Embedded System Software 과제 3

(과제 수행 결과 보고서)

과목명: [CSE4116] 임베디드시스템소프트웨어

담당교수: 서강대학교 컴퓨터공학과 박 성 용

학번 및 이름: 120230200, 김지섭

개발기간: 2023. 05. 20. - 2023. 05. 30.

최 종 보 고 서

I. 개발 목표

Module programming, 디바이스 드라이버 구현,interrupt 등, 배운 내용을 활용하여 간단한 stopwatch 프로그 램을 작성한다.

II. 개발 범위 및 내용

- 자신들이 설계한 개발 목표를 달성하기 위하여 어떠한 내용의 개발을 수행할 지 그 범위와 개발 내용을 기술할 것.

가. 개발 범위

Timer Device Driver 및 Test 용 Example Application 개발

1. Example Application

   실행 예시

   ```
   ./app stopwatch
   ```

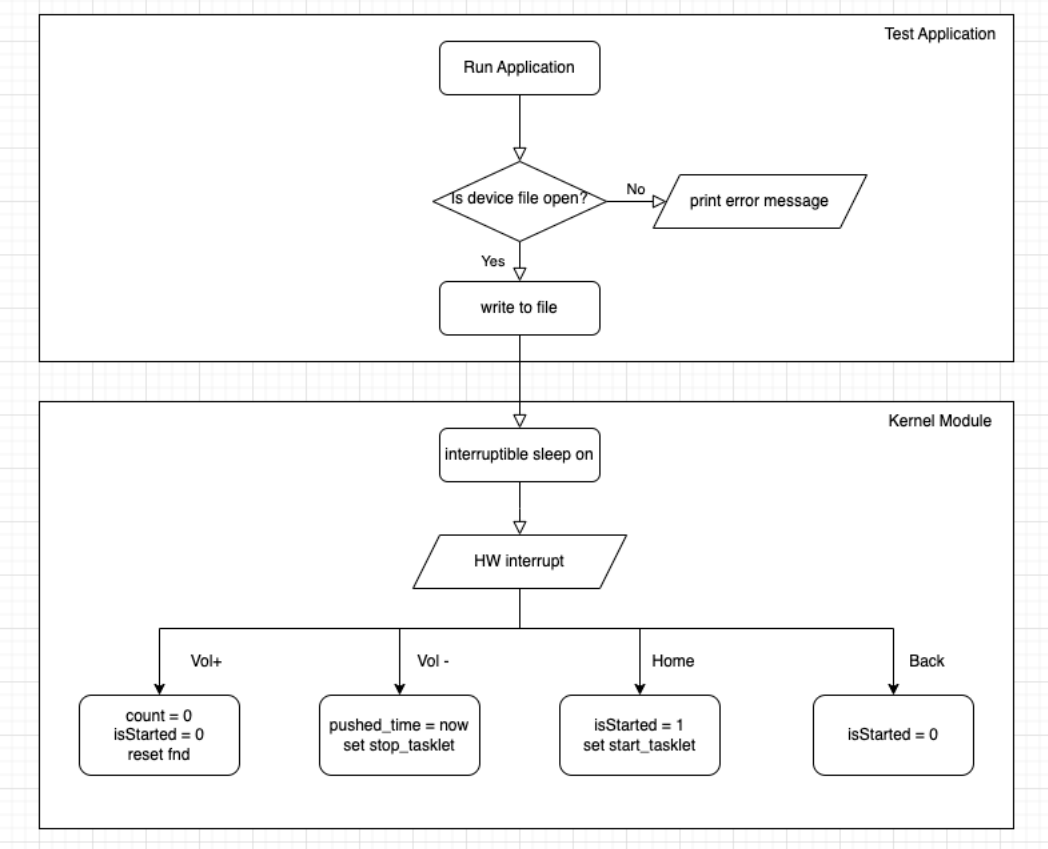
2. Stopwatch module

   - fpga_fnd 에 현재 측정된 시간을 출력한다.
     * fnd 부분의 앞의 두 자리는 분(60 분),뒤의 두 자리는 초(60 초)를 표시한다.
     * Reset, Stop 시, fpga_led 의 불을 꺼준다.(0000)
     * 키 입력은 Interrupt 를 사용하여 수행한다.
   - Home 버튼 -> start
     *1 초마다 fnd 의 정보를 갱신한다.(timer 사용)
   - Back 버튼 -> pause
     * 일시 정지
   - Vol+ 버튼 -> reset
     * fnd 출력 값 및 시간이 모두 초기상태로 돌아간다
   - Vol- 버튼 -> stop
     * 3 초 이상 누르고 있을 시, application 종료
     * fnd 를 0000 으로 초기화 한다.

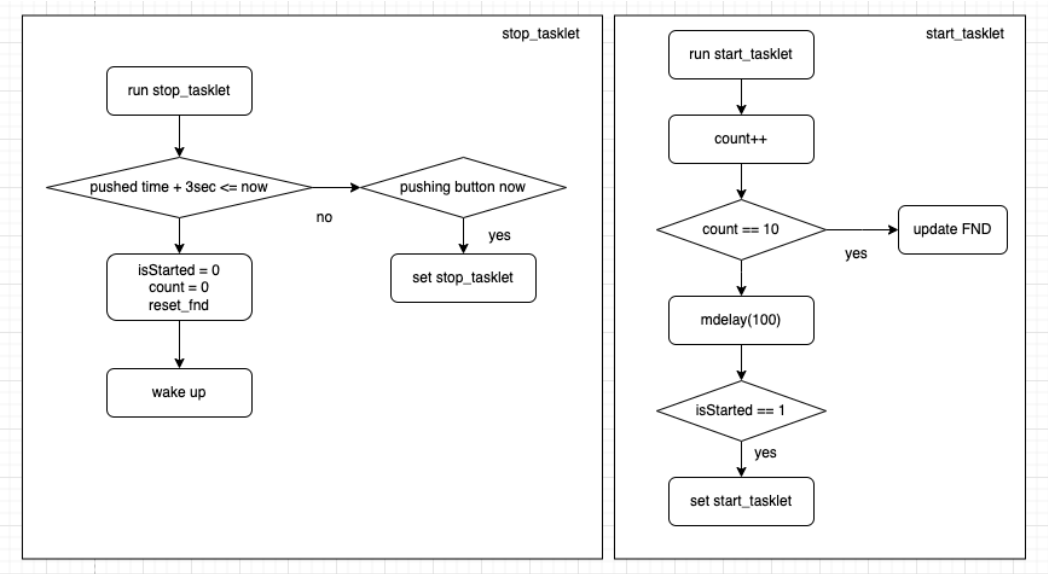
나. 개발 내용

1. Flow Chart

Application and module



Buttom half Logic



2. Example Application 구현

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
int main(void){
    int fd;
    int retn;
    char buf[2] = {0,};

    fd = open("/dev/stopwatch", O_RDWR);
    if(fd < 0) {
        perror("/dev/stopwatch");
        exit(-1);
    }
        else { printf("< inter Device has been detected > \n"); }

    retn = write(fd, buf, 2);
    close(fd);

    return 0;
}

```

Test application 코드 입니다. device file open 중 에러 발생시, error message 를 출력하고, application 을 종료합니다.

3. Stopwatch module 구현

```
void reset(unsigned long unused) {
    isStarted = 0;
    count = 0;
    reset_fnd();
}
```

```
// Vol+
irqreturn_t inter_handler3(int irq, void* dev_id,struct pt_regs* reg) {
        printk(KERN_ALERT "interrupt3!!! = %x\n", gpio_get_value(IMX_GPIO_NR(2, 15)));
        tasklet_schedule(&reset_tasklet);
        return IRQ_HANDLED;
}
```

Vol+ 버튼을 눌렀을 때 동작하는 interrupt handler 입니다.

Reset 을 수행하기 위해 pause 와 마찬가지로 isStarted 를 0 으로 수정합니다.

이때, 완전한 초기화를 위해 count 를 0 으로 초기화하고, fnd 를 reset 합니다.

```
void stop(unsigned long unused) {
    if (pushed_time + 3000000000 <= ktime_to_ns(ktime_get())) {
        isStarted = 0;
        count = 0;
        reset_fnd();
        __wake_up(&wq_write, 1, 1, NULL);
    }
    if(!gpio_get_value(IMX_GPIO_NR(5, 14))) {
        tasklet_schedule(&stop_tasklet);
    }
}
```

```
// Vol-
irqreturn_t inter_handler4(int irq, void* dev_id, struct pt_regs* reg) {
    printk(KERN_ALERT "interrupt4!!! = %x\n", gpio_get_value(IMX_GPIO_NR(5, 14)));
    pushed_time = ktime_to_ns(ktime_get());
    tasklet_schedule(&stop_tasklet);
    return IRQ_HANDLED;
}
```

Vol- 버튼을 눌렀을 때, 동작하는 interrupt handler 입니다.

3 초 이상 버튼이 눌려졌을 때, stop 을 수행해야 하고, 버튼을 누르고 있는 시간 동안 stopwatch 의 기능이 동작해야 하기 때문에, 주요 기능을 bottom half 로 구현하였습니다.

처음 interrupt 가 발생되었을 때, pushed time 을 저장하고, 지속적으로 버튼이 눌리고 있다면, tasklet 을 반복하여, 3 초 이상이 되면 값들을 초기화 하고, process 를 wakeup 시킵니다.

```
void start(unsigned long unsued) {
    count++;
    printk(KERN_ALERT "count : %d\n", count);
    if(count == 10) {
        update_fnd();
    }
    mdelay(100);
    if(isStarted) {
        tasklet_schedule(&start_tasklet);
    }
}
```

```
//Home
irqreturn_t inter_handler1(int irq, void* dev_id, struct pt_regs* reg) {
    printk(KERN_ALERT "interrupt1!!! = %x\n", gpio_get_value(IMX_GPIO_NR(1, 11)));
    isStarted = 1;
    tasklet_schedule(&start_tasklet);
    return IRQ_HANDLED;
}
```

Home 버튼을 눌렀을 때 동작하는 interrupt handler 입니다.

Stopwatch 를 실행시키기 위해 isStarted 를 1 로 수정하고, tasklet 을 set 합니다.

이후, tasklet 은 0.1 초를 대기한 후, count 를 1 씩 증가시킵니다.

isStarted 가 1 인 경우, tasklet 을 반복실행할 수 있도록 set 하고, isStarted 가 0 인 경우, tasklet 를 추가로 실행하지 않습니다.

```
void pause(unsigned long unsued) {
    isStarted = 0;
}
```

```
//BACK
irqreturn_t inter_handler2(int irq, void* dev_id, struct pt_regs* reg) {
        printk(KERN_ALERT "interrupt2!!! = %x\n", gpio_get_value(IMX_GPIO_NR(1, 12)));
        tasklet_schedule(&pause_tasklet);
        return IRQ_HANDLED;
}
```

Back 버튼을 눌렀을 때 동작하는 interrupt handler 입니다.

동작 중인 start_tasklet 을 멈추는 것이 목적이기 때문에, isStarted 를 0 으로 변경하도록 만들었습니다.

III. 추진 일정 및 개발 방법

가. 추진 일정

구분	기능	5/24		5/25		5/26	
test application	test application						
kernel module	start						
	pause						
	reset						
	stop						

나. 개발 방법

개발해야 하는 기능들을 button 단위로 구분하여 waterfall 방식으로 개발하였습니다.

우선 기존 실습 코드를 참고하여, module 을 실행시킬 수 있는 test application 을 구현하였습니다.

이후 module 을 구현하였습니다. 이때, button 별로 수행할 기능을 각각 순차적으로 구현하였습니다.

IV. 연구 결과

Interrupt 를 활용하여, process 가 sleep 상태인 경우에도 button 입력과 관련된 동작을 수행할 수 있었습니다. 특히 bottom half 로직이 수행된 이후, 다시 tasklet 을 다시 setting 하여, tasklet 을 반복 수행하도록 사용할 수 있다는 것을 알게 되었습니다. 마지막으로 wake up 을 위한 trigger 를 설정하여, process 를 깨울 수 있었습니다.

V. 기타

Interuupt 와 Time 를 활용하여, 직접 개발해보는 경험을 통해, 강의에 대한 이해도가 높아진 것 같습니다. Top half 와 Bottom half 를 적절하게 사용하기 위해 고민하면서, 좀 더 확실하게 개념을 잡을 수 있었던 것 같습니다.