Requirements

Microcontroller
Use an Atmega32 microcontroller.
Use the 8 MHz internal oscillator as the clock source.

LCD

Use a 2x16 character LCD module with a Hitachi HD44780 controller.
Connect the module to the microcontroller's PORTB pins 0-3 and 6-7 for data transmission, and PORTC pins 0-2 for control signals.
Use PORTC pin 0 for RS (register select), pin 1 for E (enable), and pin 2 for RW (read/write).
Use a 4-bit data transfer mode to reduce the number of pins required to interface with the LCD.

Keypad

Use a 4x4 matrix keypad with 8 output pins and 4 input pins.
Connect the output pins to PORTD pins 0-7.
Connect the input pins to PORTC pins 0-3.
Use pull-up resistors to avoid floating inputs.
Implement software debouncing using a debounce delay time of 20 ms.
External Interrupt
Use an external interrupt to trigger an action when a switch is pressed.
Connect the switch to INT0 pin (PD2) with a pull-up resistor.
Implement software debouncing using a debounce delay time of 20 ms.

Keypad Debouncing

When a key is pressed, read the keypad output pins and check for a stable input state.
If the input is stable, debounce the input by waiting for a specified delay time (e.g. 20 ms).
After the delay time, read the keypad output pins again to ensure that the input is still stable.
If the input is still stable, register the key press and continue with the normal operation.
If the input is not stable, wait for another delay time before checking the input again.

uint8_t keypad_debounce(uint16_t delay_time)
The API should take a delay time as a parameter, which specifies the amount of time to wait for debounce.
The API should wait for the specified delay time using a timer or delay function.
After the delay time has elapsed, the API should read the keypad input again to ensure that the input is still stable.
If the input is stable, the API should return the pressed key.
If the input is not stable, the API should wait for another delay time before checking the input again.
The debounce delay time may need to be adjusted depending on the characteristics of the keypad used.
The debounce algorithm should be implemented in a way that does not block other tasks or interrupts in the system.

The debounce algorithm should be tested thoroughly to ensure that it works reliably under various conditions and inputs.

LED(s)

Use one or more LEDs to indicate various states or events in the system.
Connect the LED(s) to PORTA pins 0-7.
Use a current limiting resistor for each LED.

Calculator App APIS:

Implement the following APIs in the application layer in the calculator.c file and use them in the main to achieve the application requirements.

add(a, b) - Adds two numbers and returns the result.
subtract(a, b) - Subtracts two numbers and returns the result.
multiply(a, b) - Multiplies two numbers and returns the result.
divide(a, b) - Divides two numbers and returns the result.
modulus(a, b) - Calculates the modulus of two numbers and returns the result.
is_digit(c) - Checks if a character is a valid digit (0-9).
is_operator(c) - Checks if a character is a valid operator (+, -, *, /).
is_enter_key(key) - Checks if a key is the enter key (e.g. '#', '*')

Programming Environment

Use Atmel Studio or other suitable programming environment to write, compile, and upload the code to the microcontroller.
Write the code in C language.
Use appropriate libraries and header files to interface with the LCD, keypad, and external interrupt.
Write code to handle the debouncing of the keypad and external interrupt signals.
Write code to handle the input from the keypad and external switch and display the results on the LCD.
Write code to control the LED(s) based on the system state and events.

Practical Usage

The application could be used as a simple calculator that performs basic arithmetic operations such as addition, subtraction, multiplication, and division.
The keypad could be used to input the numbers and operators, while the external interrupt could be used to trigger the calculation when the equals sign is pressed.
The LED(s) could be used to indicate the current operation or error states, such as division by zero.

Deliverables

A schematic diagram of the hardware setup, including the connections between the microcontroller, LCD, keypad, external switch, and LED(s).

A well-commented C code that implements the application requirements and can be compiled and uploaded to the microcontroller using Atmel Studio or other suitable programming environment.
A report that explains the design choices, implementation details, testing procedures, and results of the project. The report should also include recommendations for future improvements and extensions of the project.
Constraints

The code should be optimized for memory usage and execution time.
The total current drawn from the microcontroller's pins should not exceed their maximum ratings.
The software debouncing delay should be long enough to filter out most of the bouncing, but not too long to affect the responsiveness of the system.
The hardware setup should be robust and avoid short circuits, voltage spikes, or other potential hazards.
The project should adhere to ethical and professional standards of conduct, including honesty, integrity, respect, and accountability.