

① Polling: ask the device if it has new data & is ready to receive new data

② Interrupt: hardware signal from device that causes processor to stop current program & execute a special program to "service" device

Lab 5 Part 2

- write a program to:

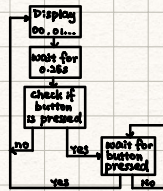
① Count from 00, 01, ... 99 & display on the hex displays

② Make a delay of 22.25 s in between each number

- use program loop to make delay

③ if key is pressed & released → stop counting

i.e: 00 → wait → 01 → ... → 99

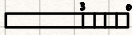


Problem: if a button is pressed & released during 0.38s, we will miss it! Looking at key data register which shows if button is up or down.

- can solve programmatically by looking at button in wait loop (complicated)


- instead, use the edge capture register. Recall the 4 memory-mapped registers for the key "parallel port"

Address

FFA0 0000  Data reg: if button i is pushed, bit i = 1 (i = 0...3)

54 Direction

58 Mask

5C  Edge capture reg: bits 3→0 start as 0000

- when a button is pressed & released, the corresponding bit in edge capture reg → 1, and it stays that way until your program

Resets it. To reset it, you must write/store a 1 into that bit.

Example program that uses this: every time key 2 is pressed & released → turn LED 0 on/off (toggle LED 0)

```
_start: LDR R0, #0xFFA0 0000 // Base address of LED
        MOV R2, #1 // keep the state that the LED will next be (After press & release)
```

// Polling loop: has human pushed & released button, will wait till button is pressed & released

// Using edge capture register

```
Poll: LDR R1, CRO, #0x5C1 // read edge capture reg into R1
        ANDS R1, #0b0100 // isolate bit 2: if bit 2 = 1, R1 = 0, if bit 2 = 0, R1 = 0
        BEQ Poll // Button is pressed & released
```