| Ousmane Toure & Jianning Chen  EECE2160 | Embedded Design: Enabling Robotics  Prelab Assignment 6 |
| --- | --- |

Prelab Assignment 6

Ousmane Toure & Jianning Chen

Toure.o@northeastern.edu

chen.jiann@northeastern.edu

Submit date: 6/10/2022

Due Date: 6/10/2022

**5.0 PreLab of Lab 5**

**a.**

char \*virtual\_base = (char \*) mmap (NULL, LW\_BRIDGE\_SPAN, (PROT\_READ |PROT\_WRITE), MAP\_SHARED, \*fd, LW\_BRIDGE\_BASE);

The first argument, which in our case is NULL, is the address. This identifies the first byte that needs to be changed

The second argument, LW\_BRIDGE\_SPAN, is the total length of bytes to change, this means we’re going to go through the whole span

The third argument, PROT\_READ |PROT\_WRITE, is the access allowed, in our case we have read and write access.

The fourth argument, MAP\_SHARED, is the access flags, means anyone else can see the memory we mapped in our case.

The fifth argument, \*fd, is the file descriptor of the file we have opened and sets it as the initial memory point.

The sixth argument, LW\_BRIDGE\_BASE, is the file offset that identifies the location to start the mapping.

**b.**

The figures are located on page 7 through 9

**c.**

The program doesn’t run and provides a segmentation fault. This is because the base address is not the physical address. Furthermore, it seems that the offset wasn’t taken into account when assigning the LEDs location.

**d.**

The program doesn’t run and provides a segmentation fault. Because the base address is set to the virtual address not the physical address. It doesn’t account for the offset needed to access the switches location, causing a segment fault to occur.

**e.**

void Write1Led(char \*pBase,int ledNum,int state){

unsigned int val,num;

if(state == 0){

//num = 0x00 | (0x00 << ledNum);

//num = ~num; same as below

num = ~(0x01 << ledNum);

val = num & RegisterRead(pBase,LEDR\_OFFSET);

RegisterWrite(pBase,LEDR\_OFFSET,val);

}

if(state == 1){

num = 0x01 << ledNum;

val = num | RegisterRead(pBase,LEDR\_OFFSET);

RegisterWrite(pBase,LEDR\_OFFSET,val);

}

else{cerr<<"Value error "; }//raise error

}

**f.**

int ReadAllSwitches(char \*pBase)

{

return \* (volatile unsigned int \*)(pBase + SW\_OFFSET);

}

# References

1. Prof. Julius Marpaung, “*Lab Report Guide*”, Northeastern University, January 6 2020.