GPIO Driver for STM32F407x

Author: Kunal Salvi

Generated by Doxygen 1.12.

Chapter 1

GPIO Driver for STM32F407VGT6

1.1 Introduction

This documentation provides an overview of the GPIO driver implementation for the STM32F407VGT6 microcontroller. The GPIO driver includes functions for setting up and controlling the GPIO pins, including setting pins high or low, reading pin or port states, writing to a port, and configuring GPIO interrupts.

1.2 Features

- Enable or disable GPIO clocks
- Initialize GPIO pins with specific configurations
- Set or clear individual GPIO pins
- Read the state of GPIO pins or the entire port
- Configure GPIO interrupts with edge selection and priority

1.3 Usage

Include the `GPIO.h` and `GPIO_Defs.h` headers in your application code to access the GPIO functions and configurations. Use the provided functions to initialize GPIO pins, control their states, and handle interrupts as needed.

Example:

```
GPIO_Pin_Init(GPIOA, 5, GPIO_Configuration.Mode.General_Purpose_Output,
      GPIO_Configuration.Output_Type.Push_Pull, GPIO_Configuration.Speed.High_Speed,
      GPIO_Configuration.Pull.No_Pull_Up_Down, 0);
GPIO_Pin_High(GPIOA, 5);    // Set GPIOA pin 5 high
```

1.4 Dependencies

- `GPIO_Defs.h`: Contains the definitions and configurations for the GPIO driver.
- `main.h`: Include this to provide the necessary microcontroller-specific includes.

1.5 Author

Author: Your Name

Date

2024-08-21

Version

1.0

Copyright

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 GPIO_Alternate_Function Struct Reference

GPIO Alternate Functions.

`#include <GPIO_Defs.h>`

Data Fields

- uint8_t None
- uint8_t Analog
- uint8_t SYS
- uint8_t MCO1
- uint8_t MCO2
- uint8_t RTC_REFIN
- uint8_t TIM_1
- uint8_t TIM_2
- uint8_t TIM_3
- uint8_t TIM_4
- uint8_t TIM_5
- uint8_t TIM_8
- uint8_t TIM_9
- uint8_t TIM_10
- uint8_t TIM_11
- uint8_t I2C_1
- uint8_t I2C_2
- uint8_t I2C_3
- uint8_t SPI_1
- uint8_t SPI_2
- uint8_t I2S_2
- uint8_t I2S_2EXT
- uint8_t SPI_3
- uint8_t I2S_EXT
- uint8_t I2S_3
- uint8_t USART_1
- uint8_t USART_2
- uint8_t USART_3

- uint8_t I2S_3EXT
- uint8_t USART_4
- uint8_t UART_5
- uint8_t USART_6
- uint8_t CAN_1
- uint8_t CAN_2
- uint8_t TIM_12
- uint8_t TIM_13
- uint8_t TIM_14
- uint8_t OTG_FS_1
- uint8_t OYG_HS_1
- uint8_t ETH_1
- uint8_t FSMC_1
- uint8_t SDIO_1
- uint8_t OTG_FS_2
- uint8_t DCMI_1
- uint8_t EVENTOUT

4.1.1 Detailed Description

GPIO Alternate Functions.

This structure defines the alternate functions that can be assigned to a GPIO pin.

Definition at line 78 of file GPIO_Defs.h.

4.1.2 Field Documentation

4.1.2.1 Analog

```
uint8_t Analog
```

GPIO pin configured as analog

Definition at line 80 of file GPIO_Defs.h.

4.1.2.2 CAN_1

```
uint8_t CAN_1
```

GPIO pin configured for CAN1

Definition at line 111 of file GPIO_Defs.h.

4.1.2.3 CAN_2

```
uint8_t CAN_2
```

GPIO pin configured for CAN2

Definition at line 112 of file GPIO_Defs.h.

4.1.2.4 DCMI_1

```
uint8_t DCMI_1
```

GPIO pin configured for DCMI

Definition at line 122 of file GPIO_Defs.h.

4.1.2.5 ETH_1

```
uint8_t ETH_1
```

GPIO pin configured for Ethernet

Definition at line 118 of file GPIO_Defs.h.

4.1.2.6 EVENTOUT

```
uint8_t EVENTOUT
```

GPIO pin configured for EVENTOUT

Definition at line 123 of file GPIO_Defs.h.

4.1.2.7 FSMC_1

```
uint8_t FSMC_1
```

GPIO pin configured for FSMC

Definition at line 119 of file GPIO_Defs.h.

4.1.2.8 I2C_1

```
uint8_t I2C_1
```

GPIO pin configured for I2C1

Definition at line 94 of file GPIO_Defs.h.

4.1.2.9 I2C_2

```
uint8_t I2C_2
```

GPIO pin configured for I2C2

Definition at line 95 of file GPIO_Defs.h.

4.1.2.10 I2C_3

```
uint8_t I2C_3
```

GPIO pin configured for I2C3

Definition at line 96 of file GPIO_Defs.h.

4.1.2.11 I2S_2

```
uint8_t I2S_2
```

GPIO pin configured for I2S2

Definition at line 99 of file GPIO_Defs.h.

4.1.2.12 I2S_2EXT

```
uint8_t I2S_2EXT
```

GPIO pin configured for I2S2EXT

Definition at line 100 of file GPIO_Defs.h.

4.1.2.13 I2S_3

```
uint8_t I2S_3
```

GPIO pin configured for I2S3

Definition at line 103 of file GPIO_Defs.h.

4.1.2.14 I2S_3EXT

```
uint8_t I2S_3EXT
```

GPIO pin configured for I2S3EXT

Definition at line 107 of file GPIO_Defs.h.

4.1.2.15 I2S_EXT

```
uint8_t I2S_EXT
```

GPIO pin configured for I2S_EXT

Definition at line 102 of file GPIO_Defs.h.

4.1.2.16 MCO1

```
uint8_t MCO1
```

GPIO pin configured as MCO1

Definition at line 82 of file GPIO_Defs.h.

4.1.2.17 MCO2

```
uint8_t MCO2
```

GPIO pin configured as MCO2

Definition at line 83 of file GPIO_Defs.h.

4.1.2.18 None

```
uint8_t None
```

No alternate function configured

Definition at line 79 of file GPIO_Defs.h.

4.1.2.19 OTG_FS_1

```
uint8_t OTG_FS_1
```

GPIO pin configured for OTG_FS1

Definition at line 116 of file GPIO_Defs.h.

4.1.2.20 OTG_FS_2

```
uint8_t OTG_FS_2
```

GPIO pin configured for OTG_FS2

Definition at line 121 of file GPIO_Defs.h.

4.1.2.21 OYG_HS_1

```
uint8_t OYG_HS_1
```

GPIO pin configured for OYG_HS1

Definition at line 117 of file GPIO_Defs.h.

4.1.2.22 RTC_REFIN

```
uint8_t RTC_REFIN
```

GPIO pin configured for RTC_REFIN

Definition at line 84 of file GPIO_Defs.h.

4.1.2.23 SDIO_1

```
uint8_t SDIO_1
```

GPIO pin configured for SDIO

Definition at line 120 of file GPIO_Defs.h.

4.1.2.24 SPI_1

```
uint8_t SPI_1
```

GPIO pin configured for SPI1

Definition at line 97 of file GPIO_Defs.h.

4.1.2.25 SPI_2

```
uint8_t SPI_2
```

GPIO pin configured for SPI2

Definition at line 98 of file GPIO_Defs.h.

4.1.2.26 SPI_3

```
uint8_t SPI_3
```

GPIO pin configured for SPI3

Definition at line 101 of file GPIO_Defs.h.

4.1.2.27 SYS

```
uint8_t SYS
```

GPIO pin configured for system functions

Definition at line 81 of file GPIO_Defs.h.

4.1.2.28 TIM_1

```
uint8_t TIM_1
```

GPIO pin configured for TIM1

Definition at line 85 of file GPIO_Defs.h.

4.1.2.29 TIM_10

```
uint8_t TIM_10
```

GPIO pin configured for TIM10

Definition at line 92 of file GPIO_Defs.h.

4.1.2.30 TIM_11

```
uint8_t TIM_11
```

GPIO pin configured for TIM11

Definition at line 93 of file GPIO_Defs.h.

4.1.2.31 TIM_12

```
uint8_t TIM_12
```

GPIO pin configured for TIM12

Definition at line 113 of file GPIO_Defs.h.

4.1.2.32 TIM_13

```
uint8_t TIM_13
```

GPIO pin configured for TIM13

Definition at line 114 of file GPIO_Defs.h.

4.1.2.33 TIM_14

```
uint8_t TIM_14
```

GPIO pin configured for TIM14

Definition at line 115 of file GPIO_Defs.h.

4.1.2.34 TIM_2

```
uint8_t TIM_2
```

GPIO pin configured for TIM2

Definition at line 86 of file GPIO_Defs.h.

4.1.2.35 TIM_3

```
uint8_t TIM_3
```

GPIO pin configured for TIM3

Definition at line 87 of file GPIO_Defs.h.

4.1.2.36 TIM_4

```
uint8_t TIM_4
```

GPIO pin configured for TIM4

Definition at line 88 of file GPIO_Defs.h.

4.1.2.37 TIM_5

```
uint8_t TIM_5
```

GPIO pin configured for TIM5

Definition at line 89 of file GPIO_Defs.h.

4.1.2.38 TIM_8

```
uint8_t TIM_8
```

GPIO pin configured for TIM8

Definition at line 90 of file GPIO_Defs.h.

4.1.2.39 TIM_9

```
uint8_t TIM_9
```

GPIO pin configured for TIM9

Definition at line 91 of file GPIO_Defs.h.

4.1.2.40 UART_5

```
uint8_t UART_5
```

GPIO pin configured for UART5

Definition at line 109 of file GPIO_Defs.h.

4.1.2.41 USART_1

```
uint8_t USART_1
```

GPIO pin configured for USART1

Definition at line 104 of file GPIO_Defs.h.

4.1.2.42 USART_2

```
uint8_t USART_2
```

GPIO pin configured for USART2

Definition at line 105 of file GPIO_Defs.h.

4.1.2.43 USART_3

```
uint8_t USART_3
```

GPIO pin configured for USART3

Definition at line 106 of file GPIO_Defs.h.

4.1.2.44 USART_4

```
uint8_t USART_4
```

GPIO pin configured for USART4

Definition at line 108 of file GPIO_Defs.h.

4.1.2.45 USART_6

```
uint8_t USART_6
```

GPIO pin configured for USART6

Definition at line 110 of file GPIO_Defs.h.

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

- GPIO_Defs.h

4.2 GPIO_Interrupt_Edge Struct Reference

GPIO Interrupt Edge Types.

`#include <GPIO_Defs.h>`

Data Fields

- uint8_t RISING_EDGE
- uint8_t FALLING_EDGE
- uint8_t RISING_FALLING_EDGE

4.2.1 Detailed Description

GPIO Interrupt Edge Types.

This structure defines the edge types for GPIO interrupts.

Definition at line 131 of file GPIO_Defs.h.

4.2.2 Field Documentation

4.2.2.1 FALLING_EDGE

`uint8_t FALLING_EDGE`

Interrupt triggered on falling edge

Definition at line 133 of file GPIO_Defs.h.

4.2.2.2 RISING_EDGE

`uint8_t RISING_EDGE`

Interrupt triggered on rising edge

Definition at line 132 of file GPIO_Defs.h.

4.2.2.3 RISING_FALLING_EDGE

`uint8_t RISING_FALLING_EDGE`

Interrupt triggered on both rising and falling edges

Definition at line 134 of file GPIO_Defs.h.

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

- GPIO_Defs.h

4.3 GPIO_Mode_Type Struct Reference

GPIO Mode Types.

`#include <GPIO_Defs.h>`

Data Fields

- uint8_t Input
- uint8_t General_Purpose_Output
- uint8_t Alternate_Function
- uint8_t Analog

4.3.1 Detailed Description

GPIO Mode Types.

This structure defines the various modes that a GPIO pin can be configured into.

Definition at line 30 of file GPIO_Defs.h.

4.3.2 Field Documentation

4.3.2.1 Alternate_Function

`uint8_t Alternate_Function`

GPIO pin configured for alternate functions (e.g., peripheral control)

Definition at line 33 of file GPIO_Defs.h.

4.3.2.2 Analog

`uint8_t Analog`

GPIO pin configured as an analog input or output

Definition at line 34 of file GPIO_Defs.h.

4.3.2.3 General_Purpose_Output

`uint8_t General_Purpose_Output`

GPIO pin configured as general-purpose output

Definition at line 32 of file GPIO_Defs.h.

4.3.2.4 Input

```
uint8_t Input
```

GPIO pin configured as input

Definition at line 31 of file GPIO_Defs.h.

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

- GPIO_Defs.h

4.4 GPIO_Output_Type Struct Reference

GPIO Output Types.

```
#include <GPIO_Defs.h>
```

Data Fields

- uint8_t Push_Pull
- uint8_t Open_Drain
- uint8_t None

4.4.1 Detailed Description

GPIO Output Types.

This structure defines the types of output that a GPIO pin can use.

Definition at line 42 of file GPIO_Defs.h.

4.4.2 Field Documentation

4.4.2.1 None

```
uint8_t None
```

No output type configured

Definition at line 45 of file GPIO_Defs.h.

4.4.2.2 Open_Drain

```
uint8_t Open_Drain
```

GPIO pin configured as open-drain output

Definition at line 44 of file GPIO_Defs.h.

4.4.2.3 Push_Pull

```
uint8_t Push_Pull
```

GPIO pin configured as push-pull output

Definition at line 43 of file GPIO_Defs.h.

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

- GPIO_Defs.h

4.5 GPIO_Pin Struct Reference

Data Fields

- uint16_t Pin_numer
- GPIO_TypeDef * Port

4.5.1 Detailed Description

Definition at line 60 of file GPIO.h.

4.5.2 Field Documentation

4.5.2.1 Pin_numer

```
uint16_t Pin_numer
```

Definition at line 62 of file GPIO.h.

4.5.2.2 Port

```
GPIO_TypeDef* Port
```

Definition at line 63 of file GPIO.h.

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

- GPIO.h

4.6 GPIO_Pull Struct Reference

GPIO Pull-Up/Pull-Down Types.

```
#include <GPIO_Defs.h>
```

Data Fields

- uint8_t No_Pull_Up_Down
- uint8_t Pull_Up
- uint8_t Pull_Down
- uint8_t None

4.6.1 Detailed Description

GPIO Pull-Up/Pull-Down Types.

This structure defines the pull-up and pull-down resistor configurations for a GPIO pin.

Definition at line 66 of file GPIO_Defs.h.

4.6.2 Field Documentation

4.6.2.1 No_Pull_Up_Down

```
uint8_t No_Pull_Up_Down
```

No pull-up or pull-down resistor configured

Definition at line 67 of file GPIO_Defs.h.

4.6.2.2 None

```
uint8_t None
```

No pull resistor configured

Definition at line 70 of file GPIO_Defs.h.

4.6.2.3 Pull_Down

```
uint8_t Pull_Down
```

GPIO pin configured with a pull-down resistor

Definition at line 69 of file GPIO_Defs.h.

4.6.2.4 Pull_Up

```
uint8_t Pull_Up
```

GPIO pin configured with a pull-up resistor

Definition at line 68 of file GPIO_Defs.h.

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

- GPIO_Defs.h

4.7 GPIO_Speed Struct Reference

GPIO Speed Types.

`#include <GPIO_Defs.h>`

Data Fields

- uint8_t Low_Speed
- uint8_t Medium_Speed
- uint8_t High_Speed
- uint8_t Very_High_Speed
- uint8_t None

4.7.1 Detailed Description

GPIO Speed Types.

This structure defines the speed options for a GPIO pin.

Definition at line 53 of file GPIO_Defs.h.

4.7.2 Field Documentation

4.7.2.1 High_Speed

`uint8_t High_Speed`

GPIO pin configured for high-speed operation

Definition at line 56 of file GPIO_Defs.h.

4.7.2.2 Low_Speed

`uint8_t Low_Speed`

GPIO pin configured for low-speed operation

Definition at line 54 of file GPIO_Defs.h.

4.7.2.3 Medium_Speed

`uint8_t Medium_Speed`

GPIO pin configured for medium-speed operation

Definition at line 55 of file GPIO_Defs.h.

4.7.2.4 None

```
uint8_t None
```

No speed configured

Definition at line 58 of file GPIO_Defs.h.

4.7.2.5 Very_High_Speed

```
uint8_t Very_High_Speed
```

GPIO pin configured for very high-speed operation

Definition at line 57 of file GPIO_Defs.h.

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

- GPIO_Defs.h

Chapter 5

File Documentation

5.1 GPIO.c File Reference

GPIO Driver Implementation for STM32F407VGT6.

```
#include "GPIO.h"
```

Macros

- #define PORT_TO_INDEX(port)
- #define GPIO_AF_SPLIT_POINT 8
- #define PIN_POS(pin)

Functions

- void EXTI0_IRQHandler (void)

  Interrupt handler for EXTI line 0.
- void EXTI1_IRQHandler (void)

  Interrupt handler for EXTI line 1.
- void EXTI2_IRQHandler (void)

  Interrupt handler for EXTI line 2.
- void EXTI3_IRQHandler (void)

  Interrupt handler for EXTI line 3.
- void EXTI4_IRQHandler (void)

  Interrupt handler for EXTI line 4.
- void EXTI9_5_IRQHandler (void)

  Interrupt handler for EXTI lines 5 to 9.
- void EXTI15_10_IRQHandler (void)

  Interrupt handler for EXTI lines 10 to 15.
- void GPIO_Interrupt_Setup (GPIO_TypeDef ∗Port, int pin, int edge_select, uint32_t priority, void(∗attach_↩ ISR))

  Configures an interrupt for a specific GPIO pin.
- int GPIO_Clock_Disable (GPIO_TypeDef ∗PORT)

  Disables the clock for a specific GPIO port.
- int GPIO_Clock_Enable (GPIO_TypeDef ∗PORT)

  Enables the clock for a specific GPIO port.
- GPIO_Status GPIO_Pin_Init (GPIO_TypeDef ∗Port, uint8_t pin, uint8_t mode, uint8_t output_type, uint8_t speed, uint8_t pull, uint8_t alternate_function)

  Initializes a specific pin with given parameters.

Variables

- void(* EXTI_ISR [16])(void)

5.1.1 Detailed Description

GPIO Driver Implementation for STM32F407VGT6.

This file provides the implementation of the GPIO driver for the STM32F407VGT6 microcontroller. It includes functions to initialize GPIO pins, handle GPIO interrupts, and manage GPIO port configurations.

Version

1.0

Date

2024-08-21

Author

Your Name

Definition in file GPIO.c.

5.1.2 Macro Definition Documentation

5.1.2.1 GPIO_AF_SPLIT_POINT

```
#define GPIO_AF_SPLIT_POINT 8
```

Definition at line 28 of file GPIO.c.

5.1.2.2 PIN_POS

```
#define PIN_POS(
                pin)
```

Value:

```
((pin) * 2)
```

Definition at line 31 of file GPIO.c.

5.1.2.3 PORT_TO_INDEX

```
#define PORT_TO_INDEX(
              port)
```

Value:

```
((port == GPIOA) ? 0 : \
(port == GPIOB) ? 1 : \
(port == GPIOC) ? 2 : \
(port == GPIOD) ? 3 : \
(port == GPIOE) ? 4 : \
(port == GPIOF) ? 5 : \
(port == GPIOG) ? 6 : \
(port == GPIOH) ? 7 : \
(port == GPIOI) ? 8 : -1)
```

Definition at line 17 of file GPIO.c.

5.1.3 Function Documentation

5.1.3.1 EXTI0_IRQHandler()

```
void EXTI0_IRQHandler (
             void )
```

Interrupt handler for EXTI line 0.

This ISR handles interrupts for pin 0, invoking the associated callback function if one is registered.

Definition at line 42 of file GPIO.c.

5.1.3.2 EXTI15_10_IRQHandler()

```
void EXTI15_10_IRQHandler (
             void )
```

Interrupt handler for EXTI lines 10 to 15.

Handles interrupts for pins 10 to 15, checking each pin for active flags.

Definition at line 98 of file GPIO.c.

5.1.3.3 EXTI1_IRQHandler()

```
void EXTI1_IRQHandler (
             void )
```

Interrupt handler for EXTI line 1.

Definition at line 50 of file GPIO.c.

5.1.3.4 EXTI2_IRQHandler()

```
void EXTI2_IRQHandler (
            void )
```

Interrupt handler for EXTI line 2.

Definition at line 58 of file GPIO.c.

5.1.3.5 EXTI3_IRQHandler()

```
void EXTI3_IRQHandler (
            void )
```

Interrupt handler for EXTI line 3.

Definition at line 66 of file GPIO.c.

5.1.3.6 EXTI4_IRQHandler()

```
void EXTI4_IRQHandler (
            void )
```

Interrupt handler for EXTI line 4.

Definition at line 74 of file GPIO.c.

5.1.3.7 EXTI9_5_IRQHandler()

```
void EXTI9_5_IRQHandler (
            void )
```

Interrupt handler for EXTI lines 5 to 9.

Handles interrupts for pins 5 to 9, checking each pin for active flags.

Definition at line 84 of file GPIO.c.

5.1.3.8 GPIO_Clock_Disable()

```
int GPIO_Clock_Disable (
            GPIO_TypeDef * PORT)
```

Disables the clock for a specific GPIO port.

Parameters

PORT	Pointer to GPIO port base address.

Returns

GPIO_SUCCESS on success, GPIO_INVALID_PORT on failure.

Definition at line 163 of file GPIO.c.

5.1.3.9 GPIO_Clock_Enable()

```
int GPIO_Clock_Enable (
            GPIO_TypeDef * PORT)
```

Enables the clock for a specific GPIO port.

Parameters

PORT	Pointer to GPIO port base address.

Returns

GPIO_SUCCESS on success, GPIO_INVALID_PORT on failure.

Definition at line 183 of file GPIO.c.

5.1.3.10 GPIO_Interrupt_Setup()

```
void GPIO_Interrupt_Setup (
            GPIO_TypeDef * Port,
            int pin,
            int edge_select,
            uint32_t priority,
            void * attach_ISR)
```

Configures an interrupt for a specific GPIO pin.

Configures the interrupt for a specific pin.

Sets up the EXTI line, enables interrupt handling, and registers a callback for a specified GPIO pin.

Parameters

Port	Pointer to GPIO port base address.
pin	Pin number (0-15) to configure.
edge_select	Interrupt edge selection (rising, falling, or both).
priority	Interrupt priority level.
attach_ISR	Pointer to the ISR function to invoke on interrupt.

Definition at line 119 of file GPIO.c.

5.1.3.11 GPIO_Pin_Init()

```
GPIO_Status GPIO_Pin_Init (
            GPIO_TypeDef * Port,
            uint8_t pin,
            uint8_t mode,
            uint8_t output_type,
            uint8_t speed,
            uint8_t pull,
            uint8_t alternate_function)
```

Initializes a specific pin with given parameters.

Parameters

Port	Pointer to GPIO port base address.
pin	Pin number to initialize (0-15).
mode	Pin mode (input, output, alternate function, analog).
output_type	Output type (push-pull, open-drain).
speed	Speed level (low, medium, high, very high).
pull	Pull-up/pull-down configuration (none, pull-up, pull-down).
alternate_function	Alternate function selection (0-15).

Definition at line 198 of file GPIO.c.

5.1.4 Variable Documentation

5.1.4.1 EXTI_ISR

```
void(* EXTI_ISR[16])(void) (
              void )
```

Definition at line 34 of file GPIO.c.

5.2 GPIO.c

Go to the documentation of this file.

```
00001
00014 #include "GPIO.h"
00015
00016 // Macro to map GPIO ports to their index for EXTI configuration
00017 #define PORT_TO_INDEX(port) ((port == GPIOA) ? 0 : \
00018                             (port == GPIOB) ? 1 : \
00019                             (port == GPIOC) ? 2 : \
00020                             (port == GPIOD) ? 3 : \
00021                             (port == GPIOE) ? 4 : \
00022                             (port == GPIOF) ? 5 : \
00023                             (port == GPIOG) ? 6 : \
00024                             (port == GPIOH) ? 7 : \
00025                             (port == GPIOI) ? 8 : -1)
00026
00027 // Macro defining the split point between AFR[0] and AFR[1] registers
00028 #define GPIO_AF_SPLIT_POINT 8
00029
00030 // Macro to calculate the bit position for a GPIO pin
00031 #define PIN_POS(pin) ((pin) * 2)
00032
00033 // Array to store function pointers for EXTI ISRs (Interrupt Service Routines)
00034 void (*EXTI_ISR[16])(void);
00035
00042 void EXTI0_IRQHandler(void) {
00043     if (EXTI_ISR[0]) EXTI_ISR[0](); // Invoke registered callback
00044     EXTI->PR |= EXTI_PR_PR0;        // Clear interrupt flag
00045 }
00046
00050 void EXTI1_IRQHandler(void) {
00051     if (EXTI_ISR[1]) EXTI_ISR[1]();
00052     EXTI->PR |= EXTI_PR_PR1;
00053 }
00054
00058 void EXTI2_IRQHandler(void) {
00059     if (EXTI_ISR[2]) EXTI_ISR[2]();
00060     EXTI->PR |= EXTI_PR_PR2;
00061 }
00062
00066 void EXTI3_IRQHandler(void) {
00067     if (EXTI_ISR[3]) EXTI_ISR[3]();
```

```
00068     EXTI->PR |= EXTI_PR_PR3;
00069 }
00070
00074 void EXTI4_IRQHandler(void) {
00075     if (EXTI_ISR[4]) EXTI_ISR[4]();
00076     EXTI->PR |= EXTI_PR_PR4;
00077 }
00078
00084 void EXTI9_5_IRQHandler(void) {
00085     for (int i = 5; i <= 9; ++i) {
00086         if ((EXTI->PR & (1 « i)) && EXTI_ISR[i]) {
00087             EXTI_ISR[i]();        // Invoke callback for pin `i`
00088             EXTI->PR |= (1 « i); // Clear interrupt flag
00089         }
00090     }
00091 }
00092
00098 void EXTI15_10_IRQHandler(void) {
00099     for (int i = 10; i <= 15; ++i) {
00100         if ((EXTI->PR & (1 « i)) && EXTI_ISR[i]) {
00101             EXTI_ISR[i]();        // Invoke callback for pin `i`
00102             EXTI->PR |= (1 « i); // Clear interrupt flag
00103         }
00104     }
00105 }
00106
00119 void GPIO_Interrupt_Setup(GPIO_TypeDef *Port, int pin, int edge_select, uint32_t priority, void
      (*attach_ISR)) {
00120     int port_data = PORT_TO_INDEX(Port);
00121
00122     if (port_data < 0 || pin < 0 || pin > 15) return; // Validate inputs
00123
00124     // Map GPIO port to EXTI line
00125     SYSCFG->EXTICR[pin / 4] |= port_data « ((pin % 4) * 4);
00126
00127     // Enable EXTI interrupt mask
00128     EXTI->IMR |= (1 « pin);
00129
00130     // Configure rising/falling edge triggers
00131     EXTI->RTSR &= ~(1 « pin); // Clear rising edge trigger
00132     EXTI->FTSR &= ~(1 « pin); // Clear falling edge trigger
00133     switch (edge_select) {
00134         case 0:
00135             EXTI->RTSR |= (1 « pin);
00136             break;
00137         case 1:
00138             EXTI->FTSR |= (1 « pin);
00139             break;
00140         case 2:
00141             EXTI->RTSR |= (1 « pin);
00142             EXTI->FTSR |= (1 « pin);
00143             break;
00144     }
00145
00146     // Register the callback function
00147     EXTI_ISR[pin] = attach_ISR;
00148
00149     // Configure NVIC for the EXTI line
00150     IRQn_Type irq = (pin <= 4) ? (IRQn_Type)(EXTI0_IRQn + pin) :
00151                     (pin <= 9) ? EXTI9_5_IRQn : EXTI15_10_IRQn;
00152
00153     NVIC_SetPriority(irq, priority); // Set interrupt priority
00154     NVIC_EnableIRQ(irq);             // Enable NVIC interrupt
00155 }
00156
00163 int GPIO_Clock_Disable(GPIO_TypeDef *PORT) {
00164     switch ((uint32_t)PORT) {
00165         case (uint32_t)GPIOA: RCC->AHB1ENR &= ~RCC_AHB1ENR_GPIOAEN; break;
00166         case (uint32_t)GPIOB: RCC->AHB1ENR &= ~RCC_AHB1ENR_GPIOBEN; break;
00167         case (uint32_t)GPIOC: RCC->AHB1ENR &= ~RCC_AHB1ENR_GPIOCEN; break;
00168         case (uint32_t)GPIOD: RCC->AHB1ENR &= ~RCC_AHB1ENR_GPIODEN; break;
00169         case (uint32_t)GPIOE: RCC->AHB1ENR &= ~RCC_AHB1ENR_GPIOEEN; break;
00170         case (uint32_t)GPIOF: RCC->AHB1ENR &= ~RCC_AHB1ENR_GPIOFEN; break;
00171         case (uint32_t)GPIOH: RCC->AHB1ENR &= ~RCC_AHB1ENR_GPIOHEN; break;
00172         default: return GPIO_INVALID_PORT;
00173     }
00174     return GPIO_SUCCESS;
00175 }
00176
00183 int GPIO_Clock_Enable(GPIO_TypeDef *PORT) {
00184     switch ((uint32_t)PORT) {
00185         case (uint32_t)GPIOA: RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN; break;
00186         case (uint32_t)GPIOB: RCC->AHB1ENR |= RCC_AHB1ENR_GPIOBEN; break;
00187         case (uint32_t)GPIOC: RCC->AHB1ENR |= RCC_AHB1ENR_GPIOCEN; break;
00188         case (uint32_t)GPIOD: RCC->AHB1ENR |= RCC_AHB1ENR_GPIODEN; break;
00189         case (uint32_t)GPIOE: RCC->AHB1ENR |= RCC_AHB1ENR_GPIOEEN; break;
00190         case (uint32_t)GPIOF: RCC->AHB1ENR |= RCC_AHB1ENR_GPIOFEN; break;
```

```
00191         case (uint32_t)GPIOH: RCC->AHB1ENR |= RCC_AHB1ENR_GPIOHEN; break;
00192         default: return GPIO_INVALID_PORT;
00193     }
00194     return GPIO_SUCCESS;
00195 }
00196
00197
00198 GPIO_Status GPIO_Pin_Init(GPIO_TypeDef *Port, uint8_t pin, uint8_t mode, uint8_t output_type, uint8_t
    speed, uint8_t pull, uint8_t alternate_function) {
00199     if (pin > 15 || mode > 3 || speed > 3 || pull > 2) return GPIO_INVALID_PIN;
00200
00201     GPIO_Clock_Enable(Port);
00202
00203     // Reset and set mode
00204     Port->MODER &= ~(3 << PIN_POS(pin));
00205     Port->MODER |= mode << PIN_POS(pin);
00206
00207     // Reset and set output type
00208     Port->OTYPER &= ~(1 << pin);
00209     if (output_type != GPIO_Configuration.Output_Type.None) {
00210         Port->OTYPER |= output_type << pin;
00211     }
00212
00213     // Reset and set speed
00214     Port->OSPEEDR &= ~(3 << PIN_POS(pin));
00215     if (speed != GPIO_Configuration.Speed.None) {
00216         Port->OSPEEDR |= speed << PIN_POS(pin);
00217     }
00218
00219     // Reset and set pull-up/pull-down
00220     Port->PUPDR &= ~(3 << PIN_POS(pin));
00221     if (pull != GPIO_Configuration.Pull.None) {
00222         Port->PUPDR |= pull << PIN_POS(pin);
00223     }
00224
00225     // Set alternate function
00226     if (mode == GPIO_Configuration.Mode.Alternate_Function) {
00227         if (pin < GPIO_AF_SPLIT_POINT) {
00228             Port->AFR[0] &= ~(0xF << (pin * 4));
00229             Port->AFR[0] |= alternate_function << (pin * 4);
00230         } else {
00231             Port->AFR[1] &= ~(0xF << ((pin - GPIO_AF_SPLIT_POINT) * 4));
00232             Port->AFR[1] |= alternate_function << ((pin - GPIO_AF_SPLIT_POINT) * 4);
00233         }
00234     }
00235
00236     return GPIO_SUCCESS;
00237 }
00238
00239
00240
```

5.3 GPIO.h

```
00001
00044 #ifndef GPIO_GPIO_H_
00045 #define GPIO_GPIO_H_
00046
00047 #include "main.h"
00048 #include "GPIO_Defs.h"
00049
00050 typedef enum {
00051     GPIO_SUCCESS = 0,
00052     GPIO_INVALID_PORT = -1,
00053     GPIO_INVALID_PIN = -2,
00054     GPIO_INVALID_MODE = -3,
00055     GPIO_INVALID_SPEED = -4,
00056 } GPIO_Status;
00057
00058
00059
00060 typedef struct GPIO_Pin
00061 {
00062     uint16_t Pin_numer;
00063     GPIO_TypeDef *Port;
00064 }GPIO_Pin;
00065
00071 __STATIC_INLINE  void GPIO_Pin_Low(GPIO_TypeDef *Port, int pin)
00072 {
00073     Port -> ODR &= ~(1 << pin);
00074 }
00075
00076
```

```
00077 __STATIC_INLINE  void GPIO_Pin_Toggle(GPIO_TypeDef *Port, int pin)
00078 {
00079     Port -> ODR ^= (1 « pin);
00080 }
00081
00087 __STATIC_INLINE  uint16_t GPIO_Port_Read(GPIO_TypeDef *Port)
00088 {
00089     return Port -> IDR;
00090 }
00091
00092
00093
00099 __STATIC_INLINE  void GPIO_Write_Port(GPIO_TypeDef *Port, uint16_t data)
00100 {
00101     Port -> ODR = data;
00102 }
00103
00110 __STATIC_INLINE  uint16_t GPIO_Read_Pin(GPIO_TypeDef *Port, uint8_t pin)
00111 {
00112     return (Port->IDR & (1«pin)) » pin;
00113 }
00114
00115
00121 __STATIC_INLINE  void GPIO_Pin_High(GPIO_TypeDef *Port, int pin)
00122 {
00123     Port -> ODR |= 1 « pin;
00124 }
00125
00126
00127
00133 int GPIO_Clock_Disable(GPIO_TypeDef *PORT);
00134
00140 int GPIO_Clock_Enable(GPIO_TypeDef *PORT);
00141
00152 GPIO_Status GPIO_Pin_Init(GPIO_TypeDef *Port, uint8_t pin, uint8_t mode, uint8_t output_type, uint8_t
      speed, uint8_t pull, uint8_t alternate_function);
00153
00154
00161 void GPIO_Interrupt_Setup(GPIO_TypeDef *Port,int pin, int edge_select, uint32_t priority, void
      (*attach_ISR));
00162
00163 #endif /* GPIO_GPIO_H_ */
00164
00165
00166
00167
00168
00169
00170
00171
00172
```

5.4 GPIO_Defs.h File Reference

GPIO Configuration Definitions for STM32F407VGT6.

```
#include "main.h"
```

Data Structures

- struct GPIO_Mode_Type

  GPIO Mode Types.
- struct GPIO_Output_Type

  GPIO Output Types.
- struct GPIO_Speed

  GPIO Speed Types.
- struct GPIO_Pull

  GPIO Pull-Up/Pull-Down Types.
- struct GPIO_Alternate_Function

GPIO Alternate Functions.

- struct GPIO_Interrupt_Edge

  GPIO Interrupt Edge Types.

- struct **GPIO_Configuration**

  GPIO Configuration Structure.

5.4.1 Detailed Description

GPIO Configuration Definitions for STM32F407VGT6.

This file contains the definitions and structures required for configuring the GPIO (General-Purpose Input/Output) pins on the STM32F407VGT6 microcontroller. It includes various configurations such as mode types, output types, speed settings, pull-up/pull-down settings, alternate functions, and interrupt edge configurations.

Version

1.0

Date

2024-08-21

Author

Your Name

Copyright

Copyright (c) 2024

Definition in file GPIO_Defs.h.

5.5 GPIO_Defs.h

Go to the documentation of this file.

```
00001
00018 #ifndef GPIO_DEFS_H_
00019 #define GPIO_DEFS_H_
00020
00021 #include "main.h"
00022
00023
00024
00030 typedef struct {
00031     uint8_t Input;
00032     uint8_t General_Purpose_Output;
00033     uint8_t Alternate_Function;
00034     uint8_t Analog;
00035 } GPIO_Mode_Type;
00036
00042 typedef struct {
00043     uint8_t Push_Pull;
00044     uint8_t Open_Drain;
00045     uint8_t None;
00046 } GPIO_Output_Type;
00047
00053 typedef struct {
00054     uint8_t Low_Speed;
```

```
00055     uint8_t Medium_Speed;
00056     uint8_t High_Speed;
00057     uint8_t Very_High_Speed;
00058     uint8_t None;
00059 } GPIO_Speed;
00060
00066 typedef struct {
00067     uint8_t No_Pull_Up_Down;
00068     uint8_t Pull_Up;
00069     uint8_t Pull_Down;
00070     uint8_t None;
00071 } GPIO_Pull;
00072
00078 typedef struct {
00079     uint8_t None;
00080     uint8_t Analog;
00081     uint8_t SYS;
00082     uint8_t MCO1;
00083     uint8_t MCO2;
00084     uint8_t RTC_REFIN;
00085     uint8_t TIM_1;
00086     uint8_t TIM_2;
00087     uint8_t TIM_3;
00088     uint8_t TIM_4;
00089     uint8_t TIM_5;
00090     uint8_t TIM_8;
00091     uint8_t TIM_9;
00092     uint8_t TIM_10;
00093     uint8_t TIM_11;
00094     uint8_t I2C_1;
00095     uint8_t I2C_2;
00096     uint8_t I2C_3;
00097     uint8_t SPI_1;
00098     uint8_t SPI_2;
00099     uint8_t I2S_2;
00100     uint8_t I2S_2EXT;
00101     uint8_t SPI_3;
00102     uint8_t I2S_EXT;
00103     uint8_t I2S_3;
00104     uint8_t USART_1;
00105     uint8_t USART_2;
00106     uint8_t USART_3;
00107     uint8_t I2S_3EXT;
00108     uint8_t USART_4;
00109     uint8_t UART_5;
00110     uint8_t USART_6;
00111     uint8_t CAN_1;
00112     uint8_t CAN_2;
00113     uint8_t TIM_12;
00114     uint8_t TIM_13;
00115     uint8_t TIM_14;
00116     uint8_t OTG_FS_1;
00117     uint8_t OYG_HS_1;
00118     uint8_t ETH_1;
00119     uint8_t FSMC_1;
00120     uint8_t SDIO_1;
00121     uint8_t OTG_FS_2;
00122     uint8_t DCMI_1;
00123     uint8_t EVENTOUT;
00124 } GPIO_Alternate_Function;
00125
00131 typedef struct {
00132     uint8_t RISING_EDGE;
00133     uint8_t FALLING_EDGE;
00134     uint8_t RISING_FALLING_EDGE;
00135 } GPIO_Interrupt_Edge;
00136
00142 static const struct GPIO_Configuration {
00143
00144     GPIO_Mode_Type Mode;
00145     GPIO_Output_Type Output_Type;
00146     GPIO_Speed Speed;
00147     GPIO_Pull Pull;
00148     GPIO_Alternate_Function Alternate_Functions;
00149     GPIO_Interrupt_Edge Interrupt_Edge;
00151 } GPIO_Configuration = {
00152     .Mode = {
00153         .Input = 0,
00154         .General_Purpose_Output = 1,
00155         .Alternate_Function = 2,
00156         .Analog = 3,
00157     },
00158     .Output_Type = {
00159         .Push_Pull = 0,
00160         .Open_Drain = 1,
00161         .None = 2,
00162     },
```

```
00163     .Speed = {
00164         .Low_Speed = 0,
00165         .Medium_Speed = 1,
00166         .High_Speed = 2,
00167         .Very_High_Speed = 3,
00168         .None = 4,
00169     },
00170     .Pull = {
00171         .No_Pull_Up_Down = 0,
00172         .Pull_Up = 1,
00173         .Pull_Down = 2,
00174         .None = 4,
00175     },
00176     .Alternate_Functions = {
00177         .None = 0,
00178         .Analog = 0,
00179         .SYS = 0,
00180         .MCO1 = 0,
00181         .MCO2 = 0,
00182         .RTC_REFIN = 1,
00183         .TIM_1 = 1,
00184         .TIM_2 = 1,
00185         .TIM_3 = 2,
00186         .TIM_4 = 2,
00187         .TIM_5 = 2,
00188         .TIM_8 = 3,
00189         .TIM_9 = 3,
00190         .TIM_10 = 3,
00191         .TIM_11 = 3,
00192         .I2C_1 = 4,
00193         .I2C_2 = 4,
00194         .I2C_3 = 4,
00195         .SPI_1 = 5,
00196         .SPI_2 = 5,
00197         .I2S_2 = 5,
00198         .I2S_2EXT = 5,
00199         .SPI_3 = 6,
00200         .I2S_EXT = 6,
00201         .I2S_3 = 6,
00202         .USART_1 = 7,
00203         .USART_2 = 7,
00204         .USART_3 = 7,
00205         .I2S_3EXT = 7,
00206         .USART_4 = 8,
00207         .UART_5 = 8,
00208         .USART_6 = 8,
00209         .CAN_1 = 9,
00210         .CAN_2 = 9,
00211         .TIM_12 = 9,
00212         .TIM_13 = 9,
00213         .TIM_14 = 9,
00214         .OTG_FS_1 = 10,
00215         .OYG_HS_1 = 10,
00216         .ETH_1 = 11,
00217         .FSMC_1 = 12,
00218         .SDIO_1 = 12,
00219         .OTG_FS_2 = 12,
00220         .DCMI_1 = 13,
00221         .EVENTOUT = 15,
00222     },
00223     .Interrupt_Edge = {
00224         .RISING_EDGE = 0,
00225         .FALLING_EDGE = 1,
00226         .RISING_FALLING_EDGE = 2,
00227     },
00228 };
00229
00230 #endif /* GPIO_DEFS_H_ */
```

Index