**GPIO Driver**

**Programmers Guide**

**Version 1.0**

**Jiang Yuchang**

**Instrument Cluster Software Group**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



O-Film Corporation

Infotainment & Instrument Cluster  
Business Unit

目录

[1 Introduction and functional overview 2](#_Toc441068598)

[2 Acronyms and abbreviations 3](#_Toc441068599)

[3 Application Interface 4](#_Toc441068600)

[3.1 Type definitions 4](#_Toc441068601)

[3.2 Function definition 10](#_Toc441068602)

[3.2.1 Port.h 10](#_Toc441068603)

[3.2.2 Dio.h 12](#_Toc441068604)

[4 Sequence diagrams 15](#_Toc441068605)

[4.1 Overall Configuration of Ports 15](#_Toc441068606)

[4.2 Set the direction of a Port Pin 16](#_Toc441068607)

[4.3 Read a value from a digital I/O 16](#_Toc441068608)

[4.4 Write a value to a digital I/O 16](#_Toc441068609)

[5 Revision history 17](#_Toc441068610)

# 1 Introduction and functional overview

This specification specifies the functionality, API and the configuration of the AUTOSAR Basic Software module PORT and DIO Driver.

This driver specification is applicable for on-chip ports and port pins. This module shall provide the service for initializing the whole PORT structure of the microcontroller. Many ports and port pins can be assigned to various functionalities, e.g.

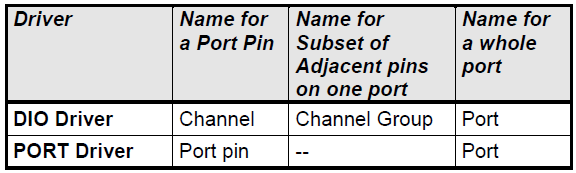
* General purpose I/O
* ADC
* SPI
* SCI
* PWM
* CAN
* LIN
* etc

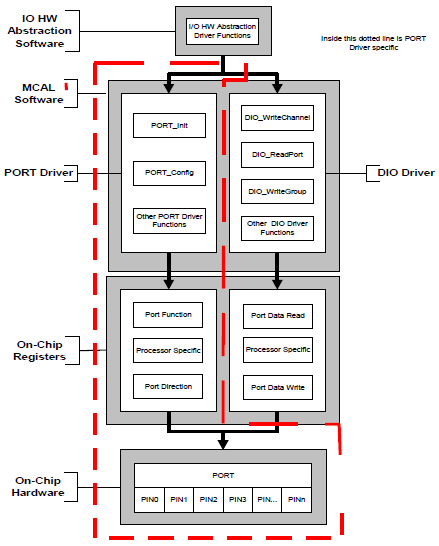
For this reason, there shall be an overall configuration and initialization of this port structure. The configuration and mode of these port pins is microcontroller and ECU dependent.

Port initialization data shall be written to each port as efficiently as possible. This PORT driver module shall complete the overall configuration and initialization of the port structure which is used in the DIO driver module. Therefore, the DIO driver works on pins and ports which are configured by the PORT driver.

The PORT driver shall be initialized prior to use of the DIO functions. Otherwise DIO functions will exhibit undefined behavior.

The diagram below identifies the PORT driver functions, and the structure of the PORT driver and DIO driver within the MCAL software layer.





# 2 Acronyms and abbreviations

|  |  |
| --- | --- |
| **Abbreviation/Acronym:** | **Description:** |
| DIO | Digital Input Output |
| ID | Identifier |
| ADC | Analog to Digital Converter |
| SPI | Serial Peripheral Interface |
| PWM | Pulse Width Modulation |
| ICU | Input Capture Unit |
| DET | Development Error Tracer |
| DEM | Diagnostic Event Manager |
| LSB | Least Significant Bit |
| MSB | Most Significant Bit |
| MCU | Micro Controller Unit |

# 3 Application Interface

## 3.1 Type definitions

/\*Type of the external data structure containing the initialization data for this module\*/

typedef struct

{

uint16 \*address;

uint16 value;

} **Port\_ConfigType**;

/\*Data type for the symbolic name of a port pin\*/

typedef uint16 **Port\_PinType**;

/\*Possible directions of a port pin\*/

typedef enum

{

PORT\_PIN\_OUT = 0,

PORT\_PIN\_IN = 1

} **Port\_PinDirectionType**;

/\*Different port pin modes\*/

typedef uint16 Port\_PinMode;

/\*Port Pin Mode enum list\*/

typedef enum

{

PORT\_PIN\_MODE\_GPIO,

PORT\_PIN\_MODE\_ALT\_OUT1,

PORT\_PIN\_MODE\_ALT\_IN1,

PORT\_PIN\_MODE\_ALT\_OUT2,

PORT\_PIN\_MODE\_ALT\_IN2,

PORT\_PIN\_MODE\_ALT\_OUT3,

PORT\_PIN\_MODE\_ALT\_IN3,

PORT\_PIN\_MODE\_ALT\_OUT4,

PORT\_PIN\_MODE\_ALT\_IN4,

PORT\_PIN\_MODE\_ALT\_OUT5,

PORT\_PIN\_MODE\_ALT\_IN5,

PORT\_PIN\_MODE\_MAX\_VALUE

} **PORT\_PIN\_MODE**;

/\*Type of the external data structure containing the pin number, pin direction register, port function control register,

\*\*port function control expansion register, port function control additional expansion register, pin bit mask\*/

typedef struct

{

Port\_PinType pin\_index; /\*pin number index\*/

uint16 \*pmc; /\*port mode control register\*/

uint16 \*pm; /\*port mode register\*/

uint16 \*pfc; /\*port function control register\*/

uint16 \*pfce; /\*port function control expansion register\*/

uint16 \*pfcae; /\*port function control additional expansion register\*/

uint16 bit\_mask; /\*port pin bit mask\*/

} **Port\_Pin\_Reg16\_Map**;

/\*This structure contains all post-build configurable parameters of the DIO driver.

\*\*A pointer to this structure is passed to the DIO driver initialization function for configuration.\*/

typedef struct

{

uint16 \*address;

uint16 value;

} **Dio\_ConfigType**;

/\*Numeric ID of a DIO channel.\*/

typedef uint16 **Dio\_ChannelType**;

/\*Numeric ID of a DIO port.\*/

typedef uint8 **Dio\_PortType**;

/\*Type for the definition of a channel group, which consists of several adjoining channels within a port.\*/

typedef struct

{

Dio\_PortType port; /\*This shall be the port on which the Channel group is defined.\*/

uint8 offset; /\*This element shall be the position of the Channel Group on the port, counted from the LSB.\*/

uint16 mask; /\*This element mask which defines the positions of the channel group.\*/

} **Dio\_ChannelGroupType**;

/\*These are the possible levels a DIO channel can have (input or output)\*/

typedef uint8 **Dio\_LevelType**;

/\*If the MCU owns ports of different port widths (e.g. 4, 8,16...Bit)

\*\*Dio\_PortLevelType inherits the size of the largest port.\*/

typedef uint16 **Dio\_PortLevelType**;

/\*Type of the external data structure containing the pin number, pin input register,

\*\*pin outpur register, pin bit mask\*/

typedef struct

{

Dio\_ChannelType pin\_index; /\*Pin number index\*/

uint16 \*p; /\*Port register\*/

uint16 \*ppr; /\*Port pin read register\*/

uint16 bit\_mask; /\*Port pin bit mask\*/

} **Pin\_Dio\_Reg16\_Map**;

/\*Type of the external data structure containing the pin number, pin input register,

\*\*pin outpur register, pin bit mask\*/

typedef struct

{

Dio\_PortType port\_index; /\*Port index\*/

uint16 \*p; /\*Port register\*/

uint16 \*ppr; /\*Port pin read register\*/

} **Port\_Dio\_Reg16\_Map**;

/\*\*\*\*\*\*\*\*\*\*\*Port Pin Index List\*\*\*\*\*\*\*\*\*\*/

typedef enum

{

/\*Port 0\*/

PORT0\_0,

PORT0\_1,

PORT0\_2,

PORT0\_3,

PORT0\_4,

PORT0\_5,

PORT0\_6,

PORT0\_7,

PORT0\_8,

PORT0\_9,

PORT0\_10,

PORT0\_11,

PORT0\_12,

PORT0\_13,

PORT0\_14,

/\*Port 1\*/

PORT1\_0,

PORT1\_1,

PORT1\_2,

PORT1\_3,

PORT1\_4,

PORT1\_5,

PORT1\_6,

PORT1\_7,

PORT1\_8,

PORT1\_9,

PORT1\_10,

PORT1\_11,

PORT1\_12,

PORT1\_13,

PORT1\_14,

PORT1\_15,

/\*Port 2\*/

PORT2\_0,

PORT2\_1,

PORT2\_2,

PORT2\_3,

PORT2\_4,

PORT2\_5,

PORT2\_6,

/\*Port 8\*/

PORT8\_0,

PORT8\_1,

PORT8\_2,

PORT8\_3,

PORT8\_4,

PORT8\_5,

PORT8\_6,

PORT8\_7,

PORT8\_8,

PORT8\_9,

PORT8\_10,

PORT8\_11,

PORT8\_12,

/\*Port 9\*/

PORT9\_0,

PORT9\_1,

PORT9\_2,

PORT9\_3,

PORT9\_4,

PORT9\_5,

PORT9\_6,

/\*Port 10\*/

PORT10\_0,

PORT10\_1,

PORT10\_2,

PORT10\_3,

PORT10\_4,

PORT10\_5,

PORT10\_6,

PORT10\_7,

PORT10\_8,

PORT10\_9,

PORT10\_10,

PORT10\_11,

PORT10\_12,

PORT10\_13,

PORT10\_14,

PORT10\_15,

/\*Port 11\*/

PORT11\_0,

PORT11\_1,

PORT11\_2,

PORT11\_3,

PORT11\_4,

PORT11\_5,

PORT11\_6,

PORT11\_7,

PORT11\_8,

PORT11\_9,

PORT11\_10,

PORT11\_11,

PORT11\_12,

PORT11\_13,

PORT11\_14,

PORT11\_15,

/\*Port 12\*/

PORT12\_0,

PORT12\_1,

PORT12\_2,

PORT12\_3,

PORT12\_4,

PORT12\_5,

/\*Port 18\*/

PORT18\_0,

PORT18\_1,

PORT18\_2,

PORT18\_3,

PORT18\_4,

PORT18\_5,

PORT18\_6,

PORT18\_7,

/\*Port 20\*/

PORT20\_0,

PORT20\_1,

PORT20\_2,

PORT20\_3,

PORT20\_4,

PORT20\_5,

/\*JPort 0\*/

JPORT0\_0,

JPORT0\_1,

JPORT0\_2,

JPORT0\_3,

JPORT0\_4,

JPORT0\_5,

JPORT0\_6,

/\*APort 0\*/

APORT0\_0,

APORT0\_1,

APORT0\_2,

APORT0\_3,

APORT0\_4,

APORT0\_5,

APORT0\_6,

APORT0\_7,

APORT0\_8,

APORT0\_9,

APORT0\_10,

APORT0\_11,

APORT0\_12,

APORT0\_13,

APORT0\_14,

APORT0\_15,

/\*APort 1\*/

APORT1\_0,

APORT1\_1,

APORT1\_2,

APORT1\_3,

APORT1\_4,

APORT1\_5,

APORT1\_6,

APORT1\_7,

APORT1\_8,

APORT1\_9,

APORT1\_10,

APORT1\_11,

APORT1\_12,

APORT1\_13,

APORT1\_14,

APORT1\_15,

IPORT0\_0

} **PORT\_PIN\_TYPE**;

/\*\*\*\*\*\*\*\*\*\*\*Port Index List\*\*\*\*\*\*\*\*\*\*/

typedef enum

{

PORT0,

PORT1,

PORT2,

PORT8,

PORT9,

PORT10,

PORT11,

PORT12,

PORT18,

PORT20,

JPORT0,

APORT0,

APORT1,

IPORT0

} **DIO\_PORT\_TYPE**;

## 3.2 Function definition

### 3.2.1 Port.h

1 void Port\_Init(const Port\_ConfigType \*ConfigPtr)

|  |  |
| --- | --- |
| Service name: | **Port\_Init** |
| Description: | Initializes the Port Driver module. |
| Sync/Async: | Synchronous |
| Input: | ConfigPtr: Pointer to configuration set |
| Output: | None |
| Return value: | None |

2 void Port\_SetPinDirection(Port\_PinType Pin, Port\_PinDirectionType Direction)

|  |  |
| --- | --- |
| Service name: | **Port\_SetPinDirection** |
| Description: | Sets the port pin direction. |
| Sync/Async: | Synchronous |
| Input: | Pin: Port Pin ID number  Direction: Port Pin Direction |
| Output: | None |
| Return value: | None |

3 void Port\_SetPinMode(Port\_PinType Pin, Port\_PinMode Mode)

|  |  |
| --- | --- |
| Service name: | **Port\_SetPinMode** |
| Description: | Sets the port pin mode. |
| Sync/Async: | Synchronous |
| Input: | Pin: Port Pin ID number  Mode: New Port Pin Mode to be set on port pin |
| Output: | None |
| Return value: | None |

4 void Port\_RefreshPortDirection(void)

|  |  |
| --- | --- |
| Service name: | **Port\_RefreshPortDirection** |
| Description: | Refreshes port direction. |
| Sync/Async: | Synchronous |
| Input: | None |
| Output: | None |
| Return value: | None |

5 void Port\_GetVersionInfo(Std\_VersionInfoType \*Versioninfo)

|  |  |
| --- | --- |
| Service name: | **Port\_GetVersionInfo** |
| Description: | Returns the version information of this module. |
| Sync/Async: | Synchronous |
| Input: | None |
| Output: | Versioninfo: Pointer to where to store the version information of this module |
| Return value: | None |

6 void Port\_Sleep(const Port\_ConfigType \*ConfigPtr)

|  |  |
| --- | --- |
| Service name: | **Port\_Sleep** |
| Description: | Updates the Port Driver Module before MCU sleep. |
| Sync/Async: | Synchronous |
| Input: | ConfigPtr: Pointer to configuration set |
| Output: | None |
| Return value: | None |

7 void Port\_Wakeup(const Port\_ConfigType \*ConfigPtr)

|  |  |
| --- | --- |
| Service name: | **Port\_Wakeup** |
| Description: | Updates the Port Driver Module after MCU wakeup. |
| Sync/Async: | Synchronous |
| Input: | ConfigPtr: Pointer to configuration set |
| Output: | None |
| Return value: | None |

8 uint8 Port\_WriteProRegister(volatile uint32 \*protCmdReg, volatile uint32 \*targetReg, volatile uint32 \*pprotStatusReg,uint32 regValue, uint8 repeatTime)

|  |  |
| --- | --- |
| Service name: | **Port\_WriteProRegister** |
| Description: | Updates write-protected register. |
| Sync/Async: | Synchronous |
| Input: | protCmdReg: Pointer to protection command register  pprotStatusReg: Pointer to port protetion status register  regValue: Value need to update targetReg  repeatTime: Repeat time if write error happened |
| Output: | targetReg: Pointer to target register which need to be updated |
| Return value: | Value to show whether write success or failure(PORT\_PROTECTION\_REGISTER\_WRITE\_SUCCESS: write success; PORT\_PROTECTION\_REGISTER\_WRITE\_FAIL: write failure) |

### 3.2.2 Dio.h

1 void Dio\_Init(const Dio\_ConfigType \*ConfigPtr)

|  |  |
| --- | --- |
| Service name: | **Dio\_Init** |
| Description: | Initializes the Dio Driver Module. |
| Sync/Async: | Synchronous |
| Input: | ConfigPtr: Pointer to configuration set |
| Output: | None |
| Return value: | None |

2 Dio\_LevelType Dio\_ReadChannel(Dio\_ChannelType ChannelId)

|  |  |
| --- | --- |
| Service name: | **Dio\_ReadChannel** |
| Description: | Returns the value of the specified DIO channel. |
| Sync/Async: | Synchronous |
| Input: | ChannelId: ID of DIO channel |
| Output: | None |
| Return value: | Value of the specified DIO channel |

3 void Dio\_WriteChannel(Dio\_ChannelType ChannelId, Dio\_LevelType Level)

|  |  |
| --- | --- |
| Service name: | **Dio\_WriteChannel** |
| Description: | Service to set a level of a channel. |
| Sync/Async: | Synchronous |
| Input: | ChannelId: ID of DIO channel  Level: Value to be written |
| Output: | None |
| Return value: | None |

4 Dio\_LevelType Dio\_FlipChannel(Dio\_ChannelType ChannelId)

|  |  |
| --- | --- |
| Service name: | **Dio\_FlipChannel** |
| Description: | Service to flip (change from 1 to 0 or from 0 to 1) the level of a channel and return the level of the channel after flip. |
| Sync/Async: | Synchronous |
| Input: | ChannelId: ID of DIO channel |
| Output: | None |
| Return value: | Value of the specified DIO channel |

5 Dio\_PortLevelType Dio\_ReadPort(Dio\_PortType PortId)

|  |  |
| --- | --- |
| Service name: | **Dio\_ReadPort** |
| Description: | Returns the level of all channels of that port. |
| Sync/Async: | Synchronous |
| Input: | PortId: ID of DIO Port |
| Output: | None |
| Return value: | Level of all channels of that port |

6 void Dio\_WritePort(Dio\_PortType PortId, Dio\_PortLevelType Level)

|  |  |
| --- | --- |
| Service name: | **Dio\_WritePort** |
| Description: | Service to set a value of the port. |
| Sync/Async: | Synchronous |
| Input: | PortId: ID of DIO Port  Level: Value to be written |
| Output: | None |
| Return value: | None |

7 Dio\_PortLevelType Dio\_ReadChannelGroup(const Dio\_ChannelGroupType \*ChannelGroupIdPtr)

|  |  |
| --- | --- |
| Service name: | **Dio\_ReadChannelGroup** |
| Description: | This Service reads a subset of the adjoining bits of a port. |
| Sync/Async: | Synchronous |
| Input: | ChannelGroupIdPtr: Pointer to ChannelGroup |
| Output: | None |
| Return value: | Level of a subset of the adjoining bits of a port |

8 void Dio\_WriteChannelGroup(const Dio\_ChannelGroupType \*ChannelGroupIdPtr, Dio\_PortLevelType Level)

|  |  |
| --- | --- |
| Service name: | **Dio\_WriteChannelGroup** |
| Description: | Service to set a subset of the adjoining bits of a port to a specified level. |
| Sync/Async: | Synchronous |
| Input: | ChannelGroupIdPtr: Pointer to ChannelGroup  Level: Value to be written |
| Output: | None |
| Return value: | None |

9 void Dio\_GetVersionInfo(Std\_VersionInfoType \*Versioninfo)

|  |  |
| --- | --- |
| Service name: | **Dio\_GetVersionInfo** |
| Description: | Returns the version information of this module. |
| Sync/Async: | Synchronous |
| Input: | None |
| Output: | Versioninfo: Pointer to where to store the version information of this module |
| Return value: | None |

10 void Dio\_Mainfunction(void)

|  |  |
| --- | --- |
| Service name: | **Dio\_Mainfunction** |
| Description: | This function will be executed by periodic task to do some digital input debounce processing. |
| Sync/Async: | Synchronous |
| Input: | None |
| Output: | None |
| Return value: | None |

11 uint8 Dio\_getDbncState(uint8 dbncChannelId)

|  |  |
| --- | --- |
| Service name: | **Dio\_getDbncState** |
| Description: | This function get dio dbnc state. |
| Sync/Async: | Synchronous |
| Input: | dbncChannelId: dio dbnc channel ID to identify different dio channel |
| Output: | None |
| Return value: | Debounced value of the specified DIO channel |

12 uint8 Dio\_getRawState(uint8 dbncChannelId)

|  |  |
| --- | --- |
| Service name: | **Dio\_getRawState** |
| Description: | Returns the version information of this module. |
| Sync/Async: | Synchronous |
| Input: | dbncChannelId: dio dbnc channel ID to identify different dio channel |
| Output: | None |
| Return value: | Raw value of the specified DIO channel |

# 4 Sequence diagrams

## 4.1 Overall Configuration of Ports



## 4.2 Set the direction of a Port Pin



## 4.3 Read a value from a digital I/O



## 4.4 Write a value to a digital I/O



# 5 Revision history

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Revision Version** | **Revision Author** | **Section** | **Date** | **Description** |
| 1.0 | Jiang Yuchang | All | 20-Jan-2016 | Initial version |
| 1.1 | Jiang Yuchang | 3.2.2 | 09-Mar-2016 | Add functions to handle digital input debounce |
|  |  |  |  |  |