GPIO Driver

version 1.0.0

Generated by Doxygen 1.8.17

1 Data Structure Index 1

1.1 Data Structures . . . 1

2 File Index 3

2.1 File List . . . 3

3 Data Structure Documentation 5

3.1 GpioConfiguration Struct Reference . . . 5

3.1.1 Detailed Description . . . 5

4 File Documentation 7

4.1 gpio.c File Reference . . . 7

4.1.1 Function Documentation . . . 7

4.1.1.1 Gpio_Dir_get() . . . 7

4.1.1.2 Gpio_Dir_set() . . . 8

4.1.1.3 Gpio_Init() . . . 8

4.1.1.4 Gpio_PinRead() . . . 9

4.1.1.5 Gpio_PinWrite() . . . 10

4.2 gpio.h File Reference . . . 10

4.2.1 Function Documentation . . . 11

4.2.1.1 Gpio_Dir_get() . . . 11

4.2.1.2 Gpio_Dir_set() . . . 11

4.2.1.3 Gpio_Init() . . . 12

4.2.1.4 Gpio_PinRead() . . . 13

4.2.1.5 Gpio_PinWrite() . . . 13

4.3 gpio_config.c File Reference . . . 14

4.3.1 Variable Documentation . . . 14

4.3.1.1 cnfgTable . . . 14

4.4 gpio_config.h File Reference . . . 14

4.4.1 Enumeration Type Documentation . . . 15

4.4.1.1 GpioAnalogMide . . . 15

4.4.1.2 GpioDigitalEnable . . . 16

4.4.1.3 GpioInterruptLevel . . . 16

4.4.1.4 GpioInterruptMask . . . 16

4.4.1.5 GpioInterruptSense . . . 17

4.4.1.6 GpioInterruptStatus . . . 17

4.4.1.7 GpioPinDirection_t . . . 17

4.4.1.8 GpioPinState_t . . . 17

4.4.1.9 GpioPullupResistors . . . 18

4.5 gpioMap.h File Reference . . . 18

4.5.1 Variable Documentation . . . 20

4.5.1.1 GpioAFSEL . . . 20

4.5.1.2 GpioAMSEL . . . 20

4.5.1.3 GpioCR 20
4.5.1.4 GpioDataDIR 21
4.5.1.5 GpioDataReg 21
4.5.1.6 GpioDEN 21
4.5.1.7 GpioIBE 21
4.5.1.8 GpioICR 22
4.5.1.9 GpioIEV 22
4.5.1.10 GpioIM 22
4.5.1.11 GpioISR 22
4.5.1.12 GpioLock 23
4.5.1.13 GpioMIS 23
4.5.1.14 GpioPCTL 23
4.5.1.15 GpioPUR 23
4.5.1.16 GpioRIS 24

Index **25**

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

GpioConfiguration
This is the configuration table structure that will be used to set the configuration parameter of every GPIO pin by defining evey entry in this structure to a specific pin and pass this table to initializing function 5

Chapter 2

File Index

2.1 File List

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

gpio.c 7
gpio.h 10
gpio_config.c 14
gpio_config.h 14
gpioMap.h 18

Chapter 3

Data Structure Documentation

3.1 GpioConfiguration Struct Reference

This is the configuration table structure that will be used to set the configuration parameter of every GPIO pin by defining evey entry in this structure to a specific pin and pass this table to initializing function.

```
#include <gpio_config.h>
```

Data Fields

- GpioPorts port

  Choose the port that you want to configure it's pin.
- GpioPins pin

  Select the pin that you want to set it's parameters.
- GpioPinDirection_t direction

  set the direction of the pin as input or outpt
- GpioPullupResistors resistor

  configure if a pull up resistor is desired to be enabled on that pin
- GpioPinState_t state

  Set the initial digital state of the pin (High,Low)

3.1.1 Detailed Description

This is the configuration table structure that will be used to set the configuration parameter of every GPIO pin by defining evey entry in this structure to a specific pin and pass this table to initializing function.

Chapter 4

File Documentation

4.1 gpio.c File Reference

```
#include "gpio.h"
```

Functions

- void Gpio_Init (const GpioConfiguration *const cnfg_table, uint8_t cnfgTable_size)

  Description: *This function is used to initiaize GPIO pins based on the configuration parameters entered for every GPIO pin in the GPIO configuration table array that resides inside gpio_config.h module the function loop through the configuration table and configure GPIO pins based on the defined configuration selected in the configuration table*
- void Gpio_Dir_set (GpioPorts port, GpioPins pin, GpioPinDirection_t direction)

  Description: *This function is used to set the direction of a GPIO pin.*
- GpioPinDirection_t Gpio_Dir_get (GpioPorts port, GpioPins pin)

  Description: *This function is used to get the direction of a certain pin.*
- void Gpio_PinWrite (GpioPorts port, GpioPins pin, GpioPinState_t state)

  Description:This *functoin is used to output a signal on the GPIO pin this signal could be logical high or low signal output*
- int32_t Gpio_PinRead (GpioPorts port, GpioPins pin)

  Description:This *functoin is used to read the current signal value going into or form a GPIO pin*
- void **Gpio_PinToggle** (GpioPorts port, GpioPins pin)

4.1.1 Function Documentation

4.1.1.1 Gpio_Dir_get()

```
GpioPinDirection_t Gpio_Dir_get (
            GpioPorts port,
            GpioPins pin )
```

Description: This function is used to get the direction of a certain pin.

PRE-CONDITION:a call to Gpio_init function must have been made so the pin is actually initialized POST-COND↩ ITION:the direction state of the pin is returned from the function

Parameters

in	*Port*	is an enum variable that takes the port name that we want to check one of it's pins direction
in	*pin*	is an enum variable that takes the pin number that we want to check it's direction state

Returns

a GpioPinDirection_t enum varibale is returned which specifies the direction state of the pin

See also

Gpio_init

4.1.1.2 Gpio_Dir_set()

```
void Gpio_Dir_set (
            GpioPorts port,
            GpioPins pin,
            GpioPinDirection_t direction )
```

Description: This function is used to set the direction of a GPIO pin.

PRE-CONDITION:a call to Gpio_init function must have been made so the pin is actually initialized POST-COND↩ ITION:the specified pin will change it's direction state to input or output pin according to configuration made

Parameters

in	*Port*	is an enum variable that takes the name of the port that contain the pin we want to modify.
in	*pin*	is an enum variable that takes the pin number that we want to modify it's direction state.
in	*direction*	is an enum varibale that takes the direction state we want to set (input,output).

Returns

void

See also

Gpio_init

4.1.1.3 Gpio_Init()

```
void Gpio_Init (
            const GpioConfiguration *const cnfg_table,
            uint8_t cnfgTable_size )
```

Description: This function is used to initiaize GPIO pins based on the configuration parameters entered for every GPIO pin in the GPIO configuration table array that resides inside gpio_config.h module the function loop through the configuration table and configure GPIO pins based on the defined configuration selected in the configuration table

PRE-CONDITION:Configure the parameters of the GPIO pin in the configuration table in gpio_config.h module POST-CONDITION:All GPIO pins listed in the configuration table structure will be initialzied

Parameters

in	*cnfg_table*	is a constant pointer to a configuration table structure defined in gpio_conig.h module this pointer is used to refrence every defined configuration of every gpio pin inside the configuration structure table
in	*cnfgTable_size*	is a uint8 varibale that hold the size of the configuration table array so the function could loop through every index of the table to initialize the coressponding defined GPIO pin

Returns

void

See also

Gpio_PinWrite

Gpio_PinRead

Gpio_PinToggle

Gpio_Dir_set

Gpio_Dir_get

4.1.1.4 Gpio_PinRead()

```
int32_t Gpio_PinRead (
            GpioPorts port,
            GpioPins pin )
```

Description:This functoin is used to read the current signal value going into or form a GPIO pin

PRE-CONDITION:a call to Gpio_init function must have been made so the pin is actually initialized POST-COND↩ ITION:the current signal value of a Gpio pin will be returned

Parameters

in	*port*	is an enum variable that takes the port name
in	*pin*	is an enum varibale that takes the pin number that we want to read it's signal value

Returns

the signal value corresponding to that pin is returned in an integer form (1,0)

See also

Gpio_init

4.1.1.5 Gpio_PinWrite()

```
void Gpio_PinWrite (
            GpioPorts port,
            GpioPins pin,
            GpioPinState_t state )
```

Description:This functoin is used to output a signal on the GPIO pin this signal could be logical high or low signal output

PRE-CONDITION:a call to Gpio_init function must have been made so the pin is actually initialized POST-COND↩ ITION:the configured GPIO pin will now output a signal based on the configured output value

Parameters

in	*port*	is an enum variable that takes the port name that we want to output a signal on one of it's pins
in	*pin*	is an enum varibale that takes the pin number that we want to output a signal to
in	*stae*	is an enum variable that takes the state of the signal that we want to output on the pin this signal could be high or low signal

Returns

void

See also

Gpio_init

4.2 gpio.h File Reference

```
#include "gpioMap.h"
```

Functions

- void Gpio_Init (const GpioConfiguration *const cnfg_table, uint8_t cnfgTable_size)

  Description: *This function is used to initiaize GPIO pins based on the configuration parameters entered for every GPIO pin in the GPIO configuration table array that resides inside gpio_config.h module the function loop through the configuration table and configure GPIO pins based on the defined configuration selected in the configuration table*
- void Gpio_Dir_set (GpioPorts port, GpioPins pin, GpioPinDirection_t direction)

  Description: *This function is used to set the direction of a GPIO pin.*
- GpioPinDirection_t Gpio_Dir_get (GpioPorts port, GpioPins pin)

  Description: *This function is used to get the direction of a certain pin.*

- void Gpio_PinWrite (GpioPorts port, GpioPins pin, GpioPinState_t state)

  Description:This *functoin is used to output a signal on the GPIO pin this signal could be logical high or low signal output*
- void **Gpio_PinToggle** (GpioPorts port, GpioPins pin)
- int32_t Gpio_PinRead (GpioPorts port, GpioPins pin)

  Description:This *functoin is used to read the current signal value going into or form a GPIO pin*

4.2.1 Function Documentation

4.2.1.1 Gpio_Dir_get()

```
GpioPinDirection_t Gpio_Dir_get (
            GpioPorts port,
            GpioPins pin )
```

Description: This function is used to get the direction of a certain pin.

PRE-CONDITION:a call to Gpio_init function must have been made so the pin is actually initialized POST-COND↩ ITION:the direction state of the pin is returned from the function

Parameters

in	*Port*	is an enum variable that takes the port name that we want to check one of it's pins direction
in	*pin*	is an enum variable that takes the pin number that we want to check it's direction state

Returns

a GpioPinDirection_t enum varibale is returned which specifies the direction state of the pin

See also

Gpio_init

4.2.1.2 Gpio_Dir_set()

```
void Gpio_Dir_set (
            GpioPorts port,
            GpioPins pin,
            GpioPinDirection_t direction )
```

Description: This function is used to set the direction of a GPIO pin.

PRE-CONDITION:a call to Gpio_init function must have been made so the pin is actually initialized POST-COND↩ ITION:the specified pin will change it's direction state to input or output pin according to configuration made

Parameters

in	*Port*	is an enum variable that takes the name of the port that contain the pin we want to modify.
in	*pin*	is an enum variable that takes the pin number that we want to modify it's direction state.
in	*direction*	is an enum varibale that takes the direction state we want to set (input,output).

Returns

void

See also

Gpio_init

4.2.1.3 Gpio_Init()

```
void Gpio_Init (
            const GpioConfiguration *const cnfg_table,
            uint8_t cnfgTable_size )
```

Description: This function is used to initiaize GPIO pins based on the configuration parameters entered for every GPIO pin in the GPIO configuration table array that resides inside gpio_config.h module the function loop through the configuration table and configure GPIO pins based on the defined configuration selected in the configuration table

PRE-CONDITION:Configure the parameters of the GPIO pin in the configuration table in gpio_config.h module
POST-CONDITION:All GPIO pins listed in the configuration table structure will be initialzied

Parameters

in	*cnfg_table*	is a constant pointer to a configuration table structure defined in gpio_conig.h module this pointer is used to refrence every defined configuration of every gpio pin inside the configuration structure table
in	*cnfgTable_size*	is a uint8 varibale that hold the size of the configuration table array so the function could loop through every index of the table to initialize the coressponding defined GPIO pin

Returns

void

See also

Gpio_PinWrite
Gpio_PinRead
Gpio_PinToggle
Gpio_Dir_set
Gpio_Dir_get

4.2.1.4 Gpio_PinRead()

```
int32_t Gpio_PinRead (
            GpioPorts port,
            GpioPins pin )
```

Description:This functoin is used to read the current signal value going into or form a GPIO pin

PRE-CONDITION:a call to Gpio_init function must have been made so the pin is actually initialized POST-COND↩ ITION:the current signal value of a Gpio pin will be returned

Parameters

in	*port*	is an enum variable that takes the port name
in	*pin*	is an enum varibale that takes the pin number that we want to read it's signal value

Returns

the signal value corresponding to that pin is returned in an integer form (1,0)

See also

Gpio_init

4.2.1.5 Gpio_PinWrite()

```
void Gpio_PinWrite (
            GpioPorts port,
            GpioPins pin,
            GpioPinState_t state )
```

Description:This functoin is used to output a signal on the GPIO pin this signal could be logical high or low signal output

PRE-CONDITION:a call to Gpio_init function must have been made so the pin is actually initialized POST-COND↩ ITION:the configured GPIO pin will now output a signal based on the configured output value

Parameters

in	*port*	is an enum variable that takes the port name that we want to output a signal on one of it's pins
in	*pin*	is an enum varibale that takes the pin number that we want to output a signal to
in	*stae*	is an enum variable that takes the state of the signal that we want to output on the pin this signal could be high or low signal

Returns

void

See also

Gpio_init

4.3 gpio_config.c File Reference

```
#include "gpio_config.h"
```

Functions

- const GpioConfiguration * **config_get** (void)

Variables

- const GpioConfiguration cnfgTable [ ]

  The following is the Gpio configuration table it is simply an array of Gpioconfiguration structure where every entry in this array is a structure that will configure a GPIO pin according to the entered parameters in the structure.

4.3.1 Variable Documentation

4.3.1.1 cnfgTable

```
const GpioConfiguration cnfgTable[ ]
```

Initial value:

```
= {
{ Portf, pin3, gpio_output, gpio_pullupdisable, gpio_high },
{ Portf, pin1, gpio_output, gpio_pullupdisable, gpio_high },
{ Portf, pin2, gpio_output, gpio_pullupdisable, gpio_high }, }
```

The following is the Gpio configuration table it is simply an array of Gpioconfiguration structure where every entry in this array is a structure that will configure a GPIO pin according to the entered parameters in the structure.

4.4 gpio_config.h File Reference

```
#include <stdint.h>
#include <stdlib.h>
```

Data Structures

- struct GpioConfiguration

  This is the configuration table structure that will be used to set the configuration parameter of every GPIO pin by defining evey entry in this structure to a specific pin and pass this table to initializing function.

Macros

- #define **NumberOfPorts** 8
- #define **NumberOfPins** 8

Enumerations

- enum GpioPorts {
  PortA, **Portb**, **Portc**, **Portd**,
  Porte, **Portf** }

  Defining all GPIO PORTS that exists on the Board.
- enum GpioPins {
  pin0, **pin1**, **pin2**, **pin3**,
  pin4, **pin5**, **pin6**, **pin7** }

  Defining every pin that exist in every GPIO port.
- enum GpioPinState_t { gpio_low, gpio_high }

  Defining The possible Logical states of any GPIO pin.
- enum GpioPinDirection_t { gpio_input, gpio_output }

  Defining Possible directions of a GPIO pin.
- enum GpioPullupResistors { gpio_pullupdisable, gpio_pullupenable }

  Configuring GPIO pin to have a pull u resistor.
- enum GpioDigitalEnable { gpio_digitalDisable, gpio_digitalEnable }

  Configure GPIO pin to detec digital signals logic.
- enum GpioAnalogMide { gpio_analogDisable, gpio_analogEnable }

  Configure GPIO pin to detect analog signal by working in Analog mode.
- enum GpioInterruptStatus { gpio_noInterrupt, gpio_triggeredInterrupt }

  Defining Interrupt status of a GPIO pin.
- enum GpioInterruptSense { gpio_edgeSensetive, gpio_levelSensetive }

  Defining Type of signal to fire an interrupt on a GPIO pin.
- enum GpioInterruptLevel { gpio_fallingEdge, gpio_risingEdge, gpio_bothEdges }

  Defining which signal edge to cause an interrupt if GPIO pin configured as an edge triggered.
- enum GpioInterruptMask { gpio_maskInterrupt, gpio_trggerInterrupt }

  Defining Interrupt Mask.

Functions

- const GpioConfiguration * **config_get** (void)

4.4.1 Enumeration Type Documentation

4.4.1.1 GpioAnalogMide

```
enum GpioAnalogMide
```

Configure GPIO pin to detect analog signal by working in Analog mode.

Enumerator

gpio_analogDisable	analog mode disabled (GPIO can't detect analog signals)
gpio_analogEnable	analog mode enabled (GPIO can detect analog signals)

4.4.1.2 GpioDigitalEnable

`enum GpioDigitalEnable`

Configure GPIO pin to detec digital signals logic.

Enumerator

gpio_digitalDisable	digital function is disabled on this GPIO (can't detect any digital Logic)
gpio_digitalEnable	digital function is enabled on this GPIO (can detect digital Logic)

4.4.1.3 GpioInterruptLevel

`enum GpioInterruptLevel`

Defining which signal edge to cause an interrupt if GPIO pin configured as an edge triggered.

Enumerator

gpio_fallingEdge	interrupt is triggred on falling edge of signal
gpio_risingEdge	interrupt is triggered on rising edge of signal
gpio_bothEdges	Interrupt occur on any edge change.

4.4.1.4 GpioInterruptMask

`enum GpioInterruptMask`

Defining Interrupt Mask.

Enumerator

gpio_maskInterrupt	mask the Activated interrupt
gpio_trggerInterrupt	send the interrupt signal to the NVIC

4.4.1.5 GpioInterruptSense

`enum GpioInterruptSense`

Defining Type of signal to fire an interrupt on a GPIO pin.

Enumerator

gpio_edgeSensetive	set GPIO to detect edge of any signal on it's pin to generate an interrupt
gpio_levelSensetive	set GPIO to detect Level of any signal on it's pin to generate an interrupt

4.4.1.6 GpioInterruptStatus

`enum GpioInterruptStatus`

Defining Interrupt status of a GPIO pin.

Enumerator

gpio_noInterrupt	No interrupt is active corresponding to that GPIO pin.
gpio_triggeredInterrupt	An interrupt is active corresponding to that GPIO pin.

4.4.1.7 GpioPinDirection_t

`enum GpioPinDirection_t`

Defining Possible directions of a GPIO pin.

Enumerator

gpio_input	set GPIO pin as input pin (0)
gpio_output	set GPIO pin as output pin (1)

4.4.1.8 GpioPinState_t

`enum GpioPinState_t`

Defining The possible Logical states of any GPIO pin.

Enumerator

gpio_low	set GPIO pin to low state (0)
gpio_high	set GPIO pin to high state(1)

4.4.1.9 GpioPullupResistors

```
enum GpioPullupResistors
```

Configuring GPIO pin to have a pull u resistor.

Enumerator

gpio_pullupdisable	disable pull up resistor for that GPIO
gpio_pullupenable	enable pull up resistor for that GPIO

4.5 gpioMap.h File Reference

```
#include "gpio_config.h"
```

Macros

- #define GPIO_O_DATA 0x00000000
  The following are defines for the GPIO register offsets.
- #define **GPIO_O_DIR** 0x00000400
- #define **GPIO_O_IS** 0x00000404
- #define **GPIO_O_IBE** 0x00000408
- #define **GPIO_O_IEV** 0x0000040C
- #define **GPIO_O_IM** 0x00000410
- #define **GPIO_O_RIS** 0x00000414
- #define **GPIO_O_MIS** 0x00000418
- #define **GPIO_O_ICR** 0x0000041C
- #define **GPIO_O_AFSEL** 0x00000420
- #define **GPIO_O_DR2R** 0x00000500
- #define **GPIO_O_DR4R** 0x00000504
- #define **GPIO_O_DR8R** 0x00000508
- #define **GPIO_O_ODR** 0x0000050C
- #define **GPIO_O_PUR** 0x00000510
- #define **GPIO_O_PDR** 0x00000514
- #define **GPIO_O_SLR** 0x00000518
- #define **GPIO_O_DEN** 0x0000051C
- #define **GPIO_O_LOCK** 0x00000520
- #define **GPIO_O_CR** 0x00000524
- #define **GPIO_O_AMSEL** 0x00000528

- #define **GPIO_O_PCTL** 0x0000052C
- #define **GPIO_O_ADCCTL** 0x00000530
- #define **GPIO_O_DMACTL** 0x00000534
- #define **GPIO_O_SI** 0x00000538
- #define **GPIO_O_DR12R** 0x0000053C
- #define **GPIO_O_WAKEPEN** 0x00000540
- #define **GPIO_O_WAKELVL** 0x00000544
- #define **GPIO_O_WAKESTAT** 0x00000548
- #define **GPIO_O_PP** 0x00000FC0
- #define **GPIO_O_PC** 0x00000FC4
- #define PortA 0x40004000

  The following are gpio register base address.
- #define **PortB** 0x40005000
- #define **PortC** 0x40006000
- #define **PortD** 0x40007000
- #define **PortE** 0x40024000
- #define **PortF** 0x40025000
- #define RCGCGPIO ∗((volatile uint32_t∗)(0x400FE608))

  clock register for gpio

Variables

- static volatile uint32_t ∗const GpioDataReg [NumberOfPorts]

  The following are pointer array that maps to the data register of every Gpio port.
- static volatile uint32_t ∗const GpioDataDIR [NumberOfPorts]

  The following are pointer array that maps to the data direction register of every Gpio port.
- static volatile uint32_t ∗const GpioISR [NumberOfPorts]

  The following are pointer array that maps to the interrupt sense register of every Gpio port.
- static volatile uint32_t ∗const GpioIBE [NumberOfPorts]

  The following are pointer array that maps to the interrupt both edges register of every Gpio port.
- static volatile uint32_t ∗const GpioIEV [NumberOfPorts]

  The following are pointer array that maps to the interrupt event register of every Gpio port.
- static volatile uint32_t ∗const GpioIM [NumberOfPorts]

  The following are pointer array that maps to the interrupt Mask register of every Gpio port.
- static volatile uint32_t ∗const GpioRIS [NumberOfPorts]

  The following are pointer array that maps to the interrupt status register of every Gpio port.
- static volatile uint32_t ∗const GpioMIS [NumberOfPorts]

  The following are pointer array that maps to the Masked interrupt status register of every Gpio port.
- static volatile uint32_t ∗const GpioICR [NumberOfPorts]

  The following are pointer array that maps to the interrupt clear register of every Gpio port.
- static volatile uint32_t ∗const GpioAFSEL [NumberOfPorts]

  The following are pointer array that maps to the Alternate function select register of every Gpio port.
- static volatile uint32_t ∗const GpioPUR [NumberOfPorts]

  The following are pointer array that maps to the Pull up Resistor select register of every Gpio port.
- static volatile uint32_t ∗const GpioDEN [NumberOfPorts]

  The following are pointer array that maps to the Digital Enable register of every Gpio port.
- static volatile uint32_t ∗const GpioLock [NumberOfPorts]

  The following are pointer array that maps to the GPIO Lock register of every Gpio port.
- static volatile uint32_t ∗const GpioCR [NumberOfPorts]

  The following are pointer array that maps to the GPIO Commit register of every Gpio port.
- static volatile uint32_t ∗const GpioAMSEL [NumberOfPorts]

  The following are pointer array that maps to the Analog Mode select of every Gpio port.
- static volatile uint32_t ∗const GpioPCTL [NumberOfPorts]

  The following are pointer array that maps to the Port control register of every Gpio port.

4.5.1 Variable Documentation

4.5.1.1 GpioAFSEL

```
volatile uint32_t* const GpioAFSEL[NumberOfPorts]  [static]
```

Initial value:

```
={
(uint32_t*)((PortA)+(GPIO_O_AFSEL)),
(uint32_t*)((PortB)+(GPIO_O_AFSEL)),
(uint32_t*)((PortC)+(GPIO_O_AFSEL)),
(uint32_t*)((PortD)+(GPIO_O_AFSEL)),
(uint32_t*)((PortE)+(GPIO_O_AFSEL)),
(uint32_t*)((PortF)+(GPIO_O_AFSEL)),
}
```

The following are pointer array that maps to the Alternate function select register of every Gpio port.

4.5.1.2 GpioAMSEL

```
volatile uint32_t* const GpioAMSEL[NumberOfPorts]  [static]
```

Initial value:

```
={
(uint32_t*)((PortA)+(GPIO_O_AMSEL)),
(uint32_t*)((PortB)+(GPIO_O_AMSEL)),
(uint32_t*)((PortC)+(GPIO_O_AMSEL)),
(uint32_t*)((PortD)+(GPIO_O_AMSEL)),
(uint32_t*)((PortE)+(GPIO_O_AMSEL)),
(uint32_t*)((PortF)+(GPIO_O_AMSEL)),
}
```

The following are pointer array that maps to the Analog Mode select of every Gpio port.

4.5.1.3 GpioCR

```
volatile uint32_t* const GpioCR[NumberOfPorts]  [static]
```

Initial value:

```
={
(uint32_t*)((PortA)+(GPIO_O_CR)),
(uint32_t*)((PortB)+(GPIO_O_CR)),
(uint32_t*)((PortC)+(GPIO_O_CR)),
(uint32_t*)((PortD)+(GPIO_O_CR)),
(uint32_t*)((PortE)+(GPIO_O_CR)),
(uint32_t*)((PortF)+(GPIO_O_CR)),
}
```

The following are pointer array that maps to the GPIO Commit register of every Gpio port.

4.5.1.4 GpioDataDIR

```
volatile uint32_t* const GpioDataDIR[NumberOfPorts]  [static]
```

Initial value:

```
={
(uint32_t*)((PortA)+(GPIO_O_DIR)),
(uint32_t*)((PortB)+(GPIO_O_DIR)),
(uint32_t*)((PortC)+(GPIO_O_DIR)),
(uint32_t*)((PortD)+(GPIO_O_DIR)),
(uint32_t*)((PortE)+(GPIO_O_DIR)),
(uint32_t*)((PortF)+(GPIO_O_DIR)),
}
```

The following are pointer array that maps to the data direction register of every Gpio port.

4.5.1.5 GpioDataReg

```
volatile uint32_t* const GpioDataReg[NumberOfPorts]  [static]
```

Initial value:

```
={
(uint32_t*)((PortA)+(GPIO_O_DATA)+(0x3fc)),
(uint32_t*)((PortB)+(GPIO_O_DATA)+(0x3fc)),
(uint32_t*)((PortC)+(GPIO_O_DATA)+(0x3fc)),
(uint32_t*)((PortD)+(GPIO_O_DATA)+(0x3fc)),
(uint32_t*)((PortE)+(GPIO_O_DATA)+(0x3fc)),
(uint32_t*)((PortF)+(GPIO_O_DATA)+(0x3fc)),
}
```

The following are pointer array that maps to the data register of every Gpio port.

4.5.1.6 GpioDEN

```
volatile uint32_t* const GpioDEN[NumberOfPorts]  [static]
```

Initial value:

```
={
(uint32_t*)((PortA)+(GPIO_O_DEN)),
(uint32_t*)((PortB)+(GPIO_O_DEN)),
(uint32_t*)((PortC)+(GPIO_O_DEN)),
(uint32_t*)((PortD)+(GPIO_O_DEN)),
(uint32_t*)((PortE)+(GPIO_O_DEN)),
(uint32_t*)((PortF)+(GPIO_O_DEN)),
}
```

The following are pointer array that maps to the Digital Enable register of every Gpio port.

4.5.1.7 GpioIBE

```
volatile uint32_t* const GpioIBE[NumberOfPorts]  [static]
```

Initial value:

```
={
(uint32_t*)((PortA)+(GPIO_O_IBE)),
(uint32_t*)((PortB)+(GPIO_O_IBE)),
(uint32_t*)((PortC)+(GPIO_O_IBE)),
(uint32_t*)((PortD)+(GPIO_O_IBE)),
(uint32_t*)((PortE)+(GPIO_O_IBE)),
(uint32_t*)((PortF)+(GPIO_O_IBE)),
}
```

The following are pointer array that maps to the interrupt both edges register of every Gpio port.

4.5.1.8 GpioICR

```
volatile uint32_t* const GpioICR[NumberOfPorts]  [static]
```

Initial value:

```
={
(uint32_t*)((PortA)+(GPIO_O_ICR)),
(uint32_t*)((PortB)+(GPIO_O_ICR)),
(uint32_t*)((PortC)+(GPIO_O_ICR)),
(uint32_t*)((PortD)+(GPIO_O_ICR)),
(uint32_t*)((PortE)+(GPIO_O_ICR)),
(uint32_t*)((PortF)+(GPIO_O_ICR)),
}
```

The following are pointer array that maps to the interrupt clear register of every Gpio port.

4.5.1.9 GpioIEV

```
volatile uint32_t* const GpioIEV[NumberOfPorts]  [static]
```

Initial value:

```
={
(uint32_t*)((PortA)+(GPIO_O_IEV)),
(uint32_t*)((PortB)+(GPIO_O_IEV)),
(uint32_t*)((PortC)+(GPIO_O_IEV)),
(uint32_t*)((PortD)+(GPIO_O_IEV)),
(uint32_t*)((PortE)+(GPIO_O_IEV)),
(uint32_t*)((PortF)+(GPIO_O_IEV)),
}
```

The following are pointer array that maps to the interrupt event register of every Gpio port.

4.5.1.10 GpioIM

```
volatile uint32_t* const GpioIM[NumberOfPorts]  [static]
```

Initial value:

```
={
(uint32_t*)((PortA)+(GPIO_O_IM)),
(uint32_t*)((PortB)+(GPIO_O_IM)),
(uint32_t*)((PortC)+(GPIO_O_IM)),
(uint32_t*)((PortD)+(GPIO_O_IM)),
(uint32_t*)((PortE)+(GPIO_O_IM)),
(uint32_t*)((PortF)+(GPIO_O_IM)),
}
```

The following are pointer array that maps to the interrupt Mask register of every Gpio port.

4.5.1.11 GpioISR

```
volatile uint32_t* const GpioISR[NumberOfPorts]  [static]
```

Initial value:

```
={
(uint32_t*)((PortA)+(GPIO_O_IS)),
(uint32_t*)((PortB)+(GPIO_O_IS)),
(uint32_t*)((PortC)+(GPIO_O_IS)),
(uint32_t*)((PortD)+(GPIO_O_IS)),
(uint32_t*)((PortE)+(GPIO_O_IS)),
(uint32_t*)((PortF)+(GPIO_O_IS)),
}
```

The following are pointer array that maps to the interrupt sense register of every Gpio port.

4.5.1.12 GpioLock

```
volatile uint32_t* const GpioLock[NumberOfPorts]  [static]
```

Initial value:

```
={
(uint32_t*)((PortA)+(GPIO_O_LOCK)),
(uint32_t*)((PortB)+(GPIO_O_LOCK)),
(uint32_t*)((PortC)+(GPIO_O_LOCK)),
(uint32_t*)((PortD)+(GPIO_O_LOCK)),
(uint32_t*)((PortE)+(GPIO_O_LOCK)),
(uint32_t*)((PortF)+(GPIO_O_LOCK)),
}
```

The following are pointer array that maps to the GPIO Lock register of every Gpio port.

4.5.1.13 GpioMIS

```
volatile uint32_t* const GpioMIS[NumberOfPorts]  [static]
```

Initial value:

```
={
(uint32_t*)((PortA)+(GPIO_O_MIS)),
(uint32_t*)((PortB)+(GPIO_O_MIS)),
(uint32_t*)((PortC)+(GPIO_O_MIS)),
(uint32_t*)((PortD)+(GPIO_O_MIS)),
(uint32_t*)((PortE)+(GPIO_O_MIS)),
(uint32_t*)((PortF)+(GPIO_O_MIS)),
}
```

The following are pointer array that maps to the Masked interrupt status register of every Gpio port.

4.5.1.14 GpioPCTL

```
volatile uint32_t* const GpioPCTL[NumberOfPorts]  [static]
```

Initial value:

```
={
(uint32_t*)((PortA)+(GPIO_O_PCTL)),
(uint32_t*)((PortB)+(GPIO_O_PCTL)),
(uint32_t*)((PortC)+(GPIO_O_PCTL)),
(uint32_t*)((PortD)+(GPIO_O_PCTL)),
(uint32_t*)((PortE)+(GPIO_O_PCTL)),
(uint32_t*)((PortF)+(GPIO_O_PCTL)),
}
```

The following are pointer array that maps to the Port control register of every Gpio port.

4.5.1.15 GpioPUR

```
volatile uint32_t* const GpioPUR[NumberOfPorts]  [static]
```

Initial value:

```
={
(uint32_t*)((PortA)+(GPIO_O_PUR)),
(uint32_t*)((PortB)+(GPIO_O_PUR)),
(uint32_t*)((PortC)+(GPIO_O_PUR)),
(uint32_t*)((PortD)+(GPIO_O_PUR)),
(uint32_t*)((PortE)+(GPIO_O_PUR)),
(uint32_t*)((PortF)+(GPIO_O_PUR)),
}
```

The following are pointer array that maps to the Pull up Resistor select register of every Gpio port.

4.5.1.16 GpioRIS

```
volatile uint32_t* const GpioRIS[NumberOfPorts]  [static]
```

Initial value:

```
={
(uint32_t*)((PortA)+(GPIO_O_RIS)),
(uint32_t*)((PortB)+(GPIO_O_RIS)),
(uint32_t*)((PortC)+(GPIO_O_RIS)),
(uint32_t*)((PortD)+(GPIO_O_RIS)),
(uint32_t*)((PortE)+(GPIO_O_RIS)),
(uint32_t*)((PortF)+(GPIO_O_RIS)),
}
```

The following are pointer array that maps to the interrupt status register of every Gpio port.

Index

cnfgTable
    gpio_config.c, 14

gpio.c, 7
    Gpio_Dir_get, 7
    Gpio_Dir_set, 8
    Gpio_Init, 8
    Gpio_PinRead, 9
    Gpio_PinWrite, 10
gpio.h, 10
    Gpio_Dir_get, 11
    Gpio_Dir_set, 11
    Gpio_Init, 12
    Gpio_PinRead, 12
    Gpio_PinWrite, 13
gpio_analogDisable
    gpio_config.h, 16
gpio_analogEnable
    gpio_config.h, 16
gpio_bothEdges
    gpio_config.h, 16
gpio_config.c, 14
    cnfgTable, 14
gpio_config.h, 14
    gpio_analogDisable, 16
    gpio_analogEnable, 16
    gpio_bothEdges, 16
    gpio_digitalDisable, 16
    gpio_digitalEnable, 16
    gpio_edgeSensetive, 17
    gpio_fallingEdge, 16
    gpio_high, 18
    gpio_input, 17
    gpio_levelSensetive, 17
    gpio_low, 18
    gpio_maskInterrupt, 16
    gpio_noInterrupt, 17
    gpio_output, 17
    gpio_pullupdisable, 18
    gpio_pullupenable, 18
    gpio_risingEdge, 16
    gpio_trggerInterrupt, 16
    gpio_triggeredInterrupt, 17
    GpioAnalogMide, 15
    GpioDigitalEnable, 16
    GpioInterruptLevel, 16
    GpioInterruptMask, 16
    GpioInterruptSense, 16
    GpioInterruptStatus, 17
    GpioPinDirection_t, 17
    GpioPinState_t, 17
    GpioPullupResistors, 18
gpio_digitalDisable
    gpio_config.h, 16
gpio_digitalEnable
    gpio_config.h, 16
Gpio_Dir_get
    gpio.c, 7
    gpio.h, 11
Gpio_Dir_set
    gpio.c, 8
    gpio.h, 11
gpio_edgeSensetive
    gpio_config.h, 17
gpio_fallingEdge
    gpio_config.h, 16
gpio_high
    gpio_config.h, 18
Gpio_Init
    gpio.c, 8
    gpio.h, 12
gpio_input
    gpio_config.h, 17
gpio_levelSensetive
    gpio_config.h, 17
gpio_low
    gpio_config.h, 18
gpio_maskInterrupt
    gpio_config.h, 16
gpio_noInterrupt
    gpio_config.h, 17
gpio_output
    gpio_config.h, 17
Gpio_PinRead
    gpio.c, 9
    gpio.h, 12
Gpio_PinWrite
    gpio.c, 10
    gpio.h, 13
gpio_pullupdisable
    gpio_config.h, 18
gpio_pullupenable
    gpio_config.h, 18
gpio_risingEdge
    gpio_config.h, 16
gpio_trggerInterrupt
    gpio_config.h, 16
gpio_triggeredInterrupt
    gpio_config.h, 17

GpioAFSEL
gpioMap.h, 20
GpioAMSEL
gpioMap.h, 20
GpioAnalogMide
gpio_config.h, 15
GpioConfiguration, 5
GpioCR
gpioMap.h, 20
GpioDataDIR
gpioMap.h, 20
GpioDataReg
gpioMap.h, 21
GpioDEN
gpioMap.h, 21
GpioDigitalEnable
gpio_config.h, 16
GpioIBE
gpioMap.h, 21
GpioICR
gpioMap.h, 21
GpioIEV
gpioMap.h, 22
GpioIM
gpioMap.h, 22
GpioInterruptLevel
gpio_config.h, 16
GpioInterruptMask
gpio_config.h, 16
GpioInterruptSense
gpio_config.h, 16
GpioInterruptStatus
gpio_config.h, 17
GpioISR
gpioMap.h, 22
GpioLock
gpioMap.h, 22
gpioMap.h, 18
GpioAFSEL, 20
GpioAMSEL, 20
GpioCR, 20
GpioDataDIR, 20
GpioDataReg, 21
GpioDEN, 21
GpioIBE, 21
GpioICR, 21
GpioIEV, 22
GpioIM, 22
GpioISR, 22
GpioLock, 22
GpioMIS, 23
GpioPCTL, 23
GpioPUR, 23
GpioRIS, 23
GpioMIS
gpioMap.h, 23
GpioPCTL
gpioMap.h, 23
GpioPinDirection_t
gpio_config.h, 17
GpioPinState_t
gpio_config.h, 17
GpioPullupResistors
gpio_config.h, 18
GpioPUR
gpioMap.h, 23
GpioRIS
gpioMap.h, 23