ii)This function writes values to LEDs and instructs them to turn on or off.

iii) mmap:

#include <sys/mman.h> void \* mmap (void \*addr, size\_t len, int prot, int flags, int fd, off\_t offset);

Asks the kernel to map len bytes of the object represented by the file descriptor fd, starting at offset bytes into the file, into memory. If addr is included, it indicates a preference to use that starting address in memory. The access permissions are dictated by prot, and additional behavior can be given by flags. On success, this call returns a pointer to the memory mapped area. In the case of an error, it returns -1.

vi) munmap

#include <sys/mman.h>  
int munmap (void \*addr, size\_t len);   
A call to munmap( ) removes any mappings that contain pages located anywhere in the process address space starting at addr.   
On success, munmap( ) returns 0; on failure, it returns −1.

3.4

#include <math.h>

void userio\_ledSet(unsigned char \*pBase, unsigned int ledNr, unsigned int state);

{

/\*address of LED

param value to show on LED

\*/

int led\_addr, value=0;

printf(“Please enter which LED you want to turn on (as a number from 0 to 7)—>”);

scanf(“%d”,&ledNr);

printf(“\nPlease enter the state you want the LED to be (1 for on and 0 for off)—>”);

scanf(“%d”,&state);

if (state)

}

int led\_addr= pBase + LED1\_OFFSET+ledNr\*4;

value = value/pow(2 Nr)%2 ;

REG\_WRITE(pBase, led\_addr, value);

}

else {

return 0;

}

}

The address distance between two nearby LEDs is 4.

3.5

void userio\_switchGet(unsigned char pBase, int value)

{

REG\_READ(pBase, SW1\_OFFSET);

REG\_READ(pBase, SW2\_OFFSET);

REG\_READ(pBase, SW3\_OFFSET);

REG\_READ(pBase, SW5\_OFFSET);

REG\_READ(pBase, SW6\_OFFSET);

REG\_READ(pBase, SW7\_OFFSET);

REG\_READ(pBase, SW8\_OFFSET);

}