Task Description: Implementing an External Interrupt Driver on Atmega32

Background

The Atmega32 microcontroller has several external interrupts that allow it to respond to external events, such as a button press, sensor input, or other signals. In this task, you will be implementing an external interrupt driver for one of these external interrupts on the Atmega32 microcontroller.

Task Requirements

To complete this task, you will need to implement an external interrupt driver for the external interrupts available on the Atmega32 microcontroller. Specifically, your driver will contain three files External_INT_Interface.h, External_INT_Reg.h and External_INT.c → Note the following should be included in the External_INT_Interface.h →

```
/************* Modes ***********************/

// Modes that will be used for the external interrupt configuration

#define LOW_LEVEL_MODE  0

#define FALLING_EDGE_MODE  1

#define RISING_EDGE_MODE   2

#define FALLING_AND_RISING_EDGE_MODE  3

/********** Enable Ext Interrupts ***********/

#define ENABLE_INT0 6

#define ENABLE_INT1 7

#define ENABLE_INT2 5

#define DISABLE_INT 0
```

```
/*******************************************/

#define Pin_Int0 2

#define Pin_Int1 3

#define Pin_Int2 2

#define GIE 7

//Struct that contains the external interrupt configuration options.

typedef struct

{

uint8 Enable_INT0_Interrupt;

uint8 Enable_INT1_Interrupt;

uint8 Enable_INT2_Interrupt;

uint8 INT0_Triggering_Mode;

uint8 INT1_Triggering_Mode;

uint8 INT2_Triggering_Mode;

}EXT_Int_Conf;

/*********** External_Interrupt APIS ***************/

void EXT_INT_Init(void);

void EXT_INT_SET_CONFIG(void);

/* Note that interrupt_callback_t is a pointer to function that should
defined by the developer this pointer to function take void and returns
void */

void external_interrupt_register_callback interrupt_callback_t
callback)
```

1. Configure the external interrupt pin(s) and associated interrupt control registers on the Atmega32 to enable interrupt detection on the external interrupt pin(s).
2. Set up the interrupt service routine (ISR) for the external interrupt. When the external interrupt is triggered, the ISR should be called and execute the necessary code to handle the interrupt event.
3. Provide an interface for the user to register a callback function to be called when the external interrupt is triggered. This callback function should be called from within the ISR to allow the user to handle the interrupt event.--> Please Register in the application callback function called External_Interrupt_Notification() and use it to blink any LED on the board for 1 second.

Deliverables

Your final submission should include:

1. The source code for your external interrupt driver implementation, including the interrupt service routine and any necessary configuration code.
2. A demonstration of your driver in action, showing how the user can register a callback function and respond to an external interrupt event.

Evaluation Criteria

Your submission will be evaluated based on the following criteria:

1. Correctness and functionality of your external interrupt driver implementation.
2. Clarity and readability of your source code, including comments and variable names.
3. Demonstrated understanding of the Atmega32 microcontroller and external interrupts.
4. Quality of your demonstration and ability to answer questions about your implementation.

Resources

To complete this task, you may find the following resources useful:

- The Atmega32 datasheet and other relevant Atmel documentation
- Online forums and tutorials on implementing external interrupts on the Atmega32
- Any additional hardware or tools needed to interface with the Atmega32 microcontroller

Good luck with your implementation!