

# DIGITAL\_IO

Home

## Apps

Here is a list of all modules:

- License Terms and Copyright Information
  - Abbreviations and Definitions
  - Overview
  - Architecture Description
  - APP Configuration Parameters
  - Enumerations
  - Data structures
  - Methods
  - Usage
  - Release History
- 
- 
-

# DIGITAL\_IO

Home

## License Terms and Copyright Information

### License Terms and Copyright Information

Copyright (c) 2015, Infineon Technologies AG All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. Neither the name of the copyright holders nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT

(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

To improve the quality of the software, users are encouraged to share modifications, enhancements or bug fixes with Infineon Technologies AG ([dave@infineon.com](mailto:dave@infineon.com)).

---

# DIGITAL\_IO

Home

## Abbreviations and Definitions

### Abbreviations and Definitions

Abbreviations:	
DAVE™	Digital Application Virtual Engineer
APP	DAVE™ Application
API	Application Program Interface
GUI	Graphical User Interface
MCU	Microcontroller Unit
SW	Software
HW	Hardware
LLD	Low Level Driver
IO	Input Output
GPIO	General Purpose Input Output

Definitions:	
Singleton	Only single instance of the APP is permitted
Sharable	Resource sharing with other APPs permitted
initProvider	Provides the initialization routine
Physical connectivity	Hardware inter/intra peripheral (constant) signal connection
Conditional connectivity	Constrained hardware inter/intra peripheral signal connection
Aggregation	Indicates consumption of low level (dependent) DAVE™ APPs



# DIGITAL\_IO

Home

## Overview

### Overview

The **DIGITAL\_IO** APP provides the following functionalities:

- Configure and control a port pin as general purpose Input or Input/Output.
- Controls various IO pad characteristics.
- Connection of port pin to other peripherals (using the DAVE™ HW Signal Connections).

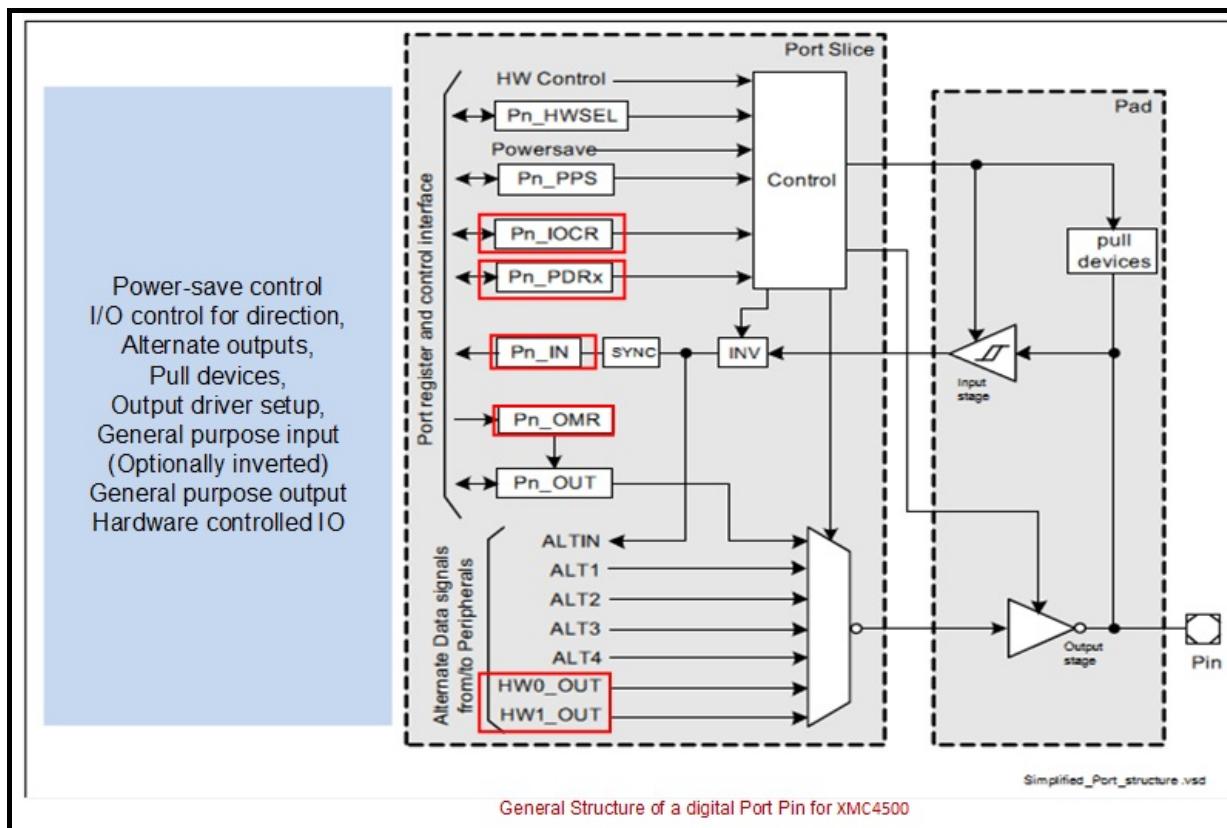
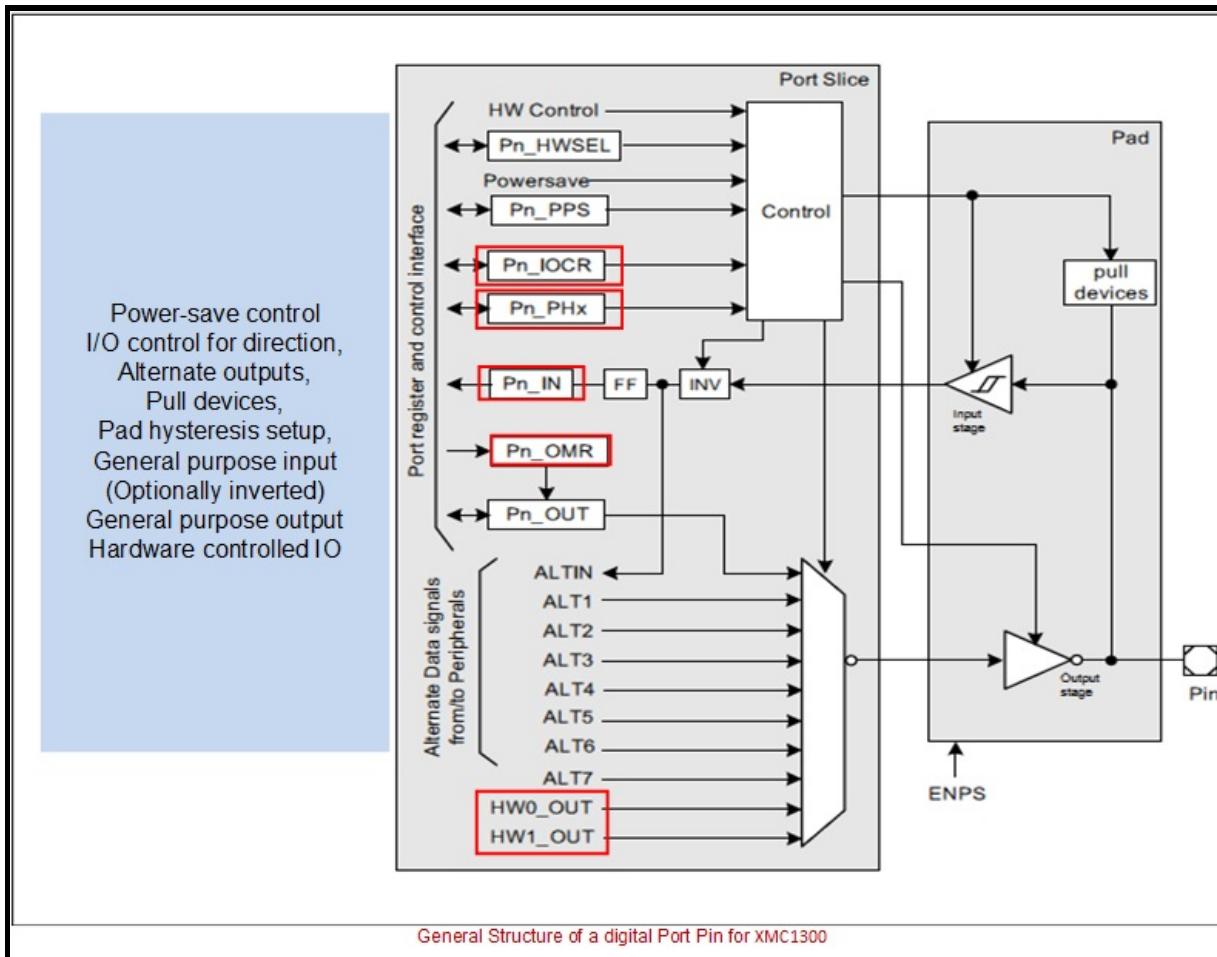


Figure 1



**Figure 2**

The red marked registers shown in above figures are modified by the **DIGITAL\_IO APP**.

### Input Operation :

Reads actual voltage level at the port pin and it can be inverted optionally.

- Configures internal weak pull-up or pull down devices in the pad.
- Configures input pad hysteresis.

### Input/Output Operation :

The output driver is activated and drives the value supplied through the output multiplexer to the port pin.

- Configures between open-drain and push-pull.
- Configures pad driver strength.
- Please refer APP Configuration Parameters section for more information.

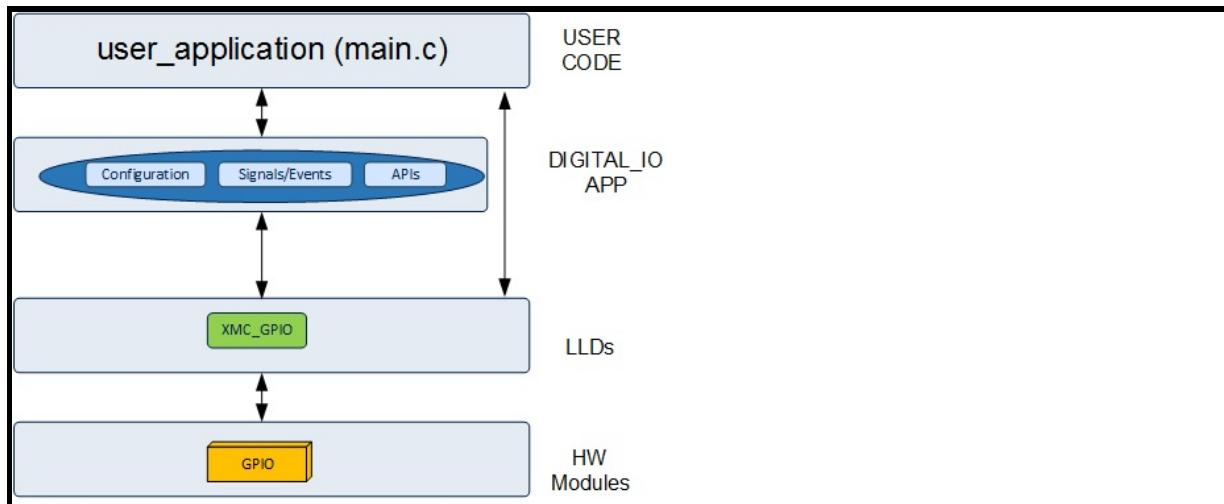
### **Hardware Controlled operation :**

The pin will be controlled by the peripheral to which it is connected.

- Configures of HWSEL bitfield which selects the peripheral that drives the pin.
- Configures pad driver strength.
- Configures input pad hysteresis.
- Configures pin input data polarity.

Additionally **DIGITAL\_IO** APP provides APIs to configure port pin.

Figure 3, shows how the APP is structured in DAVE™. XMC controllers provide the PORT module to configure pin. The LLD layer provides abstraction for these hardware modules. The **DIGITAL\_IO** APP uses XMC\_GPIO LLDs for the functionality.



**Figure 3 : Hardware and Software connectivity of **DIGITAL\_IO** APP**

**Note:**This APP is intended to configure digital I/O ports, however for analog I/O operation refer ANALOG\_IO APP.

### **Supported Devices**

*The APP supports below devices:*

1. XMC4800 / XMC4700 / XMC4300 Series
2. XMC4500 Series
3. XMC4400 Series
4. XMC4200 / XMC4100 Series
5. XMC1400 Series
6. XMC1300 Series
7. XMC1200 Series
8. XMC1100 Series

### **References:**

1. XMC4800 / XMC4700 / XMC4300 Reference Manual
2. XMC4500 Reference Manual
3. XMC4400 Reference Manual
4. XMC4200 / XMC4100 Reference Manual
5. XMC1400 Reference Manual
6. XMC1300 Reference Manual
7. XMC1200 Reference Manual
8. XMC1100 Reference Manual

### **Limitations**

None.

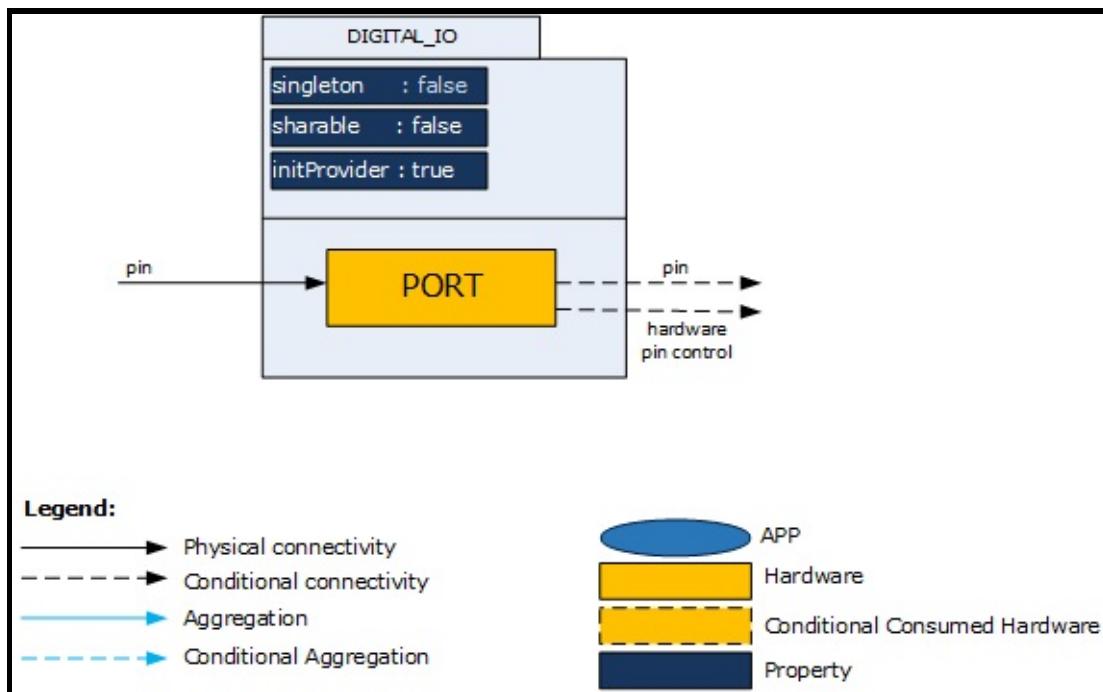
---

# DIGITAL\_IO

Home

## Architecture Description

### Architecture Description



**Figure 1 :** Architecture of **DIGITAL\_IO** APP

The above diagram represents the internal software architecture of the **DIGITAL\_IO** APP. A **DIGITAL\_IO** APP instance exists in a DAVE™ project with fixed attributes as shown. Each instance of this APP configures PORT pin in the MCU. The **DIGITAL\_IO** APP also provides output signal for inter-peripheral connections.

An instantiated APP (after code generation) generates a specific data structure with the GUI configuration. The name of this data structure can be modified by changing the APP instance label (e.g. change label from default **DIGITAL\_IO\_0** to **INPUT\_READ**).

## Signals:

The following table presents the signals provided by the APP for connection. It also gives the flexibility to configure and extend the connectivity to other APPs.

Signal Name	Input/Output	Availability	Description
pin	Input/Output	When pin direction is <i>Input</i> or <i>Input/Output</i>	GPIO functionality.
hardware pin control	Input/Output	When pin direction is <i>Hardware Controlled</i>	Peripheral controlled functionality.



# DIGITAL\_IO

Home

## APP Configuration Parameters

### App Configuration Parameters

The screenshot displays two configuration panels side-by-side, both titled "General Settings".

**XMC4xxx Device Panel:**

- General Settings:** Pin direction is set to "Input".
- Input Settings:** Mode is set to "Tristate".
- Output Settings:** Mode is set to "Push Pull", Initial output level is "High", and Driver strength is "Don't Care".

**XMC1xxx Device Panel:**

- General Settings:** Pin direction is set to "Input".
- Input Settings:** Mode is set to "Tristate" and Hysteresis is set to "Standard".
- Output Settings:** Mode is set to "Push Pull" and Initial output level is "Low".

Figure 1: General Settings

# DIGITAL\_IO

Home

## Enumerations

enum **DIGITAL\_IO\_STATUS** {  
 **DIGITAL\_IO\_STATUS\_OK** = 0U,  
 **DIGITAL\_IO\_STATUS\_FAILURE**  
 = 1U }  
Initialization status of  
**DIGITAL\_IO** APP. [More...](#)

typedef enum **DIGITAL\_IO\_STATUS** **DIGITAL\_IO\_STATUS\_t**  
Initialization status of  
**DIGITAL\_IO** APP.

## Enumeration Type Documentation

### enum DIGITAL\_IO\_STATUS

Initialization status of **DIGITAL\_IO** APP.

#### Enumerator:

*DIGITAL\_IO\_STATUS\_OK*      0=Status OK

*DIGITAL\_IO\_STATUS\_FAILURE* 1=Status Failed

Definition at line **91** of file **DIGITAL\_IO.h**.

# DIGITAL\_IO

Home

Data Structures

## Data structures

## Data Structures

---

struct **DIGITAL\_IO**  
Initialization data structure of  
**DIGITAL\_IO** APP. More...

typedef struct **DIGITAL\_IO** **DIGITAL\_IO\_t**  
Initialization data structure of  
**DIGITAL\_IO** APP.

---



# DIGITAL\_IO

Home

## Methods

DAVE\_APP\_VERSION\_t **DIGITAL\_IO\_GetAppVersion** (void)  
Get DIGITAL\_IO APP version.

**DIGITAL\_IO\_STATUS\_t** **DIGITAL\_IO\_Init** (const **DIGITAL\_IO\_t** \*const handler)  
Function to initialize the port pin as per UI settings.

**\_STATIC\_INLINE void** **DIGITAL\_IO\_SetOutputHigh** (const **DIGITAL\_IO\_t** \*const handler)  
Function to set port pin high.

**\_STATIC\_INLINE void** **DIGITAL\_IO\_SetOutputLow** (const **DIGITAL\_IO\_t** \*const handler)  
Function to reset port pin.

**\_STATIC\_INLINE void** **DIGITAL\_IO\_ToggleOutput** (const **DIGITAL\_IO\_t** \*const handler)  
Function to Toggle port pin.

**\_STATIC\_INLINE uint32\_t** **DIGITAL\_IO\_GetInput** (const **DIGITAL\_IO\_t** \*const handler)  
Function to read input level of port pin.

## Methods

## Function Documentation

**DAVE\_APP\_VERSION\_t DIGITAL\_IO\_GetAppVersion( void )**

Get **DIGITAL\_IO** APP version.

**Returns:**

DAVE\_APP\_VERSION\_t APP version information (major, minor and patch number)

**Description:**

The function can be used to check application software compatibility with a specific version of the APP.

Example Usage:

```
#include <DAVE.h>

int main(void)
{
    DAVE_STATUS_t init_status;
    DAVE_APP_VERSION_t version;

    // Initialize DIGITAL_IO APP:
    // DIGITAL_IO_Init() is called from within DAVE_Init().
    init_status = DAVE_Init();
    if(init_status == DAVE_STATUS_SUCCESS)
    {
        version = DIGITAL_IO_GetAppVersion();
        if (version.major != 4U) {
            // Probably, not the right version.
        }
    }

    // More code here
```

```
    while(1) {  
        }  
        return (1);  
    }
```

**Returns:**

DAVE\_APP\_VERSION\_t APP version information (major, minor and patch number)

Definition at line 77 of file [DIGITAL\\_IO.c](#).

---

**\_\_STATIC\_INLINE uint32\_t DIGITAL\_IO\_GetInput ( const DIGITAL\_I**

---

Function to read input level of port pin.

**Parameters:**

**handler** Pointer pointing to APP data structure. Refer [DIGITAL\\_IO\\_t](#) for details.

**Returns:**

uint32\_t input logic level. Range:0-1

**Description:**

This function reads the Pn\_IN register and returns the current logical value at the GPIO pin.

**Related APIs:**

None

**Example Usage:**

```
#include <DAVE.h> //Declarations from DAVE Code Generation (includes SFR declaration)  
int main(void)
```

```
{  
    DAVE_STATUS_t status;  
    uint32_t pin_status;  
    status = DAVE_Init(); // (DAVE_STATUS_t)DIGITAL_IO_Init(&DIGITAL_IO_0) is called within DAVE_Init()  
    if(status == DAVE_STATUS_SUCCESS)  
    {  
        XMC_DEBUG("DAVE Apps initialization success\n");  
    }  
    else  
    {  
        XMC_DEBUG(("DAVE Apps initialization failed with status %d\n", status));  
        while(1U)  
        {  
        }  
    }  
    //Placeholder for user application code. The while loop below can be replaced with user application code.  
    while(1U)  
    {  
        pin_status = DIGITAL_IO_GetInput(&DIGITAL_IO_0);  
        if(pin_status == 1)  
        {  
            // Add application code here  
        }  
        else  
        {  
            // Add application code here  
        }  
    }  
    return (1);  
}
```

Definition at line 419 of file **DIGITAL\_IO.h**.

References **DIGITAL\_IO::gpio\_pin**, and **DIGITAL\_IO::gpio\_port**.

## **DIGITAL\_IO\_STATUS\_t DIGITAL\_IO\_Init ( const DIGITAL\_IO\_t \*cons**

Function to initialize the port pin as per UI settings.

### **Parameters:**

**handler** Pointer pointing to APP data structure. Refer **DIGITAL\_IO\_t** for details.

### **Returns:**

**DIGITAL\_IO\_STATUS\_t DIGITAL\_IO** APP status. Refer **DIGITAL\_IO\_STATUS\_t** structure for details.

### **Description:**

This function initializes GPIO port registers IOCR,PDISC,OMR,PDR/PHCR to configure pin direction,initial output level, and pad driver strength/hysteresis.

### **Related APIs:**

None

### **Example Usage:**

```
#include <DAVE.h> //Declarations from DAVE Code Generation (includes SFR declaration)
int main(void)
{
    DAVE_STATUS_t status;
    status = DAVE_Init(); // (DAVE_STATUS_t)DIGITAL_IO_Init(&DIGITAL_IO_0) is called within DAVE_Init()
    if(status == DAVE_STATUS_SUCCESS)
    {
        XMC_DEBUG("DAVE Apps initialization success\
```

```
n");
}
else
{
    XMC_DEBUG(("DAVE Apps initialization failed
with status %d\n", status));
    while(1U)
    {
    }
}
//Placeholder for user application code. The w
hile loop below can be replaced with user applica
tion code.
while(1U)
{
}
return 1U;
}
```

**Parameters:**

**handler** Pointer pointing to APP data structure.

**Returns:**

DIGITAL\_IO\_STATUS\_t **DIGITAL\_IO** APP status.

Definition at line **95** of file **DIGITAL\_IO.c**.

References **DIGITAL\_IO\_STATUS\_OK**, **DIGITAL\_IO::gpio\_config**,  
**DIGITAL\_IO::gpio\_pin**, **DIGITAL\_IO::gpio\_port**, and  
**DIGITAL\_IO::hwctrl**.

---

**\_\_STATIC\_INLINE void DIGITAL\_IO\_SetOutputHigh ( const DIGITAL**

---

Function to set port pin high.

**Parameters:**

**handler** Pointer pointing to APP data structure. Refer [DIGITAL\\_IO\\_t](#) for details.

**Returns:**

None

**Description:**

This function configures port output modification register Pn\_OMR, to make port pin to high level.

**Related APIs:**

[DIGITAL\\_IO\\_SetOutputLow\(\)](#)

Example Usage:

```
#include <DAVE.h> //Declarations from DAVE Code Generation (includes SFR declaration)
int main(void)
{
    DAVE_STATUS_t status;
    status = DAVE_Init(); // (DAVE_STATUS_t)DIGITAL_IO_Init(&DIGITAL_IO_0) is called within DAVE_Init()
    if(status == DAVE_STATUS_SUCCESS)
    {
        XMC_DEBUG("DAVE Apps initialization success\n");
    }
    else
    {
        XMC_DEBUG(("DAVE Apps initialization failed with status %d\n", status));
        while(1U)
        {
        }
    }
    //Placeholder for user application code. The while loop below can be replaced with user applica
```

```
tion code.  
    DIGITAL_IO_SetOutputHigh(&DIGITAL_IO_0);  
    while(1U)  
    {  
        // Add application code here  
    }  
  
    return (1);  
}
```

Definition at line **266** of file **DIGITAL\_IO.h**.

References **DIGITAL\_IO::gpio\_pin**, and **DIGITAL\_IO::gpio\_port**.

---

```
__STATIC_INLINE void DIGITAL_IO_SetOutputLow ( const DIGITAL_
```

Function to reset port pin.

**Parameters:**

**handler** Pointer pointing to APP data structure. Refer **DIGITAL\_IO\_t** for details.

**Returns:**

None

**Description:**

This function configures port output modification register Pn\_OMR, to make port pin to low level.

**Related APIs:**

**DIGITAL\_IO\_SetOutputHigh()**

Example Usage:

```
#include <DAVE.h>/Declarations from DAVE Code  
Generation (includes SFR declaration)  
int main(void)
```

```

{
    DAVE_STATUS_t status;
    status = DAVE_Init(); // (DAVE_STATUS_t)DIGITAL_IO_Init(&DIGITAL_IO_0) is called within DAVE_Init()
    if(status == DAVE_STATUS_SUCCESS)
    {
        XMC_DEBUG("DAVE Apps initialization success\n");
    }
    else
    {
        XMC_DEBUG(("DAVE Apps initialization failed with status %d\n", status));
        while(1U)
        {
        }
    }
    //Placeholder for user application code. The while loop below can be replaced with user application code.
    DIGITAL_IO_SetOutputLow(&DIGITAL_IO_0);
    while(1U)
    {
        // Add application code here
    }

    return (1);
}

```

Definition at line 313 of file **DIGITAL\_IO.h**.

References **DIGITAL\_IO::gpio\_pin**, and **DIGITAL\_IO::gpio\_port**.

---

**\_\_STATIC\_INLINE void DIGITAL\_IO\_ToggleOutput (const DIGITAL\_**

---

Function to Toggle port pin.

**Parameters:**

**handler** Pointer pointing to APP data structure. Refer [DIGITAL\\_IO\\_t](#) for details.

**Returns:**

None

**Description:**

This function configures port output modification register Pn\_OMR, to toggle port pin.

**Related APIs:**

[DIGITAL\\_IO\\_SetOutputLow\(\)](#), [DIGITAL\\_IO\\_SetOutputHigh\(\)](#)

Example Usage:

```
#include <DAVE.h> //Declarations from DAVE Code Generation (includes SFR declaration)
int main(void)
{
    DAVE_STATUS_t status;
    uint32_t delay_count;;
    status = DAVE_Init(); // (DAVE_STATUS_t)DIGITAL_IO_Init(&DIGITAL_IO_0) is called within DAVE_Init()
    if(status == DAVE_STATUS_SUCCESS)
    {
        XMC_DEBUG("DAVE Apps initialization success\n");
    }
    else
    {
        XMC_DEBUG(("DAVE Apps initialization failed with status %d\n", status));
        while(1U)
```

```
    }
}

//Placeholder for user application code. The while loop below can be replaced with user application code.
while(1U)
{
    DIGITAL_IO_ToggleOutput(&DIGITAL_IO_0); //toggles : 1 -> 0 (if initial output level is logic 1)
    //Add application code here
    for(delay_count = 0;delay_count<0xfffff;delay_count++);
        DIGITAL_IO_ToggleOutput(&DIGITAL_IO_0); //toggles : 0 -> 1
        //Add application code here
        for(delay_count = 0;delay_count<0xfffff;delay_count++);
    }
    return (1);
}
```

Definition at line [365](#) of file **DIGITAL\_IO.h**.

References **DIGITAL\_IO::gpio\_pin**, and **DIGITAL\_IO::gpio\_port**.

---

# DIGITAL\_IO

Home

## Usage

### Usage

The below example demonstrates LED Toggling.

Hardware requirement:

- XMC4000 Application Kit : CPU\_45A-V3(CPU Board XMC4500 General Purpose).

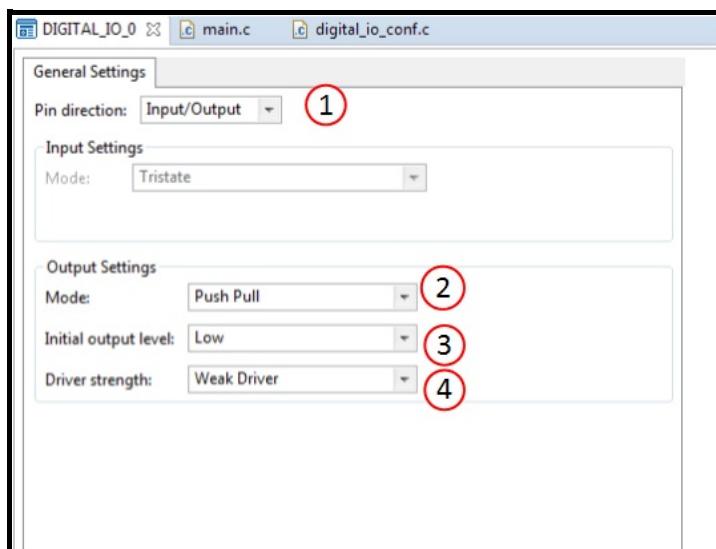
**Step 1:** Instantiate APP and select Pin direction as Input/Output.

**Step 2:** Select Mode Push-pull.

**Step 3:** Select *Initial output level* to Low.

**Step 4:** Select Desired *Driver strength* to Weak Driver.(only available for XMC4X device).

**Step 5:** Select pin by using *Manual Pin Allocator*. (Figure shows for XMC4500 device).



**Figure 1:** Deployment of **DIGITAL\_IO** APP for output mode

The screenshot shows a software window titled "Manual Pin Allocator". At the top, there is a filter dropdown set to "ALL". Below the filter is a table with three columns: "App Instance Name", "App Pin Name", and "Pin Number (Port)". A single row is present in the table. The "App Instance Name" column contains "DIGITAL\_IO\_0" with a circled number "5" next to it. The "App Pin Name" column contains "pin". The "Pin Number (Port)" column contains "#12 ( P3.9 )". There are icons for saving and adding new entries at the top right of the table area.

App Instance Name	App Pin Name	Pin Number (Port)
DIGITAL_IO_0	pin	#12 ( P3.9 )

**Figure 2:** Manual Pin Assignment

## Generate code

Files are generated here: `<project\_name>/Dave/Generated/` ('project\_name' is the name chosen by the user during project creation). APP instance definitions and APIs are generated only after code generation.

**Note:** Code must be explicitly generated for every change in the GUI configuration.

**Important:** Any manual modification to APP specific files will be overwritten by a subsequent code generation operation.

## Sample Application (main.c)

```
#include <DAVE.h> //Declarations from
DAVE Code Generation (includes SFR declaration)
int main(void)
{
    DAVE_STATUS_t status;
    uint32_t delay_count;;
    status = DAVE_Init(); // (DAVE_STATUS_t)DIGITAL
    _IO_Init(&DIGITAL_IO_0) is called within DAVE_Init()
    if(status == DAVE_STATUS_SUCCESS)
    {
```

```

        XMC_DEBUG("DAVE Apps initialization success\n");
    }
}
else
{
    XMC_DEBUG(("DAVE Apps initialization failed with status %d\n", status));
    while(1U)
    {
    }
}
//Placeholder for user application code. The while loop below can be replaced with user application code.
while(1U)
{
    DIGITAL_IO_ToggleOutput(&DIGITAL_IO_0); //toggles level at pin
    for(delay_count = 0;delay_count<0xfffff;delay_count++);
}
return 1U;
}

```

## Build and Run the Project

### **Observation :**

Observe LED1(P3.9) is blinking.

---

# DIGITAL\_IO

Home

## Release History

### Release History



# DIGITAL\_IO

[Home](#)

[Data Structures](#)

[Data Structure Index](#)

[Data Fields](#)

## Data Structures

Here are the data structures with brief descriptions:

**DIGITAL\_IO** Initialization data structure of **DIGITAL\_IO APP**



# DIGITAL\_IO

[Home](#)

[Data Structures](#)

[Data Structure Index](#)

[Data Fields](#)

[Data Fields](#)

## DIGITAL\_IO Struct Reference

[Data structures](#)

---

## Detailed Description

Initialization data structure of **DIGITAL\_IO** APP.

Definition at line **112** of file **DIGITAL\_IO.h**.

```
#include <DIGITAL_IO.h>
```

## Data Fields

```
XMC_GPIO_PORT_t *const gpio_port  
const XMC_GPIO_CONFIG_t gpio_config  
    const uint8_t gpio_pin  
const XMC_GPIO_HWCTRL_t hwctrl
```

## Field Documentation

**const XMC\_GPIO\_CONFIG\_t DIGITAL\_IO::gpio\_config**

mode, initial output level and pad driver strength / hysteresis

Definition at line [115](#) of file **DIGITAL\_IO.h**.

Referenced by **DIGITAL\_IO\_Init()**.

**const uint8\_t DIGITAL\_IO::gpio\_pin**

pin number

Definition at line [116](#) of file **DIGITAL\_IO.h**.

Referenced by **DIGITAL\_IO\_GetInput()**, **DIGITAL\_IO\_Init()**,  
**DIGITAL\_IO\_SetOutputHigh()**, **DIGITAL\_IO\_SetOutputLow()**, and  
**DIGITAL\_IO\_ToggleOutput()**.

**XMC\_GPIO\_PORT\_t\* const DIGITAL\_IO::gpio\_port**

port number

Definition at line [114](#) of file **DIGITAL\_IO.h**.

Referenced by **DIGITAL\_IO\_GetInput()**, **DIGITAL\_IO\_Init()**,  
**DIGITAL\_IO\_SetOutputHigh()**, **DIGITAL\_IO\_SetOutputLow()**, and  
**DIGITAL\_IO\_ToggleOutput()**.

**const XMC\_GPIO\_HWCTRL\_t DIGITAL\_IO::hwctrl**

Hardware port control

Definition at line [117](#) of file **DIGITAL\_IO.h**.

Referenced by **DIGITAL\_IO\_Init()**.

---

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

- **DIGITAL\_IO.h**
- 



# DIGITAL\_IO

Home

Data Structures

Data Structure Index

Data Fields

## Data Structure Index

D

D

DIGITAL\_IO

D



# DIGITAL\_IO

Home		
Data Structures	Data Structure Index	Data Fields
All	Variables	

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- gpio\_config : [DIGITAL\\_IO](#)
  - gpio\_pin : [DIGITAL\\_IO](#)
  - gpio\_port : [DIGITAL\\_IO](#)
  - hwctrl : [DIGITAL\\_IO](#)
-

# DIGITAL\_IO

Home		
Data Structures	Data Structure Index	Data Fields
All	Variables	

- gpio\_config : DIGITAL\_IO
  - gpio\_pin : DIGITAL\_IO
  - gpio\_port : DIGITAL\_IO
  - hwctrl : DIGITAL\_IO
-

# DIGITAL\_IO

Home

File List

Globals

## File List

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

[DIGITAL\\_IO.c](#) [code]

[DIGITAL\\_IO.h](#) [code]

# DIGITAL\_IO

[Home](#)

[File List](#)

[Globals](#)

[Functions](#)

## DIGITAL\_IO.c File Reference

## Detailed Description

**Date:**

2015-08-25

NOTE: This file is generated by DAVE. Any manual modification done to this file will be lost when the code is regenerated.

Definition in file **DIGITAL\_IO.c**.

```
#include "digital_io.h"
```

## Functions

DAVE_APP_VERSION_t	<b>DIGITAL_IO_GetAppVersion</b> (void) Get <b>DIGITAL_IO</b> APP version.
<b>DIGITAL_IO_STATUS_t</b>	<b>DIGITAL_IO_Init</b> (const <b>DIGITAL_IO_t</b> *const handler) Function to initialize the port pin as per UI settings.

## Function Documentation

**DIGITAL\_IO\_STATUS\_t DIGITAL\_IO\_Init ( const DIGITAL\_IO\_t \*cons**

Function to initialize the port pin as per UI settings.

**Parameters:**

**handler** Pointer pointing to APP data structure.

**Returns:**

DIGITAL\_IO\_STATUS\_t **DIGITAL\_IO** APP status.

Definition at line [95](#) of file **DIGITAL\_IO.c**.

References **DIGITAL\_IO\_STATUS\_OK**, **DIGITAL\_IO::gpio\_config**, **DIGITAL\_IO::gpio\_pin**, **DIGITAL\_IO::gpio\_port**, and **DIGITAL\_IO::hwctrl**.

[Go to the source code of this file.](#)

---

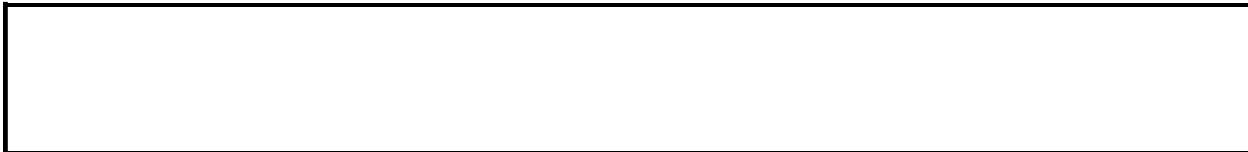
---

# DIGITAL\_IO

Home	
File List	Globals
All	Functions

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- DIGITAL\_IO\_GetAppVersion() : [DIGITAL\\_IO.c](#)
  - DIGITAL\_IO\_Init() : [DIGITAL\\_IO.c](#)
- 



# DIGITAL\_IO

Home	
File List	Globals
All	Functions

- DIGITAL\_IO\_GetAppVersion() : [DIGITAL\\_IO.c](#)
  - DIGITAL\_IO\_Init() : [DIGITAL\\_IO.c](#)
- 



# DIGITAL\_IO

Home

File List

Globals

## DIGITAL\_IO.h

```
00001 /*
00002 *
00003 * @file digital_io.h
00004 * @date 2015-12-22
00005 *
00006 * NOTE:
00007 * This file is generated by DAVE. Any manual
00008 * modification done to this file will be lost when
00009 * the code is regenerated.
00010 *
00011 * ****
00012 * ****
00013 * ****
00014 * ****
00015 * ****
00016 * ****
00017 * ****
00018 * ****
00019 * ****
```

ain the above copyright notice, this list of conditions and the following

00020 \* disclaimer.

00021 \*

00022 \* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the

00023 \* following disclaimer in the documentation and/or other materials provided with the distribution.

00024 \*

00025 \* Neither the name of the copyright holders nor the names of its contributors may be used to endorse or promote

00026 \* products derived from this software without specific prior written permission.

00027 \*

00028 \* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,

00029 \* INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE

00030 \* DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,

00031 \* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR

00032 \* SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,

00033 \* WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE

00034 \* USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

```
00035  *
00036  * To improve the quality of the software, u
00037  * users are encouraged to share modifications, enhanc
00038  * ements or bug fixes
00039  * with Infineon Technologies AG (dave@infin
00040  * eon.com).
00041  ****
00042  *
00043  * 2015-02-16
00044  *      - Initial version
00045  *
00046  * 2015-04-22
00047  *      - XMC_ASSERT is added in static inlin
00048  * e functions.<br>
00049  * 2015-06-20
00050  *      - Version check added for XMCLib depe
00051  * ndency.<br>
00052  * 2015-12-22
00053  *      - Added hardware controlled IO featur
00054  *
00055  * @endcond
00056  *
00057  */
00058
00059 /*****
00060  * HEADER FILES
00061  ****
```

```
*****
***** */
00062
00063 #ifndef DIGITAL_IO_H
00064 #define DIGITAL_IO_H
00065
00066 #include <xmc_gpio.h>
00067 #include <DAVE_common.h>
00068 #include "digital_io_conf.h"
00069
00070 /*****
***** */
***** */
***** */

00071 * MACROS
00072 ****
***** */
***** */

00073 #if (((XMC_LIB_MAJOR_VERSION == 2U) && \
00074           (XMC_LIB_MINOR_VERSION >= 0U) && \
00075           (XMC_LIB_PATCH_VERSION >= 0U)))
00076 #error "DIGITAL_IO requires XMC Peripheral Library v2.0.0 or higher"
00077 #endif
00078
00079 /*****
***** */
***** */

00080 * ENUMS
00081 ****
***** */

00082
00091 typedef enum DIGITAL_IO_STATUS
00092 {
00093     DIGITAL_IO_STATUS_OK = 0U,
00094     DIGITAL_IO_STATUS_FAILURE = 1U
00095 } DIGITAL_IO_STATUS_t;
```

```
00096
00101 /*****
00102 ****
00103 ****
00104 ****
00105 * DATA STRUCTURES
00106 ****
00107 ****
00108 ****
00109 ****
00110 ****
00111 ****
00112 typedef struct DIGITAL_IO
00113 {
00114     XMC_GPIO_PORT_t *const gpio_port;
00115     const XMC_GPIO_CONFIG_t gpio_config;
00116     const uint8_t gpio_pin;
00117     const XMC_GPIO_HWCTRL_t hwctrl;
00118 } DIGITAL_IO_t;
00119
00120
00121 /*****
00122 ****
00123 ****
00124 ****
00125 ****
00126 ****
00127 * API Prototypes
00128 ****
00129 ****
00130 ****
00131 #ifdef __cplusplus
00132 extern "C" {
00133 #endif
00134
00135 DAVE_APP_VERSION_t DIGITAL_IO_GetAppVersion(
00136     void);
00137
00138
00139
00140
00141
00142
00143
00144
00145
00146
00147
00148
00149
00150
00151
00152
00153
00154
00155
00156
00157
00158
00159
00160
00161
00162
00163
00164
00165
00166
00167
00168
00169
00170
00171
00172
00173
00174
00175
00176
00177
00178
00179
00180
00181
```

```
00222 DIGITAL_IO_STATUS_t DIGITAL_IO_Init(const DI
GITAL_IO_t *const handler);
00223
00266 __STATIC_INLINE void DIGITAL_IO_SetOutputHigh(
const DIGITAL_IO_t *const handler)
00267 {
00268     XMC_ASSERT("DIGITAL_IO_SetOutputHigh: hand
ler null pointer", handler != NULL);
00269     XMC_GPIO_SetOutputHigh(handler->gpio_port,
handler->gpio_pin);
00270 }
00271
00313 __STATIC_INLINE void DIGITAL_IO_SetOutputLow(
const DIGITAL_IO_t *const handler)
00314 {
00315     XMC_ASSERT("DIGITAL_IO_SetOutputLow: handl
er null pointer", handler != NULL);
00316     XMC_GPIO_SetOutputLow(handler->gpio_port,h
andler->gpio_pin);
00317 }
00318
00365 __STATIC_INLINE void DIGITAL_IO_ToggleOutput(
const DIGITAL_IO_t *const handler)
00366 {
00367     XMC_ASSERT("DIGITAL_IO_ToggleOutput: handl
er null pointer", handler != NULL);
00368     XMC_GPIO_ToggleOutput(handler->gpio_port,
handler->gpio_pin);
00369 }
00370
00419 __STATIC_INLINE uint32_t DIGITAL_IO_GetInput(
const DIGITAL_IO_t *const handler)
00420 {
00421     XMC_ASSERT("DIGITAL_IO_GetInput: handler n
ull pointer", handler != NULL);
00422     return XMC_GPIO_GetInput(handler->gpio_port
, handler->gpio_pin);
```

```
00423 }
00424
00429 #ifdef __cplusplus
00430 }
00431 #endif
00432
00433 /* Include APP extern file */
00434 #include "digital_io_extern.h"
00435
00436
00437 #endif /* DIGITAL_IO_H */
```



# DIGITAL\_IO

Home

File List

Globals

## DIGITAL\_IO.c

Go to the documentation of this file.

```
00001
00051 /*****
***** 
***** 
00052 * HEADER FILES
00053 *****
***** 
***** 
***** /
00054 #include "digital_io.h"
00055
00056 /*****
***** 
***** 
00057 * MACROS
00058 *****
***** 
***** 
***** /
00059
00060 /*****
***** 
***** 
***** 
00061 * LOCAL DATA
00062 *****
***** 
***** 
***** /
00063
00064 /*****
```

```
*****
*****
00065 * LOCAL ROUTINES
00066 ****
*****
***** */
00067
00068 /* ****
*****
***** */
00069 * API IMPLEMENTATION
00070 ****
*****
***** */
00071
00077 DAVE_APP_VERSION_t DIGITAL_IO_GetAppVersion(
void)
00078 {
00079     DAVE_APP_VERSION_t version;
00080
00081     version.major = (uint8_t)DIGITAL_IO_MAJOR_
VERSION;
00082     version.minor = (uint8_t)DIGITAL_IO_MINOR_
VERSION;
00083     version.patch = (uint8_t)DIGITAL_IO_PATCH_
VERSION;
00084
00085     return (version);
00086 }
00087
00088
00095 DIGITAL_IO_STATUS_t DIGITAL_IO_Init(const DI
GITAL_IO_t *const handler)
00096 {
00097     XMC_ASSERT("DIGITAL_IO_Init: handler null
pointer", handler != NULL);
00098
```

```
00099  /* Initializes input / output characteristics */
00100  XMC_GPIO_Init(handler->gpio_port, handler->
00101  gpio_pin, &handler->gpio_config);
00102  /*Configure hardware port control*/
00103  XMC_GPIO_SetHardwareControl(handler->gpio_
00104  port, handler->gpio_pin, handler->hwctrl);
00105  return (DIGITAL_IO_STATUS_OK);
00106 }
```

