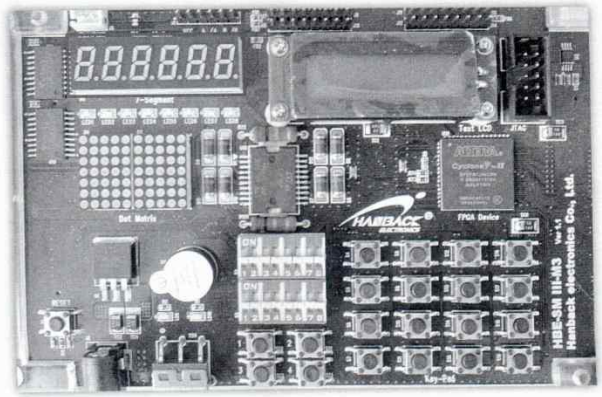
```
[root@SMIII-SV210 ~]$./led
 -------------------------------------
  8bit LED IO Interface Procedure
 -------------------------------------
  [l] left shift
  [r] right shift
  [q] exit
 -------------------------------------
```

키보드의 'l' 이나 'r' 을 누르면 led가 이동하는 것을 볼 수 있을 것이다. 'q' 를 누르면 프로그램이 종료된다.

11.1.2 7-Segment 제어

HBE-SMIII-SV210 보드에는 6자리를 가지는 7-Segment 모듈이 실장되어 있다. 이 7-Segment LED는 데이터 버스에 연결되어 있으며, 두개의 제어번지에 즉, 각 자리(그리드)를 제어하는 번지와 각 부분(세그먼트)를 제어하는 번지에 의해 7-Semgnet LED를 제어할 수 있다.



[그림 11-3] 7-Segment 위치

원하는 자리에 원하는 부분(세그먼트)를 켜기 위한 데이터를 출력하면 해당 데이터에 따라 7-Segment에 숫자나 영문자를 표시할 수 있으며, 데이터 구성하는 방법은 다음과 같다.

- 그리드 선택 번지 : 0x88000030
- 세그먼트 선택 번지 : 0x88000032

그리드 선택 번지의 데이터 형식은 다음과 같다.

- 0x01 - SEG1을 선택
- 0x02 - SEG2를 선택
- 0x04 - SEG3을 선택
- 0x08 - SEG4를 선택
- 0x10 - SEG5를 선택
- 0x20 - SEG6을 선택

다음으로 세그먼트 데이터 형식은 다음과 같다.

- 숫자 0 : 0xfc
- 숫자 1 : 0x60
- 숫자 2 : 0xda
- 숫자 3 : 0xf2
- 숫자 4 : 0x66
- 숫자 5 : 0xb6
- 숫자 6 : 0xbe
- 숫자 7 : 0xe4
- 숫자 8 : 0xfe
- 숫자 9 : 0xf6
- 영문 a : 0xfa
- 영문 b : 0x3e
- 영문 c : 0x1a
- 영문 d : 0x7a
- 영문 e : 0x9e
- 영문 f : 0x8e

다음과 같이 /working/mmap/2.segment 디렉토리에서 segment.c를 작성한다.

```
root@hanback-desktop:~# mkdir -p /working/mmap/2.segment
root@hanback-desktop:~# cd /working/mmap/2.segment/
root@hanback-desktop:/working/mmap/2.segment# vi segment.c
```

---------- 다음과 같이 작성한다 ----------

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/mman.h>
#include <fcntl.h>

unsigned short Getsegcode(short x);

#define FPGA_BASEADDRESS      0x88000000
#define SEG_GRID_OFFSET      0x30
#define SEG_DATA_OFFSET      0x32

int main()
{
   int fd;
   volatile int i,j;
   unsigned int count=1,num,temp1,temp2;
   unsigned short data[6];
   unsigned short digit[6]={0x20, 0x10, 0x08, 0x04, 0x02, 0x01};
   unsigned short *addr_fpga;
   unsigned short *addr_grid, *addr_data;

   if((fd=open("/dev/mem",O_RDWR|O_SYNC)) < 0) {
      printf("mem open fail\n");
      exit(1);
   }

   addr_fpga= (unsigned short *)mmap(NULL, 4096,
PROT_READ|PROT_WRITE, MAP_SHARED, fd, FPGA_BASEADDRESS);

   addr_grid=addr_fpga+SEG_GRID_OFFSET/sizeof(unsigned short);
   addr_data=addr_fpga+SEG_DATA_OFFSET/sizeof(unsigned short);

   if(*addr_grid == (unsigned short)-1 || *addr_data==(unsigned
short)-1) {
      close(fd);
      printf("mmap error\n");
      exit(1);
   }

   printf("- 7 Segment\n");
   while(count!=0) {
      num=0;
      printf("Input count value: [1-999999] (0 : exit program)
\n");
```

```
        scanf("%d", &count);
        while(num<=count ) {  // 카운트 값 조절
            data[5]=Getsegcode(num/100000);
            temp1=num%100000;
            data[4]=Getsegcode(temp1/10000);
            temp2=temp1%10000;
            data[3]=Getsegcode(temp2/1000);
            temp1=temp2%1000;
            data[2]=Getsegcode(temp1/100);
            temp2=temp1%100;
            data[1]=Getsegcode(temp2/10);
            data[0]=Getsegcode(temp2%10);

            for(j=0;j<20;j++) {    // 값 조정으로 카운트 속도 조절
                for(i=0;i<6;i++) {
                    *addr_grid = digit[i];
                    *addr_data = data[i];
                    usleep(1000);
                }
            }
            num++;
        }
        *addr_grid = ~digit[0];
        *addr_data = 0;
    }

    munmap(addr_fpga,4096);
    close(fd);
    return 0;
}

unsigned short Getsegcode(short x)
{
    unsigned short code;
    switch (x) {
        case 0x0 : code = 0xfc; break;
        case 0x1 : code = 0x60; break;
        case 0x2 : code = 0xda; break;
        case 0x3 : code = 0xf2; break;
        case 0x4 : code = 0x66; break;
        case 0x5 : code = 0xb6; break;
        case 0x6 : code = 0xbe; break;
        case 0x7 : code = 0xe4; break;
        case 0x8 : code = 0xfe; break;
        case 0x9 : code = 0xf6; break;

        case 0xa : code = 0xfa; break;
        case 0xb : code = 0x3e; break;
```

```
        case 0xc : code = 0x1a; break;
        case 0xd : code = 0x7a; break;
        case 0xe : code = 0x9e; break;
        case 0xf : code = 0x8e; break;
        default : code = 0; break;
    }
    return code;
}
------------------------- 저장하고 종료한다 -------------------------
```

위에서 작성한 소스를 컴파일하기 위한 Makefile을 작성한다.

```
root@hanback-desktop:/working/mmap/2.segment# vi Makefile
------------------------- 다음과 같이 작성한다 -------------------------
CC = arm-linux-gcc

CFLAGS = -DNO_DEBUG

EXEC=segment

OBJS=$(EXEC).o

####### Implicit rules

.SUFFIXES: .cpp .cxx .cc .C .c

.cpp.o:
    $(CXX) -c $(CXXFLAGS) -o $@ $<

.cxx.o:
    $(CXX) -c $(CXXFLAGS) -o $@ $<

.cc.o:
    $(CXX) -c $(CXXFLAGS) -o $@ $<

.C.o:
    $(CXX) -c $(CXXFLAGS) -o $@ $<

.c.o:
    $(CC) -c $(CFLAGS) -o $@ $<

all: $(EXEC)
```

```
$(EXEC): $(OBJS)
   $(CC) -o $@ $^

clean:
   rm -f $(OBJS) $(EXEC)
-------------------------- 저장하고 종료한다 --------------------------
```

make 명령어를 실행하여 소스를 컴파일하고, 실행파일(segment)를 /tftpboot에 복사한다.

```
root@hanback-desktop:/working/mmap/2.segment# make
arm-linux-gcc -c -DNO_DEBUG -o segment.o segment.c
arm-linux-gcc -o segment segment.o
root@hanback-desktop:/working/mmap/2.segment# ls
Makefile segment segment.c segment.o
root@hanback-desktop:/working/mmap/2.segment# cp segment /tftpboot/
```

타겟보드에서 실행파일(segment)를 다운받고, 권한을 변경한 후 실행한다.

```
[root@SMIII-SV210 ~]$tftp -r segment -g 192.168.0.100
[root@SMIII-SV210 ~]$ls -l
-rw-r--r--   1 root   root      7489 Feb 1 20:06 segment
[root@SMIII-SV210 ~]$chmod 777 segment
[root@SMIII-SV210 ~]$./segment
- 7 Segment
Input count value: [1-999999] (0 : exit program)
100
```

숫자를 입력하고 엔터키를 누르면 7-Segment의 숫자가 증가되는 것을 볼 수 있다.