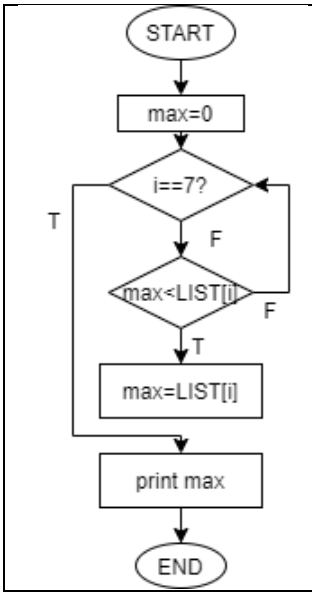
전자 HW 설계 – 실습 보고서

이름: 양해찬 (2016124145)

✓ Part I

동작 원리



비교할 숫자들이 저장되어 있는 LIST 배열을 선언하고 큰 숫자를 찾는 루틴 전에 가장 max에 0을 저장한다. 그리고 변수 i에 배열의 개수를 저장하고 순서에 맞게 배열에서 숫자를 꺼내서 max 와 비교한 후 해당 수가 max 보다 크다면 max에 그 숫자를 넣어준다. i가 7(배열 개수)이 되어 루틴을 마치면 max에는 LIST 배열의 가장 큰 숫자가 저장된다. 마지막으로 printf를 이용해서 Monitor program 인터페이스에 display한다.

구현 코드 설명

```
#include <stdio.h>

int LIST[8] = {7, 4, 5, 3, 6, 1, 8, 2};
//TEST NUNBERS
int main(){
    int max=0;//max초기화
    int i;
    for(i=1;i<=LIST[0];i++){
        if(max<LIST[i]) max=LIST[i];
    }
    printf("max=%d",max);//max출력
}
```

결과 및 토의

Terminal

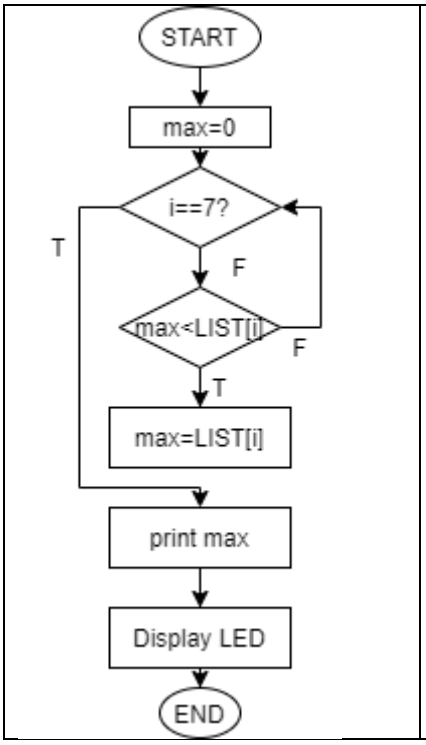
```
JTAG UART link established using cable "DE-SoC [USB-1]", device 2, instance 0x00
max=8
```

LIST에서 가장 큰 8이 출력되고 있다.

전자 HW 설계 – 실습 보고서

✓ Part II

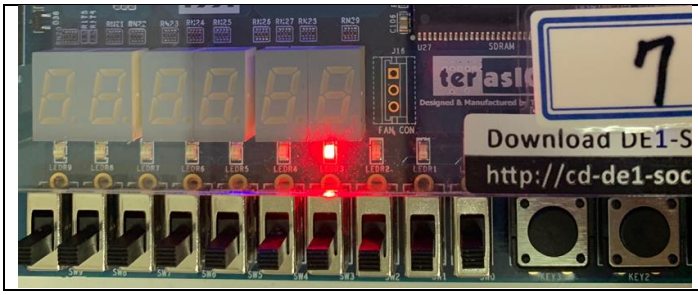
동작 원리



Part1은 값을 확인하기 위해 printf를 사용했지만 Part2 부터 DE1-Soc의 IO를 이용해서 데이터 값을 확인한다. IO를 이용하기 위해서 포인터를 사용하여 Device의 주소 값을 저장해야만 한다. 다른 포인터와는 다르게 IO를 통해 input output하기 위해서는 static volatile int로 선언하여 컴파일 할 때 optimize 하지 않도록 해야 한다. 그렇게 선언된 포인터 변수에 최종적으로 원하는 데이터를 넣어주게 되면 IO를 통해 출력을 확인할 수 있다. Part2는 Part1에서 구한 리스트 중 가장 큰 수인 MAX 변수를 binary로 LEDR을 통해 출력한다.

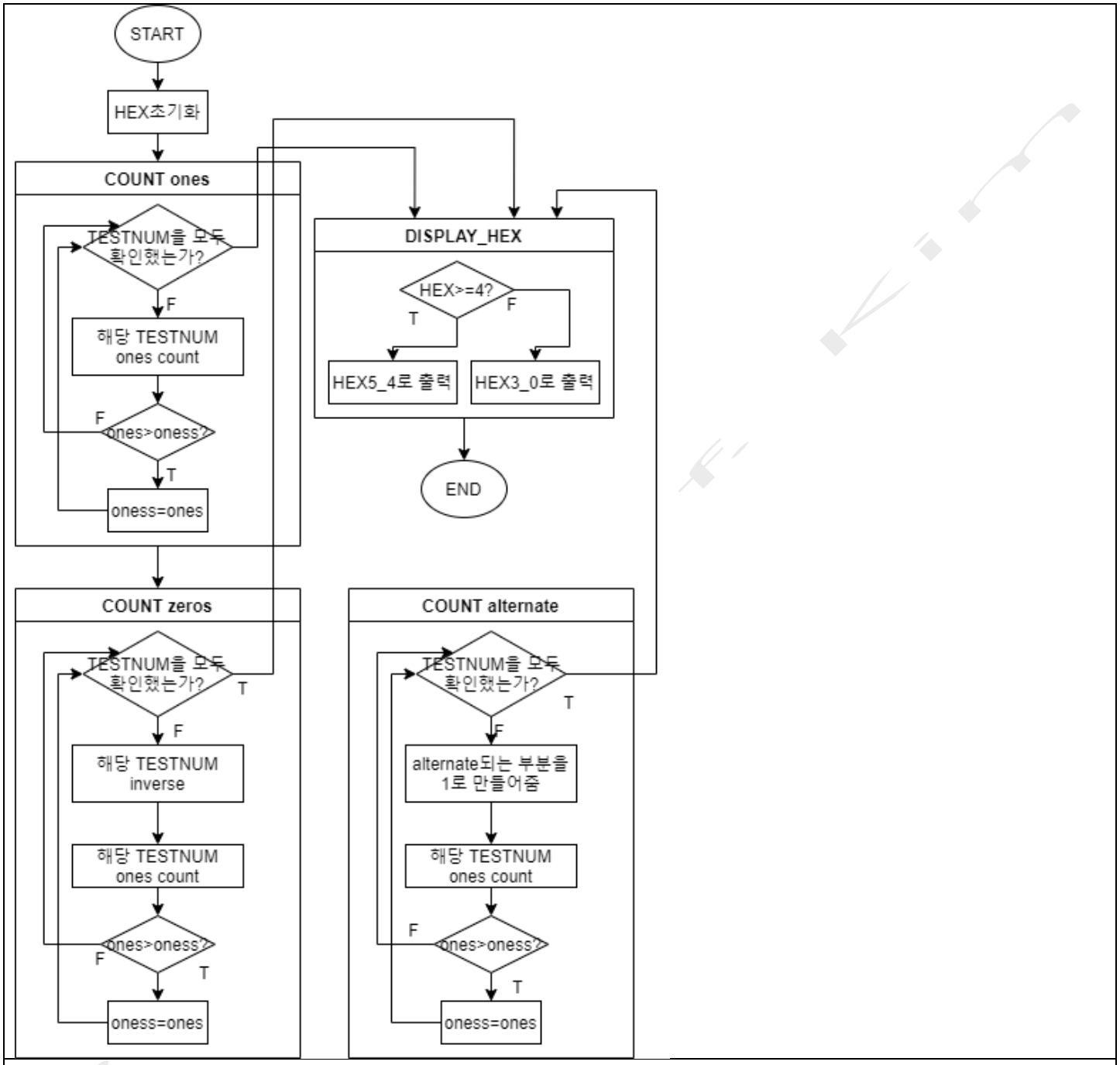
구현 코드 설명

```
#include "address_map_nios2.h"

static volatile int *LEDR_ptr = (int*) LEDR_BASE;
//LEDR_BASE주소 복사
int LIST[8] = {7, 4, 5, 3, 6, 1, 8, 2};
//TEST_NUMBERS
void display_led(int num){
    *LEDR_ptr=num;//LED출력
}
int main(){
    int max=0;
    int i;
    for(i=1;i<=LIST[0];i++){
        if(max<LIST[i]) max=LIST[i];
    }
    display_led(max);
    //max를 LED로 출력
}
```

	Part1에서의 출력과 동일한 8의 binary 즉 100이 LEDR로 잘 출력 되고 있다.

✓ Part III

동작 원리.



주어진 16진수 32bits 숫자들을 2진수로 변환하였을 때, 각각의 16진수 숫자들의 1의 연속된 개수를 count하는 COUNT_ONES, 0의 연속된 개수를 count하는 COUNT_ZEROS, 연속된 10의 개수를 count하는 COUNT_ALTERNATE 함수를 수행하고 ONES, ZEROS, ALTERNATE에서 나온 숫자들 중 가장 큰 수를 HEX5_0에 DISPLAY한다.
ZEROS_COUNT함수에서는 zeros를 구하기 전 testnum을 inverse해서 ones를 구하는 방법으로 구현했고, COUNT_ALTERNATE에서는 ALTERNATE하는 부분을 ones로 만들어 이후 ones를 구하는 방법으로 구현했다.
HEX5_4에는 ATLERNATE의 최대값을, HEX3_2에는 ZEROS의 최대값을 그리고 HEX1_0에는 ONES의 최대값을 표시한다.

구현 코드 설명

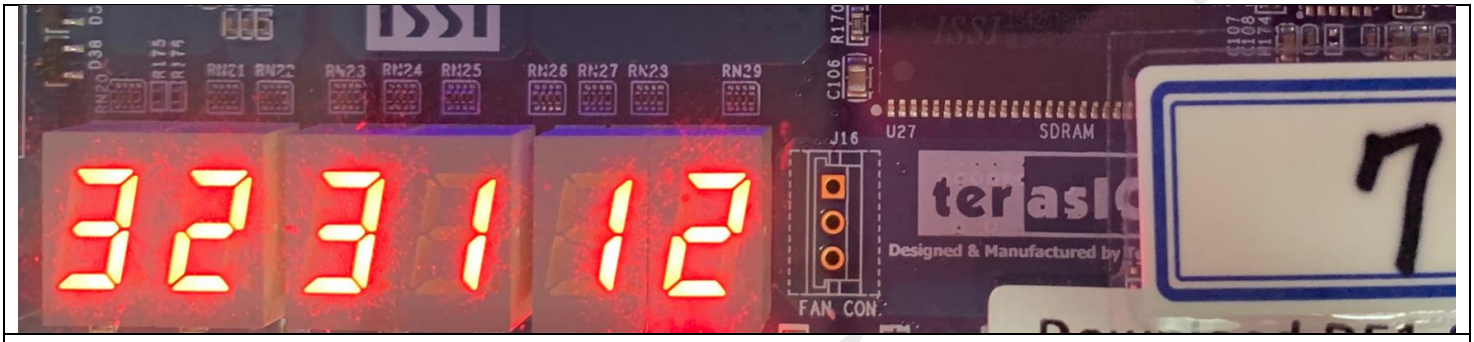
```
#define HEX3_HEX0_BASE          0xFF200020
#define HEX5_HEX4_BASE          0xFF200030
#include <stdio.h>


unsigned int TEST_NUM[] = { 0x0000e000, 0x3fabedef, 0x00000001, 0x00000002,
                            0x75a5a5a5, 0x01ffc000, 0x03ffc000, 0x55555555,
                            0x77777777, 0x08888888,0x00000000 };
unsigned int seg_code[] = { 0b00111111, 0b00000110, 0b01011011, 0b01001111,
                             0b01100110, 0b01101101, 0b01111101, 0b00000111,
                             0b01111111, 0b01100111 };
                             //segment code
static volatile int* HEX3_HEX0_ptr = (int*)HEX3_HEX0_BASE;
static volatile int* HEX5_HEX4_ptr = (int*)HEX5_HEX4_BASE;
//HEX에 diplay하기위한 함수
void display_HEX(int num, int hex) {
    int hex_tmp = hex % 4;//hex3-0과 5-4는 주소값이 다르므로
    int NT,NO;//NT=10의자리수 NO=1의자리수
    unsigned int output_num;//hex에 저장할 segmentcode
    if (num >= 10) {     //10이상일때
        NT = num / 10;  //입력값을 10으로 나눠 10의자리수를 구함
        NO = num - 10*NT;   //1의자리수 구함
    }
    else {          //10미만
        NO = num;   //1의자리 그대로출력
        NT = 0;     //10의자리는 0
    }
    unsigned int output_num_T = seg_code[NT] << (hex_tmp * 8);
    //10의자리 segment code부분
    unsigned int output_num_O = seg_code[NO] << ((hex_tmp-1) * 8);
    //1의자리 segment code부분
    output_num=output_num_O|output_num_T|* HEX3_HEX0_ptr;
    //10자리 1의자리 합침
    if (hex > 3) {//HEX4나 HEX5로 출력할 때
        output_num=output_num_O|output_num_T;
        *HEX5_HEX4_ptr = output_num;
    }
    else {//HEX3-0
        output_num=output_num_O|output_num_T|* HEX3_HEX0_ptr;
        *HEX3_HEX0_ptr = output_num;
    }
}
int COUNT_ONES() {//ones
```

```
    int i, j, ones = 0, ones_tmp = 0, oness = 0;
    //ones:testnum하나에서의 가장 큰 ones
    //ones_tmp:testnum하나에서의 현재ones값
    //oness:최종 ones값
    unsigned int l;//testnum임시저장용
    unsigned int k = 0x80000000;
    for (i = 0;i < sizeof(TEST_NUM) / sizeof(int);i++) {
        l = TEST_NUM[i];
        if(l==0x0) break;//0x0이면 break
        ones = 0;
        for (j = 0;j < 32;j++) {
            if ((l & k) == 0) {
                if (ones_tmp > ones) {
                    ones = ones_tmp;
                }
                ones_tmp = 0;
            }
            else {
                ones_tmp++;
            }
            l = l << 1;
        }
        if (ones > oness) oness = ones;
    }
    return oness;
}
int COUNT_ZEROS() {
    int i, j, ones = 0, ones_tmp = 0, oness = 0;
    //ones:testnum하나에서의 가장 큰 ones
    //ones_tmp:testnum하나에서의 현재ones값
    //oness:최종 ones값
    unsigned int l;
    unsigned int k = 0x80000000;
    for (i = 0;i < sizeof(TEST_NUM) / sizeof(int);i++) {
        if(TEST_NUM[i]==0x0) break;//0x0이면 break
        l = ~TEST_NUM[i];//zeros는 ones에서 testnum만 inverse해서 구하면된다
        ones = 0;
        for (j = 0;j < 32;j++) {
            if ((l & k) == 0) {
                if (ones_tmp > ones) {
                    ones = ones_tmp;
                }
                ones_tmp = 0;
            }
            else {
```

```
            ones_tmp++;
        }
        l = l << 1;
    }
    if (ones > oness) oness = ones;
  }
  return oness;
}

int COUNT_ALTERNATE() {
  int i, j, ones = 0, ones_tmp = 0, oness = 0;
  unsigned int l, m;//토글되는 비트를 1로 만들어주기위함
  unsigned int k = 0x80000000;
  //ones:testnum하나에서의 가장 큰 ones
  //ones_tmp:testnum하나에서의 현재ones값
  //oness:최종 ones값
  for (i = 0;i < sizeof(TEST_NUM) / sizeof(int);i++) {
    if(TEST_NUM[i]==0x0) break;//0x0이면 break
    if ((TEST_NUM[i] << 31) == 0) l = TEST_NUM[i];
    else    l = ~TEST_NUM[i];
    m = l << 1;
    l = l ^ m;
    ones = 0;
    for (j = 0;j < 32;j++) {
      if ((l & k) == 0) {
        if (ones_tmp > ones) {
          ones = ones_tmp;
        }
        ones_tmp = 0;
      }
      else {
        ones_tmp++;
      }
      l = l << 1;
    }
    if (ones > oness) oness = ones;
  }
  if (oness > 0) oness++;
  return oness;
}

int main() {
  * HEX3_HEX0_ptr=0;
  * HEX5_HEX4_ptr =0;
  int ones=COUNT_ONES();
```

```
    int zeros=COUNT_ZEROS();
    int alter=COUNT_ALTERNATE();
    display_HEX(ones, 1);//ones를 HEX1,0에 출력
    display_HEX(zeros, 3);//zeros를 HEX3,2에 출력
    display_HEX(alter, 5);//alternate를 HEX5,4에 출력
    printf("ONES: %d ₩n",ones);
    printf("ZEROS: %d ₩n",zeros);
    printf("ALTERNATES: %d ₩n",alter);
}
```

결과 및 토의



0x55555555의 alternate 32, 0x00000001의 zeros 31, 0x03ffc000의 ones 12가 잘 출력되고 있다,

Terminal

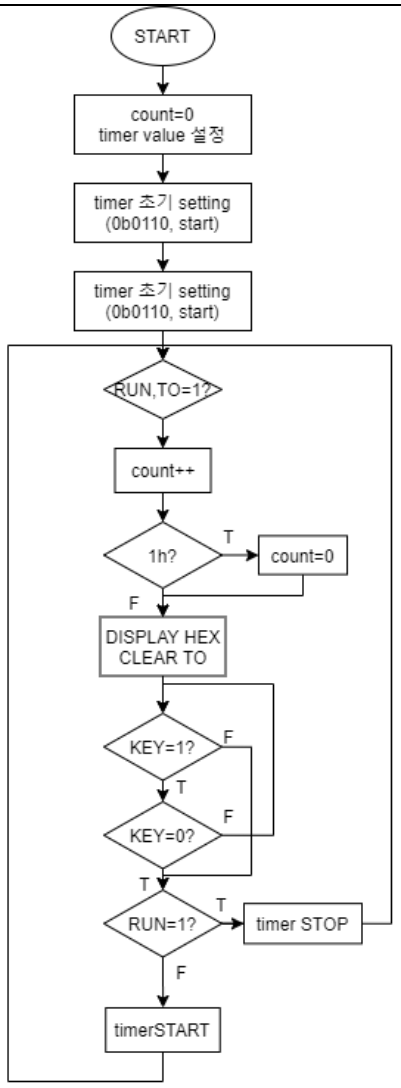
```
JTAG UART link established using cable "DE-SoC [USB-1]", device 2, instance 0x00
ONES: 12
ZEROS: 31
ALTERNATES: 32
```

Terminal에도 동일한 결과가 출력된다.

전자 HW 설계 – 실습 보고서

✓ Part IV

동작 원리



Polled I/O를 이용하여 Timer와 Key의 register 변화를 확인하면서 Timer를 Key를 누를 때마다STOP/START 동작 시키며 HEX에 그 시간을 표시하는 프로그램을 설계한다. HEX5_4에는 mm(분), HEX3_2에는 ss(초), HEX1_0에는 ms(밀리초)를 표시한다. DISPLAY_HEX함수는 part3과 동일하다. TIMER의 interval time을 0.01초로 설정하고 TO가 1이 될 때 마다 count를 해준다. 그 후 TO를 초기화해서 계속 반복한다.

TimerValue에서 가장 작게 표시할 시간을 설정해주고 Timer Control register를 0b0110으로 세팅해준다. 초기 Display에 다른 숫자가 출력되지 않도록 HEX 출력값은 0으로 초기화 해준다.

기본적인 register 설정이나 출력 설정이 끝났다면 while(1), 무한 반복 구문을 통하여 두가지를 확인하는데 첫번째는 현재 Run과 TO의 상태를 확인한다. RUN과 TO 상태가 1이면 count를 1씩 증가 시키며 count가 360000 즉, 59:59:99를 지난 1시간이 되면 다시 0부터 시작할 수 있도록 초기화 시킨다. 또한 HEX에 시간을 표시해주며 TO가 1이면 주어진 시간이 Time Out되었으므로 다시 시간이 흐르도록 TO를 0으로 초기화 한다. 두번째는 Key의 입력을 받는 구문인데 key입력은 계속 기다려야 하므로 while (READ_KEY())를 사용해 입력을 기다린다. 키 입력을 받았다면 다시 키 입력이 0이 될 때 (손에서 뗄 때) RUN의 상태를 확인하여 STOP할지 START할지 정해준다.

.

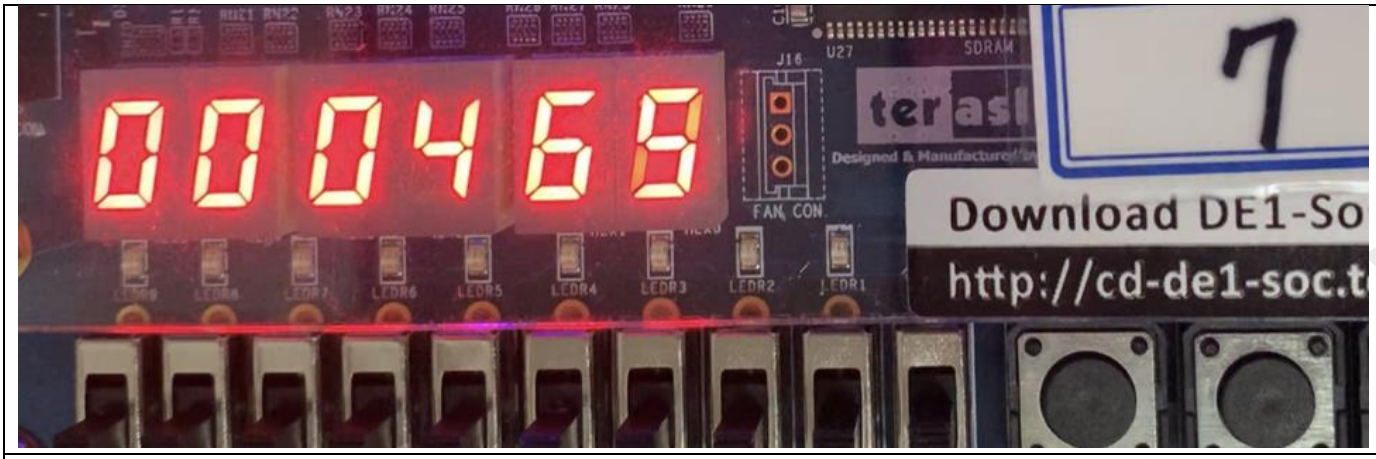
구현 코드 설명

```
#define HEX3_HEX0_BASE          0xFF200020
#define HEX5_HEX4_BASE          0xFF200030
#define TIMER_BASE              0xFF202000
#define KEY_BASE                0xFF200050
unsigned int seg_code[] = { 0b00111111, 0b00000110, 0b01011011, 0b01001111,
                            0b01100110, 0b01101101, 0b01111101, 0b00000111,
                            0b01111111, 0b01100111 };
static volatile int *KEY_ptr = (int*)KEY_BASE;
static volatile int *TIMER_ptr = (int*)TIMER_BASE;
static volatile int *HEX3_HEX0_ptr = (int*)HEX3_HEX0_BASE;
```

```
static volatile int *HEX5_HEX4_ptr = (int*)HEX5_HEX4_BASE;
static volatile int *TIMER_SET = (int*)(TIMER_BASE + 4);
static volatile int *TIMER_pl = (int*)(TIMER_BASE + 8);
static volatile int *TIMER_ph = (int*)(TIMER_BASE + 12);

void TIMER_pl_ph(int ms) {
    int pl = ms * 100000;
    int ph = pl >> 16;
    *TIMER_pl = pl;
    *TIMER_ph = ph;//timer value 설정
}

char TIMER_TO() {// TO
    return (*TIMER_ptr) & 0b01;
}
void CLEAR_TIMER_TO() {// TO초기화
    *TIMER_ptr = 0;
}
void Timer_SET(int set) {// Timer setting
    *TIMER_SET = set;
}

char READ_KEY() {// Key 눌렸는지
    return (*KEY_ptr);
}

char READ_RUN() {//RUN 인지
    return (*TIMER_ptr) & 0b10;
}

// HEX3_2: sec, HEX1_0: msec
void DISPLAY_HEX3_0(int num) {
    unsigned int OS;
    unsigned int TS;
    num = num % 6000;//분단위 버림
    OS = seg_code[num / 1000];//10초단위
    TS = OS << 8;
    num = num % 1000;//10초단위 버림
    OS = seg_code[num / 100];//1초단위
    TS = TS | OS;//10초 1초 단위 합침
    TS = TS << 8; //HEX한칸씩 밀어냄
    num = num % 100;//초단위 버림
    OS = seg_code[num / 10];//0.1초단위
    TS = TS | OS;//0.1초 단위까지 합침
    TS = TS << 8;//HEX한칸씩 밀어냄
```

```
    OS = seg_code[num % 10];//0.01초단위
    TS = TS | OS;//최종적으로 합침
    *HEX3_HEX0_ptr = TS;//HEX에 입력
}
//HEX5_4: min
void DISPLAY_HEX5_4(int num) {
    unsigned int TM;
    unsigned int OM;
    num = num / 6000;//분단위
    num = num % 60;
    TM = seg_code[num / 10]; //10분단위
    OM = seg_code[num % 10]; //1분단위
    TM = TM << 8;//HEX5
    TM = (TM | OM);//합침
    *HEX5_HEX4_ptr = TM;
}

//main 함수
int main()
{
    unsigned int count = 0; // COUNT 초기화

    TIMER_pl_ph(10);          // timer value = 10ms
    Timer_SET(0b0110);        // timer start

    DISPLAY_HEX3_0(0);
    DISPLAY_HEX5_4(0);        // HEX OFF

    while (1) {
        if (READ_RUN() && TIMER_TO()) {
            count++; // 1씩 카운트
            if (count == 6000 * 60) count = 0;// 한 시간이 되면 초기화
            DISPLAY_HEX3_0(count); // 시간 Display
            DISPLAY_HEX5_4(count);
            CLEAR_TIMER_TO();     // TO 초기화
        }
            while (READ_KEY()) { // Key 누름 확인
            while (READ_KEY()) { // Key 뗌 확인
            }
            if (READ_RUN()) Timer_SET(0b1010); // RUN이 1이면 STOP
            else Timer_SET(0b0110); // RUN이 0이면 START
        }
    }
}
```

결과 및 토의



TIMER 흐르는 중

STOP//하단 구현 동영상 링크

https://www.youtube.com/watch?v=W_TUszdTNmo