Document Name	Specification of DIO Driver (SWS)
Document Author	HEX CLAN
Document Status	Published
Version Release Date	4/11/2023

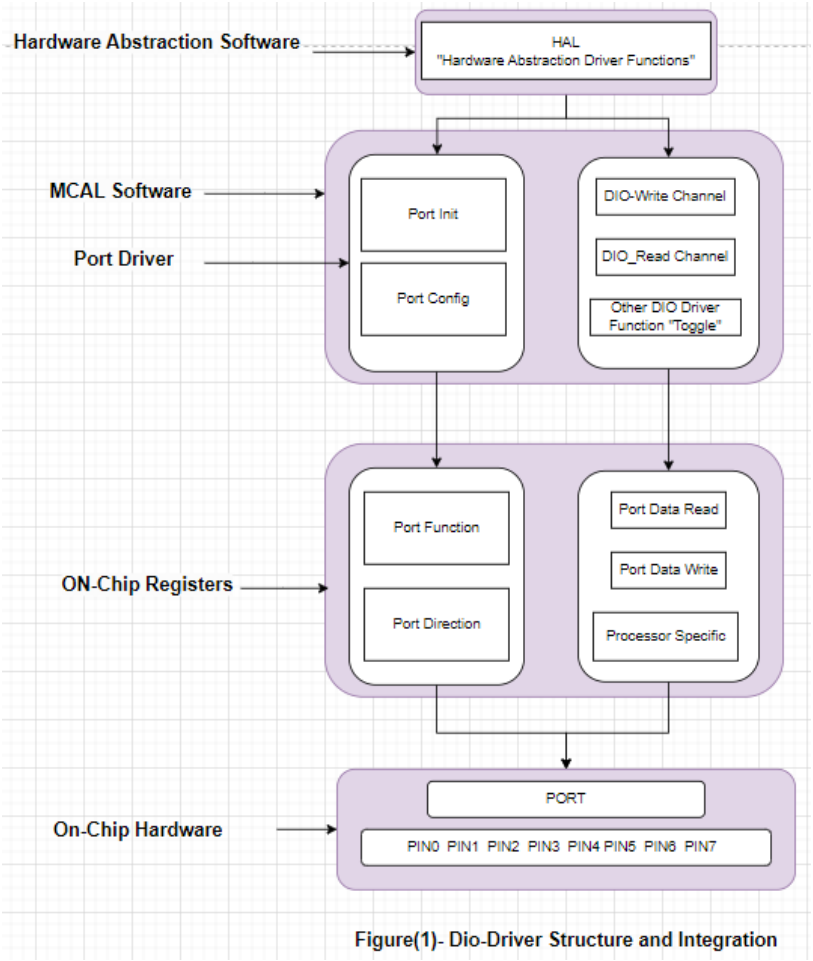
1) Introduction and functional overview

This specification specifies the functionality, API and the configuration of the Basic Software module DIO Driver. This specification is applicable to drivers only for on chip DIO pins and ports. The DIO Driver provides services for reading and writing to/from

- **DIO Channels** (Pins)
- **DIO Ports**

The behavior of those services is synchronous.

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



Figure(1)- Dio-Driver Structure and Integration

2) Acronyms and abbreviations

Acronyms and abbreviations	Description
DIO channel	Represents a single general-purpose digital input/output pin
DIO Port	Represents several DIO channels that are grouped by hardware (typically controlled by one hardware register). Example: Port A (8 bit)
STD-High	Represent the Bit Value High = 1
STD-Low	Represent the Bit Value Low = 0
Physical Level (Input)	Two states possible: LOW/HIGH. A bit value '0' represents a LOW, a bit value '1' represents a HIGH.
Physical Level (Output)	Two states possible: LOW/HIGH. A bit value '0' represents a LOW, a bit value '1' represents a HIGH.
ADC	Analog to Digital Converter
CLCD	Character Liquid Crystal Display
ICU	Input Capture Unit
PWM	Pulse Width Modulation
UART	Universal Asynchronous Recover transmitter
U8	Unsigned Character Data Type of size (8-Bits)
Void	Function does not return any value
(Data -type (Var)*)	The Variable points to the address, of size (Datatype), in data assigned to it.

3) Constraints and assumptions

3.1- Limitations: No limitations

3.2- Applicability to car domains: No restrictions

4) Dependencies to other modules Port Driver Modules

Many ports and port pins are assigned by the PORT Driver Module to various functionalities as for example:

- ADC
- CLCD
- TIMERS (ICU/PWM)
- UART

5) Functional specification

5.1- General Behavior

5.1.1- Background & Rationale

The DIO Driver abstracts the access to the microcontroller's hardware pins.

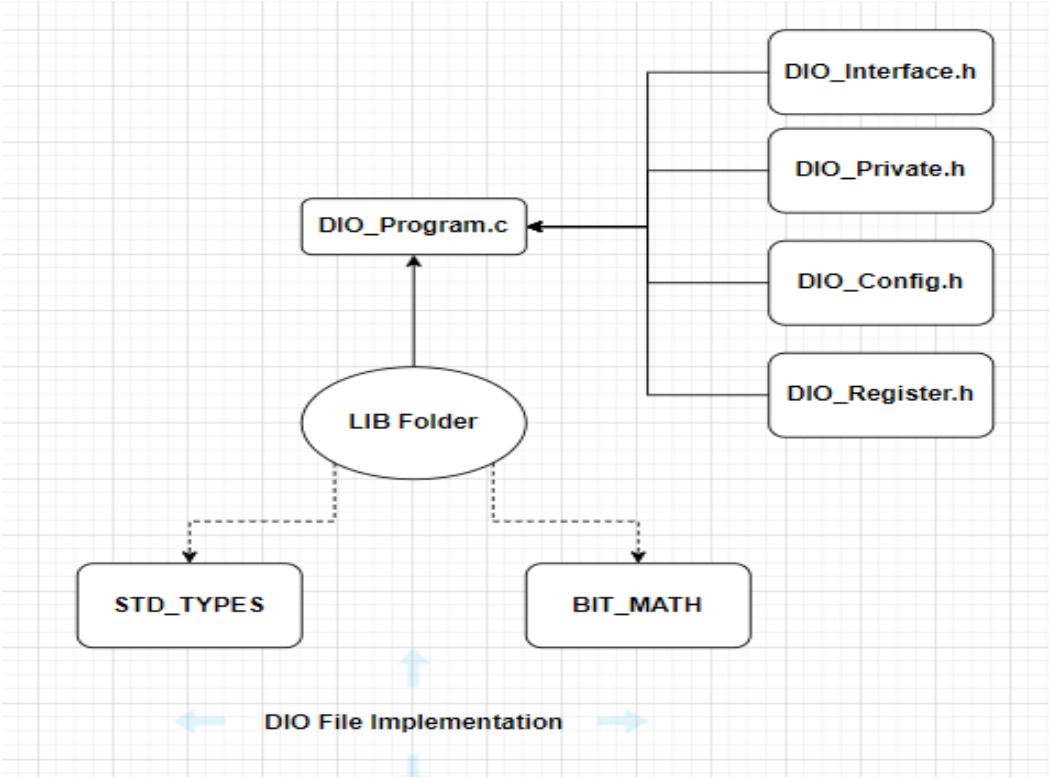
5.1.2- Requirements.

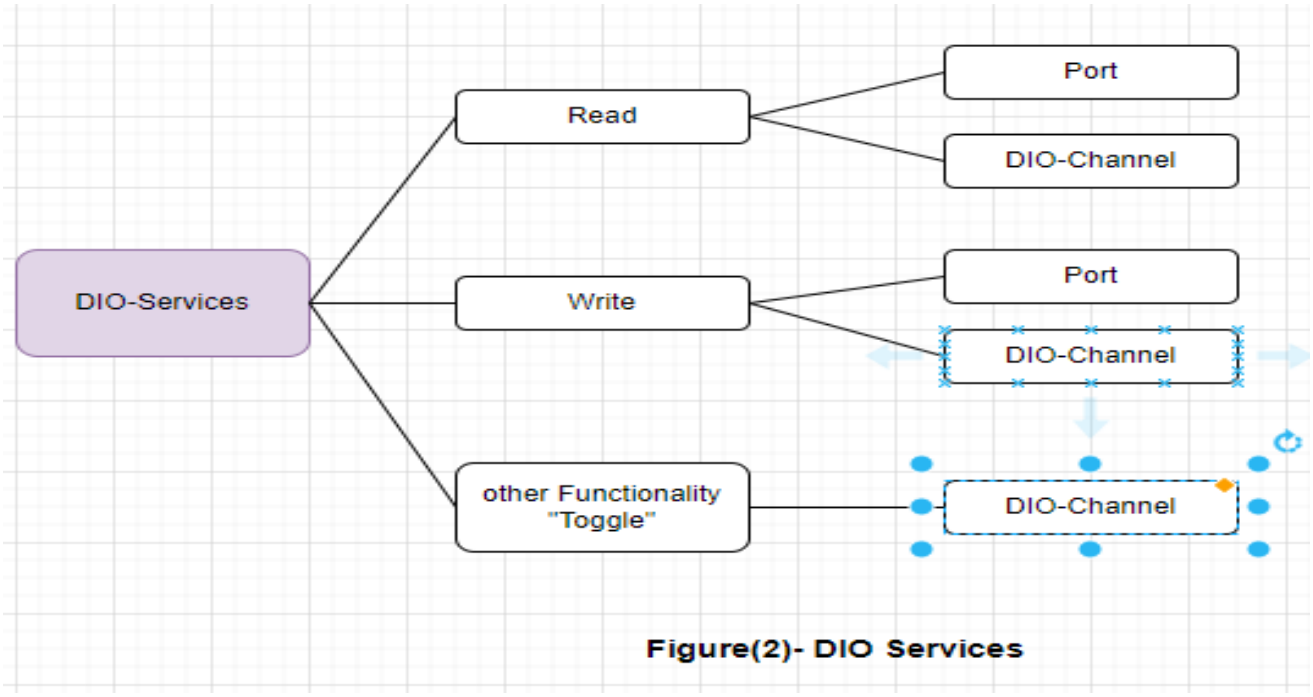
The Dio SWS shall define functions allowing Port/Channel based read and write access to the internal general purpose I/O ports.

- [SWS_Dio_00051] The Dio module shall not buffer data when providing read and write services.
  → The Dio SWS shall define synchronous read/write services.
- [SWS_Dio_00089] Values used by the DIO Driver for the software level of Channels are either STD_HIGH or STD_LOW.
- [SWS_Dio_00060] All read and write functions of the Dio module shall be re-entrant.
  →The DIO Driver may be accessed by different upper layer handlers or drivers. These upper layer modules may access the driver concurrently.
- [SWS_Dio_00026] [The configuration process for Dio module shall provide symbolic names for each configured DIO channel, port.

The DIO Driver provides the following services:

- The Dio SWS shall define functions to modify the levels of output channels individually or a port.
- The Dio SWS shall define functions to read the level of input and output individually or a port





Figure(2)- DIO Services

5.2- DIO write service

5.2.1- Background & Rationale

The DIO Driver provides services to transfer data to the microcontroller's pins

5.2.2- Requirements

[SWS_Dio_00064] The Dio module’s write functions shall work on input and output channels.
[SWS_Dio_00070] If a Dio write function is used on an input channel, it shall have no effect on the physical output level.
[SWS_Dio_00109] If supported by hardware, the Dio module shall set/clear the output data latch of an input channel so that the required level is output from the pin when the port driver configures the pin as a DIO output pin.

5.2.2.1- DIO channel write service

[SWS_Dio_00006] The DIO-Write Pin Direction
→the function shall set the level of a single DIO channel to Input or Output.
[SWS_Dio_00007] The DIO-Write Pin Value
→ the function shall set the level of a single DIO channel to STD-High or STD-Low.

5.2.2.2- DIO port write service

[SWS_Dio_00009] The DIO-Write Port Direction

→the function shall set simultaneously set the levels of all channels to Input or Output.
[SWS_Dio_00010] The DIO-Write Port Value
→the function shall set simultaneously set the levels of all channels to STD-High or STD-Low.

5.3- *DIO Read Service*

5.3.1- Background & Rationale

The DIO Driver provides services to transfer data from the microcontroller's pins

5.3.2- Requirements

[SWS_Dio_00065] The Dio module’s write functions shall work on input and output channels.

5.3.2.1- DIO channel Read service

[SWS_Dio_00011] The DIO-Read Pin Value

→the function shall Read the level of a single DIO. A bit value (0) indicates that the corresponding channel is physical STD_LOW, a bit value (1) indicates that the corresponding channel is physical STD_HIGH.

5.4- DIO Toggle

5.4.1- Background & Rationale

The DIO Driver provides services to toggle data from the microcontroller's pins

5.4.2- Requirements

[SWS_Dio_00066] The Dio module's write functions shall work on input and output channels.

5.4.2.1- DIO channel Toggle service

[SWS_Dio_00012] The DIO-Toggle Pin Value

→the function shall Toggle the level of a single DIO. Change the bit value(0) that corresponds physical STD_LOW, to a bit value (1) that the corresponds to physical STD_HIGH.

5.5- Error classification

5.5.1- Development Errors

Type of Error	Related Error Code	Error Value
ErrorState_t	• Invalid channel requested • Invalid port requested	0

5.5.2 Runtime Errors There are no runtime errors.

5.5.3 Transient Faults There are no transient faults.

5.5.4 Production Errors There are no production errors.

5.5.5 Extended Production Errors There are no extended production errors.

6) API specification

6.1 Imported types

In this chapter all types included from the following modules are listed:

Module	Header File	Imported types
LIB	STD_TYPES.h	U8 (typedef)
	STD_TYPES.h	OK (Error State)
	STD_TYPES.h	NOK (Error State)
	STD_TYPES.h	STD_High
	STD_TYPES.h	STD_Low

6.2 Type Definitions

6.2.1- Dio_PortType

Name	PORT(X)_ID	
Kind	Type	
Range	0 → No. of Ports	Shall Cover All DIO-Ports
Description	Numeric ID of DIO-Port	
Available Via	DIO_Interface.h	

[SWS_Dio_00018] Parameters of type PORT(X)_ID contain the numeric ID of a DIO port.
[SWS_Dio_00181] The mapping of ID is implementation specific but not configurable.
[SWS_Dio_00020] For parameter values of type PORT(X)_ID, the user shall use the symbolic names provided by the configuration description.

6.2.2- Dio_ChannelType

Name	PIN(X)	
Kind	Type	
Range	0 → No. of Pin	Shall Cover All DIO-Pin in a Port
Description	Numeric ID of each DIO-Pin in each Port	
Available Via	DIO_Interface.h	

[SWS_Dio_00015] Parameters of type PIN(X) contain the numeric ID of a DIO channel.
[SWS_Dio_00180] The mapping of the ID is implementation specific but not configurable.
[SWS_Dio_00017] ⌈For parameter values of type PIN(X), the Dio's user shall use the symbolic names provided by the configuration description.

6.3- Function definitions

Function Name	MCAL_DIO_u8_SetPortDirection
Syntax	ErrorState_t MCAL_DIO_u8_SetPortDirection(Dio_ConfigPin*Pset);
Synch/Asynch	Synchronous
Reentrancy	Reentrant
Parameters (In)	(Struct*) Pset | • u8 Port_Id; • u8 Pin_Id;

	• u8 dir; • u8 level;
Parameters (Out)	None
Parameters (In/Out)	None
Return Value	ErrorState_t | • OK • NOK
Description	Sets the Direction of the Port Post Build.
Available Via	DIO_Interface.h

Function Name	MCAL_DIO_u8_SetPinDirection
Syntax	ErrorState_t MCAL_DIO_u8_SetPinDirection(Dio_ConfigPin*Pset);
Synch/Asynch	Synchronous
Reentrancy	Reentrant
Parameters (In)	(Struct*) Pset | • u8 Port_Id; • u8 Pin_Id; • u8 dir; • u8 level;
Parameters (Out)	None
Parameters (In/Out)	None
Return Value	ErrorState_t | • OK • NOK
Description	Sets the Direction of the Pin Post Build.
Available Via	DIO_Interface.h

Function Name	MCAL_DIO_u8_SetPortValue
Syntax	ErrorState_t

	MCAL_DIO_u8_ SetPortValue (Dio_ConfigPin*Pset);
Synch/Asynch	Synchronous
Reentrancy	Reentrant
Parameters (In)	(Struct*) Pset | • u8 Port_Id; • u8 Pin_Id; • u8 dir; • u8 level;
Parameters (Out)	None
Parameters (In/Out)	None
Return Value	ErrorState_t | • OK • NOK
Description	Sets the Value of the Port Physical Level Post Build.
Available Via	DIO_Interface.h

Function Name	MCAL_DIO_u8_SetPinValue
Syntax	ErrorState_t MCAL_DIO_u8_ SetPortValue (Dio_ConfigPin*Pset);
Synch/Asynch	Synchronous
Reentrancy	Reentrant
Parameters (In)	(Struct*) Pset | • u8 Port_Id; • u8 Pin_Id; • u8 dir; • u8 level;
Parameters (Out)	None
Parameters (In/Out)	None
Return Value	ErrorState_t | • OK • NOK
Description	Sets the Value of the Pin Physical Level Post Build.
Available Via	DIO_Interface.h

Function Name	MCAL_DIO_u8_GetPinValue
Syntax	ErrorState_t MCAL_DIO_u8_GetPinValue(Dio_ConfigPin *Pset);
Synch/Asynch	Synchronous
Reentrancy	Reentrant
Parameters (In)	(Struct*) Pset | • u8 Port_Id; • u8 Pin_Id; • u8 dir; • u8 level;
Parameters (Out)	(Struct*) Pset | • u8 u8*Pret_value;
Parameters (In/Out)	None
Return Value	ErrorState_t | • OK • NOK
Description	Gets the Value of the Pin Physical Level Post Build.
Available Via	DIO_Interface.h

Function Name	MCAL_DIO_u8_TogglePinValue
Syntax	ErrorState_t MCAL_DIO_u8_TogglePinValue(Dio_ConfigPin *Pset);
Synch/Asynch	Synchronous
Reentrancy	Reentrant
Parameters (In)	(Struct*) Pset | • u8 Port_Id; • u8 Pin_Id; • u8 dir; • u8 level;
Parameters (Out)	None
Parameters (In/Out)	None
Return Value	ErrorState_t | • OK • NOK

Description	Change the Value of the Pin Physical Level From Physical High Logic(1) to Physical Low Logic(0) and vise versa.
Available Via	DIO_Interface.h

6.4 Expected Interfaces

This chapter lists all functions the Dio module requires from other modules.

9.6.1 Mandatory Interfaces

None.

7) Sequence Diagrams

The diagrams below show the sequences when calling the Dio_ReadChannel and Dio_WriteChannel services, they show normal operation mode.

7.1- Write a value to a digital I/O

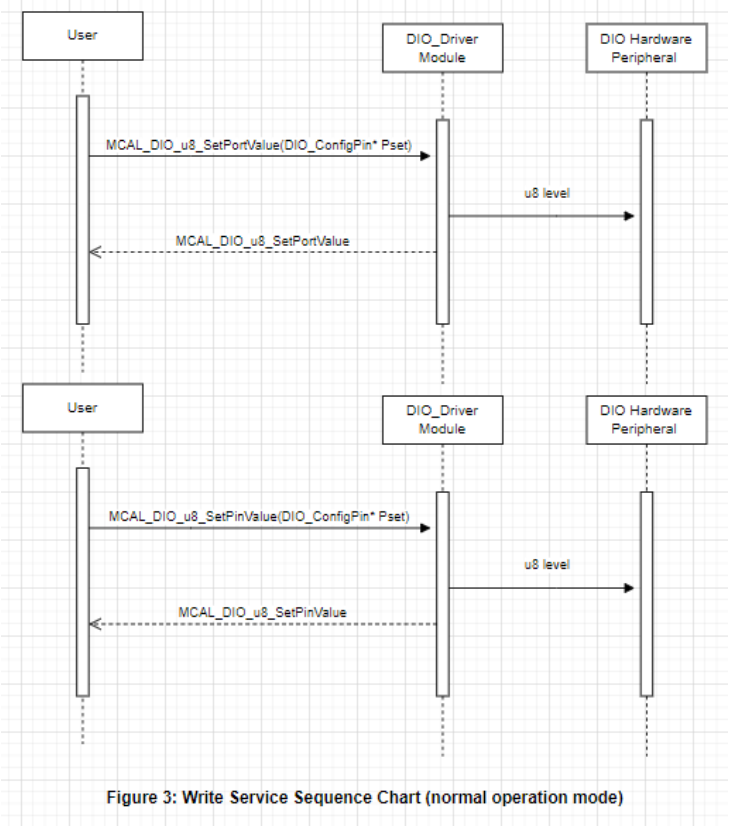


Figure 3: Write Service Sequence Chart (normal operation mode)

7.2- Write a value to a digital I/O

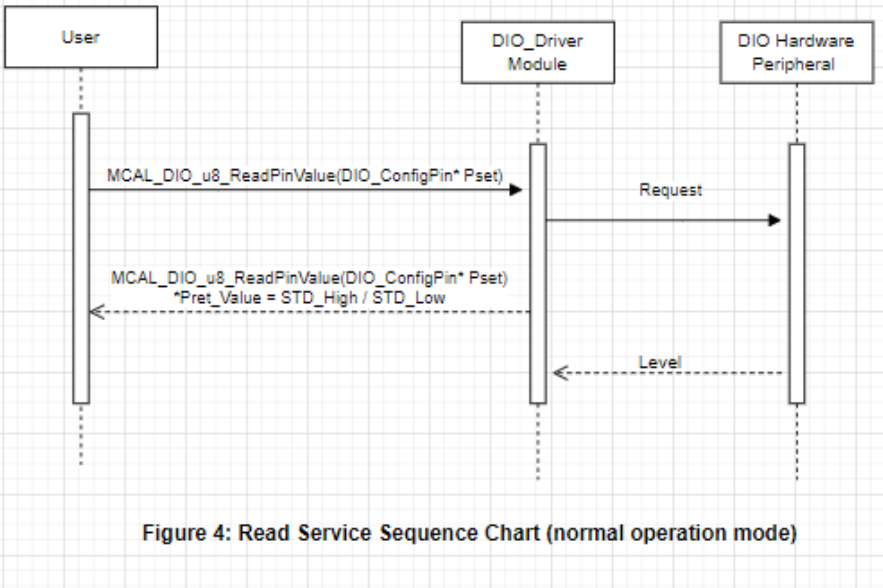


Figure 4: Read Service Sequence Chart (normal operation mode)