapperGoToAppConnection (uint8_t, int16_t, int16_t)
- void wrapperGoToWifiConnection (uint8_t, int16_t, int16_t)

Variables

- UIManager * uiInstance = nullptr

4.27.1 Detailed Description

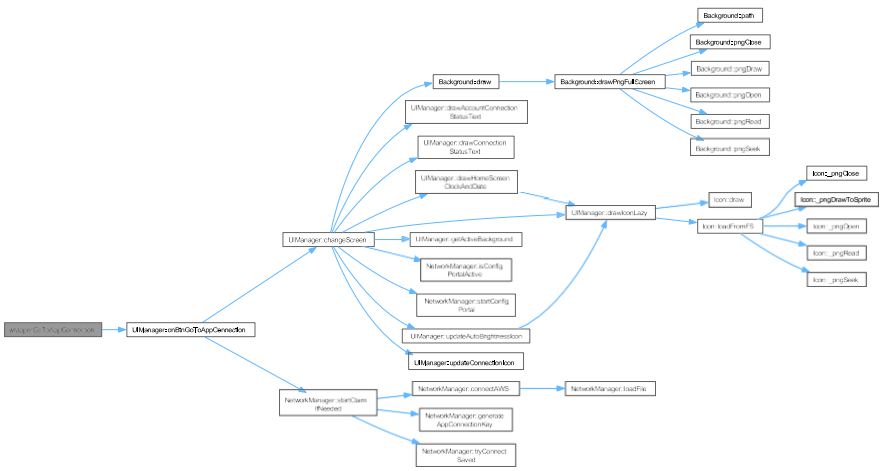
Implementation of the UIManager class.

4.27.2 Function Documentation

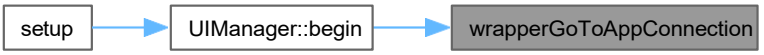
4.27.2.1 wrapperGoToAppConnection()

```
void wrapperGoToAppConnection (
            uint8_t ,
            int16_t ,
            int16_t )
```

Here is the call graph for this function:



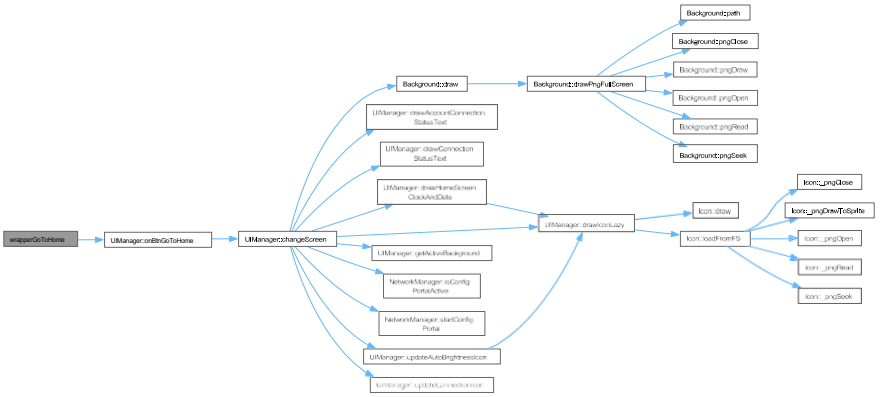
Here is the caller graph for this function:



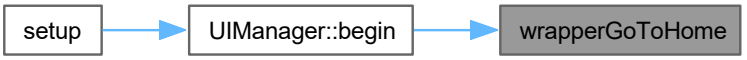
4.27.2.2 wrapperGoToHome()

```
void wrapperGoToHome (
            uint8_t ,
            int16_t ,
            int16_t )
```

Here is the call graph for this function:



Here is the caller graph for this function:

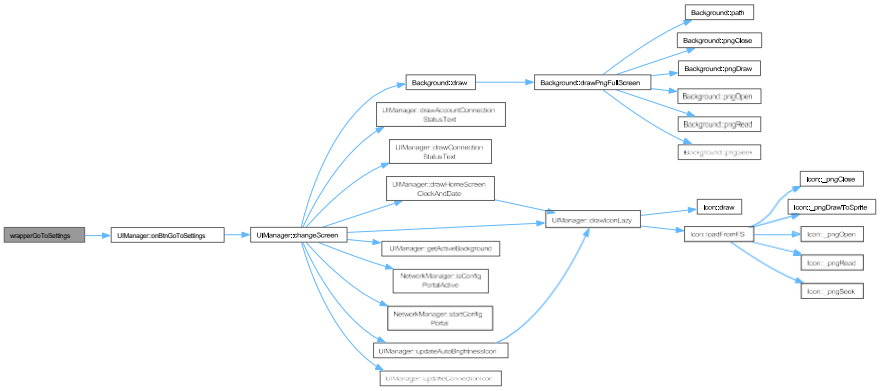


4.27.2.3 wrapperGoToSettings()

```
void wrapperGoToSettings (
            uint8_t ,
```

```
            int16_t ,
            int16_t )
```

Here is the call graph for this function:

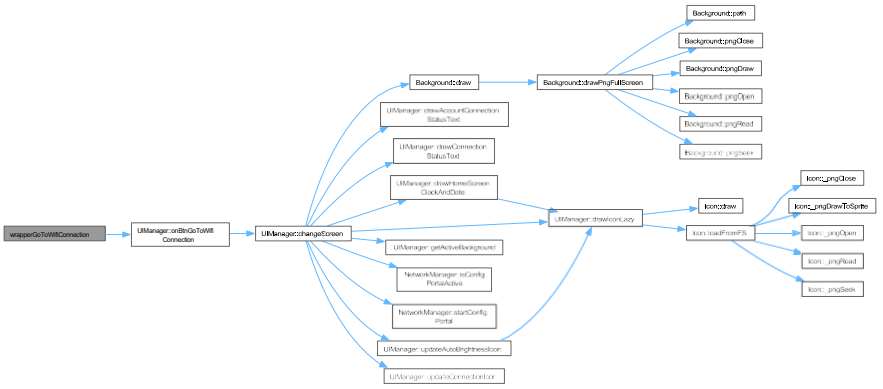


Here is the caller graph for this function:

4.27.2.4 wrapperGoToWifiConnection()

```
void wrapperGoToWifiConnection (
            uint8_t ,
            int16_t ,
            int16_t )
```

Here is the call graph for this function:

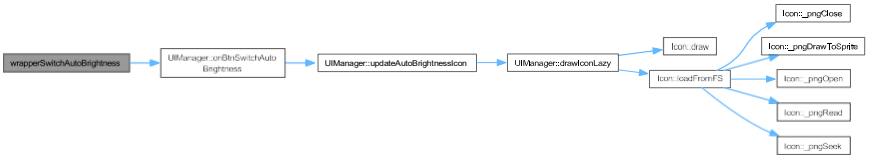


Here is the caller graph for this function:

4.27.2.5 wrapperSwitchAutoBrightness()

```
void wrapperSwitchAutoBrightness (
            uint8_t ,
             int16_t ,
             int16_t )
```

Here is the call graph for this function:



Here is the caller graph for this function:

4.27.3 Variable Documentation

4.27.3.1 uiInstance

```
UIManager* uiInstance = nullptr
```

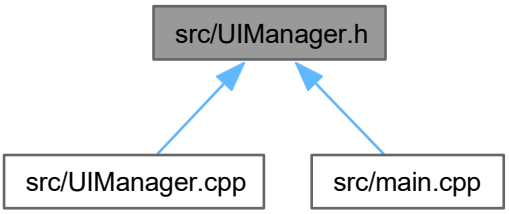
4.28 src/UIManager.h File Reference

Manages the User Interface, screens, and touch interactions.

```
#include <Arduino.h>
#include <LittleFS.h>
#include <PNGdec.h>
#include <TFT_eSPI.h>
#include "Background.h"
#include "Config.h"
#include "Globals.h"
#include "Icon.h"
#include "NetworkManager.h"
#include "SensorManager.h"
#include "autoBrightnessOff_Sprite.h"
#include "autoBrightnessOn_Sprite.h"
#include "settings_sprite.h"
#include "wifiFalse_Sprite.h"
#include "wifitrue_sprite.h"
```

Include dependency graph for UIManager.h:

This graph shows which files directly or indirectly include this file:



Classes

- class UIManager

  Controls the display, rendering logic, and user input.

4.28.1 Detailed Description

Manages the User Interface, screens, and touch interactions.

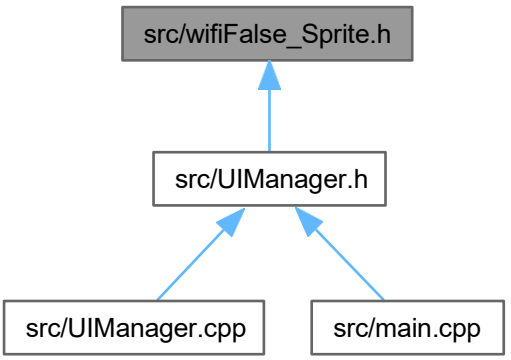
4.29 UIManager.h

Go to the documentation of this file.

```
00001
00006 #pragma once
00007 #include <Arduino.h>
00008 #include <LittleFS.h>
00009 #include <PNGdec.h>
00010 #include <TFT_eSPI.h>
00011
00012 #include "Background.h"
00013 #include "Config.h"
00014 #include "Globals.h"
00015 #include "Icon.h"
00016 #include "NetworkManager.h"
00017 #include "SensorManager.h"
00018
00019 #include "autoBrightnessOff_Sprite.h"
00020 #include "autoBrightnessOn_Sprite.h"
00021 #include "settings_sprite.h"
00022 #include "wifiFalse_Sprite.h"
00023 #include "wifitrue_sprite.h"
00024
00029 class UIManager {
00030 public:
00036   UIManager(SensorManager *sensorMgr, NetworkManager *networkMgr);
00037
00041   void begin();
00042
00046   void update();
00047
00052   void changeScreen(SCREEN s);
00053
00054   // Button Callbacks
00055   void onBtnGoToSettings();
00056   void onBtnGoToHome();
00057   void onBtnGoToWifiConnection();
00058   void onBtnGoToAppConnection();
00059   void onBtnSwitchAutoBrightness();
00060
00061 private:
00062   TFT_eSPI tft;
00063   PNG png;
00064   SensorManager *_sensorMgr;
00065   NetworkManager *_networkMgr;
00066
00067   Background *bgHome;
00068   Background *bgSettings;
00069   Background *bgAccount;
00070
00071   TFT_eSprite settingsIconSprite;
00072   SCREEN currentScreen;
00073
00074   uint32_t lastDrawMs = 0;
00075   int lastDrawnMinute = -1;
00076   int lastDrawnDay = -1;
00077   bool homeStaticDrawn = false;
00078
00079   Background *getActiveBackground();
00080   void drawIconLazy(const char *path, int x, int y, uint16_t bg = TFT_BLACK);
00081   void updateAutoBrightnessIcon(bool status);
00082   void updateConnectionIcon(bool status);
00083   void drawConnectionStatusText();
00084   void drawAccountConnectionStatusText();
00085   void drawHomeScreenDynamicData();
00086   void drawHomeScreenClockAndDate();
00087 };
```

4.30 src/wifiFalse_Sprite.h File Reference

This graph shows which files directly or indirectly include this file:



Variables

- const unsigned short wifiFalse_Sprite[900] PROGMEM

4.30.1 Variable Documentation

4.30.1.1 PROGMEM

```
const unsigned short wifiFalse_Sprite [900] PROGMEM
```

4.31 wifiFalse_Sprite.h

Go to the documentation of this file.

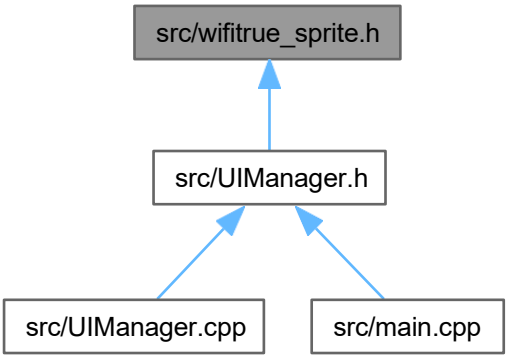
```
00001 // Generated by   : ImageConverter 565 Online
00002 // Generated from : wifiFalse_Sprite.png
00003 // Time generated : Fri, 05 Dec 25 15:14:00 +0100  (Server timezone: CET)
00004 // Image Size     : 30x30 pixels
00005 // Memory usage   : 1800 bytes
00006
00007 #if defined(__AVR__)
00008 #include <avr/pgmspace.h>
00009 #elif defined(__PIC32MX__)
00010 #define PROGMEM
00011 #elif defined(__arm__)
00012 #define PROGMEM
00013 #endif
00014
00015 const unsigned short
00016     wifiFalse_Sprite[900] PROGMEM =
00017         {
00018             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00019             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00020             0x0000, 0x0000, 0x0000, 0x0000, // 0x0010 (16) pixels
00021             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00022             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
```

```
00023             0x0000, 0x0000, 0x0000, 0x0000, // 0x0020 (32) pixels
00024             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00025             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00026             0x0000, 0x0000, 0x0000, 0x0000, // 0x0030 (48) pixels
00027             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00028             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00029             0x0000, 0x0000, 0x0000, 0x0000, // 0x0040 (64) pixels
00030             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00031             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00032             0x0000, 0x0000, 0x0000, 0x0000, // 0x0050 (80) pixels
00033             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00034             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00035             0x0000, 0x0000, 0x0000, 0x0000, // 0x0060 (96) pixels
00036             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00037             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00038             0x0000, 0x0000, 0x0000, 0x0000, // 0x0070 (112) pixels
00039             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00040             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00041             0x0000, 0x0000, 0x0000, 0x0000, // 0x0080 (128) pixels
00042             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0xF800,
00043             0xF800, 0xF800, 0xF800, 0x0000, 0x0000, 0x0000,
00044             0x0000, 0x0000, 0x0000, 0x0000, // 0x0090 (144) pixels
00045             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00046             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00047             0x0000, 0x0000, 0xF800, 0xF800, // 0x00A0 (160) pixels
00048             0xF800, 0xF800, 0xF800, 0xF800, 0xF800, 0xF800,
00049             0xF800, 0xF800, 0xF800, 0xF800, 0xF800, 0xF800,
00050             0x0000, 0x0000, 0x0000, 0x0000, // 0x00B0 (176) pixels
00051             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00052             0x0000, 0x0000, 0x0000, 0x0000, 0xF800, 0xF800,
00053             0xF800, 0xF800, 0xF800, 0xF800, // 0x00C0 (192) pixels
00054             0xF800, 0xF800, 0xF800, 0xF800, 0xF800, 0xF800,
00055             0xF800, 0xF800, 0xF800, 0xF800, 0xF800, 0xF800,
00056             0x0000, 0x0000, 0x0000, 0x0000, // 0x00D0 (208) pixels
00057             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00058             0xF800, 0xF800, 0xF800, 0xF800, 0xF800, 0xF800,
00059             0xF800, 0xF800, 0xF800, 0xF800, // 0x00E0 (224) pixels
00060             0xF800, 0xF800, 0xF800, 0xF800, 0xF800, 0xF800,
00061             0xF800, 0xF800, 0xF800, 0xF800, 0xF800, 0xF800,
00062             0x0000, 0x0000, 0x0000, 0x0000, // 0x00F0 (240) pixels
00063             0x0000, 0x0000, 0x0000, 0xF800, 0xF800, 0xF800,
00064             0xF800, 0xF800, 0xF800, 0xF800, 0xF800, 0xF800,
00065             0xF800, 0xF800, 0x0000, 0x0000, // 0x0100 (256) pixels
00066             0xF800, 0xF800, 0xF800, 0xF800, 0xF800, 0xF800,
00067             0xF800, 0xF800, 0xF800, 0xF800, 0xF800, 0x0000,
00068             0x0000, 0x0000, 0x0000, 0xF800, // 0x0110 (272) pixels
00069             0xF800, 0xF800, 0xF800, 0xF800, 0xF800, 0xF800,
00070             0xF800, 0xF800, 0x0000, 0x0000, 0x0000, 0x0000,
00071             0x0000, 0x0000, 0x0000, 0x0000, // 0x0120 (288) pixels
00072             0x0000, 0x0000, 0xF800, 0xF800, 0xF800, 0xF800,
00073             0xF800, 0xF800, 0xF800, 0xF800, 0xF800, 0x0000,
00074             0x0000, 0xF800, 0xF800, 0xF800, // 0x0130 (304) pixels
00075             0xF800, 0xF800, 0xF800, 0xF800, 0x0000, 0x0000,
00076             0xF800, 0xF800, 0xF800, 0xF800, 0xF800, 0xF800,
00077             0xF800, 0xF800, 0xF800, 0xF800, // 0x0140 (320) pixels
00078             0x0000, 0x0000, 0xF800, 0xF800, 0xF800, 0xF800,
00079             0xF800, 0xF800, 0xF800, 0x0000, 0x0000, 0xF800,
00080             0xF800, 0xF800, 0xF800, 0xF800, // 0x0150 (336) pixels
00081             0x0000, 0x0000, 0xF800, 0xF800, 0xF800, 0xF800,
00082             0xF800, 0xF800, 0xF800, 0xF800, 0xF800, 0xF800,
00083             0xF800, 0xF800, 0xF800, 0xF800, // 0x0160 (352) pixels
00084             0x0000, 0x0000, 0xF800, 0xF800, 0xF800, 0xF800,
00085             0xF800, 0x0000, 0x0000, 0x0000, 0xF800, 0xF800,
00086             0xF800, 0x0000, 0xF800, 0xF800, // 0x0170 (368) pixels
00087             0xF800, 0xF800, 0xF800, 0xF800, 0xF800, 0xF800,
00088             0xF800, 0xF800, 0xF800, 0xF800, 0xF800, 0xF800,
00089             0xF800, 0xF800, 0xF800, 0xF800, // 0x0180 (384) pixels
00090             0x0000, 0xF800, 0xF800, 0xF800, 0x0000, 0x0000,
00091             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0xF800,
00092             0xF800, 0xF800, 0xF800, 0xF800, // 0x0190 (400) pixels
00093             0xF800, 0xF800, 0xF800, 0xF800, 0xF800, 0xF800,
00094             0xF800, 0xF800, 0xF800, 0xF800, 0xF800, 0xF800,
00095             0xF800, 0xF800, 0xF800, 0x0000, // 0x01A0 (416) pixels
00096             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00097             0x0000, 0x0000, 0xF800, 0xF800, 0xF800, 0xF800,
00098             0xF800, 0xF800, 0xF800, 0x0000, // 0x01B0 (432) pixels
00099             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00100             0x0000, 0xF800, 0xF800, 0xF800, 0xF800, 0xF800,
00101             0xF800, 0xF800, 0x0000, 0x0000, // 0x01C0 (448) pixels
00102             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00103             0x0000, 0xF800, 0xF800, 0xF800, 0xF800, 0xF800,
00104             0x0000, 0x0000, 0x0000, 0x0000, // 0x01D0 (464) pixels
00105             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00106             0xF800, 0xF800, 0xF800, 0xF800, 0xF800, 0x0000,
00107             0x0000, 0x0000, 0x0000, 0x0000, // 0x01E0 (480) pixels
00108             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00109             0xF800, 0xF800, 0x0000, 0x0000, 0x0000, 0xF800,
```

```
00110             0xF800, 0xF800, 0xF800, 0xF800, // 0x01F0 (496) pixels
00111             0xF800, 0xF800, 0xF800, 0x0000, 0x0000, 0x0000,
00112             0xF800, 0xF800, 0x0000, 0x0000, 0x0000, 0x0000,
00113             0x0000, 0x0000, 0x0000, 0x0000, // 0x0200 (512) pixels
00114             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00115             0x0000, 0x0000, 0xF800, 0xF800, 0xF800, 0xF800,
00116             0xF800, 0xF800, 0xF800, 0xF800, // 0x0210 (528) pixels
00117             0xF800, 0xF800, 0x0000, 0x0000, 0x0000, 0x0000,
00118             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00119             0x0000, 0x0000, 0x0000, 0x0000, // 0x0220 (544) pixels
00120             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0xF800,
00121             0xF800, 0xF800, 0xF800, 0xF800, 0xF800, 0xF800,
00122             0xF800, 0xF800, 0xF800, 0xF800, // 0x0230 (560) pixels
00123             0xF800, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00124             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00125             0x0000, 0x0000, 0x0000, 0x0000, // 0x0240 (576) pixels
00126             0x0000, 0x0000, 0x0000, 0xF800, 0xF800, 0xF800,
00127             0xF800, 0xF800, 0xF800, 0xF800, 0xF800, 0xF800,
00128             0xF800, 0xF800, 0xF800, 0x0000, // 0x0250 (592) pixels
00129             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00130             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00131             0x0000, 0x0000, 0x0000, 0x0000, // 0x0260 (608) pixels
00132             0x0000, 0x0000, 0xF800, 0xF800, 0xF800, 0x0000,
00133             0x0000, 0x0000, 0x0000, 0xF800, 0xF800, 0xF800,
00134             0x0000, 0x0000, 0x0000, 0x0000, // 0x0270 (624) pixels
00135             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00136             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00137             0x0000, 0x0000, 0x0000, 0x0000, // 0x0280 (640) pixels
00138             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00139             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00140             0x0000, 0x0000, 0x0000, 0x0000, // 0x0290 (656) pixels
00141             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00142             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00143             0x0000, 0x0000, 0x0000, 0x0000, // 0x02A0 (672) pixels
00144             0x0000, 0xF800, 0xF800, 0xF800, 0xF800, 0x0000,
00145             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00146             0x0000, 0x0000, 0x0000, 0x0000, // 0x02B0 (688) pixels
00147             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00148             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00149             0x0000, 0x0000, 0x0000, 0xF800, // 0x02C0 (704) pixels
00150             0xF800, 0xF800, 0xF800, 0x0000, 0x0000, 0x0000,
00151             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00152             0x0000, 0x0000, 0x0000, 0x0000, // 0x02D0 (720) pixels
00153             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00154             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00155             0x0000, 0xF800, 0xF800, 0xF800, // 0x02E0 (736) pixels
00156             0xF800, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00157             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00158             0x0000, 0x0000, 0x0000, 0x0000, // 0x02F0 (752) pixels
00159             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00160             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00161             0xF800, 0xF800, 0x0000, 0x0000, // 0x0300 (768) pixels
00162             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00163             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00164             0x0000, 0x0000, 0x0000, 0x0000, // 0x0310 (784) pixels
00165             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00166             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00167             0x0000, 0x0000, 0x0000, 0x0000, // 0x0320 (800) pixels
00168             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00169             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00170             0x0000, 0x0000, 0x0000, 0x0000, // 0x0330 (816) pixels
00171             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00172             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00173             0x0000, 0x0000, 0x0000, 0x0000, // 0x0340 (832) pixels
00174             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00175             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00176             0x0000, 0x0000, 0x0000, 0x0000, // 0x0350 (848) pixels
00177             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00178             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00179             0x0000, 0x0000, 0x0000, 0x0000, // 0x0360 (864) pixels
00180             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00181             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00182             0x0000, 0x0000, 0x0000, 0x0000, // 0x0370 (880) pixels
00183             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00184             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00185             0x0000, 0x0000, 0x0000, 0x0000, // 0x0380 (896) pixels
00186             0x0000, 0x0000, 0x0000, 0x0000};
```

4.32 src/wifitrue_sprite.h File Reference

This graph shows which files directly or indirectly include this file:



Variables

- const unsigned short wifitrue_sprite[900] PROGMEM

4.32.1 Variable Documentation

4.32.1.1 PROGMEM

```
const unsigned short wifitrue_sprite [900] PROGMEM
```

4.33 wifitrue_sprite.h

Go to the documentation of this file.

```
00001 // Generated by   : ImageConverter 565 Online
00002 // Generated from : wifi-true-sprite.png
00003 // Time generated : Fri, 05 Dec 25 14:57:08 +0100  (Server timezone: CET)
00004 // Image Size     : 30x30 pixels
00005 // Memory usage   : 1800 bytes
00006
00007 #if defined(__AVR__)
00008 #include <avr/pgmspace.h>
00009 #elif defined(__PIC32MX__)
00010 #define PROGMEM
00011 #elif defined(__arm__)
00012 #define PROGMEM
00013 #endif
00014
00015 const unsigned short
00016     wifitrue_sprite[900] PROGMEM =
00017         {
00018             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00019             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00020             0x0000, 0x0000, 0x0000, 0x0000, // 0x0010 (16) pixels
00021             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00022             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
```

```
00023             0x0000, 0x0000, 0x0000, 0x0000, // 0x0020 (32) pixels
00024             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00025             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00026             0x0000, 0x0000, 0x0000, 0x0000, // 0x0030 (48) pixels
00027             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00028             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00029             0x0000, 0x0000, 0x0000, 0x0000, // 0x0040 (64) pixels
00030             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00031             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00032             0x0000, 0x0000, 0x0000, 0x0000, // 0x0050 (80) pixels
00033             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00034             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00035             0x0000, 0x0000, 0x0000, 0x0000, // 0x0060 (96) pixels
00036             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00037             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00038             0x0000, 0x0000, 0x0000, 0x0000, // 0x0070 (112) pixels
00039             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00040             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00041             0x0000, 0x0000, 0x0000, 0x0000, // 0x0080 (128) pixels
00042             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x428C,
00043             0x428C, 0x428C, 0x428C, 0x0000, 0x0000, 0x0000,
00044             0x0000, 0x0000, 0x0000, 0x0000, // 0x0090 (144) pixels
00045             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00046             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00047             0x0000, 0x0000, 0x428C, 0x428C, // 0x00A0 (160) pixels
00048             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00049             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00050             0x0000, 0x0000, 0x0000, 0x0000, // 0x00B0 (176) pixels
00051             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00052             0x0000, 0x0000, 0x0000, 0x0000, 0x428C, 0x428C,
00053             0x428C, 0x428C, 0x428C, 0x428C, // 0x00C0 (192) pixels
00054             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00055             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00056             0x0000, 0x0000, 0x0000, 0x0000, // 0x00D0 (208) pixels
00057             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00058             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00059             0x428C, 0x428C, 0x428C, 0x428C, // 0x00E0 (224) pixels
00060             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00061             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00062             0x0000, 0x0000, 0x0000, 0x0000, // 0x00F0 (240) pixels
00063             0x0000, 0x0000, 0x0000, 0x428C, 0x428C, 0x428C,
00064             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00065             0x428C, 0x428C, 0x0000, 0x0000, // 0x0100 (256) pixels
00066             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00067             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x0000,
00068             0x0000, 0x0000, 0x0000, 0x428C, // 0x0110 (272) pixels
00069             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00070             0x428C, 0x428C, 0x0000, 0x0000, 0x0000, 0x0000,
00071             0x0000, 0x0000, 0x0000, 0x0000, // 0x0120 (288) pixels
00072             0x0000, 0x0000, 0x428C, 0x428C, 0x428C, 0x428C,
00073             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x0000,
00074             0x0000, 0x428C, 0x428C, 0x428C, // 0x0130 (304) pixels
00075             0x428C, 0x428C, 0x428C, 0x428C, 0x0000, 0x0000,
00076             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00077             0x428C, 0x428C, 0x428C, 0x428C, // 0x0140 (320) pixels
00078             0x0000, 0x0000, 0x428C, 0x428C, 0x428C, 0x428C,
00079             0x428C, 0x428C, 0x428C, 0x0000, 0x0000, 0x428C,
00080             0x428C, 0x428C, 0x428C, 0x428C, // 0x0150 (336) pixels
00081             0x0000, 0x0000, 0x428C, 0x428C, 0x428C, 0x428C,
00082             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00083             0x428C, 0x428C, 0x428C, 0x428C, // 0x0160 (352) pixels
00084             0x0000, 0x0000, 0x428C, 0x428C, 0x428C, 0x428C,
00085             0x428C, 0x0000, 0x0000, 0x0000, 0x428C, 0x428C,
00086             0x428C, 0x0000, 0x428C, 0x428C, // 0x0170 (368) pixels
00087             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00088             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00089             0x428C, 0x428C, 0x428C, 0x428C, // 0x0180 (384) pixels
00090             0x0000, 0x428C, 0x428C, 0x428C, 0x0000, 0x0000,
00091             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x428C,
00092             0x428C, 0x428C, 0x428C, 0x428C, // 0x0190 (400) pixels
00093             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00094             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00095             0x428C, 0x428C, 0x428C, 0x0000, // 0x01A0 (416) pixels
00096             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00097             0x0000, 0x0000, 0x428C, 0x428C, 0x428C, 0x428C,
00098             0x428C, 0x428C, 0x428C, 0x0000, // 0x01B0 (432) pixels
00099             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00100             0x0000, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00101             0x428C, 0x428C, 0x0000, 0x0000, // 0x01C0 (448) pixels
00102             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00103             0x0000, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00104             0x0000, 0x0000, 0x0000, 0x0000, // 0x01D0 (464) pixels
00105             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00106             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x0000,
00107             0x0000, 0x0000, 0x0000, 0x0000, // 0x01E0 (480) pixels
00108             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00109             0x428C, 0x428C, 0x0000, 0x0000, 0x0000, 0x428C,
```

```
00110             0x428C, 0x428C, 0x428C, 0x428C, // 0x01F0 (496) pixels
00111             0x428C, 0x428C, 0x428C, 0x0000, 0x0000, 0x0000,
00112             0x428C, 0x428C, 0x0000, 0x0000, 0x0000, 0x0000,
00113             0x0000, 0x0000, 0x0000, 0x0000, // 0x0200 (512) pixels
00114             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00115             0x0000, 0x0000, 0x428C, 0x428C, 0x428C, 0x428C,
00116             0x428C, 0x428C, 0x428C, 0x428C, // 0x0210 (528) pixels
00117             0x428C, 0x428C, 0x0000, 0x0000, 0x0000, 0x0000,
00118             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00119             0x0000, 0x0000, 0x0000, 0x0000, // 0x0220 (544) pixels
00120             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x428C,
00121             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00122             0x428C, 0x428C, 0x428C, 0x428C, // 0x0230 (560) pixels
00123             0x428C, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00124             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00125             0x0000, 0x0000, 0x0000, 0x0000, // 0x0240 (576) pixels
00126             0x0000, 0x0000, 0x0000, 0x428C, 0x428C, 0x428C,
00127             0x428C, 0x428C, 0x428C, 0x428C, 0x428C, 0x428C,
00128             0x428C, 0x428C, 0x428C, 0x0000, // 0x0250 (592) pixels
00129             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00130             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00131             0x0000, 0x0000, 0x0000, 0x0000, // 0x0260 (608) pixels
00132             0x0000, 0x0000, 0x428C, 0x428C, 0x428C, 0x0000,
00133             0x0000, 0x0000, 0x0000, 0x428C, 0x428C, 0x428C,
00134             0x0000, 0x0000, 0x0000, 0x0000, // 0x0270 (624) pixels
00135             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00136             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00137             0x0000, 0x0000, 0x0000, 0x0000, // 0x0280 (640) pixels
00138             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00139             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00140             0x0000, 0x0000, 0x0000, 0x0000, // 0x0290 (656) pixels
00141             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00142             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00143             0x0000, 0x0000, 0x0000, 0x0000, // 0x02A0 (672) pixels
00144             0x0000, 0x428C, 0x428C, 0x428C, 0x428C, 0x0000,
00145             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00146             0x0000, 0x0000, 0x0000, 0x0000, // 0x02B0 (688) pixels
00147             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00148             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00149             0x0000, 0x0000, 0x0000, 0x428C, // 0x02C0 (704) pixels
00150             0x428C, 0x428C, 0x428C, 0x0000, 0x0000, 0x0000,
00151             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00152             0x0000, 0x0000, 0x0000, 0x0000, // 0x02D0 (720) pixels
00153             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00154             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00155             0x0000, 0x428C, 0x428C, 0x428C, // 0x02E0 (736) pixels
00156             0x428C, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00157             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00158             0x0000, 0x0000, 0x0000, 0x0000, // 0x02F0 (752) pixels
00159             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00160             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00161             0x428C, 0x428C, 0x0000, 0x0000, // 0x0300 (768) pixels
00162             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00163             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00164             0x0000, 0x0000, 0x0000, 0x0000, // 0x0310 (784) pixels
00165             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00166             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00167             0x0000, 0x0000, 0x0000, 0x0000, // 0x0320 (800) pixels
00168             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00169             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00170             0x0000, 0x0000, 0x0000, 0x0000, // 0x0330 (816) pixels
00171             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00172             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00173             0x0000, 0x0000, 0x0000, 0x0000, // 0x0340 (832) pixels
00174             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00175             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00176             0x0000, 0x0000, 0x0000, 0x0000, // 0x0350 (848) pixels
00177             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00178             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00179             0x0000, 0x0000, 0x0000, 0x0000, // 0x0360 (864) pixels
00180             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00181             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00182             0x0000, 0x0000, 0x0000, 0x0000, // 0x0370 (880) pixels
00183             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00184             0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
00185             0x0000, 0x0000, 0x0000, 0x0000, // 0x0380 (896) pixels
00186             0x0000, 0x0000, 0x0000, 0x0000};
```