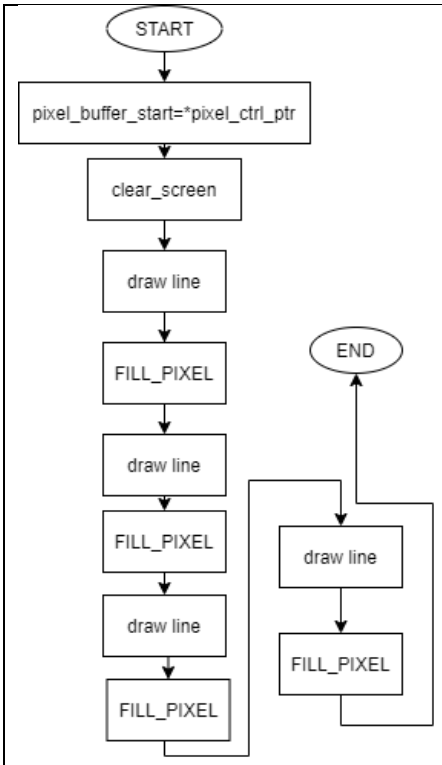
전자 HW 설계 – 실습 보고서

이름: 양해찬 (2016124145)

✓ Part I

동작 원리



Part1은 de1 soc와 VGA모니터를 연결해 모니터로 직선들을 출력하는 코드이다. 320*240 pixel 의 size를 가지고, RGB에 각각 값을 할당해서 원하는 색을 만들어 낼 수 있다.

픽셀 버퍼 주소에 원하는 RGB 값을 넣는 매크로를 정의하고 그 매크로를 draw_line 함수에서 사용 할 수 있다.

구현 코드 설명

```
        #define PIXEL_BUF_CTRL_BASE      0xFF203020

#define SCREEN_WIDTH 320
#define SCREEN_HEIGHT 240
//표시할 스크린의 크기 상수 정의
#define PIXEL(r,g,b) (short int)(((((r)&0x1f)<<11)|(((g)&0x3f)<<5)|(((b)&0x1f)))
#define FILL_PIXEL(x,y,r,g,b) *(short int*)(pixel_buffer_start +(((y)&0xff)<<10)+ (((x)&0x1ff)<<1))=PIXEL(r,g,b)

volatile int pixel_buffer_start;//화면 저장
volatile int* pixel_ctrl_ptr = (int*)PIXEL_BUF_CTRL_BASE;
//전역 번수 전언
void clear_screen(int r, int g, int b){//인수로 받은 rgb값으로 clear
    int x, y;
    for(x=0;x<SCREEN_WIDTH+1;x++){
        for(y=0;y<SCREEN_HEIGHT+1;y++){
            FILL_PIXEL(x,y,r,g,b);
        }
    }
```

```
}
int abs(int x){//절댓값
    if(x<0)
    return -x;
    else
    return x;
}
void draw_line(int x1, int y1, int x2, int y2, int r, int g, int b) {
    //선을 그리는 함수, 강의자료에 있는 알고리즘 사용함 설명 생략
    int tmp, x, y;
    int grad = (abs(x2 - x1) < abs(y2 - y1));
    if (grad) {
        tmp = x2;
        x2 = y2;
        y2 = tmp;
        tmp = x1;
        x1 = y1;
        y1 = tmp;
    }
    if (x1 > x2) {
        tmp = x1;
        x1 = x2;
        x2 = tmp;
        tmp = y1;
        y1 = y2;
        y2 = tmp;
    }
    int dx = abs(x2 - x1);
    int dy = abs(y2 - y1);
    int err = -(dx / 2);
    y = y1;
    int y_step;
    if (y1 < y2) y_step = 1;
    else y_step = -1;
    for (x = x1;x <= x2;x++) {
        if (grad) {
            FILL_PIXEL(y, x, r, g, b);
        }
        else FILL_PIXEL(x, y, r, g, b);
        err = err + dy;
        if (err >= 0) {
            y += y_step;
            err -= dx;
        }
    }
```

```
}
int main(void) {
    pixel_buffer_start = *pixel_ctrl_ptr;//화면에 그림
    clear_screen(0xFF, 0xFF, 0xFF);//흰색으로 화면clear
    draw_line(0,0,150,150,0x00,0x00,0xFF);//파란선
    draw_line(150,150,319,0,0x00,0xFF,0x00);//초록선
    draw_line(0,239,319,239,0xFF,0x00,0x00);//빨간선
    draw_line(319,0,0,239,0xFF,0x00,0xFF);//파란선
}
```

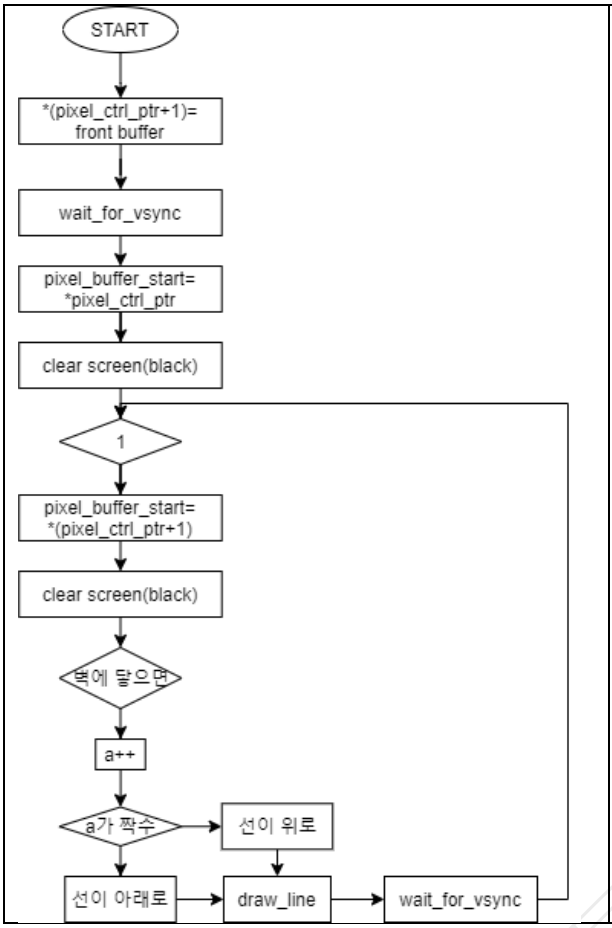
결과 및 토의

```
draw_line(0,0,150,150,0x00,0x00,0xFF);//파란선
draw_line(150,150,319,0,0x00,0xFF,0x00);//초록선
draw_line(0,239,319,239,0xFF,0x00,0x00);//빨간선
draw_line(319,0,0,239,0xFF,0x00,0xFF);//파란선
에 해당하는 선들이 잘 출력되고 있다.
```

전자 HW 설계 – 실습 보고서

✓ Part II

동작 원리



Part2는 Part1에서 사용했던 draw_line 함수를 그대로 사용해 화면 중앙에 움직이는 선을 만들고, 그것이 화면의 위아래로 움직이며 화면 끝까지 갔을 때에는 직전의 이동방향과 반대방향으로 같은 속도로 이동하게 하는 코드이다.

도중에 화면이 깜빡이지 않게 하기 위해 wait for vsync함수를 이용해 선을 그릴 때 마다 호출하면 깜빡이지 않고 자연스럽게 움직일 수 있다.

선이 움직이는 방법은 선이 벽에 닿을 때 마다 선언한 변수에 1씩 더해주고, 이것이 짝수인지 홀수인지 판단하고, 이것에 맞게 방향을 설정해 움직인다.

구현 코드 설명

```
#define PIXEL_BUF_CTRL_BASE      0xFF203020

#define SCREEN_WIDTH 320
#define SCREEN_HEIGHT 240
//표시할 스크린의 크기 상수 정의
#define PIXEL(r,g,b) (short int)((((r)&0x1f)<<11)|(((g)&0x3f)<<5)|(((b)&0x1f)))
#define FILL_PIXEL(x,y,r,g,b) *(short int*)(pixel_buffer_start +(((y)&0xff)<<10)+ (((x)&0x1ff)<<1))=PIXEL(r,g,b)

volatile int pixel_buffer_start;//화면 저장
volatile int* pixel_ctrl_ptr = (int*)PIXEL_BUF_CTRL_BASE;
//전역 번수 전언
short int front_buffer[512*256]; //front buffer를 위한 메모리 할당
short int back_buffer[512*256]; //back buffer를 위한 메모리 할당
void clear_screen(int r, int g, int b){//인수로 받은 rgb값으로 clear
    int x, y;
    for(x=0;x<SCREEN_WIDTH+1;x++){
```

```
        for(y=0;y<SCREEN_HEIGHT+1;y++){
            FILL_PIXEL(x,y,r,g,b);
        }
    }
}
int abs(int x){//절댓값
    if(x<0)
    return -x;
    else
    return x;
}
void draw_line(int x1, int y1, int x2, int y2, int r, int g, int b){
    //선을 그리는 함수, 강의자료에 있는 알고리즘 사용함 설명 생략
    int tmp,x,y;
    int grad=(abs(x2-x1)<abs(y2-y1));
    if(grad){
        tmp=x2;
        x2=y2;
        y2=tmp;
        tmp=x1;
        x1=y1;
        y1=tmp;
    }
    if(x1>x2){
        tmp=x1;
        x1=x2;
        x2=tmp;
        tmp=y1;
        y1=y2;
        y2=tmp;
    }
    int dx=abs(x2-x1);
    int dy=abs(y2-y1);
    int err=-(dx/2);
    y=y1;
    int y_step;
    if(y1<y2) y_step=1;
    else y_step=-1;
    for(x=x1;x<=x2;x++){
        if(grad){
            FILL_PIXEL(y,x,r,g,b);
        }
        else FILL_PIXEL(x,y,r,g,b);
        err=err+dy;
        if(err>=0){
```

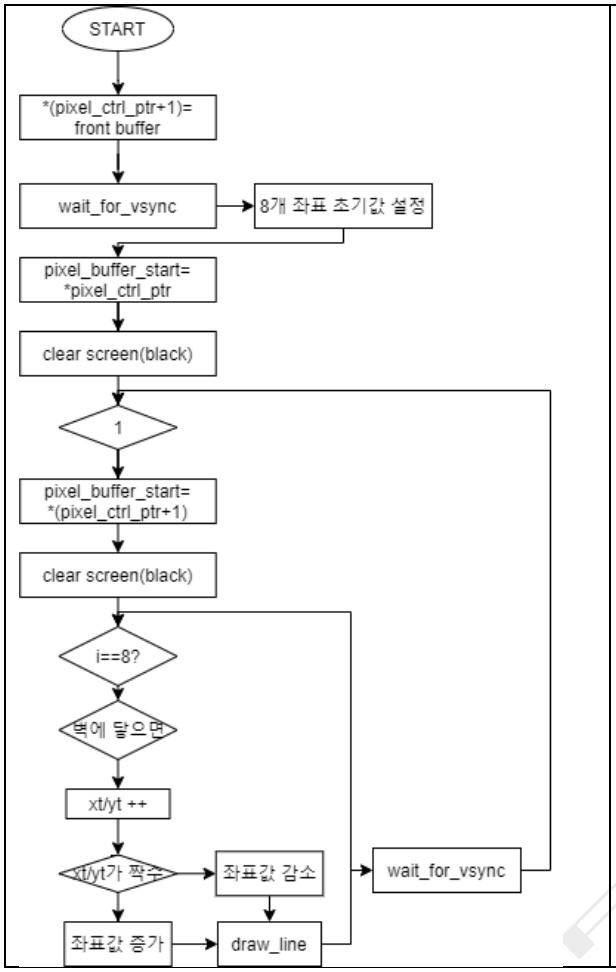
```
            y+=y_step;
            err-=dx;
        }
    }

}

void wait_for_vsync() {
    register int status;
    *pixel_ctrl_ptr = 1;
    //synchronization process 시작(swappint)
    status = *(pixel_ctrl_ptr + 3);//S
    while ((status & 0x01) != 0)//S=0(swapping이 끝날때)까지 대기
    status = *(pixel_ctrl_ptr + 3);
}

int main(void) {
    *(pixel_ctrl_ptr+1)=front_buffer; //front buffer로 주소 할당
    wait_for_vsync();                 //vsync 대기
    pixel_buffer_start = *pixel_ctrl_ptr;
    clear_screen(0x00, 0x00, 0x00); //화면 검은색으로 clear
    *(pixel_ctrl_ptr+1)=back_buffer;//back buffer로 주소 할당
    int x = 60, y = 0, a=-1;//초기 좌표
    while (1) {
        pixel_buffer_start = *(pixel_ctrl_ptr+1);
        clear_screen(0x00, 0x00, 0x00); //검은색으로 clear
        if(y>=SCREEN_HEIGHT||y<=0) a++; //screen 끝까지 가면 a++
        if(a%2!=0) y-=1;//a가 홀수이면 선이 위로 올라오게
        else y+=1;//짝수이면 아래로
        draw_line(x, y, x + 200, y, 0xff, 0xff, 0xff);//선 그림
        wait_for_vsync(); //vsync 대기
    }
}
```

결과 및 토의

https://www.youtube.com/watch?v=J69OUdBrCxw
선이 위아래로 bounce하며 잘 움직인다.

✓ Part III

동작 원리.



Part3은 Part2와 Part1을 응용해 만들었고, draw_line과 wait_for_sync 함수 등을 그대로 사용했다.

8개의 임의의 좌표를 찍고 그 점들에 작은 정사각형을 그려주고, 그 점들이 움직이며 벽과 부딪히면 이동하던 방향과 반대방향으로 움직이게 된다. 이때 draw_line함수를 이용해서 좌표들끼리 연결해서 직선도 같이 움직이게 한다.
clear_screen함수를 사용해 배경은 검은색으로 설정했고
draw square함수를 사용해 정사각형을 그렸다.
점, 선 모두 파란색으로 설정했다.

점들이 벽과 부딪혀 이동하는 원리는 Part2에서와 유사한데, 좌표 8개에 대해 x좌표, y좌표에 각각 8개짜리 배열을 선언하고, x y배열에 해당하는 원소 8개짜리 배열을 각각 선언했다. 각각의 좌표 원소들이 초기 설정값에 맞게 이동하다가 벽에 닿으면 각각의 좌표에 맞는 xt/yt 배열의 원소값을 1증가 시키고 이것의 홀/짝 여부에 따라 방향을 설정한다.

구현 코드 설명

```
#define PIXEL_BUF_CTRL_BASE      0xFF203020

#define SCREEN_WIDTH 320
#define SCREEN_HEIGHT 240
//표시할 스크린의 크기 상수 정의
#define PIXEL(r,g,b) (short int)(((((r)&0x1f)<<11)|((((g)&0x3f)<<5)|((((b)&0x1f)))
#define FILL_PIXEL(x,y,r,g,b) *(short int*)(pixel_buffer_start +(((y)&0xff)<<10)+ (((x)&0x1ff)<<1))=PIXEL(r,g,b)

volatile int pixel_buffer_start;//화면 저장
volatile int* pixel_ctrl_ptr = (int*)PIXEL_BUF_CTRL_BASE;//전역 번수 전언
short int front_buffer[512 * 256]; //front buffer를 위한 메모리 할당
short int back_buffer[512 * 256];   //back buffer를 위한 메모리 할당
void clear_screen(int r, int g, int b) {//인수로 받은 rgb값으로 clear
```

```
    int x, y;
    for (x = 0;x < SCREEN_WIDTH + 1;x++) {
        for (y = 0;y < SCREEN_HEIGHT + 1;y++) {
            FILL_PIXEL(x, y, r, g, b);
        }
    }
}
int abs(int x) {//절댓값
    if (x < 0)
        return -x;
    else
        return x;
}
void draw_line(int x1, int y1, int x2, int y2, int r, int g, int b) {
    //선을 그리는 함수, 강의자료에 있는 알고리즘 사용함 설명 생략
    int tmp, x, y;
    int grad = (abs(x2 - x1) < abs(y2 - y1));
    if (grad) {
        tmp = x2;
        x2 = y2;
        y2 = tmp;
        tmp = x1;
        x1 = y1;
        y1 = tmp;
    }
    if (x1 > x2) {
        tmp = x1;
        x1 = x2;
        x2 = tmp;
        tmp = y1;
        y1 = y2;
        y2 = tmp;
    }
    int dx = abs(x2 - x1);
    int dy = abs(y2 - y1);
    int err = -(dx / 2);
    y = y1;
    int y_step;
    if (y1 < y2) y_step = 1;
    else y_step = -1;
    for (x = x1;x <= x2;x++) {
        if (grad) {
            FILL_PIXEL(y, x, r, g, b);
        }
        else FILL_PIXEL(x, y, r, g, b);
```

```
        err = err + dy;
        if (err >= 0) {
            y += y_step;
            err -= dx;
        }
    }

}

void draw_square(int x1, int y1, int x2, int y2, int r, int g, int b) {
    int x, y;//정사각형을 그리는 함수
    for (x = x1; x <= x2; x++) {
        for (y = y1; y <= y2; y++) {
            FILL_PIXEL(x, y, r, g, b);
        }
    }
}
void wait_for_vsync() {
    register int status;
    *pixel_ctrl_ptr = 1;
    //synchronization process 시작(swappint)
    status = *(pixel_ctrl_ptr + 3);//S
    while ((status & 0x01) != 0)//S=0(swapping이 끝날때)까지 대기
        status = *(pixel_ctrl_ptr + 3);
}

int main(void) {
    *(pixel_ctrl_ptr + 1) = front_buffer;   //front buffer로 주소 할당
    wait_for_vsync();                       //vsync 대기
    pixel_buffer_start = *pixel_ctrl_ptr;
    clear_screen(0x00, 0x00, 0x00);     //화면 검은색으로 clear
    *(pixel_ctrl_ptr + 1) = back_buffer;//back buffer로 주소 할당
    int x[8] = { 20,50,200,170,300,23,156,69 };
    int y[8] = { 30,200,220,14,66,37,99,102 };//초기 좌표들
    int xt[8] = { 0,1,0,1,0,1,0,1 };
    int yt[8] = { 1,0,1,0,1,0,1,0 };//벽에 닿았는지 여부
    int i;
    while (1) {
        pixel_buffer_start = *(pixel_ctrl_ptr + 1);
        clear_screen(0x00, 0x00, 0x00);//화면 검은색으로 clear
        if (y[0] >= SCREEN_HEIGHT || y[0] <= 0) yt[0]++;
        if (yt[0] % 2 != 0) y[0] -= 1;
        if (yt[0] % 2 == 0) y[0] += 1;
        if (x[0] >= SCREEN_WIDTH || x[0] <= 0) xt[0]++;
        if (xt[0] % 2 != 0) x[0] -= 1;
```

```
        if (xt[0] % 2 == 0) x[0] += 1;      //점 X0Y0에 대한 이동설정
        for (i = 1;i < 8;i++) {
            if (y[i] >= SCREEN_HEIGHT || y[i] <= 0) yt[i]++;
            if (yt[i] % 2 != 0) y[i] -= 2;
            if (yt[i] % 2 == 0) y[i] += 2;
            if (x[i] >= SCREEN_WIDTH || x[i] <= 0) xt[i]++;
            if (xt[i] % 2 != 0) x[i] -= 2;
            if (xt[i] % 2 == 0) x[i] += 2; //나머지 점들에 대한 이동 설정
            draw_square(x[i] - 1, y[i] - 1, x[i] + 1, y[i] + 1, 0x00, 0x00, 0xFF);//각 점마다 정사각형
            draw_line(x[i - 1], y[i - 1], x[i], y[i], 0x00, 0x00, 0xFF);//점끼리 이은 선
        }
        draw_line(x[7], y[7], x[0], y[0], 0x00, 0x00, 0xFF);//x0 y0과 x7 y7이어줌
        wait_for_vsync();//vsync 대기
    }
}
```

결과 및 토의

https://www.youtube.com/watch?v=hDzx7TpBnB8

점들이 연결되어 선으로 표시되고 있고 각 점은 정사각형으로 이루어 진 채 벽에 부딪힐 때 마다 반대방향으로 이동하며 잘 수행되는 것을 볼 수 있다.