External Interrupt Driver for ATmega32 - Explanation

1. Introduction

This document explains the implementation of the External Interrupt (EXI) driver for the ATmega32 microcontroller. It includes the source code along with a detailed explanation of how the external interrupt system works in ATmega32.

2. EXI_Enable Function

The `EXI_Enable` function enables the specified external interrupt (INT0, INT1, INT2) and configures the interrupt triggering condition.

- `GICR` is the General Interrupt Control Register. It is used to enable INT0, INT1, and INT2.
- `MCUCR` controls the interrupt sense control for INT0 and INT1 (rising edge, falling edge, etc.).
- `MCUCSR` configures INT2.
- `sei()` enables global interrupts by setting the I-bit in `SREG`.

The function uses `WRITE_BIT` macro to write each bit individually based on the selected edge source.

3. EXI_Disable Function

The `EXI_Disable` function disables the specified external interrupt by clearing its corresponding bit in the `GICR` register.

4. EXI_SetCallBack and ISR Functions

The `EXI_SetCallBack` function assigns a user-defined function to be called when a specific interrupt is triggered.

- `ApfuncEXTINT[]` is a static array of function pointers.
- The ISR functions call the appropriate function from this array based on which interrupt occurred.

5. Header Files Overview

The `EXI_interface.h` file defines public macros and functions for enabling/disabling interrupts and setting

callbacks.

The `EXI_private.h` file contains private macros including register bit positions and vector definitions, and defines the `ISR()` macro which is used to define an interrupt service routine.

6. Conclusion

This EXI driver provides a modular way to configure and manage external interrupts in the ATmega32. By using callbacks and separating interface/private headers, the code is reusable and scalable for larger applications.