

Certainly! Let's break down the provided code and explain it in more detail:

```
```c
#include <reg52.h>
```
```

This line includes the necessary header file for the 8051 microcontroller family, which provides definitions for the specific registers and bit manipulation macros.

```
```c
sbit led_1 = P2^0;
sbit button = P3^1;
```
```

These lines declare variables `led_1` and `button` as single-bit variables (`sbit`). `led_1` represents the LED connected to pin P2.0, and `button` represents the button connected to pin P3.1.

```
```c
void delay(unsigned int milliseconds) {
 unsigned int i, j;
 for(i = 0; i < milliseconds; i++)
 for(j = 0; j < 125; j++); // Delay approximation for 1 ms
}
```
```

This function `delay()` creates a delay in milliseconds. It uses nested loops to approximate the delay. The delay is not very precise but is often sufficient for simple applications like debouncing.

```
```c
void main(void) {
```
```

This is the main function where the program execution begins.

```
```c
 while(1) {
```
```

This is an infinite loop that continuously polls the button state and controls the LED accordingly.

```
```c
 if(button == 0) {
```
```

This condition checks if the button is pressed. Since the STC microcontroller reads the button as 0 when it's pressed due to pull-up resistors, this condition checks for a low (0) state.

```
```\n    delay(20); // Debounce delay\n```\n
```

This line adds a debounce delay of 20 milliseconds to filter out any bouncing when the button is pressed.

```
```\n    if(button == 1) {\n```\n
```

This condition checks again if the button is unpressed after the debounce delay.

```
```\n        led_1 = !led_1; // Toggle LED state\n```\n
```

This line toggles the state of the LED. If the LED is currently on, it turns it off, and vice versa.

```
```\n        while(button == 1); // Wait for button release\n```\n
```

This loop waits until the button is released. It ensures that the LED state is toggled only once per button press.

```
```\n        delay(20); // Debounce delay for button release\n```\n
```

This line adds a debounce delay after the button is released to prevent detecting multiple button releases due to bouncing.

```
```\n    }\n}\n}\n```\n
```

This closing brace marks the end of the `while(1)` loop and the end of the `main()` function. The program will continuously loop here, polling the button state and controlling the LED accordingly.