Agenda

- Bit Banding
-

Bit Banding

- One bit in bitband region is mapped to one word in bitband alias region.
- SRAM Bit Banding
    - bitband region 0x20000000 mapped to alias region 0x22000000
- Peripheral Bit Banding
    - bitband region 0x40000000 mapped to alias region 0x42000000
- Alias word address = bb_alias_base + (regr-bb_base)<<5 + bit<<2

LPC1768 Interrupt Handling

- CM3 architecture have single table for exceptions as well as interrupts.
    - slot 1 to 15 -- exceptions
    - slot 16 onwards -- peripheral interrupts
- Steps for handing interrupt
    - step1: Implement ISR for the interrupt and place it in appropriate slot in vector table.
    - step2: enable interrupt in NVIC.
    - step3: enable interrupt in peripheral registers.

External Interrupt Programming

External Interrupt registers

- EXTINT: Flag Register
    - When interrupt arrives from EINT pin, corresponding bit in this regr is set.
    - The interrupt flag bit in this register is cleared by writing 1 to it.
    - Once ISR begin execution, it should clear the interrupt flag.
    - Once interrupt mode is changed (in EXTMODE), the interrupt flag must be cleared.
- EXTMODE: edge-sensitive or level-sensitive.
    - 0: level sensitive
    - 1: edge sensitive

- EXTPOLAR: polariy (rising edge/falling edge) or (low level/high level)
    - For level sensitive
        - 0: low level
        - 1: high level
    - For edge sensitive
        - 0: falling edge
        - 1: rising edge
- In LPC1768, most of the pins have 4 functions. Appropriate fn should be activated using PINSEL register.
    - PINSEL4[25:24] = 01 --> P2.12 as EINT2

External Interrupt cct

- When switch is pressed
    - Low level
    - Falling edge
- When switch is released
    - High level
    - Rising edge
- Switch --> ~~P3.13~~ --> P2.12 (EINT2)

Programming

- eint2_init();
    - Enable EINT2 for P2.12.
        - PINSEL4 &= ~(BV(25) | BV(24));
        - PINSEL4 |= BV(24);
    - Enable EINT2 interrupt.
        - NVIC_EnableIRQ(EINT2_IRQn);
    - Configure EINT2 as Rising edge.
        - EXTMODE |= BV(2);
        - EXTPOLAR |= BV(2);
        - EXTINT |= BV(2);
- EINT2_IRQHandler();
    - Beep the buzzer.
        - buzzer_beep(2000);
    - Clear the interrupt.

        - EXTINT |= BV(2);

volatile keyword

- while(eint2_flag == 0); (-O3)
    - LDR r7, =eint2_flag
    - LDR r0, [r7]
    - check:
    - CMP r0, #0
    - BREQ check
- while(eint2_flag == 0); (-O0)
    - LDR r7, =eint2_flag
    - check:
    - LDR r0, [r7]
    - CMP r0, #0
    - BREQ check
- eint2_flag = 1; (-Ox)
    - MOV r1, #1
    - LDR r8, =eint2_flag
    - STR [r8], r1
- while(eint2_flag == 0); (-Ox and eint2_flag as volatile)
    - LDR r7, =eint2_flag
    - check:
    - LDR r0, [r7]
    - CMP r0, #0
    - BREQ check
- volatile keyword disable the optimization i.e. it force compiler to fetch the variable each time from the RAM (not to use its copy in the register).
- volatile implies that variable can suddenly change (out of current execution context).
    - The variable may be modified in ISR.
    - The variable may be modified in another thread of execution.
    - The variable may be modified due to memory mapped IO device.
- volatile int i = 1;
    - Do not optimize "i", because it may be modified outside current execution context.
- const int i = 1;
    - const tells compiler that the variable is not allowed to be modified.
    - Compiler doesn't allow using any operator on that variable, which may modify value of that variable. For example:
        - i++;

        - ++i;
        - i+=3;
        - i=1;

- volatile const int i = 1;
    - Due to const, compiler doesn't allow using any operator on that variable, which may modify value of that variable.
    - Due to volatile, compiler won't optimize access to the variable.
    - const volatile ensure variable cannot be modified in the code, but it can be modified by the memory mapped IO device.

- static int i = 1;
    - static is same as global, except limited scope.
    - static variable is accessible in the function/file in which it is declared.

- static void func(void);
    - static function is accessible in the file in which it is declared.

- static volatile const int i = 1;
    - static variable is accessible in the function/file in which it is declared.
    - const volatile ensure variable cannot be modified in the code, but it can be modified by the memory mapped IO device.