TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – GJ (박현우)
uc820@naver.com

목차

DSP C 프로그래밍

1) FOURIER TRANSFORM으로 DFT 구현하기 1

CODE

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <math.h>

#include <GL/glut.h>

#define SLICE   4

void draw_omega_sin(void);
void draw_spectrum(void);

float common_angles[5] = {15.0, 30.0, 45.0, 60.0, 75.0};
float freq_table[5] = {1000.0, 2400.0, 5000.0, 24000.0, 77000.0};

float theta = 0.0;

typedef struct complex
{
    float cosx[32][32];
    float isinx[32][32];
} c;
```

```c
void display(void)
{
    glClearColor(0.0, 0.0, 0.0, 1.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();

    //gluLookAt(0.0, 0.0, 3.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

    glColor3f(1, 0, 0);

    glBegin(GL_LINE_LOOP);
    glVertex3f(100.0, 0.0, 0.0);
    glVertex3f(-100.0, 0.0, 0.0);
    glEnd();

    glColor3f(0.0, 1.0, 0.0);

    glBegin(GL_LINE_LOOP);
    glVertex3f(0.0, 100.0, 0.0);
    glVertex3f(0.0, -100.0, 0.0);
    glEnd();

    //draw_omega_sin();
    draw_spectrum();
    glutSwapBuffers();
}

void reshape(int w, int h)
{
    GLfloat n_range = 100.0f;

    if(h == 0)
        h = 1;

    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    if(w <= h)
        glOrtho(-n_range, n_range, -n_range * h / w, n_range * h / w, -n_range, n_range);
    else
        glOrtho(-n_range * w / h, n_range * w / h, -n_range, n_range, -n_range, n_range);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
```

```c
void keyboard(unsigned char key, int x, int y)
{
    switch(key)
    {
        case 27:
            exit(0);
            break;
    }
}

void set_rand_amplitude(float *amp)
{
    *amp = rand() % 3 + 3;
}

void set_angle_with_common_angles(float *angle)
{
    *angle = common_angles[rand() % 5];
}

void angle2radian(float *angle, float *radian)
{
    *radian = *angle * M_PI / 180.0;
}

void radian2angle(float *angle, float *radian)
{
    *angle = *radian * 180.0 / M_PI;
}

void set_rand_frequency(float *freq)
{
    *freq = freq_table[rand() % 5];
}

void calc_period(float *freq, float *period)
{
    *period = 1 / (*freq);
}

void calc_angular_velocity(float *freq, float *ang_vel)
{
    *ang_vel = 2 * M_PI * (*freq);
}
```

1) FOURIER TRANSFORM으로 DFT 구현하기 2

CODE

```
void cos_sim(float amplitude, float ang_vel, float period)
{
    int cnt = 0;
    float step, t = 0.0;

    t = step = get_step(SLICE, period);

    while(cnt++ < 36)
    {
        printf("%.1fcos(%f * %.8f) = %f\n", amplitude, ang_vel,
                t, amplitude * cos(ang_vel * t));
        t += step;
    }
}

void sin_sim(float amplitude, float ang_vel, float period)
{
    int cnt = 0;
    float step, t = 0.0;

    t = step = get_step(SLICE, period);

    while(cnt++ < 36)
    {
        printf("%.1fsin(%f * %.8f) = %f\n", amplitude, ang_vel,
                t, amplitude * sin(ang_vel * t));
        t += step;
    }
}
```

```
void draw_spectrum(void)
{
    float x = 0, x2 = 0, y2, cx, cy;
    float t, step = 0.0;
    int i, j, cnt = 0, cache = 0;
    float period, freq = 100.0;
    float res_real[32] = {0};
    float res_image[32] = {0};
    float y[32] = {0};
    c exp = {0};

    calc_period(&freq, &period);  // period = 1 / 100
    step = get_step(SLICE, period);  // step =  period / 4   == 1/400

    for(i = 0; i < SLICE; i++)
    {
        for(j = 0; j < SLICE; j++)
        {
            exp.cosx[i][j] = cos(-(2 * M_PI * j * i) / SLICE);
            exp.isinx[i][j] = sin(-(2 * M_PI * j * i) / SLICE);
            printf("exp.cosx[%d][%d] = %f\n", i, j, exp.cosx[i][j]);
        }
    }

    if(t > period)
        t = 0.0;

    i = 0;
    t = 0.0;
    for(; i < SLICE; t += step)   // t =0, t = 1/400 ...
    {
        if(t > period)
        {
            break;
            t = 0.0;
        }
        y[i] = 10 * cos(200 * M_PI * t);

        printf("y[%d] = %f\n", i++, y[i]);

    }

    for(i = 0; i < SLICE; i++)
        for(j = 0; j < SLICE; j++)
        {
            res_real[i] += y[j] * exp.cosx[i][j];
            res_image[i] += y[j] * exp.isinx[i][j];
            printf("res_image[%d] = %f\n", i, res_image[i]);
        }
    for(i = 0; i < SLICE; i++)
        for(j = 0; j < SLICE; j++)
        {
            res_real[i] += y[j] * exp.cosx[i][j];
            res_image[i] += y[j] * exp.isinx[i][j];
            printf("res_image[%d] = %f\n", i, res_image[i]);
        }

    for(i = 0; i < SLICE; i++)
    {
        glBegin(GL_POINTS);
        glVertex2f(i * 10, res_real[i] * 3);
        glEnd();

        glBegin(GL_LINE_STRIP);
        glVertex2f(i * 10, res_real[i] * 3);
        glVertex2f(i * 10, 0);
        glEnd();
    }
#if 0
    glBegin(GL_LINE_STRIP);

    for(j = 0; j < SLICE; j++)
    {
        glVertex2f(j * 10, res_real[j] * 2);
        glVertex2f(j * 10, 0);
    }

    glEnd();
#endif
#if 0
    //if(cache && !(cnt % 16))
    if(cache)
    {
        glBegin(GL_POINTS);
        glVertex2f(cx * 4000, cy * 6);
        glVertex2f(t * 4000, y2 * 6);
        glEnd();

        glBegin(GL_LINE_STRIP);
        glVertex2f(t * 4000, y2 * 6);
        glVertex2f(t * 4000, 0);
        glEnd();
    }

    cache = 1;
    cx = t;
    cy = y2;
    cnt++;
}
#endif
```

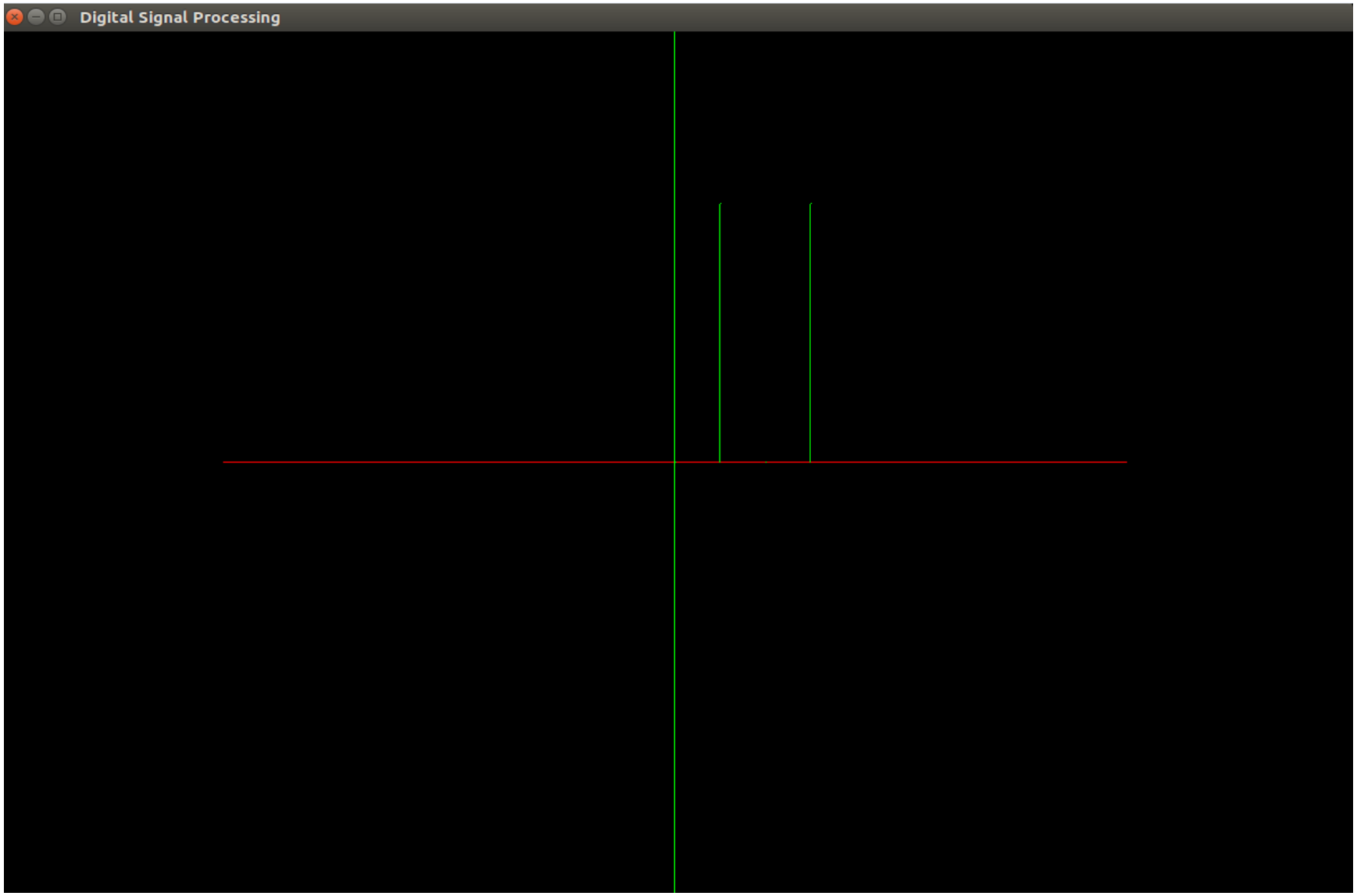
1) FOURIER TRANSFORM으로 DFT 구현하기 3

CODE

```
void draw_omega_sin(void)
{
    float amp, angle, period, freq, rad, omega, t, step = 0.0;
    float radius = 3.0;
    float x = 0, x2 = 0, y2, cx, cy;
    float tmp;
    int cnt = 0, cache = 0;

    srand(time(NULL));

    amp = 10;
    angle = 45.0;
    freq = 100.0;

    angle2radian(&angle, &rad);
    calc_period(&freq, &period);
    calc_angular_velocity(&freq, &omega);
    t = step = get_step(SLICE, period);

    printf("step = %f\n", step);

    if(t > period)
        t = 0.0;

    //glLineStipple(1, 0xFFEE);
    //glEnable(GL_LINE_STIPPLE);
    //glBegin(GL_POINTS);
    for(; ; t += step)
    {
        if(t > period)
            //if(t > 3 * period)
        {
            break;
            t = 0.0;
        }

        y2 = amp * sin(omega * t);

        if(cache && !(cnt % 16))
        {
            glBegin(GL_POINTS);
            glVertex2f(cx * 6000, cy * 6);
            glVertex2f(t * 6000, y2 * 6);
            glEnd();

            glBegin(GL_LINE_STRIP);
            //glVertex2f(cx * 4000, cy * 2);
            //glVertex2f(cx * 4000, 0);
            glVertex2f(t * 6000, y2 * 6);
            glVertex2f(t * 6000, 0);
            glEnd();
        }
        else
        {
            cache = 1;
            cx = t;
            cy = y2;
            cnt++;
        }
        //glEnd();
        //glDisable(GL_LINE_STIPPLE);
}

int main(int argc, char **argv)
{
    float amplitude, angle, period, frequency, radian, angular_velocity;
    float step = 0.0;

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE);
    glutInitWindowSize(1200, 800);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("Digital Signal Processing");

    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMainLoop();

    return 0;
}
```

1) FOURIER TRANSFORM으로 DFT 구현하기 4

DFT - SIMULATION



1) FOURIER TRANSFORM으로 FFT 구현하기 1

CODE

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include     <math.h>

#include <GL/glut.h>

#define SLICE           8
#define HALF_SLICE      (SLICE >> 1)

int glob = 4;
int count = 0;

#define TWID_FACTOR     (SLICE >> 1) + 1

typedef struct complex
{
    double re;
    double im;
} c;

void draw_omega_sin(void);
void draw_spectrum(void);

float common_angles[5] = {15.0, 30.0, 45.0, 60.0, 75.0};
float freq_table[5] = {1000.0, 2400.0, 5000.0, 24000.0, 77000.0};

float theta = 0.0;
```

```c
void display(void)
{
    glClearColor(0.0, 0.0, 0.0, 1.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();

    //gluLookAt(0.0, 0.0, 3.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

    glColor3f(1, 0, 0);

    glBegin(GL_LINE_LOOP);
    glVertex3f(100.0, 0.0, 0.0);

    glVertex3f(-100.0, 0.0, 0.0);
    glEnd();

    glColor3f(0.0, 1.0, 0.0);

    glBegin(GL_LINE_LOOP);
    glVertex3f(0.0, 100.0, 0.0);
    glVertex3f(0.0, -100.0, 0.0);
    glEnd();

    //draw_omega_sin();
    draw_spectrum();
    glutSwapBuffers();
}

void reshape(int w, int h)
{
    GLfloat n_range = 100.0f;

    if(h == 0)
        h = 1;

    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    if(w <= h)
        glOrtho(-n_range, n_range, -n_range * h / w, n_range * h / w, -n_range, n_range);
    else
        glOrtho(-n_range * w / h, n_range * w / h, -n_range, n_range, -n_range, n_range);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
```

```c
void keyboard(unsigned char key, int x, int y)
{
    switch(key)
    {
        case 27:
            exit(0);
            break;
    }
}

void set_rand_amplitude(float *amp)
{
    *amp = rand() % 3 + 3;
}

void set_angle_with_common_angles(float *angle)
{
    *angle = common_angles[rand() % 5];
}

void angle2radian(float *angle, float *radian)
{
    *radian = *angle * M_PI / 180.0;
}

void radian2angle(float *angle, float *radian)
{
    *angle = *radian * 180.0 / M_PI;
}

void set_rand_frequency(float *freq)
{
    *freq = freq_table[rand() % 5];
}

void calc_period(float *freq, float *period)
{
    *period = 1 / (*freq);
}

void calc_angular_velocity(float *freq, float *ang_vel)
{
    *ang_vel = 2 * M_PI * (*freq);
}

float get_step(float slice, float period)
{
    return period / slice;
}
```

1) FOURIER TRANSFORM으로 FFT 구현하기 2

CODE

```
float get_step(float slice, float period)
{
    return period / slice;
}

void cos_sim(float amplitude, float ang_vel, float period)
{
    int cnt = 0;
    float step, t = 0.0;

    t = step = get_step(SLICE, period);

    while(cnt++ < 36)
    {
        printf("%.1fcos(%f * %.8f) = %f\n", amplitude, ang_vel,
                t, amplitude * cos(ang_vel * t));
        t += step;
    }
}

void sin_sim(float amplitude, float ang_vel, float period)
{
    int cnt = 0;
    float step, t = 0.0;

    t = step = get_step(SLICE, period);

    while(cnt++ < 36)
    {
        printf("%.1fsin(%f * %.8f) = %f\n", amplitude, ang_vel,
                t, amplitude * sin(ang_vel * t));
        t += step;
    }
}
```

```
void draw_spectrum(void)
{
    float t, step = 0.0;
    float period, freq = 100.0;

    c y[8] = {0};
    int ix;
    int ju;
    int iy;
    int i;
    double x[8] = {0};

    int tst;
    double temp_re;
    double temp_im;
    int iheight;
    int istart;
    int j;
    double twid_re;
    double dv0[5] = {0};

    double twid_im;
    double dv1[5] = {0};

    int ihi;

    calc_period(&freq, &period);
    step = get_step(SLICE, period);

    if(t > period)
        t = 0.0;

    i = 0;
    t = 0.0;
    for(; i < SLICE; t += step)
    {
        //if(t > 3 * period)
        if(t > period)
        {
            break;
            t = 0.0;
        }

        x[i] = 10 * sin(200 * M_PI * t);
        printf("x[%d] = %f\n", i++, x[i]);

    }

    t = 0;
    step = 2 * M_PI / SLICE;
```

```
for(i = 0; i < 5; i++)
{
    dv0[i] = cos(t);
    dv1[i] = -sin(t);
    t += step;
}

t = 0;

ix = 0;
ju = 0;
iy = 0;
printf("Before Reverse Order\n");
for (i = 0; i < 7; i++) {
    y[iy].re = x[ix];
    y[iy].im = 0.0;

    printf("y[%d].re = %lf\t", iy, y[iy].re);
    printf("y[%d].im = %lf\n", iy, y[iy].im);

    iy = 8;
    tst = 1;
    while (tst) {
        iy >>= 1;
        ju ^= iy;
        tst = ((ju & iy) == 0);
    }

    iy = ju;
    ix++;
}
y[iy].re = x[ix];
y[iy].im = 0.0;

printf("\nAfter Reverse Order\n");
for(i = 0; i < SLICE; i++)
{
    printf("y[%d].re = %lf\t", i, y[i].re);
    printf("y[%d].im = %lf\n", i, y[i].im);
}
printf("\nN-2 First Butterfly\n");
```

1) FOURIER TRANSFORM으로 FFT 구현하기 3

CODE

```
for (i = 0; i <= 7; i += 2) {
    temp_re = y[i + 1].re;
    temp_im = y[i + 1].im;
    y[i + 1].re = y[i].re - y[i + 1].re;
    y[i + 1].im = y[i].im - y[i + 1].im;
    y[i].re += temp_re;
    y[i].im += temp_im;

    printf("y[%d].re = %lf\t", i, y[i].re);
    printf("y[%d].im = %lf\n", i, y[i].im);
    printf("y[%d].re = %lf\t", i+1, y[i+1].re);
    printf("y[%d].im = %lf\n", i+1, y[i+1].im);
}

iy = 2;
ix = 4;
ju = 2;
iheight = 5;
while (ju > 0) {
    // 0 ~ 4
    printf("\nN-%d Butterfly(처음은 짝수 오더)\n", glob);
    for (i = 0; i < iheight; i += ix) {
        temp_re = y[i + iy].re;
        temp_im = y[i + iy].im;
        y[i + iy].re = y[i].re - temp_re;
        y[i + iy].im = y[i].im - temp_im;
        y[i].re += temp_re;
        y[i].im += temp_im;

        printf("y[%d].re = %lf\t", i, y[i].re);
        printf("y[%d].im = %lf\n", i, y[i].im);
        printf("y[%d].re = %lf\t", i+iy, y[i+iy].re);
        printf("y[%d].im = %lf\n", i+iy, y[i+iy].im);
    }
    printf("\n");

    istart = 1;
    printf("\nN-%d Butterfly(처음은 홀수 오더)\n", glob);
    for (j = ju; j < 4; j += ju) {
        printf("twid_re = dv0 =  cos(2 * pi * f * t / fftN의절반)\n");
        printf("twid_im = dv1 = -sin(2 * pi * f * t / fftN의절반)\n");
        twid_re = dv0[j];
        twid_im = dv1[j];
        i = istart;
        ihi = istart + iheight;
        while (i < ihi) {
            temp_re = twid_re * y[i + iy].re - twid_im * y[i + iy].im;
            temp_im = twid_re * y[i + iy].im + twid_im * y[i + iy].re;
            y[i + iy].re = y[i].re - temp_re;
            y[i + iy].im = y[i].im - temp_im;
            y[i].re += temp_re;
            y[i].im += temp_im;

            printf("y[%d].re = %lf\t", i, y[i].re / HALF_SLICE);
            printf("y[%d].im = %lf\n", i, y[i].im / HALF_SLICE);
            printf("y[%d].re = %lf\t", i+iy, y[i+iy].re / HALF_SLICE);
            printf("y[%d].im = %lf\n", i+iy, y[i+iy].im / HALF_SLICE);

            i += ix;
        }

        istart++;
    }

    ju /= 2;
    iy = ix;
    ix <<= 1;
    iheight -= iy;

    if(ju > 0)
    {
        count++;
        printf("\nFinished N-%d Butterfly\nNow Starting N-%d Butterfly", glob, glob *= 2);
    }
    else
        printf("\nFinished N-%d Butterfly\n", glob);
}

//printf("OK");

for(i = 0; i < SLICE; i++)
{
    glBegin(GL_LINE_STRIP);

    if(y[i].re == 0 && y[i].im == 0)
        continue;

    glVertex2f(i * 10, y[i].re / HALF_SLICE);
    glVertex2f(i * 10, 0);

    glEnd();

    glBegin(GL_LINE_STRIP);

    glVertex2f(i * 10, y[i].im / HALF_SLICE);
    glVertex2f(i * 10, 0);

    glEnd();
}
```

```
void draw_omega_sin(void)
{
    float amp, angle, period, freq, rad, omega, t, step = 0.0;
    float radius = 3.0;
    float x = 0, x2 = 0, y2, cx, cy;
    float tmp;
    int cnt = 0, cache = 0;

    srand(time(NULL));

    amp = 10;
    angle = 45.0;
    freq = 100.0;

    angle2radian(&angle, &rad);
    calc_period(&freq, &period);
    calc_angular_velocity(&freq, &omega);
    t = step = get_step(SLICE, period);

    printf("step = %f\n", step);

    if(t > period)
        t = 0.0;

    //glLineStipple(1, 0xFFEE);
    //glEnable(GL_LINE_STIPPLE);
    //glBegin(GL_POINTS);
    for(; ; t += step)
    {
        if(t > period)
            //if(t > 3 * period)
        {
            break;
            t = 0.0;
        }

        y2 = amp * sin(omega * t);

        if(cache && !(cnt % 16))
        {
            glBegin(GL_POINTS);
            glVertex2f(cx * 6000, cy * 6);
            glVertex2f(t * 6000, y2 * 6);
            glEnd();
```

1) FOURIER TRANSFORM으로 FFT 구현하기 4

CODE

```
            glBegin(GL_LINE_STRIP);
            //glVertex2f(cx * 4000, cy * 2);
            //glVertex2f(cx * 4000, 0);
            glVertex2f(t * 6000, y2 * 6);
            glVertex2f(t * 6000, 0);
            glEnd();
        }

        cache = 1;
        cx = t;
        cy = y2;
        cnt++;
    }
    //glEnd();
    //glDisable(GL_LINE_STIPPLE);
}

int main(int argc, char **argv)
{
    float amplitude, angle, period, frequency, radian, angular_velocity;
    float step = 0.0;

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE);
    glutInitWindowSize(1200, 800);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("Digital Signal Processing");

    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMainLoop();

    return 0;
}
```

1) FOURIER TRANSFORM으로 FFT 구현하기 5

FFT - SIMULATION