Task: Implement the DIO module for AVR ATMEGA32 target

Description: In this task, you will be implementing the DIO (digital input/output) module for the AVR ATMEGA32 microcontroller. The DIO module is responsible for controlling the digital pins on the microcontroller, which can be used for a variety of purposes such as controlling LEDs, reading switches, and interfacing with other digital devices.

Requirements:

Implement the following interfaces for the DIO module:

```
void DIO_vSetPinDirection(uint8 Copy_u8PORT,uint8Copy_u8PinNumber,uint8
copy_u8state);
void DIO_vWritePin(uint8Copy_u8PORT,uint8Copy_u8PinNumber
,uint8Copy_u8value);
void DIO_vTogglePin(uint8 Copy_u8PORT,uint8 Copy_u8PinNumber);
void DIO_vSetPortDirection(uint8 Copy_u8PORT,uint8 copy_u8state);
void DIO_vWritePort(uint8 Copy_u8PORT,uint8 Copy_u8value);
uint8 DIO_u8GetPinValue(uint8 Copy_u8PORT,uint8 Copy_u8PinNumber);
void DIO_vTogglrPort(uint8 Copy_u8PORT);
void DIO_vWritePortValue(uint8 Copy_u8PORT,uint8 Copy_u8value);
```

Use the following files provided in the task:
**MCAL DIO DRIVER:**
a. **DIO_REG.h**: This header file defines the registers of the DIO Module for the AVR Microcontroller.
The definition should follow the following implementation:
#define PORTA_BASE
#define PORTB_BASE
#define PORTC_BASE
#define PORTD_BASE
Where they reference the base address for the Dio Registers and the reference should be in a form of pointer to structure.
The structure shall be defined as:
typedef struct
{
uint8 Pin;
uint8 DDR;
uint8 Port;
}DIO_Regs;

b. **DIO_Interface.h**: Header File that Contains the external interfaces that will be used by other modules, also should include any Macros that shall be used by the user.
c. **DIO.c**: this file should contain the Implementation of the DIO APIs which achieve their functionalities.

**LIB FOLDER ----->** this is a helper Library that will be used by different drivers and modules, Shall contain the following files.

a. Datatypes.h: Header file that contains all the typedefs for the different data types that will be used.
b. Calcbit.h: Header file that contains bit manipulation objects like macros and they are:
   - `#define setbit(reg,bin_no)`
   - `#define clearbit(reg,bin_no)`
   - `#define getbit(reg,bit_no)`
   - `#define togglebit(reg,bit_no)`

Use the following guidelines when implementing the DIO module:
a. Use the AVR ATMEGA32 datasheet and avoid magic numbers in the implementation by using Macros.
b. Use bitwise operations to manipulate the individual bits in the registers.

Deliverables:

1. A complete implementation of the DIO module that meets the requirements outlined above.
2. A brief report detailing your implementation and any challenges or interesting aspects you encountered during the project.

2. Implement **Debouncing Handler** in the HAL Layer: Debouncing Handler shall be used to trust the input level of the data, Debounce handler shall read four High levels in raw to decide the Input level is HIGH otherwise the oInput shall be LOW.
uint8 **Debounce_Check**(uint8 Input_Signal);