|  |  |
| --- | --- |
| 년도-학기 | 2021년 1학기 |
| 과목명 | 임베디드시스템설계 |

|  |  |
| --- | --- |
| **LAB번호** | **제목** |
| 4 | Lab4: Character LCD Driver |

|  |  |
| --- | --- |
| 실험 일자 | 2021년 3월 26일 |
| 제출자 이름 | 강\*\* |
| 제출자 학번 | 201803\*\*\*\* |
| 팀원 이름 |  |
|  |  |

**Chapter 1. 프로그램의 동작 방식 설명**

과정을 설명하기 위해 응용프로그램의 동작을 설명하고 드라이버에 대한 설명을 한다.

**- 응용프로그램 동작**

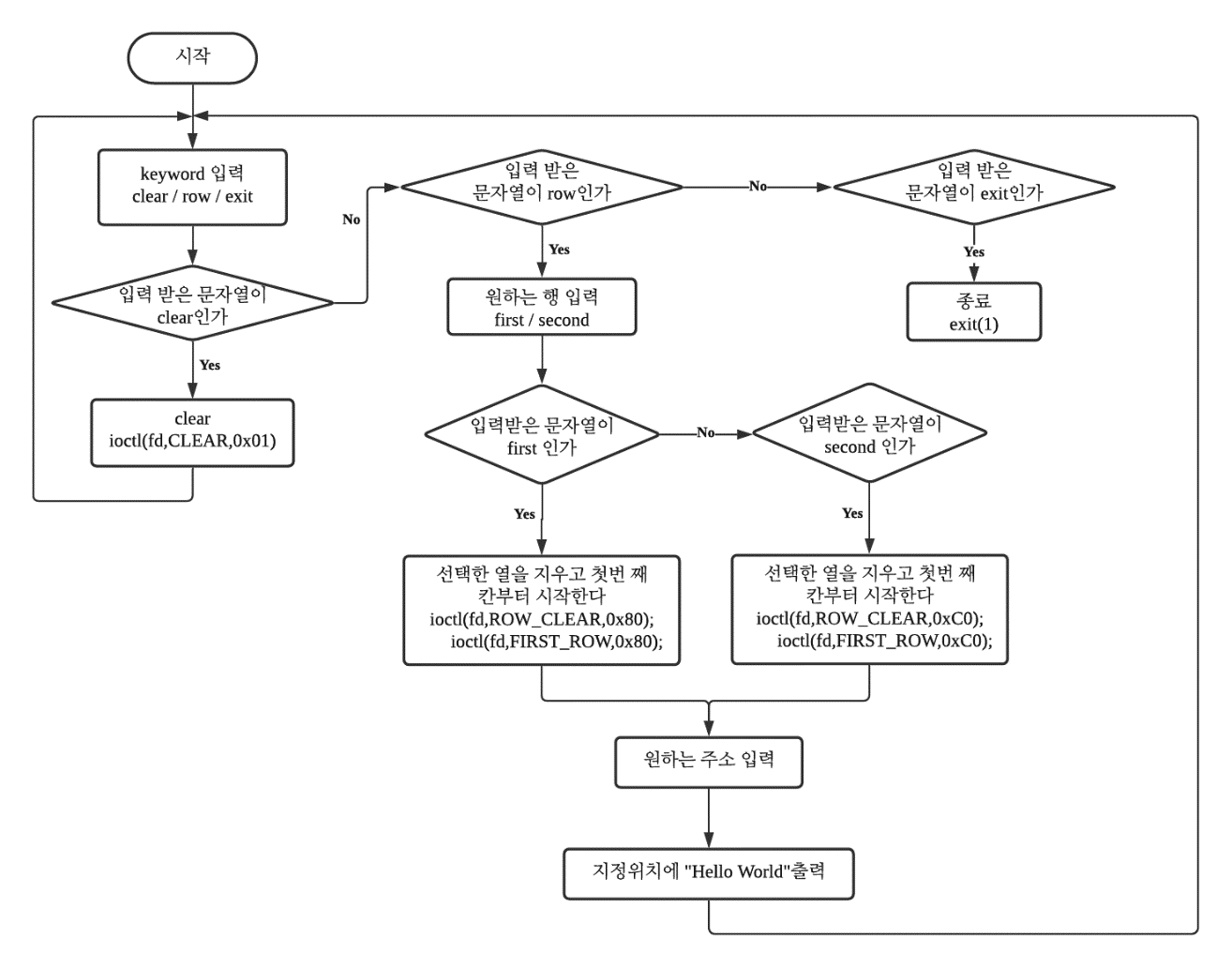
Clear 기능, 표시되는 줄 지정, 시작 주소 지정의 3가지 기능을 수행하기 위해 2가지 키워드 clear과 row를 정했다. (exit는 프로그램을 종료하기 위해 추가한 키워드로 설명에서는 제외하였다.) 처음에 키워드를 입력할 수 있도록 안내 문장을 출력한다.

키워드로 clear을 입력하면 ioctl(fd,CLEAR,0x01)를 실행해 (1)clear 기능을 수행하도록 했다.

키워드로 row를 입력하면 (2)표시되는 줄을 지정하기위한 단계로 넘어간다. first 와 second 둘 중 하나를 입력할 수 있도록 안내 문장을 프린트한다. first/second를 입력하면 첫째 줄/둘째 줄이 지워지고 가장 왼쪽부터 표시되도록 한다.

(첫번째 줄을 지울 때는 ioctl(fd,ROW\_CLEAR,0x80) 두번째 줄을 지울 때는 ioctl(fd,ROW\_CLEAR,0xC0)을 실행했다. 줄의 가장 왼쪽부터 표시되도록 할 때는 ioctl(fd,FIRST\_ROW,0x80), ioctl(fd,SECOND\_ROW,0xC0)을 실행했다.)

그 후 (3)시작 주소 지정을 위한 안내문장이 출력된다. 이 때 시작하길 원하는 위치의 주소를 입력하면 그 위치부터 표시되도록 지정한다. 시작 위치 주소를 set.add 변수로 받아 ioctl(fd,START\_ADDRESS,set.add)을 실행해 표시될 위치를 지정했다. 이후“Hello World”를 출력하는 write함수를 통해 앞서 정한 위치에 메시지가 제대로 출력되는지 확인한다.



**- 드라이버 동작**

Ioctl 함수의 두번째 인자 cmd가 무엇인지에 따라 다른 기능을 구현하도록 했다. cmd는 매크로함수를 이용해 설정하였다.

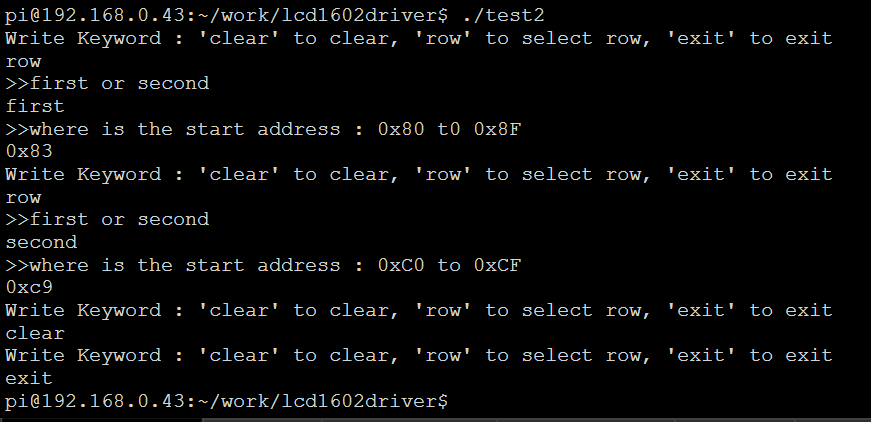
cmd가 CLEAR 일 때 LCD\_Command(0x01)을 실행해 lcd 컨트롤러의 clear 기능을 수행한다.

cmd가 FIRST\_ROW 일 때 LCD\_Command(0x80)을 실행해 시작을 첫째 줄 첫번째 칸으로 지정한다. cmd가 SECOND\_ROW 일 때 LCD\_Command(0xC0)을 실행해 시작을 둘째 줄 첫번째 칸으로 지정한다.

cmd가 START\_ADDRESS 일 때 응용프로그램 파일에서 입력 받은 주소 값을 gdata 값으로 받아 LCD\_Command(gdata)를 실행해 시작할 위치를 지정한다.

다음은 추가적으로 구현한 기능이다. 응용프로그램을 실행하면 row를 선택하게 된다. first와 second 중 선택하여 입력하면 first일 때 0x80, second일 때 0xC0를 ioctl(fd,ROW\_CLEAR, )의 세번째 인자로 넘겨 선택된 줄을 지우도록 하였다. 따라서 cmd가 ROW\_CLEAR 일 때, LCD\_Command(0x80) 혹은 LCD\_Command(0xC0)을 실행하고 LCD\_String(“ “)을 실행해 공백으로 한 줄을 지우도록 했다.

**Chapter 2. 결과**



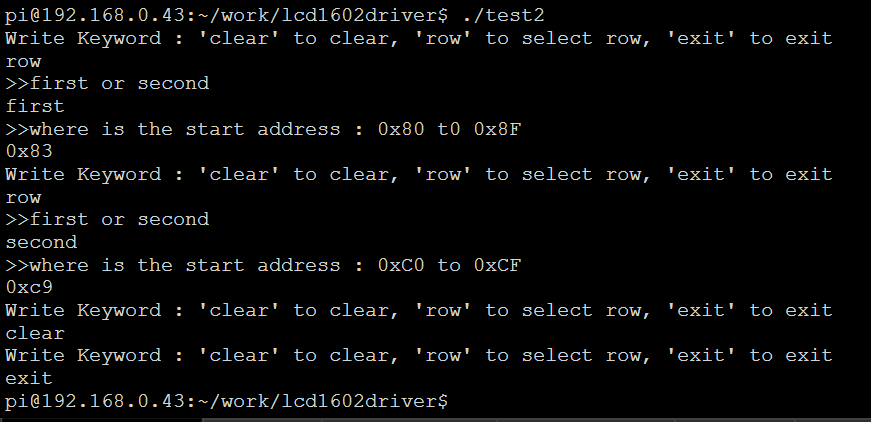
열을 선택하고 시작 위치를 지정하는 과정이다. 메시지가 지정된 위치에 입력되는 것을 확인하기 위해 원하는 주소를 입력하면 write 함수로 “Hello World”가 바로 해당 위치에 출력되도록 했다.

키워드로 row를 입력하면 first 와 second 중 하나를 선택한다. first를 선택하면 첫 번째 줄이 지워지고 첫 번째 칸(0x80)을 지정한다. 그 후 원하는 주소를 입력하면 해당 위치로 시작위치가 변경되고, write 함수로 인해 LCD에“Hello World”가 출력된다. 마찬가지로 second를 선택하면 같은 방식으로 두 번째 줄이 지워지고 그 줄의 첫 번째 칸(0xC0)을 지정한다. 그 후 시작할 위치의 주소를 입력하면, LCD에 “Hello World”가 지정한 위치에 출력된다.

텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명

0x83(첫째 줄, 4번째 칸), 0xc9(둘째 줄, 10번째 칸)부터 메시지가 출력됨을 볼 수 있다.

clear기능을 수행하는 코드이다. clear를 입력하면 lcd 전체가 지워진다.

**텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명**

디스플레이에 쓰여 있던 메세지가 사라졌음을 볼 수 있다.

**Chapter 3. 결론 및 Discussion**

응용 프로그램에서의 명령으로 Ioctl 함수에서 정해 놓은 기능을 사용할 수 있었다. 사용자와 드라이버의 역할을 분리해서 생각할 수 있었던 실습인 것 같다.

어려운 점보다는 해도 될지 고민했던 점이 있었다. Ioctl 함수만을 수정한다고 생각했는데, 과제 구현을 위해서 필요한 수정은 자유롭게 가능하였다. 그리고 헤더파일을 생성하는 것을 처음 해봐서 임의로 파일을 추가해도 되는건지 겁이 났고, 검색을 해보고 교수님께 질문도 드렸다. 그런데 응용프로그램과 lcd1602driver.c 둘 다 매크로 함수를 적었더니 헤더파일 없이도 실행할 수 있었다.

아쉬운 점은 과제를 할 때는 LCD의 칸에 대한 주소를 알고 있음을 가정하고 사용자가 입력하도록 했지만, 만약 사용자가 주소를 모른다면 현재 작성한 프로그램 상으로는 사용하기 어려워진다는 점이었다. 이럴 경우, 몇 번째 칸인지를 입력하고 조건문이나 case문 등을 통해 주소와 매칭시킬 수 있다면, 칸 번호만으로 원하는 위치에서 메시지를 출력할 수 있겠다는 생각이 들었다.

그리고 또 아쉬운 점은 (2)기능과 (3)기능을 순차적으로 수행하는 점이었다. 행을 선택하고 위치(주소) 입력을 통한 기능을 보여주기 위해서는 순차적인 진행이 좋을 것 같다고 생각했다. 하지만 처음 행을 선택하면 첫번째 칸부터 시작할 수 있는데, 그대로 첫번째 칸에서 시작하려고 해도 주소를 또 입력해야 한다는 점이 조금 아쉬웠다. 위치주소 만을 고려하거나, 처음부터 행과 시작위치를 동시에 입력 받는다면 아쉬운 점이 해결될 것 같다는 생각이 들었다.

**부록**

**응용프로그램(테스트) 소스코드**

#include <fcntl.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <string.h>

#include <sys/ioctl.h>

#define IOCTL\_MAGIC 'G'

#define CLEAR \_IOW(IOCTL\_MAGIC, 1 ,int)

#define FIRST\_ROW \_IOW(IOCTL\_MAGIC, 2 ,int)

#define SECOND\_ROW \_IOW(IOCTL\_MAGIC, 3 ,int)

#define START\_ADDRESS \_IOW(IOCTL\_MAGIC, 4 ,int)

#define ROW\_CLEAR \_IOW(IOCTL\_MAGIC, 5, int)

int main()

{

int fd;

fd=open("/dev/lcd1602",O\_RDWR);

if (fd < 0) {

printf("Device open error : %s\n","/dev/lcd1602");

exit(1);

}

char row[30];

int address;

char wbuf[30];

char keyword[30];

while(1){

printf("Write Keyword : 'clear' to clear, 'row' to select row");

printf(", 'exit' to exit \n");

scanf("%s",keyword);

if( strcmp("clear",keyword)==0 ){

ioctl(fd,CLEAR,0x01);

}

else if( strcmp("row",keyword)==0 ){

printf(">>first or second \n");

scanf("%s",row);

if( strcmp("first",row)==0 ){

ioctl(fd,ROW\_CLEAR,0x80);

ioctl(fd,FIRST\_ROW,0x80);

printf(">>where is the start address : 0x80 t0 0x8F \n");

scanf("%x",&address);

if(address > 0x8F || address < 0x80){

printf("write address 0x80 t0 0x8F \n");

exit(1);

}

else if(address > 0x7F && address < 0x90){

ioctl(fd,START\_ADDRESS,address);

strcpy(wbuf,"Hello World");

write(fd, wbuf, strlen(wbuf));

sleep(1);

}

}

else if( strcmp("second",row)==0 ){

ioctl(fd,ROW\_CLEAR,0xC0);

ioctl(fd,SECOND\_ROW,0xC0);

printf(">>where is the start address : 0xC0 to 0xCF \n");

scanf("%x",&address);

if(address < 0xC0 || address > 0xCF){

printf("write address 0xC0 t0 0xCF \n");

exit(1);

}

else if(address > 0xBF && address < 0xD0){

ioctl(fd,START\_ADDRESS,address);

strcpy(wbuf,"Hello World");

write(fd, wbuf, strlen(wbuf));

sleep(1);

}

}

}

else if( strcmp("exit",keyword)==0 ){

exit(1);

}

}

return 0;

close(fd);

}

**드라이버 소스 코드**

#include "linux/version.h"

#include <linux/module.h>

#include <linux/fs.h>

#include <asm/uaccess.h> /\* for put\_user \*/

#include <linux/ioport.h>

#include <asm/io.h>

#include <linux/delay.h>

#include <linux/string.h>

#include <linux/uaccess.h>

MODULE\_LICENSE("GPL");

#define SUCCESS 0

#define DEVICE\_NAME "lcd1602" // Dev name as it appears in /proc/devices

#define BUF\_LEN 80 // Max length of the message from the device

#define GPFSEL0 0

#define GPFSEL1 4

#define GPFSEL2 8

#define GPSET0 0x1c

#define GPCLR0 0x28

#define DATA4 18

#define DATA5 19

#define DATA6 20

#define DATA7 21

#define RS 26 /\* Define Register Select pin \*/

#define EN 16 /\* Define Enable signal pin \*/

static unsigned PORT = 0xFE200000;

static unsigned RANGE = 0xB0;

unsigned cmd\_word;

unsigned out\_word;

volatile u8 \*addr;

static int Major; /\* Major number assigned to our device driver \*/

static int Device\_Open = 0; /\* Is device open?

\* Used to prevent multiple access to device \*/

static char msg[BUF\_LEN]; /\* The msg the device will give when asked \*/

static char \*msg\_Ptr;

void LCD\_Command( unsigned char cmnd )

{

writel((u32)(cmnd>>4 & 0x0F)<<DATA4,addr+GPSET0); /\* sending upper nibble \*/

writel((u32)((~cmnd)>>4 & 0x0F)<<DATA4,addr+GPCLR0);

writel((u32)1<<RS,addr+GPCLR0); /\* RS=0, command reg. \*/

writel((u32)1<<EN,addr+GPSET0); /\* Enable pulse \*/

udelay(1);

writel((u32)1<<EN,addr+GPCLR0);

udelay(200);

writel((u32)(cmnd & 0x0F)<<DATA4,addr+GPSET0); /\* sending lower nibble \*/

writel((u32)((~cmnd) & 0x0F)<<DATA4,addr+GPCLR0);

writel((u32)1<<EN,addr+GPSET0); /\* Enable pulse \*/

udelay(1);

writel((u32)1<<EN,addr+GPCLR0);

mdelay(2);

}

void LCD\_Char( unsigned char data )

{

writel((u32)(data>>4 & 0x0F)<<DATA4,addr+GPSET0); /\* sending upper nibble \*/

writel((u32)(~data>>4 & 0x0F)<<DATA4,addr+GPCLR0);

writel((u32)1<<RS,addr+GPSET0); /\* RS=1, data reg. \*/

writel((u32)1<<EN,addr+GPSET0); /\* Enable pulse \*/

udelay(1);

writel((u32)1<<EN,addr+GPCLR0);

udelay(200);

writel((u32)(data & 0x0F)<<DATA4,addr+GPSET0); /\* sending lower nibble \*/

writel((u32)(~data & 0x0F)<<DATA4,addr+GPCLR0);

writel((u32)1<<EN,addr+GPSET0); /\* Enable pulse \*/

udelay(1);

writel((u32)1<<EN,addr+GPCLR0);

mdelay(2);

}

void LCD\_String (char \*str) /\* Send string to LCD function \*/

{

int i;

for(i=0;str[i]!=0;i++) /\* Send each char of string till the NULL \*/

{

LCD\_Char (str[i]);

}

}

void gpio\_output(u8 pin\_number)

{

u8 pin\_position;

u32 value;

pin\_position = pin\_number % 10;

switch(pin\_number/10) {

case 0:

value=readl(addr+GPFSEL0);

value |= (u32)1<<(3\*pin\_position);

writel(value,addr+GPFSEL0);

break;

case 1:

value=readl(addr+GPFSEL1);

value |= (u32)1<<(3\*pin\_position);

writel(value,addr+GPFSEL1);

break;

case 2:

value=readl(addr+GPFSEL2);

value |= (u32)1<<(3\*pin\_position);

writel(value,addr+GPFSEL2);

break;

default:

break;

}

}

void LCD\_Init (void) /\* LCD Initialize function \*/

{

gpio\_output(DATA4);

gpio\_output(DATA5);

gpio\_output(DATA6);

gpio\_output(DATA7);

gpio\_output(RS);

gpio\_output(EN);

mdelay(20); /\* LCD Power ON delay always >15ms \*/

LCD\_Command(0x02); /\* send for 4 bit initialization of LCD \*/

LCD\_Command(0x28); /\* 2 line, 5\*7 matrix in 4-bit mode \*/

LCD\_Command(0x0c); /\* Display on cursor off\*/

LCD\_Command(0x06); /\* Increment cursor (shift cursor to right)\*/

LCD\_Command(0x01); /\* Clear display screen\*/

mdelay(2);

}

static int device\_open(struct inode \*inode, struct file \*file)

{

msg\_Ptr = msg;

sprintf(msg,"Not implemented\n");

if (Device\_Open)

return -EBUSY;

Device\_Open++;

try\_module\_get(THIS\_MODULE);

return SUCCESS;

}

static int device\_release(struct inode \*inode, struct file \*file)

{

Device\_Open--; //make ready for the next caller

module\_put(THIS\_MODULE);

return 0;

}

static ssize\_t device\_read(struct file \*filp,

char \*buffer,

size\_t length,

loff\_t \* offset)

{

int bytes\_read = 1;

int index;

u32 res; // status word to read

msg\_Ptr = msg;

res = readl(addr+0x34);

msg[3] = res & 0xFF;

msg[2] = (res >> 8) & 0xFF;

msg[1] = (res >> 16) & 0xFF;

msg[0] = (res >> 24) & 0xFF;

msg[4] = 0;

index = 4;

while (length && (index >= 0))

{

put\_user(\*(msg\_Ptr++), buffer++);

length--;

index--;

bytes\_read++;

}

return bytes\_read;

}

static ssize\_t device\_write(struct file \*filp, const char \*buff, size\_t len, loff\_t \* off)

{

u8 string[30];

memcpy(string,buff,len);

string[len]=0x0;

//LCD\_Command(0x80); /\*Go to 1st line home position \*/

LCD\_String(string);

return len;

}

#define IOCTL\_MAGIC 'G'

#define CLEAR \_IOW(IOCTL\_MAGIC, 1 ,int)

#define FIRST\_ROW \_IOW(IOCTL\_MAGIC, 2 ,int)

#define SECOND\_ROW \_IOW(IOCTL\_MAGIC, 3 ,int)

#define START\_ADDRESS \_IOW(IOCTL\_MAGIC, 4 ,int)

#define ROW\_CLEAR \_IOW(IOCTL\_MAGIC, 5, int)

static long device\_ioctl(struct file \*filp, unsigned int cmd,unsigned long gdata)

{

switch(cmd){

case CLEAR:

LCD\_Command(gdata);

printk("%lx\n",gdata);

break;

case FIRST\_ROW:

LCD\_Command(gdata);

printk("%lx\n",gdata);

break;

case SECOND\_ROW:

LCD\_Command(gdata);

printk("%lx\n",gdata);

break;

case START\_ADDRESS:

LCD\_Command(gdata);

printk("%lx\n",gdata);

break;

case ROW\_CLEAR:

LCD\_Command(gdata);

LCD\_String(" ");

}

return 0;

}

static struct file\_operations fops = {

.owner = THIS\_MODULE,

.open = device\_open,

.release = device\_release,

.read = device\_read,

.write = device\_write,

.unlocked\_ioctl = device\_ioctl,

};

int init\_module(void)

{

Major = register\_chrdev(0,DEVICE\_NAME, &fops);

if (Major < 0) {

printk(KERN\_ALERT "Registering char device failed with %d\n", Major);

return Major;

}

addr = ioremap(PORT, RANGE);

LCD\_Init();

LCD\_String("Hanyang Univ.");

LCD\_Command(0xC0); /\* Go to 2nd line\*/

LCD\_String("LCD1602 driver"); /\* Write string on 2nd line\*/

return SUCCESS;

}

void cleanup\_module(void)

{

unregister\_chrdev(Major, DEVICE\_NAME);

}