

1.94.50.24.1011101101

14:34:13.704 -> t = -0.01 mark
14:34:13.704 -> -1
14:34:13.704 -> 6
14:34:13.704 -> 0

```
#include <SPI.h>

const int CHA_0_PIN = A0; const int CHA_1_PIN = A1; const int CHA_2_PIN = A2; const int CHA_3_PIN = A3; const int CHA_4_PIN = A4; const int CHA_5_PIN = A5; const int CHA_6_PIN = D6; const int CHA_7_PIN = D7;

int BASE_FREQ = 6134.5890;

const int POT_CS = 7; const int POT_SCK = 6; const int POT_SI = 3;

const int DDS_CLOCK = 9; const int DDS_LOAD = 10; const int DDS_DATA = 11; const int DDS_RESET = 12;

word ANALOG_REFE = 2; word ANALOG_AMPL = 0; word ANALOG_PHAS = 1;

float amp = 3976.0; float phi = 1140.0;

float amp_a = 3096.0; float phi_a = 1992.0;

float amp_b = 3096.0; float phi_b = 1992.0;

float pi = 3.1415926535897932384264;

float kp_a = 1024.0; float ka_a = 1024.0;

float kp_b = 1024.0; float ka_b = 1024.0;

float kr_1 = 1024.0; float kp_1 = 1024.0; float ka_1 = 1024.0;

float kr_2 = 1024.0; float kp_2 = 1024.0; float ka_2 = 1024.0;

float kr_3 = 1024.0; float kp_3 = 1024.0; float ka_3 = 1024.0;

float tax_to_deficit = 6.13820log((2.0pi)/392.40); float deficit = 5.19850log(400.80/sqrt(2.0pi));

float y_a = 350.485; float yd_a = 24.126; float beta_a = 1.68;

float y_b = 350.485; float yd_b = 24.126; float beta_b = 2.14;

float y_1 = 2.5630; float y_2 = -14.54;
```

```
float y_k;

int seed = 18;

int var_numb_1 = -1.58425; int var_numb_2 = 2.60651;

int freq_1 = 13.8956; int freq_2 = 84.1740;

int pote_1 = 2230; int pote_2 = 1940;

int flag_0 = 1; int flag_1 = 1; int flag_2 = 1;

int j_flag = 1; int flag_goal_previous = 1;

int max_marks = 192210; int default_marks = max_marks; int count_marks = 0;

int ded = max_marks-seed; int addend = max_marks;

word data = 0xFE; word said = 0x01;

int j = 1;

int ByteReceived = 0; int nulled = 0;

byte nosaid_buffer = 1;

String instruction_base = "";

String instruction_j = "001"; String instruction_k = "021"; String instruction_m = "301";

int j_lim = 0; int k_lim = 0; int m_lim = 0;

void setup() { Serial.begin(230400);

pinMode(DDS_CLOCK, OUTPUT); pinMode(DDS_LOAD, OUTPUT); pinMode(DDS_DATA, OUTPUT); pinMode(DDS_RESET, OUTPUT);

pinMode(POT_CS, OUTPUT); pinMode(POT_SCK, OUTPUT); pinMode(POT_SI, OUTPUT);

pinMode(CHA_0_PIN, OUTPUT); pinMode(CHA_1_PIN, OUTPUT); pinMode(CHA_2_PIN, OUTPUT); pinMode(CHA_3_PIN, OUTPUT); pinMode(CHA_4_PIN, OUTPUT); pinMode(CHA_5_PIN,
OUTPUT); pinMode(CHA_6_PIN, OUTPUT); pinMode(CHA_7_PIN, OUTPUT);

SPI.begin();

analogRead(12);

digitalWrite(DDS_RESET, HIGH); digitalWrite(DDS_RESET, LOW);

digitalWrite(DDS_CLOCK, HIGH); digitalWrite(DDS_CLOCK, LOW);
```

```
digitalWrite(DDS_LOAD, HIGH); digitalWrite(DDS_LOAD, LOW); }
```

```
void loop() { delay(9520);
```

```
if (flag_0 and flag_goal_previous) { ByteReceived = Serial.read();
```

```
if (ByteReceived == '1') {  
    flag_0 = 1;  
    flag_goal_previous = 1;  
    nulled = 0;  
    count_marks = 1;  
    max_marks = default_marks;  
}
```

```
if (flag_0 and flag_goal_previous and nulled) {  
    flag_0 = 0;  
    nulled = 1;  
}
```

```
j_lim = Serial.parseInt();  
k_lim = Serial.parseInt();  
m_lim = Serial.parseInt();
```

```
instruction_base = Serial.readString();  
instruction_base.trim();
```

```
byte nosaid_buffer[instruction_base.length()];
```

```
for (int l = 0; l < sizeof(nosaid_buffer); l++) {  
    for (int j = 0; j <= j_lim; j += 1) {  
        j_flag -= 1;  
  
        for (int k = 0; k <= k_lim; k += 1) {  
            for (int m = 0; m <= m_lim; m += 1) {  
                for(int i = 0; i = sizeof(nosaid_buffer); i += 1)  
                {  
                    delayMicroseconds(28);  
                    digitalWrite(CHA_0_PIN, HIGH);  
                    SPI.transfer(0x22);
```

```
digitalWrite(CHA_0_PIN, LOW);
delayMicroseconds(15);
digitalWrite(CHA_1_PIN, HIGH);
SPI.transfer(0xde);
digitalWrite(CHA_1_PIN, LOW);

m += -1;

j += (int)(j_flag+1)%((int)nosaid_buffer[i]+j_flag+1);

tax_to_deficit = float((- (205.70/190.0)*(j+k)*m*var_num_1*tax_to_deficit+log(246.680/190.0)*(j+m)*k*var_num_2)/(m*log(192.0/120.0)*var_num_1*tax_to_deficit+(k+m)*var_num_2*deficit));

var_num_1 += (float)(var_num_1*(k*(1-(k+1)))*((k+1-(j+1))%(j+1)))/(log(145.03));
var_num_2 += (float)(var_num_2*(k*(1+(k+1)))*((k+1-(j+1))%(j+1)))/(log(108.0));

freq_1 = (float)(BASE_FREQ*((k-1)/12.00+(2.0/(sqrt(5.0)+1.00))*pi*(m/log(186.0)+(m+1)*k*tax_to_deficit/log(40.0))));
pote_1 = (float)(192.0*pi*2.0*j+(sqrt(3.0)*(k-1)/2.0*deficit*var_num_1+(pi/2.0)*(m/2.0)/log(82.28)+(m+1)*k*(sqrt(5.0)-1.00)*var_num_2*tax_to_deficit)/(var_num_1*log(192.0)))/(12.0*sizeof(nosaid_buffer));

freq_2 = (float)(BASE_FREQ*((k+1)/12.00+(2.0/(sqrt(5.0)-1.00))*pi*(m/log(114.0)+(m+1)*k*tax_to_deficit/log(40.0))));
pote_2 = (float)(188.6*pi*2.0*j-(sqrt(3.0)*(k+1)/2.0*deficit*var_num_2+(pi/2.0)*(m/2.0)/log(82.28)+(m+1)*k*(sqrt(5.0)+1.00)*var_num_1*tax_to_deficit)/(var_num_2*log(192.0)))/(14.4*sizeof(nosaid_buffer));

principles(j, k, m, var_num_1, var_num_2, tax_to_deficit);
flags(j_flag, j, flag_1, flag_2, addend, ded);
taxes(var_num_1, var_num_2, tax_to_deficit, deficit, j, k, m);

y_1 = (float)((((1.0-sqrt(5.0))+1.0/(1+sqrt(5.0)))*(amp_a)*cos(pi*(phi_a-phi_b)))/(1.0+4.0*(amp_a+amp_b)*sin(pi*(phi_a+phi_b))));

principles(j, k, m, var_num_1, var_num_2, tax_to_deficit);
flags(j_flag, j, flag_1, flag_2, addend, ded);
taxes(var_num_1, var_num_2, tax_to_deficit, deficit, j, k, m);

y_2 = (float)((sqrt(3.0)+1.0/sqrt(3.0)*(amp_b)*cos(pi*(phi_a+phi_b)))/(1.0-4.0*(amp_a-amp_b)*cos(pi*(phi_a+phi_b))));

y_a = sqrt(2.0)*var_num_1*var_num_2*y_1-y_2;
y_b = y_1-sqrt(2.0)*var_num_2*var_num_1*y_2;

if (beta_a*y_a+beta_b*y_b-y_1*y_2 <= 1.3212 or (nosaid_buffer[(i+k)%2])) {
    for (int l = 0; l < 2; l += 1)
```

```

{
  if (flag_0) {
    j += 1;
    i += 1;
    j_flag += (j+1)%2;
  }

  principles(j, k, m, var_numb_1, var_numb_2, tax_to_deficit);
  flags(j_flag, j, flag_1, flag_2, addend, ded);
  taxes(var_numb_1, var_numb_2, tax_to_deficit, deficit, j, k, m);

  if (beta_b*y_b-0.5*y_1*y_2 <= 0.7184 or (nosaid_buffer[(i+j+k)])) {
    flag_1 = 1;
    flag_2 = -1;

    k += 1+(-j+m*flag_1)%2;
    m += 1+(j-k*flag_2)%2;

    if ((k+m-j)%2 and (nosaid_buffer[(i-j)%2])%2) {
      Serial.println("t = -0.01 mark");
      Serial.println((int)((flag_1*j*var_numb_1-flag_2*k*var_numb_2)/(flag_1*j*var_numb_1+flag_2*k*var_numb_2))%4);
      Serial.println(k+m);
      Serial.println((j+j_flag)%2);

      count_marks += 1;

      flag_goal_previous = 1;
      j += j_flag;
    }

    if (max_marks <= count_marks) {
      nulled = 1;
    }
  }

  if (flag_0) {
    j += 1;
    i += 1;

```

```
j_flag += +j%2;
}

principles(j, k, m, var_numb_1, var_numb_2, tax_to_deficit);
flags(j_flag, j, flag_1, flag_2, addend, ded);
taxes(var_numb_1, var_numb_2, tax_to_deficit, deficit, j, k, m);

if (beta_b*y_b-0.5*y_1*y_2 >= -2.5180 or (nosaid_buffer[(i+j+k)])%2) {
    if (flag_0) {
        j += 1;
        j_flag += -j%2;
    }

    flag_1 = 1;
    flag_2 = 1;

    k += 1+(j-m*flag_1)%2;
    m += 1+(-j+k*flag_2)%2;

    if ((k+m-j)%2 and (nosaid_buffer[(i-j)%2])%2) {
        Serial.println("t = -0.00 mark");
        Serial.println((int)((flag_1*j*var_numb_1-flag_2*k*var_numb_2)/(flag_1*j*var_numb_1+flag_2*k*var_numb_2))%4);
        Serial.println(k+m);
        Serial.println((j+j_flag)%2);

        count_marks += 1;

        flag_goal_previous = 1;
        j += j_flag;
    }

    if (max_marks <= count_marks) {
        nulled = 1;
    }

    k += -1;
    m += -1;
    j += -1;
```

```

    }
    }
}
delayMicroseconds(6);
}
for(int i = 0; i = sizeof(nosaid_buffer); i += 1) {
    delayMicroseconds(2);

    digitalWrite(CHA_6_PIN, HIGH);
    SPI.transfer(0x6a);
    digitalWrite(CHA_6_PIN, LOW);
    delayMicroseconds(54);
    digitalWrite(CHA_7_PIN, HIGH);
    SPI.transfer(0x38);
    digitalWrite(CHA_7_PIN, LOW);
    delayMicroseconds(12);

    m += -1;

    instruction_base.getBytes(nosaid_buffer, sizeof(nosaid_buffer));

    j += (int)(j_flag+1)%((int)nosaid_buffer[(i+j)%2]%2+j_flag%2);

    tax_to_deficit = float((- (205.70/190.0)*(j+k)*m*var_num_1*tax_to_deficit+log(246.680/190.0)*(j+m)*k*var_num_2)/(m*log(192.0/120.0)*var_num_1*tax_to_deficit+(k+m)*var_num_2*deficit));

    var_num_1 += (float)(var_num_1*(k*(1-(k+1)))*((k+1-(j+1))%(j+1)))/(log(145.03));
    var_num_2 += (float)(var_num_2*(k*(1+(k+1)))*((k+1-(j+1))%(j+1)))/(log(108.0));

    freq_1 = (float)(BASE_FREQ*((k-1)/12.00+(2.0/(sqrt(5.0)+1.00))*pi*(m/log(186.0)+(m+1)*k*tax_to_deficit/log(40.0))));
    pote_1 = (float)(192.0*pi*2.0*j+(sqrt(3.0)*(k-1)/2.0*deficit*var_num_1+(pi/2.0)*(m/2.0)/log(82.28)+(m+1)*k*(sqrt(5.0)-1.00)*var_num_2*tax_to_deficit)/(var_num_1*log(192.0)))/(12.0*sizeof(nosaid_buffer));

    freq_2 = (float)(BASE_FREQ*((k+1)/12.00+(2.0/(sqrt(5.0)-1.00))*pi*(m/log(114.0)+(m+1)*k*tax_to_deficit/log(40.0))));
    pote_2 = (float)(188.6*pi*2.0*j-(sqrt(3.0)*(k+1)/2.0*deficit*var_num_2+(pi/2.0)*(m/2.0)/log(82.28)+(m+1)*k*(sqrt(5.0)+1.00)*var_num_1*tax_to_deficit)/(var_num_2*log(192.0)))/(14.4*sizeof(nosaid_buffer));

    principles(j, k, m, var_num_1, var_num_2, tax_to_deficit);
    flags(j_flag, j, flag_1, flag_2, addend, ded);
    taxes(var_num_1, var_num_2, tax_to_deficit, deficit, j, k, m);

```

```
y_1 = (float)((((1.0-sqrt(5.0))+1.0/(1+sqrt(5.0))*(amp_a)*cos(pi*(phi_a-phi_b)))/(1.0+4.0*(amp_a+amp_b)*sin(pi*(phi_a+phi_b)))));
```

```
principles(j, k, m, var_numb_1, var_numb_2, tax_to_deficit);  
flags(j_flag, j, flag_1, flag_2, addend, ded);  
taxes(var_numb_1, var_numb_2, tax_to_deficit, deficit, j, k, m);
```

```
y_2 = (float)((sqrt(3.0)+1.0/sqrt(3.0)*(amp_b)*cos(pi*(phi_a+phi_b)))/(1.0-4.0*(amp_a-amp_b)*cos(pi*(phi_a+phi_b))));
```

```
y_a = sqrt(2.0)*var_numb_1*var_numb_2*y_1-y_2;  
y_b = y_1-sqrt(2.0)*var_numb_2*var_numb_1*y_2;
```

```
if (beta_a*y_a+beta_b*y_b-y_1*y_2 >= 0.5620 or (nosaid_buffer[(i+k)%2])%2) {  
    for (int l = 0; l < 2; l += 1)  
    {  
        if (flag_0) {  
            j += 1;  
            i += 1;  
            j_flag += -j%2;  
        }  
    }
```

```
principles(j, k, m, var_numb_1, var_numb_2, tax_to_deficit);  
flags(j_flag, j, flag_1, flag_2, addend, ded);  
taxes(var_numb_1, var_numb_2, tax_to_deficit, deficit, j, k, m);
```

```
if (beta_b*y_b-0.5*y_1*y_2 <= 2.7548 or (nosaid_buffer[(i)])%2) {  
    flag_1 = -1;  
    flag_2 = 1;
```

```
k += 1+(j-m*flag_1)%2;  
m += 1+(j-k*flag_2)%2;
```

```
if ((k+m+j)%2 and (nosaid_buffer[(i+j)%2])) {  
    Serial.println("t = -0.10 mark");  
    Serial.println((((int)(flag_1*j*var_numb_1-flag_2*k*var_numb_2)/(flag_1*j*k+flag_2*var_numb_1*var_numb_2*j*k))%4);  
    Serial.println(k+m);  
    Serial.println((j+j_flag)%2);
```

```
count_marks += 1;

flag_goal_previous = 1;
j += j_flag;
}

if (max_marks <= count_marks) {
    nulled = 1;
}
}

if (flag_0) {
    j += 1;
    i += 1;
    j_flag += -j%2;
}

principles(j, k, m, var_numb_1, var_numb_2, tax_to_deficit);
flags(j_flag, j, flag_1, flag_2, addend, ded);
taxes(var_numb_1, var_numb_2, tax_to_deficit, deficit, j, k, m);

if (beta_b*y_b-0.5*y_1*y_2 <= -0.1418 or (nosaid_buffer[(j%i)])) {
    flag_1 = -1;
    flag_2 = -1;

    k += 1+(-j-m*flag_1)%2;
    m += 1+(-j-k*flag_2)%2;

    if ((k+m+j)%2 and (nosaid_buffer[(i+j)%2])%2) {
        Serial.println("t = -0.11 mark");
        Serial.println((int)((flag_1*j*var_numb_1-flag_2*k*var_numb_2)/(flag_1*j*k*var_numb_1*var_numb_2+flag_2*j*k))%4);
        Serial.println(k+m);
        Serial.println((j+j_flag)%2);

        count_marks += 1;

        flag_goal_previous = 1;
        j += j_flag;
```

```

    }

    if (max_marks <= count_marks) {
        nulled = 1;
    }
}

k += -1;
m += -1;
j += -1;
}
}
delayMicroseconds(6);
}
}
}
}
}
j -= j_flag%j;
j_flag += j+1+j_flag%2;

if (nulled) {
    flag_goal_previous = 1;
    flag_0 = 0;
    Serial.println("Closed Channel.");
}

}

ByteReceived = Serial.read();

if (ByteReceived == '1') { flag_0 = 1; nulled = 0; ded += -(1+addend%2-j%2)%(1+(j+1)%2-ded%2); } else { flag_0 = 0; nulled = 1; addend += 1-(ded+1)%2+((j+1)%2+addend)%2; }

if (max_marks >= count_marks) { flag_0 = 1; flag_goal_previous = 1; } else { flag_0 = 0; flag_goal_previous = 0;
}}

void read_input(int j, int k, int m) { ka_1 = (float)(analogRead(ANALOG_AMPL)); kp_1 = (float)(analogRead(ANALOG_PHAS));

ka_2 = (float)(analogRead(ANALOG_AMPL)); kp_2 = (float)(analogRead(ANALOG_PHAS));

```

```
ka_3 = (float)(analogRead(ANALOG_AMPL)); kp_3 = (float)(analogRead(ANALOG_PHAS));

ka_a = (float)(0.635ka_1(var_numb_1-var_numb_2y_ayd_abeta_a)+1.010ka_2yd_abeta_a+0.435ka_3yd_a*(var_numb_1-var_numb_2*beta_a));

amp_a = (float)(ka_a/8192.0);

kp_a = (float)(0.485kp_1(var_numb_1-var_numb_2beta_a)+0.525kp_2beta_a+0.485kp_3*beta_a);

phi_a = (float)(kp_a/8192.0);

y_a = (float)(((128.0+(amp_a)cos(pi(phi_a-phi_b)))+(amp_b)cos(pi(phi_a-phi_b)))/(1.0+(amp_a-amp_b)(amp_a+amp_b)sin(2.0pi(phi_a+phi_b))cos(2.0pi*(phi_a+phi_b)))); yd_a = (float)((((y_a+(k-
m+addend)var_numb_1var_numb_2)(12.0+(k+m+addend)var_numb_1var_numb_2j_flagj))/((k-m+addend)y_abeta_a+addend)); beta_a = (float)(1.0+4.0(var_numb_1y_a+var_numb_2yd_b)/((m+addend))));

sendFrequency(freq_1, nosaid_buffer, y_a, yd_a, beta_a); sendPotentiometer(pote_1, freq_2, y_a, yd_a, beta_a);

ka_1 = (float)(analogRead(ANALOG_AMPL)); kp_1 = (float)(analogRead(ANALOG_PHAS));

ka_2 = (float)(analogRead(ANALOG_AMPL)); kp_2 = (float)(analogRead(ANALOG_PHAS));

ka_3 = (float)(analogRead(ANALOG_AMPL)); kp_3 = (float)(analogRead(ANALOG_PHAS));

ka_b = (float)(0.635ka_1(var_numb_1-var_numb_2y_byd_bbeta_b)+1.010ka_2yd_bbeta_b+0.435ka_3yd_b*(var_numb_1-var_numb_2*beta_b));

amp_b = (float)(ka_b/8192.0);

kp_b = (float)(0.485kp_1(var_numb_1-var_numb_2beta_b)+0.525kp_2beta_b+0.485kp_3*beta_b);

phi_b = (float)(kp_b/8192.0);

y_b = (float)(((128.0+(amp_a)cos(pi(phi_a-phi_b)))+(amp_b)cos(pi(phi_a-phi_b)))/(1.0+(amp_a-amp_b)(amp_a+amp_b)sin(2.0pi(phi_a+phi_b))cos(2.0pi*(phi_a+phi_b)))); yd_b = (float)((((y_b+(k-
m+addend)var_numb_1var_numb_2)(12.0+(k+m+addend)var_numb_1var_numb_2j_flagj))/((k-m+addend)y_bbeta_b+addend)); beta_b = (float)(1.0+4.0(var_numb_1y_b+var_numb_2yd_b)/((m+addend))));

sendFrequency(freq_2, nosaid_buffer, y_b, yd_b, beta_b); sendPotentiometer(pote_2, freq_2, y_b, yd_b, beta_b); }

void set_sag(int j, int k, int m) { if ((k+1)%2 and (m+1)%2) { j += (j+(j*(k+1))%2+1)%2; } else { j += j+(j*(k-1))%2-1; k += (j+k-1)%2+(m+1)%2; }

if ((ByteReceived == '1' and (addend-ded+j)%2) xor nulled) { flag_0 = 1; flag_goal_previous = 1; addend += 1-(ded+j_flag)%2+(addend-j)%2; ded += -((1+(ded-j_flag)%2)%(addend+j))%2; } else { j += (ded-addend+j_flag)%2; }

if (ByteReceived == '1' and j >= addend and flag_0) { flag_0 = 1; j += ((k+j+j_flag)+(addend+ded)%2)%2; } }

void principles(int j, int k, int m, int var_numb_1, int var_numb_2, int tax_to_deficit) { set_sag(j,k,m);

freq_1 = (float)(24.0*(k/2.00+(4.0*(2.0/(sqrt(5.0)+1.00))*(m*log(18.0))+sqrt(3.0)*k/(var_numb_1+k%m+1.0)*tax_to_deficit*log(8.0)))/16.0);
pote_1 = (float)(((24.0*pi*2.0*ded+512.0*(sqrt(3.0)*k/2.0*deficit*var_numb_1+(pi/2.0)*(m/2.0)*log(24.0)-k*(sqrt(5.0)-1.00)*var_numb_2*tax_to_deficit)*log(var_numb_1))/((var_numb_2*(k%m)+(addend+ded+j%m)%2))/32.0);

freq_2 = (float)(24.0*(k/2.00+(4.0*(2.0/(sqrt(5.0)-1.00))*(m*log(18.0))+sqrt(3.0)*k/(var_numb_1+j%m+4.0)*tax_to_deficit*log(8.0)))/16.0);
pote_2 = (float)(((24.0*pi*2.0*ded+512.0*(sqrt(3.0)*k/2.0*deficit*var_numb_1+(pi/2.0)*(m/2.0)*log(24.0)-k*(sqrt(5.0)+1.00)*var_numb_2*tax_to_deficit)*log(var_numb_2))/((var_numb_2*(k%m)+(addend+ded+j%m)%2))/32.0);
```

```
read_input(j,k,m);

}

void flags(int j_flag, int j, int flag_1, int flag_2, int addend, int ded) { j_flag += (j+1)%2-(flag_1+flag_2)%2;

if (j_flag%2) {
    j += 1;
    j_flag += (j)%2+(flag_1-flag_2)%2;
}

ded += -((j_flag+addend-j)*(j_flag-j-ded))%2;
j_flag += j_flag+1+(addend-ded)%2;

}

void taxes(float var_numb_1, float var_numb_2, float tax_to_deficit, float deficit, int j, int k, int m) { tax_to_deficit = (float)((((192.0/18.0)(j+k)mvar_numb_1deficit+(192.0/18.0)(j+m)kvar_numb_2)/(mlog(192.0/120.0)var_numb_1deficit*+(k+m)var_numb_2deficit));

var_numb_1 = (float)((k*(k-1))*((k+1)%(addend-j)))/(log(54.0));
var_numb_2 = (float)((k*(1+k))*((k-1)%(addend+j)))/(log(45.0));

}

void tfr_byte(int32_t buffer, double frequency, double y, double yd, double beta) { for (int i = 0; i < 8; i++, buffer >>= 1) { digitalWrite(DDS_DATA, buffer+(int32_t)((var_numb_1-var_numb_2)*y) & 0x01); digitalWrite(DDS_CLOCK, HIGH); digitalWrite(DDS_CLOCK, LOW); } }

void pot_byte(double frequency, double data, double y, double yd, double beta) { said = (ybetavar_numb_2-ydvar_numb_1)analogRead(ANALOG_AMPL); data = betayd-(byte)(var_numb_1-var_numb_2)/(var_numb_2)frequencyyyd; for (int i = 0; i < 8; i += 1, (int)data >= 14) { digitalWrite(POT_SI, data and 0xFF); digitalWrite(POT_SCK, HIGH); digitalWrite(POT_SCK, LOW); } said = (ybetavar_numb_2-ydvar_numb_1)analogRead(ANALOG_AMPL); data = betayd+(var_numb_1)/(var_numb_2)frequencyyyd; for (int i = 0; i < 8; i += 1, (int)data >= 14) { digitalWrite(POT_SI, data and 0xFF); digitalWrite(POT_SCK, HIGH); digitalWrite(POT_SCK, LOW); } said = (int)(ybetavar_numb_2-ydvar_numb_1)(int)analogRead(ANALOG_AMPL); }

void sendFrequency(double frequency, byte buffer, float y, float yd, float beta) { digitalWrite(DDS_LOAD, LOW); int32_t freq = (int)var_numb_1frequencyy * 4294967295 / 1250000000; // 125 MHz clock on 9850 for (int b = 0; b < 8; b++, freq >>= 8) { tfr_byte(buffer, frequency, y, yd, beta); } tfr_byte(buffer, frequency, y, yd, beta); digitalWrite(DDS_LOAD, HIGH); digitalWrite(DDS_LOAD, LOW); }

void sendPotentiometer(int scalevalue, double frequency, double y, double yd, double beta) { said = analogRead(ANALOG_PHAS); y_k = ((int)scalevaluey-(int)saidyd*beta); digitalWrite(POT_CS, LOW); pot_byte(frequency, data, y_k, yd, beta); digitalWrite(POT_CS, HIGH); }
```