Practical Use-Case (Classic Software)

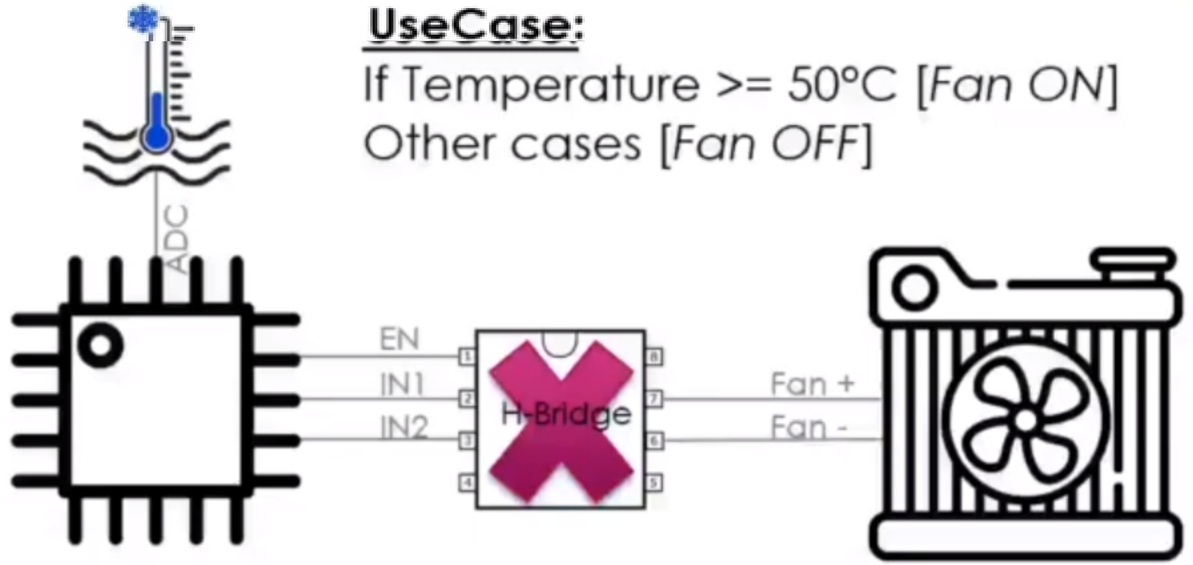
Typical Non-Autosar Software

```
Void ApplicationFunction()
{
    Codefor: ADC Pin Initialize
    Codefor: IO Pin Initialize
    Codefor: H-Bridge IC Initialize

    while(1)
    {
        Codefor: StartADCConversion

        Result = ReadADCValues;
        Temperature = Result * 0.5; // Raw to °C

        if(Temperature >= 50 )
        {
            IN1_PIN = 1; // Switch ON Motor
            IN2_PIN = 0;
        }
        else
        {
            IN1_PIN = 0; // Switch OFF Motor
            IN2_PIN = 0;
        }
    }
}
```



UseCase:

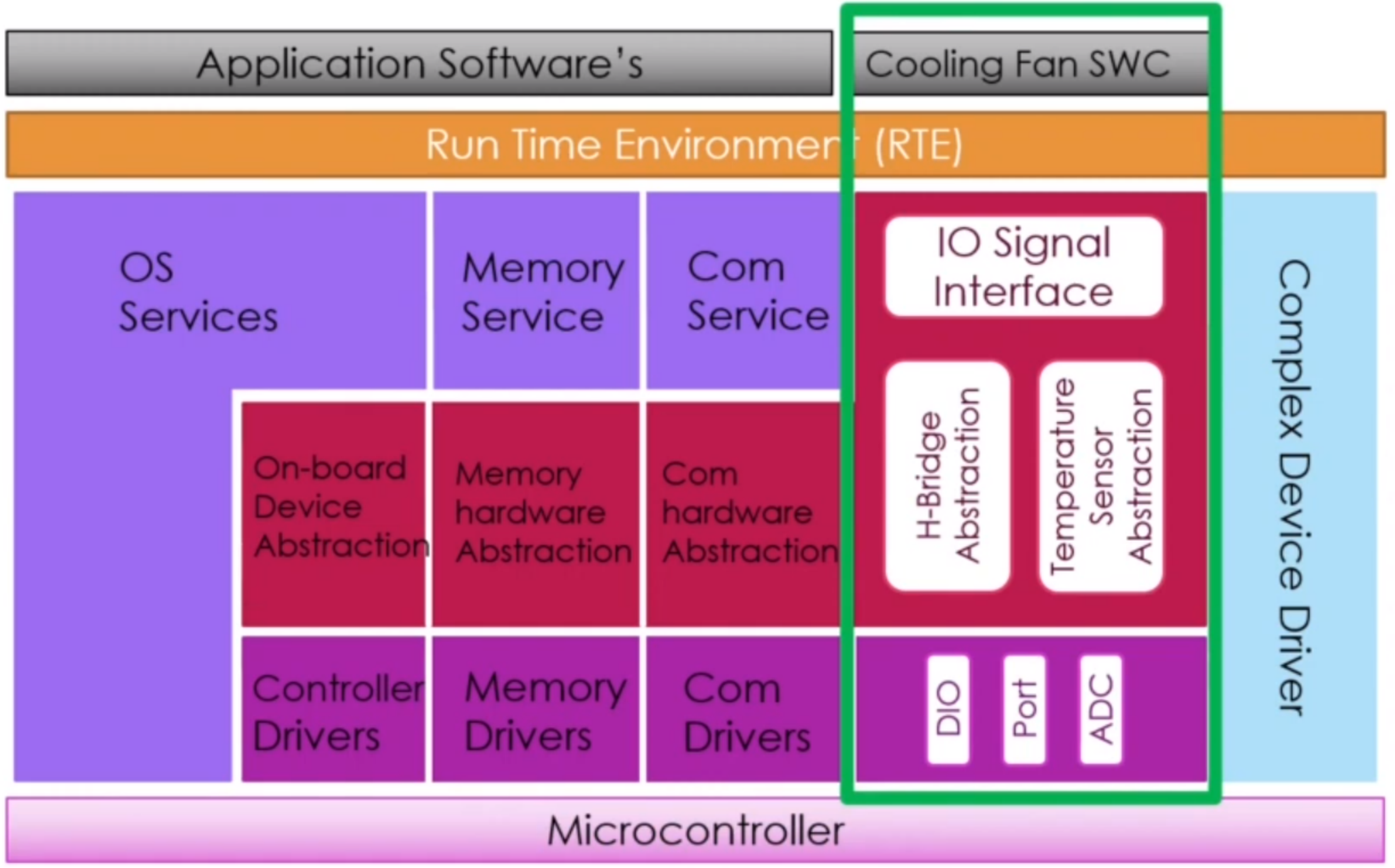
If Temperature >= 50°C [*Fan ON*]

Other cases [*Fan OFF*]

Conclusion (Non-Autosar Software):

- Application Software and hardware are tightly coupled
- Complete Software has to be changed in-case of any hardware changes
- Cost for software development is too huge
- Software not structured or reusable with different hardware's
- Customers (OEMs) become too much dependant to the suppliers for a long term supply of software and hardware during production

Cooling Fan Use-Case (Autosar Software)



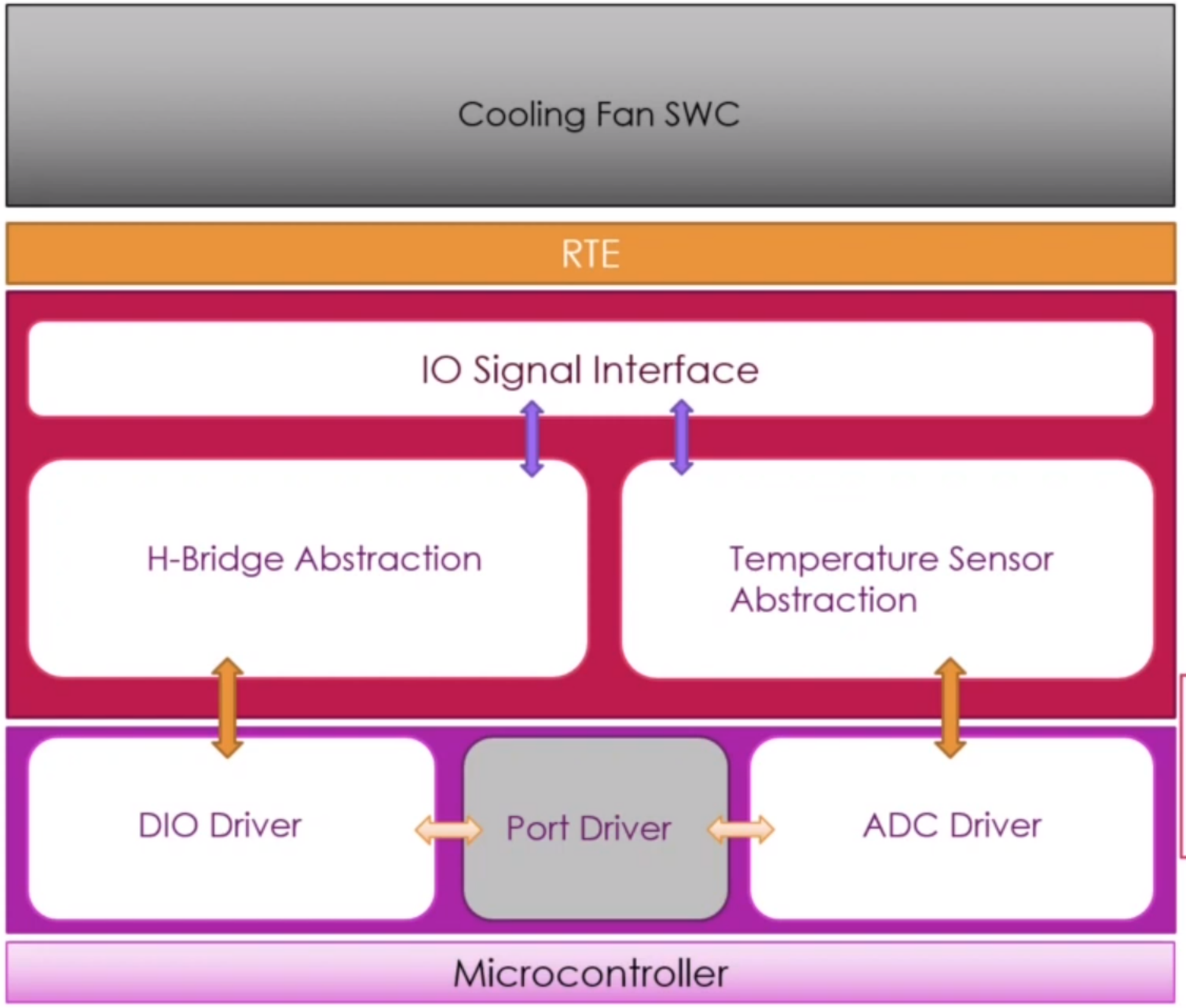
Requirements layer wise:

Application Software: (SWC)
- Control Fan based on Temperature

Drivers: (IO Abstraction)
- H-Bridge Driver
- Temperature Sensor Driver

Hardware Pins: (IO Driver)
- 3 Digital IO Ports
- 1 ADC Port

Cooling Fan Use-Case (Autosar Software)

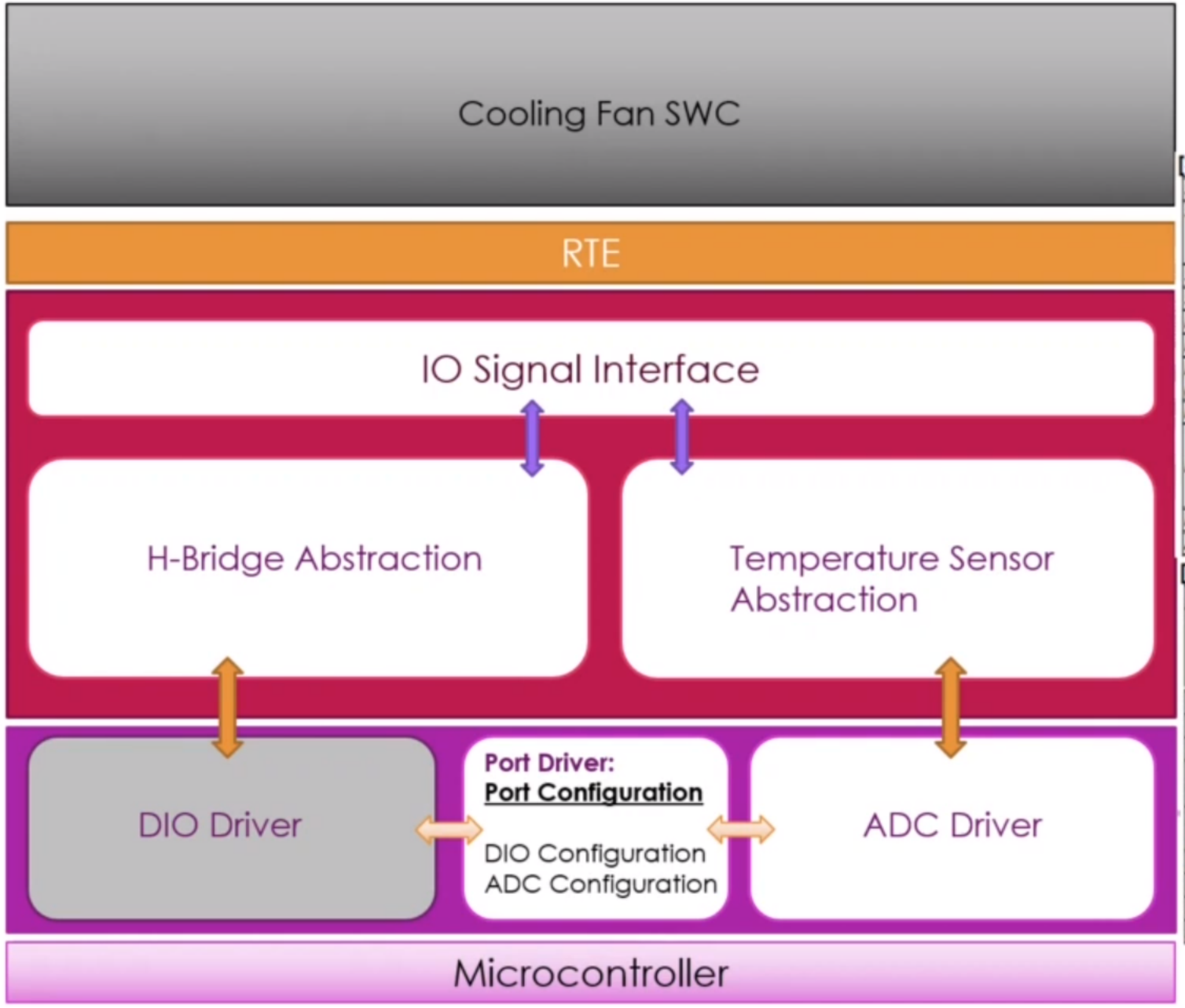


PORT Driver Configuration from AR4.4.0 standards

Parameters	Comments
PortPinId	Set ID for a port pin from (1-65535)
PortPinDirection	Pin direction IN or OUT
PortPinDirectionChangeable	Set to TRUE is port pin direction is changeable during runtime
PortPinLevelValue	Set Port pin HIGH or LOW during initialization
PortPinInitialMode	Initial Port pins as ADC, DIO, SPI, PWM etc..
PortPinMode	Change Port pins as ADC, DIO, SPI, PWM etc..

	PortPinId	PortPinDirection	PortPinInitialMode
ADC Pin	10	PORT_PIN_IN	PORT_PIN_MODE_ADC
DIO_Enable	11	PORT_PIN_OUT	PORT_PIN_MODE_DIO
DIO_IN1	12	PORT_PIN_OUT	PORT_PIN_MODE_DIO
DIO_IN2	13	PORT_PIN_OUT	PORT_PIN_MODE_DIO

Cooling Fan Use-Case (Autosar Software)



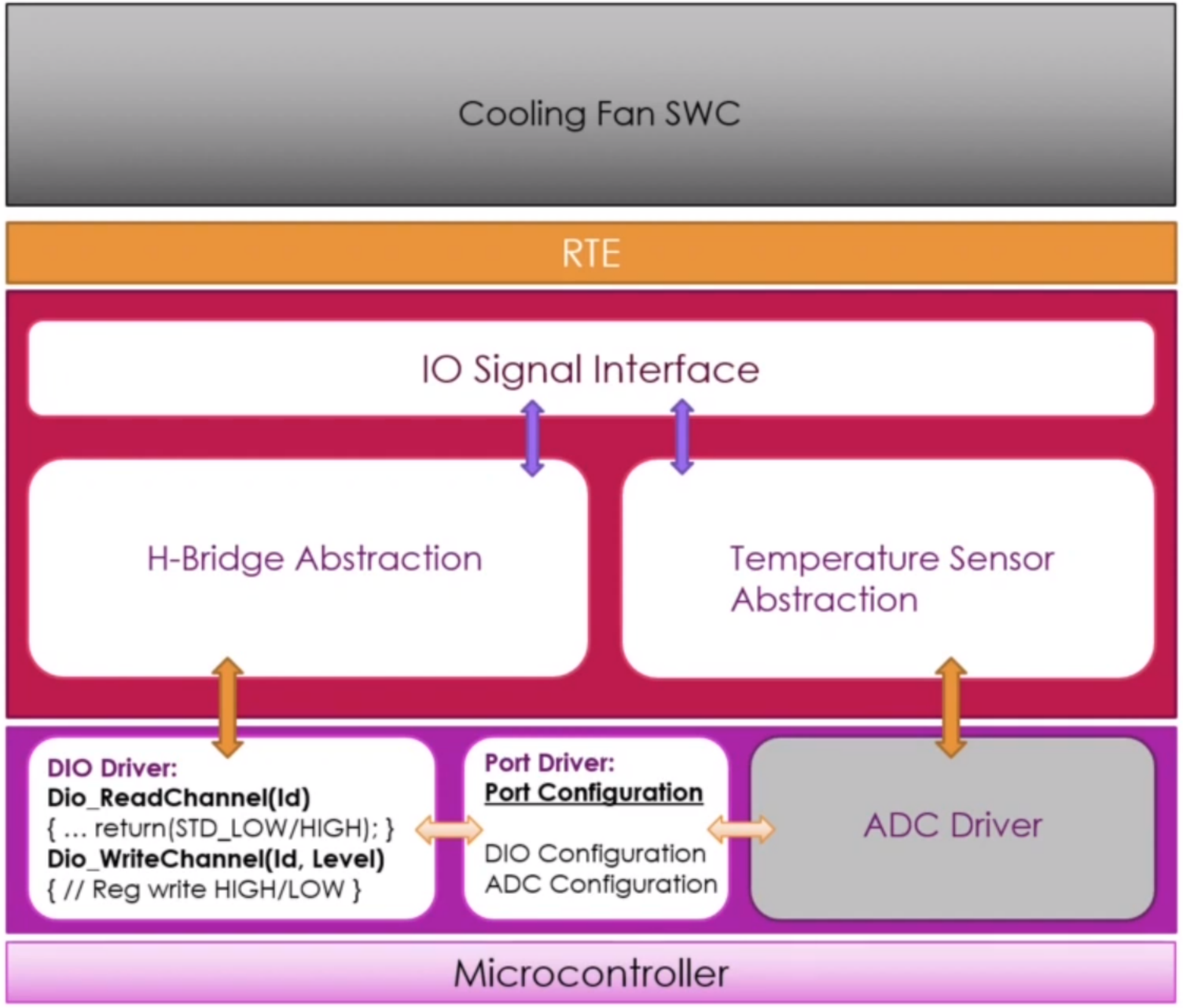
[SWS_Dio_00133] [

Service name:	Dio_ReadChannel	
Syntax:	`Dio_LevelType Dio_ReadChannel( Dio_ChannelType ChannelId )`	
Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	ChannelId	ID of DIO channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Dio_LevelType	STD_HIGH The physical level of the corresponding Pin is STD_HIGH STD_LOW The physical level of the corresponding Pin is STD_LOW
Description:	Returns the value of the specified DIO channel.	
Available via:	`Dio.h`	

[SWS_Dio_00134] [

Service name:	Dio_WriteChannel	
Syntax:	`void Dio_WriteChannel( Dio_ChannelType ChannelId, Dio_LevelType Level )`	
Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	ChannelId	ID of DIO channel
	Level	Value to be written
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Service to set a level of a channel.	
Available via:	`Dio.h`	

Cooling Fan Use-Case (Autosar Software)



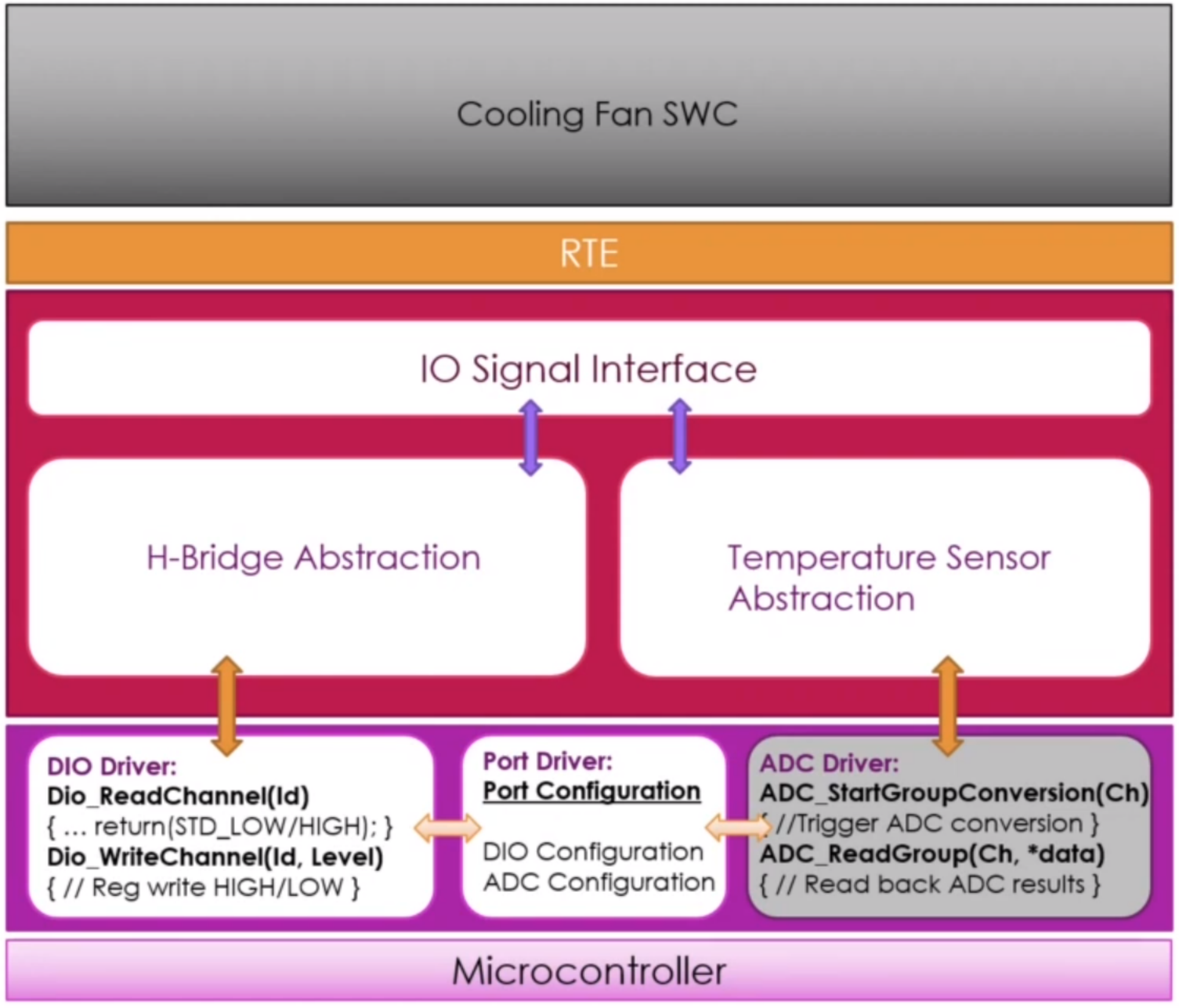
[SWS_Adc_00367] [

Service name:	Adc_StartGroupConversion	
Syntax:	`void Adc_StartGroupConversion( Adc_GroupType Group )`	
Service ID[hex]:	0x02	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	Group	Numeric ID of requested ADC Channel group.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Starts the conversion of all channels of the requested ADC Channel group.	
Available via:	`Adc.h`	

[SWS_Adc_00369] [

Service name:	Adc_ReadGroup	
Syntax:	`Std_ReturnType Adc_ReadGroup( Adc_GroupType Group, Adc_ValueGroupType* DataBufferPtr )`	
Service ID[hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	Group	Numeric ID of requested ADC channel group.
Parameters (inout):	None	
Parameters (out):	DataBufferPtr	ADC results of all channels of the selected group are stored in the data buffer addressed with the pointer.
Return value:	Std_ReturnType	E_OK: results are available and written to the data buffer E_NOT_OK: no results are available or development error occured
Description:	Reads the group conversion result of the last completed conversion round of the requested group and stores the channel values starting at the DataBufferPtr address. The group channel values are stored in ascending channel number order (in contrast to the storage layout of the result buffer if streaming access is configured).	
Available via:	`Adc.h`	

Cooling Fan Use-Case (Autosar Software)



[SWS_Adc_00367] [

Service name:	Adc_StartGroupConversion	
Syntax:	`void Adc_StartGroupConversion( Adc_GroupType Group )`	
Service ID[hex]:	0x02	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	Group	Numeric ID of requested ADC Channel group.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Starts the conversion of all channels of the requested ADC Channel group.	
Available via:	`Adc.h`	

[SWS_Adc_00369] [

Service name:	Adc_ReadGroup	
Syntax:	`Std_ReturnType Adc_ReadGroup( Adc_GroupType Group, Adc_ValueGroupType* DataBufferPtr )`	
Service ID[hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	Group	Numeric ID of requested ADC channel group.
Parameters (inout):	None	
Parameters (out):	DataBufferPtr	ADC results of all channels of the selected group are stored in the data buffer addressed with the pointer.
Return value:	Std_ReturnType	E_OK: results are available and written to the data buffer E_NOT_OK: no results are available or development error occured
Description:	Reads the group conversion result of the last completed conversion round of the requested group and stores the channel values starting at the DataBufferPtr address. The group channel values are stored in ascending channel number order (in contrast to the storage layout of the result buffer if streaming access is configured).	
Available via:	`Adc.h`	