Embedded System Software 과제 2

(과제 수행 결과 보고서)

과목명: [CSE4116] 임베디드시스템소프트웨어

담당교수: 서강대학교 컴퓨터공학과 박 성 용

학번 및 이름: 120230200, 김지섭

개발기간: 2023. 05. 12. - 2023. 05. 14.

최 종 보 고 서

I. 개발 목표

- 각 과제마다 주어지는 주제를 바탕으로 본 과제에서 추구하는 개발 목표를 설정하고 그 내용을 기술할 것.

Timer Device Driver 와 해당 Device driver 를 사용하는 Example Application 구현.
기존에 구현되어 있는 FPGA Device 들의 driver 를 참고하여, 해당 Driver 들을 insmod 하지 않아도 FPGA Devive 를 control 할 수 있는 Device Driver 개발.
Timer interval, Timer count, Timer init 를 입력받아, ioctl 명령어를 사용하여, init 으로 device 를 초기화하고, device 해당 주기마다 1 번씩 총 Timer Count 만큼 반복하는 Device driver 구현
해당 Device driver 의 기능을 활용할 수 있는 Test Application 개발

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
   - 사용자로부터 Argument 로 Device 구동옵션을 받는다.
   - ioctl 을 사용하여 Device driver 에 해당 데이터를 전달한다.
   - Ioctl 을 사용하여 Timer Device 를 구동시킨다.
   - 위의 과정에서 2 번의 ioctl 명령어를 사용한다.

실행 예시

```
./app TIMER_INTERVAL [1-100] TIMER_CNT[1-100] TIMER_INIT[0001-8000]
```

*TIMER_INTERVAL: HZ 값 1~100 (0.1~10 초)
*TIMER_CNT: device 출력 변경 횟수 (1~100)
*TIMER_INIT: FND 에 출력되는 초기 문양과 위치 (0001~8000)

2. Timer Device Driver

- 사용자 프로그램에서 ioctl 을 통하여 전달된 옵션을 기준으로 4 가지 device (FND, LED, DOT, TEXT_LCD) 의 요구사항에 따라 device 들에 동시 출력한다.
- 모든 device 의 값들이 바뀌는 시간은 TIMER_INTERVAL 을 기준으로 한다. Device driver 의 이름은 /dev/dev_driver 로 통일한다. (major number : 242)

A. LED

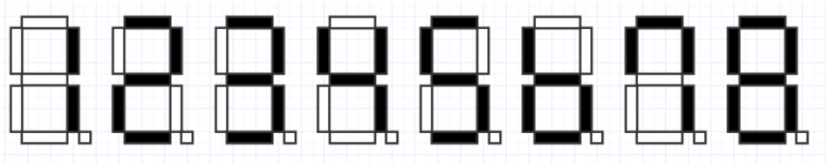
- 현재 fpga_fnd 에서 출력 중인 문양의 번호를 나타낸다.
  * D1:1 번, D2:2 번, D3:3 번, D4:4 번, D5:5 번, D6:6 번, D7:7 번, D8:8 번
- **TIMER_CNT** 횟수만큼의 출력이 끝나면 fpga_led 의 불을 꺼준다.

B. Dot

- 현재 fpga_fnd 에서 출력 중인 문양과 같은 모양의 문양을 출력한다. fpga_fnd 의 문양이 바뀐다면, fpga_dot 도 fpga_fnd 와 같은 문양으로 함께 바뀐다.
- **TIMER_CNT** 횟수만큼의 출력이 끝나면 dot 의 불을 꺼준다.

C. FND

- 초기 상태: TIMER_INIT 에서 전달된 문양과 위치에 따라 FND 에 출력한다.
- 이후 상태: TIMER_INTERVAL 마다 다음 문양를 지정된 위치에 출력한다.
- TIMER_CNT 횟수만큼의 출력이 끝나면 FND 를 0000 으로 초기화한다.
- 다음 문양이란 현재 수에서 1 증가된 수를 의미한다. 단,8 의 다음 문양은 1 이다.
- 문양이 출력되는 위치는 한번의 로테이션이 끝날때 마다 우측으로 이동한다.
- 만약 (왼쪽에서부터) 4 번째 자리에서 로테이션이 끝났다면, 1 번째 자리로 이동한다
  Ex2) 0008→0001 (**TIMER_INIT**: 0008, **TIMER_CNT**: 2)
  Ex3) 0007→0008→0001→…→0006→7000 (**TIMER_INIT**: 0007, **TIMER_CNT**: 9)

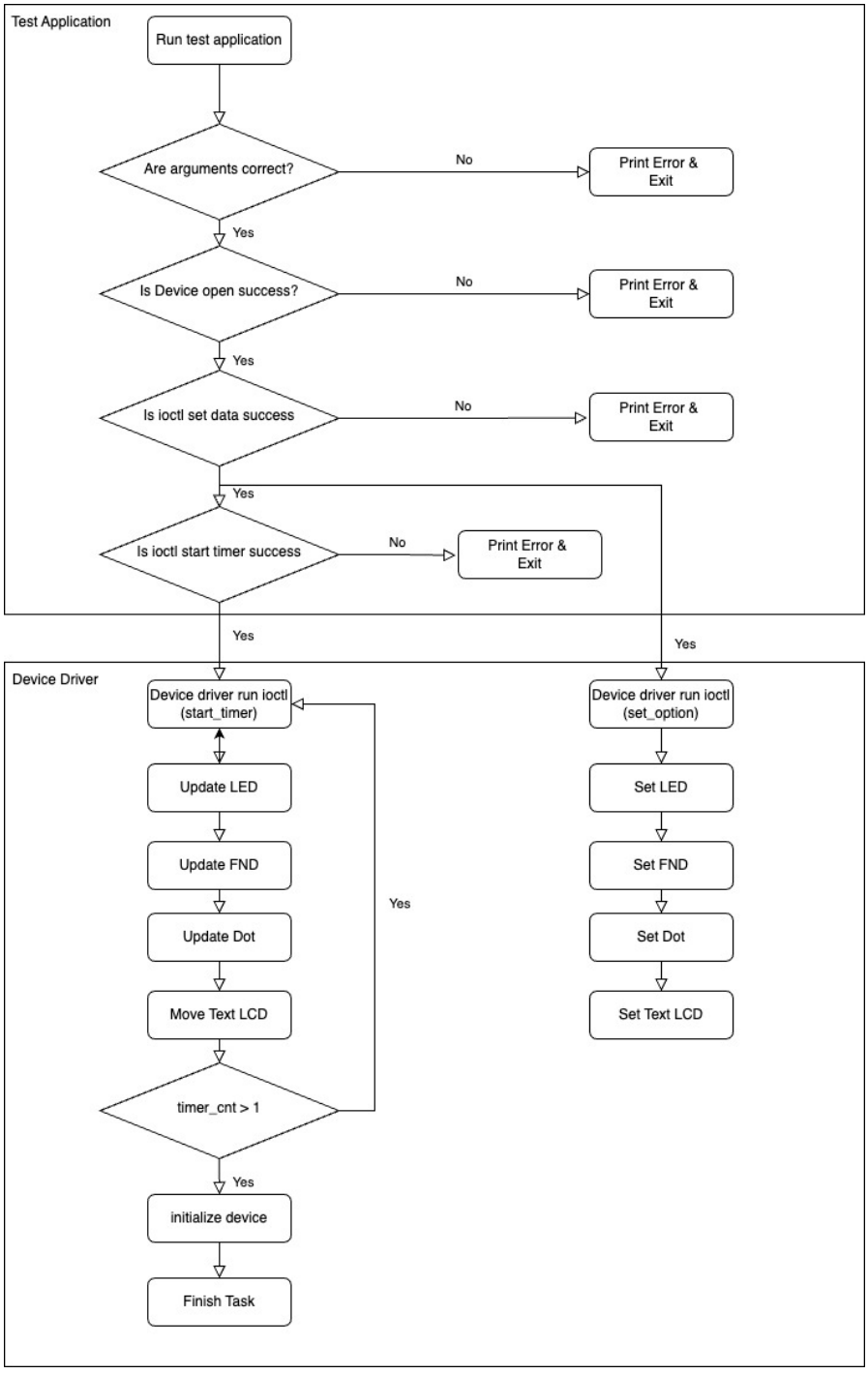


D. Text LCD

- 초기 상태: 첫 번째 줄에는 자신의 학번을, 다음줄에는 이름을 영문으로 출력한다.
- 이때 두 문장 모두 left align 시킨다. (가장 왼쪽 글자가 가장 왼쪽에 위치)
- 텍스트가 없는 나머지 칸은 “ “이 출력되도록 한다.
- 이후 상태: **TIMER_INTERVAL** 마다 오른쪽으로 한 칸씩 shift 이동을 하고, 가장 우측 글자가 가장 오른 쪽 칸에 도달한 경우, 다시 왼쪽으로 shift 이동을 시작한다. 왼쪽 shift 이동도 마찬가지로, 왼쪽 글자가 가장 왼쪽 칸에 도달한 경우, 다시 오른쪽으로 shift 이동을 시작한다. (각 줄은 다른 형태로 이동을 하게 된다.).
- **TIMER_CNT** 횟수만큼의 출력이 끝나면 LCD 의 불을 꺼준다.

나. 개발 내용

1. Flow Chart



2. Example Application 구현

Argument Error Check

```
if(argc!=4) {
    printf("please input the parameter! \n");
    printf("ex)./test_led a1\n");
    return -1;
}

data = atoi(argv[1]);
if((data < 1)||(data > 100))
{
    printf("[argv1 ERROR]Invalid range!\n");
    exit(1);
}
```

```
for(i=0;i<str_size;i++)
{
    if((argv[3][i]<0x30)||(argv[3][i])>0x38) {
        printf("[argv3 ERROR] Invalid Value!\n");
        return -1;
    }
    value.timer_char[i] = argv[3][i] - 0x30;
}
// 0이 3개 존재하는지 확인
count = 0;
for(i=0; i<str_size; i++) {
    if(value.timer_char[i] == 0) {
        count++;
    }
}

if(count != 3) {
    printf("[argv3 ERROR] Invalid Value!!\n");
    return -1;
}
```

Test application 실행 시, 입력 받는 arguments 의 값이 정해진 값과 일치하는지 확인 후, 일치하지 않는 경우 예외처리 하는 code 입니다.

Run ioctl

```
if(ioctl(dev, SET_OPTIONS, &value) < 0) {
    perror("[IOCTL ERROR] SET_OPTIONS Error\n");
    close(dev);
    return -1;
}

sleep(1);

if(ioctl(dev, START_TIMER) < 0) {
    perror("[IOCTL ERROR] START_TIMER Error\n");
    close(dev);
    return -1;
}
```

ioctl 을 사용하여 device 를 control 하는 code 입니다.

SET_OPTIONS 는 value 값을 device driver 에 넘겨, initialize 를 수행하도록 하는 option command 입니다.

START_TIMER 는 SET_OPTIONS 에서 setting 된 값을 기반으로 timer 기능을 수행하는 option command 입니다.

3. Timer Device Driver 구현

Define & mapping function

```c
// define functions...
static long dev_ioctl(struct file *file, unsigned int command, unsigned long arg);
int iom_dev_open(struct inode *minode, struct file *mfile);
int iom_dev_release(struct inode *minode, struct file *mfile);

// define file_operations structure
struct file_operations iom_dev_fops =
{
    .owner      =   THIS_MODULE,
    .open       =   iom_dev_open,
    .unlocked_ioctl      =   dev_ioctl,
    .release    =   iom_dev_release,
};
```

Device driver 에 구현된 function 을 file function 과 mapping 하기 위한 code 입니다.

Set Options

```c
static long dev_ioctl(struct file *file, unsigned int command, unsigned long arg)
{
    switch (command)
    {
    case SET_OPTIONS:
        if(copy_from_user(&value, (struct ioctl_info __user *)arg, sizeof(struct ioctl_info))) {
            return -EFAULT;
        }
        printk(KERN_INFO "Received SET_OPTIONS value1: %d, value2: %d\n", value.timer_interval, value.timer_cnt);
        set_led(value.timer_char);
        set_fnd(value.timer_char);
        set_dot(value.timer_char);
        set_text_lcd();
        break;
```

ioctl 로 SET_OPTIONS command 를 요청하였을 때, 실행 code 입니다.

Value 값을 user memory 영역에서 copy 한 후, 해당 data 를 기반으로 device 들을 setting 합니다.

Start Timer

```c
    case START_TIMER:
        printk(KERN_INFO "START TIMER\n");
        printk(KERN_INFO "Received START_TIMER value1: %d, value2: %d\n", value.timer_interval, value.timer_cnt);
        do_timer(&value);
        printk(KERN_INFO "END TIMER\n");
        break;
```

ioctl 로 START_TIMER command 를 요청하였을 때, 실행 code 입니다.

기존에 setting 되어 있는 값을 기반으로 timer 기능을 수행합니다.

Set LED

```
void write_led(unsigned char value) {
    unsigned short _s_value;
    _s_value = (unsigned short)value;
    outw(_s_value, (unsigned int)iom_fpga_dev_addr + ledAddr);
}

// when write to led device  ,call this function
void set_led(unsigned char *timer_char)
{
    unsigned char value;
    value = ledValue[find_setting_number(timer_char)];
    write_led(value);
    printk(KERN_INFO "led Set %d\n", value);
}
```

LED 를 입력받은 값으로 setting 하는 code 입니다.

Set Dot

```
void set_dot(unsigned char *timer_char)
{
    int temp_value;
    unsigned char temp;
    temp = find_setting_number(timer_char);
    temp_value = (int)temp;
    printk(KERN_INFO "temp_value : %d,", temp_value);
    write_dot(temp_value);
}
```

Dot 를 입력받은 값으로 setting 하는 code 입니다.

Update LED, Dot

```
// update led & dot
led_temp = get_led();
printk(KERN_INFO "led_temp: %d\n", led_temp);
if(led_temp == ledValue[8]) {
    write_led(ledValue[1]);
    write_dot(1);
}
else {
    for(i=1; i < 8; i++) {
        if(ledValue[i] == led_temp) {
            write_led(ledValue[++i]);
            write_dot(i);
            break;
        }
    }
}
```

본 코드에서 LED, Dot 의 set/write function 에서 사용되는 값은 아래와 같은 미리 정의된 배열 값을 활용합니다.

```
unsigned char ledValue[9] = {
    0, 128, 64, 32, 16, 8, 4, 2, 1
};
```

```
unsigned char fpga_number[10][10] = {
    {0x3e,0x7f,0x63,0x73,0x73,0x6f,0x67,0x63,0x7f,0x3e}, // 0
    {0x0c,0x1c,0x1c,0x0c,0x0c,0x0c,0x0c,0x0c,0x0c,0x1e}, // 1
    {0x7e,0x7f,0x03,0x03,0x3f,0x7e,0x60,0x60,0x7f,0x7f}, // 2
    {0xfe,0x7f,0x03,0x03,0x7f,0x7f,0x03,0x03,0x7f,0x7e}, // 3
    {0x66,0x66,0x66,0x66,0x66,0x66,0x7f,0x7f,0x06,0x06}, // 4
    {0x7f,0x7f,0x60,0x60,0x7e,0x7f,0x03,0x03,0x7f,0x7e}, // 5
    {0x60,0x60,0x60,0x60,0x7e,0x7f,0x63,0x63,0x7f,0x3e}, // 6
    {0x7f,0x7f,0x63,0x63,0x03,0x03,0x03,0x03,0x03,0x03}, // 7
    {0x3e,0x7f,0x63,0x63,0x7f,0x7f,0x63,0x63,0x7f,0x3e}, // 8
    {0x3e,0x7f,0x63,0x63,0x7f,0x3f,0x03,0x03,0x03,0x03} // 9
};
```

Set & Get FND

```c
void set_fnd(unsigned char *timer_char)
{
    unsigned short int value_short = 0;
    value_short = timer_char[0] << 12 | timer_char[1] << 8 |timer_char[2] << 4 |timer_char[3];
    outw(value_short,(unsigned int)iom_fpga_dev_addr + fndAddr);
}

void get_fnd(unsigned char* value)
{
    unsigned short int value_short = 0;

    value_short = inw((unsigned int)iom_fpga_dev_addr + fndAddr);
    value[0] =(value_short >> 12) & 0xF;
    value[1] =(value_short >> 8) & 0xF;
    value[2] =(value_short >> 4) & 0xF;
    value[3] = value_short & 0xF;
}
```

FND 에 값을 setting 하거나 FND 에서 값을 가져오는 code 입니다.

Update FND

```c
    //update fnd
    get_fnd(fnd_temp);
    fnd_data = find_setting_fnd(fnd_temp);
    for(i=0;i<4;i++){
        fnd_value[i] = 0;
    }
    if(fnd_temp[fnd_data] == 8) {
        fnd_value[fnd_data] = 1;
    }
    else {
        fnd_value[fnd_data] = fnd_temp[fnd_data]+1;
    }

    if(fnd_value[fnd_data] == value->timer_char[find_setting_fnd(value->timer_char)]) {
        temp = fnd_value[3];
        for(i=2; i>=0; i--) {
            fnd_value[i+1] = fnd_value[i];
        }
        fnd_value[0] = temp;
    }
    set_fnd(fnd_value);
```

FND 에서 값을 읽어와 update 하는 code 입니다.

처음 입력된 setting 값을 사용하여, 현재 값이 1 cycle 을 돌았다면, shift 를 수행합니다.

Set Text LCD

```c
void set_text_lcd(void)
{
    int i;
    unsigned short int _s_value = 0;
    unsigned char value[MAX_BUFF + 1];

    memset(value,0,sizeof(value));
    strncat(value, student_number, LINE_BUFF);
    strncat(value, name, LINE_BUFF);

    value[MAX_BUFF]=0;
    printk("Get Size : %d \n String : %s\n",MAX_BUFF, value);

    for(i=0; i<MAX_BUFF; i++)
    {
        _s_value = ((value[i] & 0xFF) << 8) | value[i + 1] & 0xFF;
        outw(_s_value,(unsigned int)iom_fpga_dev_addr+text_lcdAddr+i);
        i++;
    }
}
```

Text LCD 에 지정된 값을 출력하는 code 입니다.

Shift(move) Text LCD

```
    // shift
    if(st_flag == 1) {
        for(i=LINE_BUFF-2; i>=0; i--) {
            student_number[i+1] = student_number[i];
        }
        student_number[0] = ' ';
    } else {
        for(i=0; i<LINE_BUFF-1; i++) {
            student_number[i] = student_number[i+1];
        }
        student_number[LINE_BUFF-1] = ' ';
    }
```

각 line 별로 현재 이동방향의 끝이 빈칸이 아니라면, 반대방향으로 shift 하도록 Flag 를 update 하는 code 입니다.

Device off

```
    // initialize
    write_led(ledValue[0]);
    write_dot(-1);
    init_text_lcd();

    for(i = 0; i<4; i++) {
        fnd_value[i] = 0;
    }
    set_fnd(fnd_value);
    strcpy(student_number, "120230200       ");
    strcpy(name, "kimjeeseob      ");
}
```

Timer 종료 후, 초기화 code 입니다.

III. 추진 일정 및 개발 방법

- 자신들이 설정한 개발 목표를 달성하기 위한 개발 일정을 설정하고, 각 요소 문제를 해결하기 위해서 어떤 방법을 사용할 지 기술할 것.

가. 추진 일정

구분	기능	5/12	5/13	5/14
Example Application	Input data 예외처리			
	file open ioctl 사용 예제			
	예외처리			
Timer Device Driver	skeleton(LED)			
	Dot			
	FND			
	Text LCD			

나. 개발 방법

개발해야 하는 기능들을 control 해야하는 device 단위로 구분하여 waterfall 방식으로 개발하였습니다.

우선 device driver 의 skeleton code 로, led 를 ioctl 로 control 하는 driver 를 구현하였고, 이를 실행시킬 수 있는 test application prototype 을 구현하였습니다.

이후 device 별로 수행할 기능을 각각 순차적으로 구현하였습니다.

IV. 연구 결과

- 최종 연구 개발 결과를 자유롭게 기술할 것.

ioctl 을 활용하여, device driver 를 비교적 자유롭게 control 할 수 있었습니다. 단순한 read, write 기능만 존재한다면, user level 에서 구현의 복잡도가 높았을 것이라 생각합니다.
어느정도 static 한 동작을 수행해야하는 device 라면, ioctl 을 적극적으로 활용한 driver 를 구현하여, user level 개발자들의 편의를 도모할 수 있을 것 같습니다.

V. 기타

- 본 설계 프로젝트를 수행하면서 느낀 점을 요약하여 기술하라. 내용은 어떤 것이든 상관이 없으며, 본 프로젝트에 대한 문제점 제시 및 제안을 포함하여 자유롭게 기술할 것.

강의를 통해 배운 Device Driver 를 직접 개발해보는 경험을 통해, 강의에 대한 이해도가 높아진 것 같습니다. 처음엔 막막했지만, 배운 내용을 리마인드 하면서 문제를 해결할 수 있는 실마리들을 찾을 수 있었던 것 같습니다. 문제해결 능력, 개발 경험, 강의 이해 등 다양한 방면에서 도움이 많이 되는 프로젝트였던 것 같습니다.
본 프로젝트의 학번, 이름이 device driver 에 들어가 있도록 설계되어 있습니다. 과제 목적으로는 괜찮은 프로젝트일지 몰라도, device driver 의 출력 값과 형태를 좀 더 자유롭게 할 수 있도록 application level 에서 입력 받는 형태로 구현했으면 어땠을까 하는 생각이 들었습니다.