

ANALOG_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
-
-
-

ANALOG_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).

ANALOG_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
IO	Input Output
ADC	Analog to Digital Converter

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



ANALOG_IO

Home

Overview

Overview

The **ANALOG_IO APP** provides the following features:

1. Analog IO port pins.
2. Connection of port pin to other peripherals (using the DAVE™ HW Signal Connections).

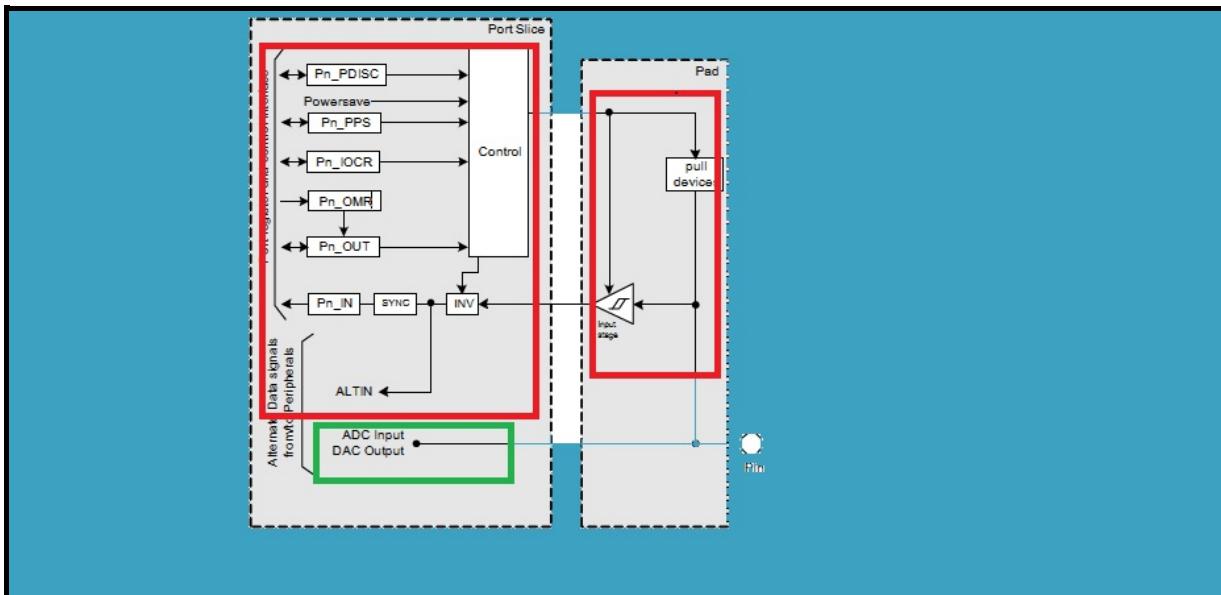


Figure 1 : Overview of the **ANALOG_IO APP**

The above figure shows the functional overview of the **ANALOG_IO APP**. The Analog port does not have output path and alternate outputs to connect other peripherals. Instead, ADC input and DAC output is connected to pin via PAD. Though Digital input path is available in Analog ports, the **ANALOG_IO** is designed in such way that, it consumes only analog pad. Therefore this APP is applicable only for analog port pin consumption. For Digital input requirements

DIGITAL_IO APP shall be used.

As shown in above figure, red color box indicates that APP does not configure any of the hardware registers. Green color box indicates that APP reserves analog input/output path.

Note:

This APP does not provide GUI configuration or APIs. It allows to select analog port pins.

Supported Devices

The APP supports below devices:

1. XMC4800 / XMC4700 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 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.



ANALOG_IO

Home

Architecture Description

Architecture Description

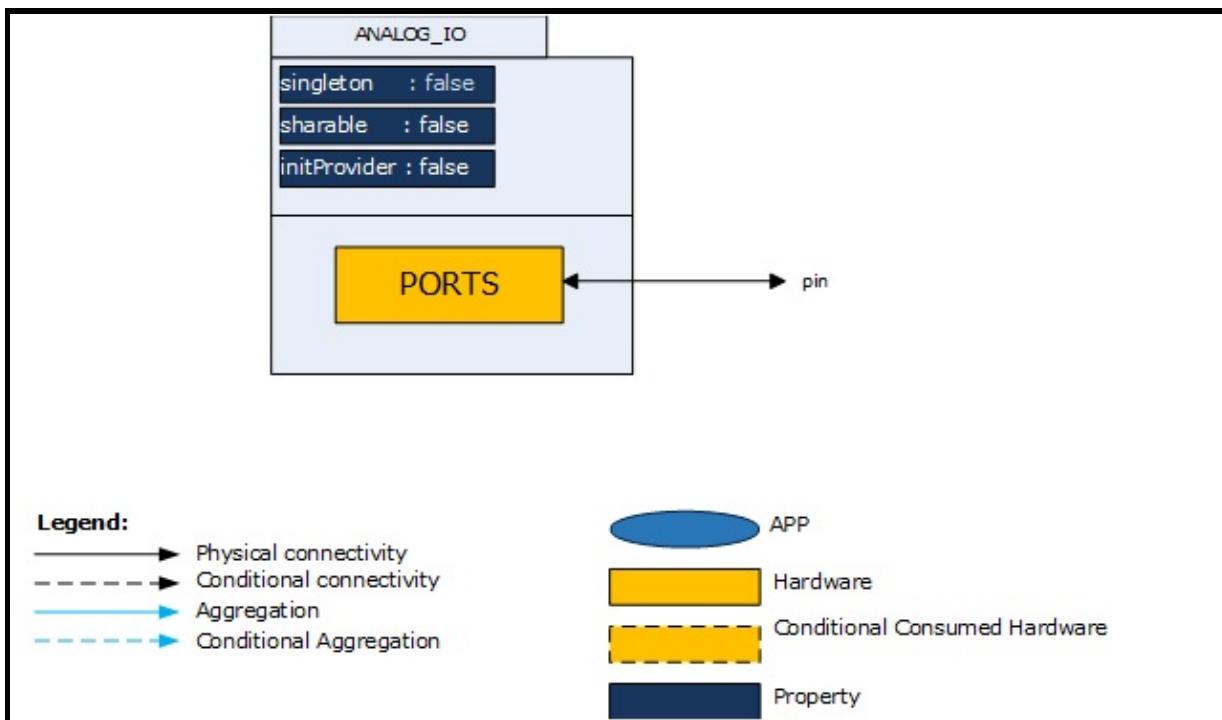


Figure 1 : Architecture of ANALOG_IO APP

The above diagram represents the internal software architecture of the ANALOG_IO APP. A ANALOG_IO APP instance exists in a DAVE™ project with fixed attributes as shown. Each instance of this APP configures one PORT pin in the MCU.

ANALOG_IO APP do not provide GUI configuration and configuration data structure. The name of this data structure can be modified by changing the APP instance label (e.g. change label from default ANALOG_IO to (VOLTAGE_MEASUREMENT)).

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.

Table 1: APP IO signals

Signal Name	Input/Output	Availability	Description
pin	Input/Output	Always	Analog Input/Output functionality.

ANALOG_IO

Home

APP Configuration Parameters

App Configuration Parameters

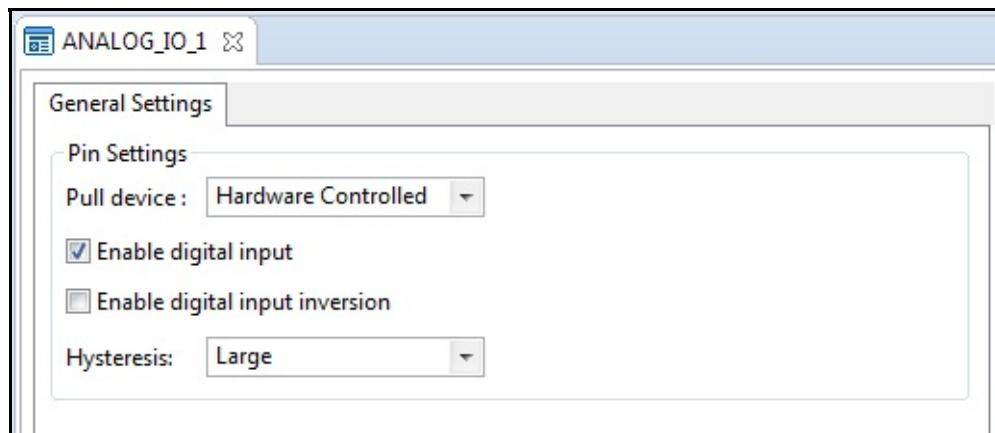


Figure 1: General Settings

ANALOG_IO

Home

Enumerations

	ANALOG_IO_STATUS {
enum	ANALOG_IO_STATUS_OK = 0, ANALOG_IO_STATUS_FAILURE
	Initialization status of ANALOG_IO
	More...
	ANALOG_IO_STATE {
enum	ANALOG_IO_STATE_NOT_INITIALIZED = 0, ANALOG_IO_STATE_INITIALIZED
	Initialization state of ANALOG_IO
	More...
typedef enum ANALOG_IO_STATUS	ANALOG_IO_STATUS_t
	Initialization status of ANALOG_IO
typedef enum ANALOG_IO_STATE	ANALOG_IO_STATE_t
	Initialization state of ANALOG_IO

Enumeration Type Documentation

enum ANALOG_IO_STATE

Initialization state of **ANALOG_IO** APP.

Enumerator:

`ANALOG_IO_STATE_NOT_INITIALIZED` State = 0, APP not even initialized once

`ANALOG_IO_STATE_INITIALIZED` State = 1, APP successfully initialized

Definition at line [102](#) of file **ANALOG_IO.h**.

enum ANALOG_IO_STATUS

Initialization status of **ANALOG_IO** APP.

Enumerator:

`ANALOG_IO_STATUS_OK` Status = 0, APP Initialization OK

`ANALOG_IO_STATUS_FAILURE` Status = 1, APP Initialization Failed

Definition at line [92](#) of file **ANALOG_IO.h**.



ANALOG_IO

Home

Data structures

typedef void(* **ANALOG_IO_CONFIG_PTR_t**)(void)

Function pointer prototype declaration
used for initialization function.

typedef struct **ANALOG_IO** **ANALOG_IO_t**

Initialization data structure of
ANALOG_IO APP.

ANALOG_IO

Home

Methods

DAVE_APP_VERSION_t **ANALOG_IO_GetAppVersion** (void)
Get ANALOG_IO APP version.

ANALOG_IO_STATUS_t **ANALOG_IO_Init** (const **ANALOG_IO_t** *const handle)
Function to initialize the port pin as per UI settings.

_STATIC_INLINE uint32_t **ANALOG_IO_GetInput** (const **ANALOG_IO_t** *const handler)
Function to read input level of port pin.

Methods

Function Documentation

DAVE_APP_VERSION_t ANALOG_IO_GetAppVersion(void)

Get **ANALOG_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 ANALOG_IO APP:
    // ANALOG_IO_Init() is called from within DAVE
    // _Init().
    init_status = DAVE_Init();

    if(init_status == DAVE_STATUS_SUCCESS)
    {
        version = ANALOG_IO_GetAppVersion();
        if (version.major != 4U) {
            // Probably, not the right version.
        }
    }
}
```

```
// More code here
while(1) {
}
return (1);
}
```

Definition at line **78** of file [ANALOG_IO.c](#).

__STATIC_INLINE uint32_t ANALOG_IO_GetInput (const ANALOG_

Function to read input level of port pin.

Parameters:

handler Pointer pointing to APP data structure. Refer [ANALOG_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)ANALOG_IO_Init(&ANALOG_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 = ANALOG_IO_GetInput(&ANALOG_IO_0);
        if(pin_status == 1)
        {
            // Add application code here
        }
        else
        {
            // Add application code here
        }
    }
    return (1);
}

```

Definition at line **290** of file **ANALOG_IO.h**.

References [ANALOG_IO::pin](#), and [ANALOG_IO::port](#).

ANALOG_IO_STATUS_t ANALOG_IO_Init (const ANALOG_IO_t *cc)

Function to initialize the port pin as per UI settings.

Parameters:

handler Pointer pointing to APP data structure. Refer [ANALOG_IO_t](#) for details.

Returns:

ANALOG_IO_STATUS_t [ANALOG_IO](#) APP status. Refer [ANALOG_IO_STATUS_t](#) structure for details.

Description:

This function initializes GPIO port registers IOCR,HWSEL to configure pin direction and 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)ANALOG_IO_Init(&ANALOG_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;
}
```

Description : Function to initialize the analog port pin properties.

Input Parameter : Handler Pointer pointing to APP data structure.

Output Parameter : ANALOG_IO_STATUS_t ([ANALOG_IO](#) APP initialization status).

Definition at line [96](#) of file [ANALOG_IO.c](#).

References [ANALOG_IO_STATE_INITIALIZED](#),
[ANALOG_IO_STATE_NOT_INITIALIZED](#),
[ANALOG_IO_STATUS_OK](#), [ANALOG_IO::config_ptr](#), and
[ANALOG_IO::state](#).

ANALOG_IO

Home

Usage

Usage

It is typically used for consuming analog port pins. The following figures shows how top level APPs like ADC_MEASUREMENT APP uses **ANALOG_IO** APP for consuming analog port pin.

1. Figure 1 shows instantiation of ADC_MEASUREMENT APP.

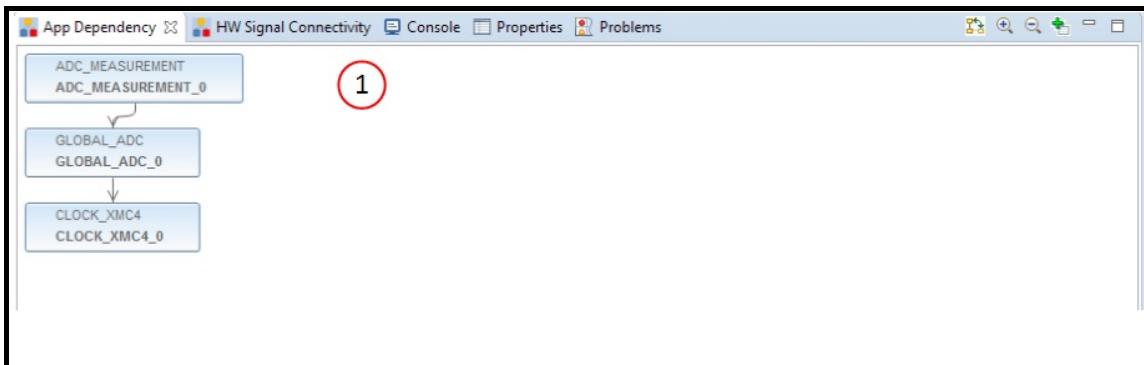


Figure 1 : Instantiation of ADC_MEASUREMENT APP

2. When Expose pin in Measurements Tab of ADC_MEASUREMENT APP is enabled, as shown in figure 2, **ANALOG_IO** APP is consumed by top level APP(ADC_MEASUREMENT APP) as shown in figure 3, it interns consumes analog port pin.



Figure 2 : ADC_MEASUREMENT APP

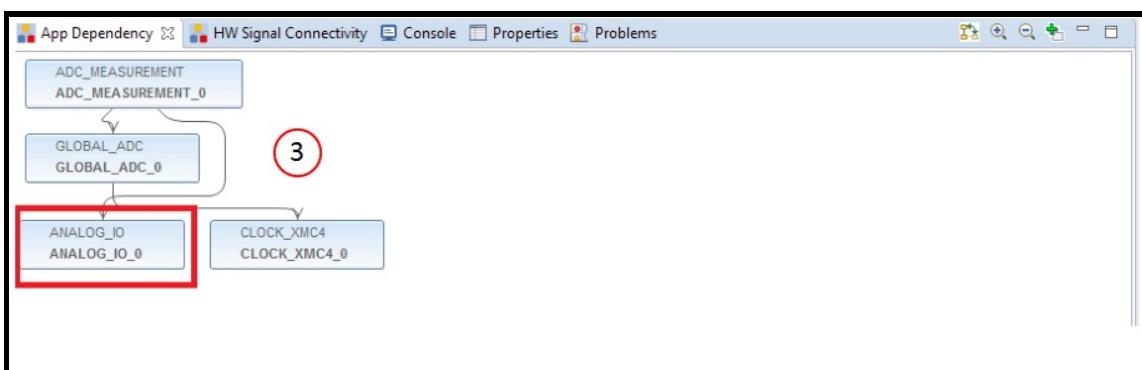


Figure 3 : ADC_MEASUREMENT APP

3. This is the one usecase of the **ANALOG_IO APP**, which shown above how top level APPs are using it.

ANALOG_IO

Home

Release History

Release History



ANALOG_IO

[Home](#)

[Data Structures](#)

[Data Structure Index](#)

[Data Fields](#)

Data Structures

Here are the data structures with brief descriptions:

[ANALOG_IO](#) Initialization data structure of [ANALOG_IO APP](#)



ANALOG_IO

[Home](#)

[Data Structures](#)

[Data Structure Index](#)

[Data Fields](#)

[Data Fields](#)

ANALOG_IO Struct Reference

Detailed Description

Initialization data structure of **ANALOG_IO** APP.

Definition at line **128** of file **ANALOG_IO.h**.

```
#include <ANALOG_IO.h>
```

Data Fields

ANALOG_IO_CONFIG_PTR_t	config_ptr
ANALOG_IO_STATE_t	* state
XMC_GPIO_PORT_t	* port
uint8_t	pin

Field Documentation

ANALOG_IO_CONFIG_PTR_t ANALOG_IO::config_ptr

Pointer to **ANALOG_IO** instance specific initialization function

Definition at line **130** of file **ANALOG_IO.h**.

Referenced by **ANALOG_IO_Init()**.

uint8_t ANALOG_IO::pin

Pin number used

Definition at line **133** of file **ANALOG_IO.h**.

Referenced by **ANALOG_IO_GetInput()**.

XMC_GPIO_PORT_t* ANALOG_IO::port

Port number used

Definition at line **132** of file **ANALOG_IO.h**.

Referenced by **ANALOG_IO_GetInput()**.

ANALOG_IO_STATE_t* ANALOG_IO::state

APP initialization state

Definition at line **131** of file **ANALOG_IO.h**.

Referenced by **ANALOG_IO_Init()**.

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

- [**ANALOG_IO.h**](#)
-



ANALOG_IO

Home

Data Structures

Data Structure Index

Data Fields

Data Structure Index

A

A

[ANALOG_IO](#)

A



ANALOG_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:

- config_ptr : [ANALOG_IO](#)
 - pin : [ANALOG_IO](#)
 - port : [ANALOG_IO](#)
 - state : [ANALOG_IO](#)
-

ANALOG_IO

Home		
Data Structures	Data Structure Index	Data Fields
All	Variables	

- config_ptr : ANALOG_IO
 - pin : ANALOG_IO
 - port : ANALOG_IO
 - state : ANALOG_IO
-

ANALOG_IO

Home

File List

Globals

File List

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

[ANALOG_IO.c](#) [code] 

[ANALOG_IO.h](#) [code] 

ANALOG_IO

[Home](#)

[File List](#)

[Globals](#)

[Functions](#)

ANALOG_IO.c File Reference

Detailed Description

Date:

2016-07-08

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 [ANALOG_IO.c](#).

```
#include "analog_io.h"
```

Functions

DAVE_APP_VERSION_t **ANALOG_IO_GetAppVersion** (void)
Get ANALOG_IO APP version.

ANALOG_IO_STATUS_t **ANALOG_IO_Init** (const **ANALOG_IO_t**
*const handle)
Function to initialize the port pin as per UI
settings.

Function Documentation

[ANALOG_IO_STATUS_t ANALOG_IO_Init \(const ANALOG_IO_t *config_ptr, ANALOG_IO_Status_t *status \)](#)

Function to initialize the port pin as per UI settings.

Description : Function to initialize the analog port pin properties.

Input Parameter : Handler Pointer pointing to APP data structure.

Output Parameter : ANALOG_IO_STATUS_t ([ANALOG_IO APP initialization status](#)).

Definition at line [96](#) of file [ANALOG_IO.c](#).

References [ANALOG_IO_STATE_INITIALIZED](#),
[ANALOG_IO_STATE_NOT_INITIALIZED](#),
[ANALOG_IO_STATUS_OK](#), [ANALOG_IO::config_ptr](#), and
[ANALOG_IO::state](#).

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

ANALOG_IO

[Home](#)

[File List](#)

[Globals](#)

[Data Structures](#)

ANALOG_IO.h File Reference

Detailed Description

Date:

2016-07-08

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 [ANALOG_IO.h](#).

```
#include <xmc_gpio.h> #include <DAVE_Common.h>
#include "analog_io_conf.h"
#include "analog_io_extern.h"
```

Data Structures

struct **ANALOG_IO**

Initialization data structure of **ANALOG_IO** APP. More...

TypeDefs

```
typedef void(* ANALOG_IO_CONFIG_PTR_t )(void)
```

Function pointer prototype declaration
used for initialization function.

```
typedef struct ANALOG_IO ANALOG_IO_t
```

Initialization data structure of
ANALOG_IO APP.

Functions

DAVE_APP_VERSION_t	ANALOG_IO_GetAppVersion (Get ANALOG_IO APP version.
ANALOG_IO_STATUS_t	ANALOG_IO_Init (const ANALOG_IO_STATUS_t *const handle) Function to initialize the port pin UI settings.
__STATIC_INLINE uint32_t	ANALOG_IO_GetInput (const ANALOG_IO_t *const handler) Function to read input level of port.
enum	ANALOG_IO_STATUS { ANALOG_IO_STATUS_OK = 0, ANALOG_IO_STATUS_FAILURE , Initialization status of ANALOG_IO. More...
enum	ANALOG_IO_STATE { ANALOG_IO_STATE_NOT_INITIALIZED = 0, ANALOG_IO_STATE_INITIALIZED , Initialization state of ANALOG_IO. More...
typedef enum ANALOG_IO_STATUS	ANALOG_IO_STATUS_t Initialization status of ANALOG_IO.
typedef enum ANALOG_IO_STATE	ANALOG_IO_STATE_t Initialization state of ANALOG_IO.

Go to the source code of this file.

ANALOG_IO

Home					
File List	Globals				
All	Functions	Typedefs	Enumerations	Enumerator	

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

- ANALOG_IO_CONFIG_PTR_t : [ANALOG_IO.h](#)
- ANALOG_IO_GetAppVersion() : [ANALOG_IO.c](#) , [ANALOG_IO.h](#)
- ANALOG_IO_GetInput() : [ANALOG_IO.h](#)
- ANALOG_IO_Init() : [ANALOG_IO.c](#) , [ANALOG_IO.h](#)
- ANALOG_IO_STATE : [ANALOG_IO.h](#)
- ANALOG_IO_STATE_INITIALIZED : [ANALOG_IO.h](#)
- ANALOG_IO_STATE_NOT_INITIALIZED : [ANALOG_IO.h](#)
- ANALOG_IO_STATE_t : [ANALOG_IO.h](#)
- ANALOG_IO_STATUS : [ANALOG_IO.h](#)
- ANALOG_IO_STATUS_FAILURE : [ANALOG_IO.h](#)
- ANALOG_IO_STATUS_OK : [ANALOG_IO.h](#)
- ANALOG_IO_STATUS_t : [ANALOG_IO.h](#)
- ANALOG_IO_t : [ANALOG_IO.h](#)

ANALOG_IO

Home					
File List	Globals				
All	Functions	Typedefs	Enumerations	Enumerator	

- ANALOG_IO_GetAppVersion() : [ANALOG_IO.c](#) , [ANALOG_IO.h](#)
 - ANALOG_IO_GetInput() : [ANALOG_IO.h](#)
 - ANALOG_IO_Init() : [ANALOG_IO.c](#) , [ANALOG_IO.h](#)
-



ANALOG_IO

Home					
File List		Globals			
All	Functions	Typedefs	Enumerations	Enumerator	

- ANALOG_IO_CONFIG_PTR_t : [ANALOG_IO.h](#)
 - ANALOG_IO_STATE_t : [ANALOG_IO.h](#)
 - ANALOG_IO_STATUS_t : [ANALOG_IO.h](#)
 - ANALOG_IO_t : [ANALOG_IO.h](#)
-

ANALOG_IO

Home					
File List	Globals				
All	Functions	Typedefs	Enumerations	Enumerator	

- ANALOG_IO_STATE : [ANALOG_IO.h](#)
 - ANALOG_IO_STATUS : [ANALOG_IO.h](#)
-

ANALOG_IO

Home					
File List		Globals			
All	Functions	Typedefs	Enumerations	Enumerator	

- ANALOG_IO_STATE_INITIALIZED : [ANALOG_IO.h](#)
- ANALOG_IO_STATE_NOT_INITIALIZED : [ANALOG_IO.h](#)
- ANALOG_IO_STATUS_FAILURE : [ANALOG_IO.h](#)
- ANALOG_IO_STATUS_OK : [ANALOG_IO.h](#)

ANALOG_IO

Home

File List

Globals

ANALOG_IO.h

Go to the documentation of this file.

```
00001
00060 /*****
***** * HEADER FILES *
00061 * HEADER FILES
00062 ****
***** */
00063
00064 #ifndef ANALOG_IO_H
00065 #define ANALOG_IO_H
00066
00067 #include <xmc_gpio.h>
00068 #include <DAVE_Common.h>
00069 #include "analog_io_conf.h"
00070
00071 /*****
***** * MACROS *
00072 * MACROS
00073 ****
***** */
00074 #if (((XMC_LIB_MAJOR_VERSION == 2U) && \
00075         (XMC_LIB_MINOR_VERSION >= 0U) && \
00076         (XMC_LIB_PATCH_VERSION >= 0U)))
00077 #error "ANALOG_IO requires XMC Peripheral Li
```

```
brary v2.0.0 or higher"
00078 #endif
00079
00080 /*****
***** ****
***** ****
00081 * ENUMS
00082 ****
***** ****
***** ****
00083
00092 typedef enum ANALOG_IO_STATUS
00093 {
00094     ANALOG_IO_STATUS_OK      = 0U,
00095     ANALOG_IO_STATUS_FAILURE = 1U
00096 } ANALOG_IO_STATUS_t;
00097
00102 typedef enum ANALOG_IO_STATE
00103 {
00104     ANALOG_IO_STATE_NOT_INITIALIZED = 0,
00105     ANALOG_IO_STATE_INITIALIZED
00106 } ANALOG_IO_STATE_t;
00107
00108
00113 /*****
***** ****
***** ****
00114 * DATA STRUCTURES
00115 ****
***** ****
***** ****
00123 typedef void (*ANALOG_IO_CONFIG_PTR_t)(void)
;
00124
00128 typedef struct ANALOG_IO
00129 {
00130     ANALOG_IO_CONFIG_PTR_t config_ptr;
```

```
00131     ANALOG_IO_STATE_t *state;
00132     XMC_GPIO_PORT_t *port;
00133     uint8_t pin;
00134 } ANALOG_IO_t;
00135
00136
00141 /*****
00141 ****
00141 ****
00141 ****
00142 * LOCAL ROUTINES
00143 ****
00143 ****
00143 ****
00143 ****
00144
00145 /*****
00145 ****
00145 ****
00145 ****
00146 * API IMPLEMENTATION
00147 ****
00147 ****
00147 ****
00148
00149 #ifdef __cplusplus
00150 extern "C" {
00151 #endif
00152
00198 DAVE_APP_VERSION_t ANALOG_IO_GetAppVersion(
void);
00199
00239 ANALOG_IO_STATUS_t ANALOG_IO_Init(const ANALOG_IO_t *const handle);
00240
00241
00290 __STATIC_INLINE uint32_t ANALOG_IO_GetInput(
const ANALOG_IO_t *const handler)
00291 {
00292     XMC_ASSERT("ANALOG_IO_GetInput: handler n
```

```
    ull pointer", handler != NULL);
00293     return XMC_GPIO_GetInput(handler->port, h
andler->pin);
00294 }
00295
00299 #ifdef __cplusplus
00300 }
00301 #endif
00302
00303 /* Include APP extern file */
00304 #include "analog_io_extern.h"
00305
00306
00307 #endif /* ANALOG_IO_H */
```

ANALOG_IO

Home

File List

Globals

ANALOG_IO.c

Go to the documentation of this file.

```
00001
00055 /*****
***** 
***** 
00056 * HEADER FILES
00057 *****
***** 
***** 
***** /
00058
00059 #include "analog_io.h"
00060
00061 /*****
***** 
***** 
00062 * MACROS
00063 *****
***** 
***** 
***** /
00064
00065 /*****
***** 
***** 
00066 * LOCAL DATA
00067 *****
***** 
***** 
***** /
00068
```

```
00069 /*****  
*****  
*****  
00070 * LOCAL ROUTINES  
00071 ****  
*****  
*****  
*****  
*****  
00072  
00073 /*****  
*****  
*****  
*****  
00074 * API IMPLEMENTATION  
00075 ****  
*****  
*****  
*****  
*****  
00076  
00077 /*Get driver version*/  
00078 DAVE_APP_VERSION_t ANALOG_IO_GetAppVersion(v  
oid)  
00079 {  
00080     DAVE_APP_VERSION_t version;  
00081  
00082     version.major = (uint8_t)ANALOG_IO_MAJOR  
_VERSION;  
00083     version.minor = (uint8_t)ANALOG_IO_MINOR  
_VERSION;  
00084     version.patch = (uint8_t)ANALOG_IO_PATCH  
_VERSION;  
00085  
00086     return (version);  
00087 }  
00088  
00096 ANALOG_IO_STATUS_t ANALOG_IO_Init(const ANAL  
OG_IO_t *const handle)  
00097 {  
00098     XMC_ASSERT("ANALOG_IO_Init: Passed handler  
is a null pointer", handle != NULL);
```

```
00099     if (*handle->state) == ANALOG_IO_STATE_NO
T_INITIALIZED)
00100     {
00101         if (handle->config_ptr != NULL)
00102         {
00103             /* Instance specific initialization fu
nction call*/
00104             handle->config_ptr();
00105         }
00106         /* Set the state variable to initialized
state*/
00107         *(handle->state) = ANALOG_IO_STATE_INITI
ALIZED;
00108     }
00109     return (ANALOG_IO_STATUS_OK);
00110 }
```

