

```
float tax_to_deficit = 3.182530log((2.0pi)/156.40); float deficit = 0.169580log(295.80/sqrt(8.0pi));
//
float y_k = 3590.218600;
float y_1 = 2800.567815; float y_2 = 2310.546700;
float y_b = 328.485; float yd_b = 30.126; float beta_b = 1.14;
float y_a = 292.918; float yd_a = 39.126; float beta_a = 2.68;
float kr_3 = 1024.0; float kp_3 = 1024.0; float ka_3 = 1024.0;
float kr_2 = 1024.0; float kp_2 = 1024.0; float ka_2 = 1024.0;
float kr_1 = 1024.0; float kp_1 = 1024.0; float ka_1 = 1024.0;
float kp_b = 1024.0; float ka_b = 1024.0;
float kp_a = 1024.0; float ka_a = 1024.0;
float pi = 3.1415926535897932386264;
float amp_b = 3096.0; float phi_b = 1992.0;
float amp_a = 3096.0; float phi_a = 1992.0;
float amp = 3976.0; float phi = 1140.0;
word ANALOG_REFE = 2; word ANALOG_AMPL = 0; word ANALOG_PHAS = 1;
const int DDS_CLOCK = 9; const int DDS_LOAD = 10; const int DDS_DATA = 11; const int DDS_RESET = 12;
const int POT_CS = 7; const int POT_SCK = 6; const int POT_SI = 3;
int BASE_FREQ = 150640.15920;
CHA_4_PIN = A4; const int CHA_5_PIN = A5; const int CHA_6_PIN = D6; const int CHA_7_PIN = D7;
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
#include <SPI.h>

21:08:59.220 -> 0
21:08:59.220 -> 0
21:08:59.220 -> 0
21:08:59.220 -> t = -0.11 mark
21:08:59.220 -> 0
21:08:59.220 -> -1
21:08:59.220 -> 0
21:08:59.220 -> t = -0.10 mark
21:08:59.188 -> 0
21:08:59.188 -> 0
21:08:59.188 -> t = -0.01 mark
1.159.62.130.9015.101101010011010010101.92.58.2620
```

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

instruction_base.trim();
instruction_base = Serial.readString();
Serial.parseInt(); m_lim = Serial.parseInt();
if (flag_0 and flag_goal_previous and max_marks >= count_marks) { j_lim = Serial.parseInt(); k_lim =
if (flag_0 and flag_goal_previous and nullled) { flag_0 = 1; nullled = 0; }
if (ByteReceived == '1') { flag_0 = 1; flag_goal_previous = 1; nullled = 0; count_marks = 1; max_marks = default_marks; }
ByteReceived = Serial.read();
void loop() { delay(9520);
digitalWrite(DDS_LOAD, HIGH); digitalWrite(DDS_LOAD, LOW); }
digitalWrite(DDS_CLOCK, HIGH); digitalWrite(DDS_CLOCK, LOW);
digitalWrite(DDS_RESET, HIGH); digitalWrite(DDS_RESET, LOW);
analogRead(12);
SPI.begin();
OUTPUT); pinMode(CHA_6_PIN, OUTPUT); pinMode(CHA_7_PIN, OUTPUT);
OUTPUT); pinMode(CHA_3_PIN, OUTPUT); pinMode(CHA_4_PIN, OUTPUT); pinMode(CHA_5_PIN,
pinMode(CHA_0_PIN, OUTPUT); pinMode(CHA_1_PIN, OUTPUT); pinMode(CHA_2_PIN,
pinMode(POT_CS, OUTPUT); pinMode(POT_SCK, OUTPUT); pinMode(POT_SI, OUTPUT);
OUTPUT); pinMode(DDS_RESET, OUTPUT);
pinMode(DDS_CLOCK, OUTPUT); pinMode(DDS_LOAD, OUTPUT); pinMode(DDS_DATA,
void setup() { Serial.begin(230400);
int max_marks = nosaid_buffer+default_marks+count_marks+seed;
int j_lim = 0; int k_lim = 0; int m_lim = 0;
String instruction_j = "001"; String instruction_k = "021"; String instruction_m = "301";
String instruction_base = "";
int nullled = 0;
byte nosaid_buffer = addend+seed; byte said = 1;
int ByteReceived = 0;
int j = 1; int k = 1; int m = 1;
int ded = default_marks+seed; int addend = ded+default_marks;
int j_flag = 1; int flag_goal_previous = 1;
int flag_0 = 0; int flag_1 = 1; int flag_2 = 1;
//
int pote_1 = 3080.2; int pote_2 = 1620.4;
int freq_1 = 1850.8956; int freq_2 = 4160.3132;
int var_numb_1 = 20.58925; int var_numb_2 = 30.90612;
int seed = 4; int default_marks = 14; int count_marks = 1;
```

```
var _numb_2 += (float)(var _numb_2*(k*(1+(k+1)))*((k+1-(j+1))%(j+1)))/(log(192.60)*8192.0);
var _numb_1 += (float)(var _numb_1*(k*(1-(k+1)))*((k+1-(j+1))%(j+1)))/(log(145.30)*8192.0);
```

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

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

```
m += -1;
```

```
delayMicroseconds(12);
```

```
digitalWrite(CHA_7_PIN, LOW);
```

```
SPI.transfer(0xc6);
```

```
digitalWrite(CHA_7_PIN, HIGH);
```

```
delayMicroseconds(16);
```

```
digitalWrite(CHA_6_PIN, LOW);
```

```
SPI.transfer(0x2b);
```

```
digitalWrite(CHA_6_PIN, HIGH);
```

```
for(int i = 0; i = sizeof(nosaid_buffer); i += 1) {
```

```
    } else {
```

```
        Serial.println("Closed Channel");
```

```
        ByteReceived = 'B';
```

```
        flag_0 = 0;
```

```
        flag_goal_previous = 0;
```

```
        if (count_marks >= max_marks and nulled and flag_0) {
```

```
            m = addend+ded-m_lim-m;
```

```
            if (m <= m_lim and flag_0 and count_marks <= max_marks) {
```

```
                k = addend+k_lim-k;
```

```
                if (k <= k_lim and flag_0 and count_marks <= max_marks) {
```

```
                    j_flag -= 1;
```

```
                    j += 1;
```

```
                    if (j <= j_lim and flag_0 and count_marks <= max_marks) {
```

```
                        for (int l = 0; l < sizeof(nosaid_buffer); l++) {
```

```
                            if (flag_0) {
```

```
                                delayMicroseconds(90);
```

```
                                digitalWrite(CHA_7_PIN, LOW);
```

```
                                SPI.transfer(0xa7);
```

```
                                digitalWrite(CHA_7_PIN, HIGH);
```

```
                                delayMicroseconds(16);
```

```
                                digitalWrite(CHA_6_PIN, LOW);
```

```
                                SPI.transfer(0xef);
```

```
                                digitalWrite(CHA_6_PIN, HIGH);
```

```
                                y_b = y_1-sqrt(2.0)*sqrt(var _numb_1*var _numb_2)*y_2;
```

```
                                y_a = sqrt(2.0)*sqrt(var _numb_1*var _numb_2)*y_1+y_2;
```

```
(1.5890);
```

```
                                y_2 = (float)((sqrt(6.0)-1.0/sqrt(8.0)*(amp_a)*cos(pi*(phi_a-phi_b)))/(1.0-0.5*(amp_a-amp_b)*cos(pi*(phi_a-phi_b)))/
```

```
                                taxes(var _numb_1, var _numb_2, tax_to_deficit, deficit, j, k, m);
```

```
                                flags(j_flag, j, flag_1, flag_2, addend, ded);
```

```
                                principles(i, j, k, m, var _numb_1, var _numb_2, tax_to_deficit);
```

```
                                amp_b)*sin(pi*(phi_a+phi_b)))/(2.1465);
```

```
                                y_1 = (float)((1.0+sqrt(5.0)+1.0/(1.0-sqrt(5.0))*(amp_b)*cos(pi*(phi_a+phi_b)))/(1.0+0.5*(amp_a-
```

```
                                taxes(var _numb_1, var _numb_2, tax_to_deficit, deficit, j, k, m);
```

```
                                flags(j_flag, j, flag_1, flag_2, addend, ded);
```

```
                                principles(i, j, k, m, var _numb_1, var _numb_2, tax_to_deficit);
```

```
                                (m+1)*k*(sqrt(5.0)+1.00)*var _numb_1*tax_to_deficit/(var _numb_2*log(192.0))/(14.4*sizeof(nosaid_buffer));
```

```
                                pote_2 = (float)(188.6*pi*2.0*j-(sqrt(3.0)*(k+1)/2.0*deficit*var _numb_2+(pi/2.0)*(m/2.0)/log(82.28)+
```

```
                                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)));
```

```
                                1.00)*var _numb_2*tax_to_deficit/(var _numb_1*log(192.0))/(12.0*sizeof(nosaid_buffer));
```

```
                                pote_1 = (float)(192.0*pi*2.0*j+(sqrt(3.0)*(k-1)/2.0*deficit*var _numb_1+(pi/2.0)*(m/2.0)/log(82.28)+(m+1)*k*(sqrt(5.0)-
```

```
                                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)));
```

```
                                ((m+k)*log(192.0/146.0)*var _numb_1*tax_to_deficit+(k-m)*var _numb_2*deficit);
```

```
                                tax_to_deficit = float(((205.70/290.0)*(j+k)*m*var _numb_1*tax_to_deficit+log(246.680/190.0)*(j+m)*k*var _numb_2)/
```

```
    }
    j += j_flag;

    }
    flag_goal_previous = 1;
    } else {
    flag_goal_previous = 0;
    flag_0 = 0;

    if (count_marks >= max_marks) {

    count_marks += 1;

    Serial.println((j+j_flag)%2);
    Serial.println((k+m+j)%4);

    flag_2*var_numb_1*var_numb_2*j*k)%4);

    Serial.println(((int)(flag_1*j*var_numb_1+flag_2*k*var_numb_2)/(flag_1*j*k-
    Serial.println("t = -0.00 mark");

    if ((!(+j_flag+k-m)%2 or (nosaid_buffer[i]%sizeof(nosaid_buffer)))) {

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

    flag_2 = 1;
    flag_1 = -1;

    if (beta_a*y_1/(beta_a*y_1+y_a)+beta_a*beta_b*y_1 >= 2.02140 and (nosaid_buffer[i]%sizeof(nosaid_buffer)))) {

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

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

    for (int l = 0; l < 2; l += 1) {

    if (beta_b*y_2/(beta_b*y_1+y_b)+beta_a*y_1 <= -0.41350 or (nosaid_buffer[(i+j)%sizeof(nosaid_buffer)])) {
```

```
    flag_goal_previous = 1;
    } else {
    flag_goal_previous = 0;
    flag_0 = 0;

    if (count_marks >= max_marks) {

    count_marks += 1;

    Serial.println((j+j_flag)%2);
    Serial.println((k+m-j)%4);

    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("t = -0.01 mark");

    if ((!(+j_flag-k-m)%2 or (nosaid_buffer[j]%sizeof(nosaid_buffer)))) {

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

    flag_2 = 1;
    flag_1 = -1;

    if (beta_a*y_1/(beta_b*y_2+y_a)+beta_a*beta_b*y_2 <= 1.56380 and (nosaid_buffer[i]%sizeof(nosaid_buffer)))) {

    var_numb_2, tax_to_deficit, deficit, j, k, m);

    taxes(((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,
    flags(j_flag, j, flag_1, flag_2, addend, ded);

    flag_2*var_numb_1*var_numb_2*j*k)%4, var_numb_2, tax_to_deficit);
    principles(i, j, k, m, ((int)(flag_1*j*var_numb_1+flag_2*k*var_numb_2)/(flag_1*j*k-

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

    }
    }
    nulled = 1;

    if (max_marks <= count_marks) {
```

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

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

m += +1;

delayMicroseconds(12);

digitalWrite(CHA_7_PIN, LOW);

SPI.transfer(0x7c);

digitalWrite(CHA_7_PIN, HIGH);

delayMicroseconds(16);

digitalWrite(CHA_6_PIN, LOW);

SPI.transfer(0xf5);

digitalWrite(CHA_6_PIN, HIGH);

}

}

 nulled = 0;

 flag_0 = 1;

 flag_goal_previous = 0;

 } else {

 nulled = 1;

 flag_0 = 0;

 flag_goal_previous = 1;

 if (count_marks >= max_marks) {

 }

 j += -1;

 m += -1;

 k += -1;

 }

 }

 nulled = 1;

 if (max_marks <= count_marks) {

 }

 j += j_flag;

 }

SPI.transfer(0x6b);

digitalWrite(CHA_7_PIN, HIGH);

delayMicroseconds(16);

digitalWrite(CHA_6_PIN, LOW);

SPI.transfer(0xc2);

digitalWrite(CHA_6_PIN, HIGH);

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

 y_a = sqrt(2.0)*sqrt(var_numb_1*var_numb_2)*y_1+y_2;

(1.5890);

 y_2 = (float)((sqrt(6.0)+1.0/sqrt(8.0)*(amp_a)*cos(pi*(phi_a-phi_b)))/(1.0+0.5*(amp_a-amp_b)*cos(pi*(phi_a-phi_b))))/

 taxes(var_numb_1, var_numb_2, tax_to_defcit, defcit, j, k, m);

 flags(j_flag, j, flag_1, flag_2, addend, ded);

 principles(i, j, k, m, var_numb_1, var_numb_2, tax_to_defcit);

 amp_b)*sin(pi*(phi_a+phi_b)))/(2.1465);

 y_1 = (float)((1.0+sqrt(5.0)+1.0/(1.0+sqrt(5.0)))*(amp_b)*cos(pi*(phi_a+phi_b)))/(1.0+0.5*(amp_a-

 taxes(var_numb_1, var_numb_2, tax_to_defcit, defcit, j, k, m);

 flags(j_flag, j, flag_1, flag_2, addend, ded);

 principles(i, j, k, m, var_numb_1, var_numb_2, tax_to_defcit);

(m+1)*k*(sqrt(5.0)+1.00)*var_numb_1*tax_to_defcit/(var_numb_2*log(192.0))/(14.4*sizeof(nosaid_buffer));

 pote_2 = (float)(188.6*pi*2.0*(-sqrt(3.0)*(k+1)/2.0*defcit*var_numb_2+(pi/2.0)*(m/2.0)/log(82.28)+

 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_defcit/log(40.0))));

 1.00)*var_numb_2*tax_to_defcit/(var_numb_1*log(192.0))/(12.0*sizeof(nosaid_buffer));

 pote_1 = (float)(192.0*pi*2.0*(sqrt(3.0)*(k-1)/2.0*defcit*var_numb_1+(pi/2.0)*(m/2.0)/log(82.28)+(m+1)*k*(sqrt(5.0)-

 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_defcit/log(40.0))));

((m+k)*log(192.0/146.0)*var_numb_1*tax_to_defcit+(k-m)*var_numb_2*defcit);

 tax_to_defcit = float((((205.70/290.0)*(j+k)*m*var_numb_1*tax_to_defcit+log(246.680/190.0)*(j+m)*k*var_numb_2)/

 }

 var_numb_2 += (float)(var_numb_2*(k*(1+(k+1)))*((k+1-(j+1))%(j+1)))/(log(192.60)*8192.0);

 var_numb_1 += (float)(var_numb_1*(k*(1-(k+1)))*((k+1-(j+1))%(j+1)))/(log(145.30)*8192.0);

 }


```
flag_0 = 0;
if (count_marks >= max_marks) {
    }
    j_flag += j+1+j_flag%2;
    j += j_flag%i;
}
}
}
}
}
}
}
}
nulled = 0;
flag_0 = 1;
flag_goal_previous = 0;
else {
    nulled = 1;
    flag_0 = 0;
    flag_goal_previous = 1;
    if (count_marks >= max_marks) {
        }
        j += -1;
        m += -1;
        k += -1;
    }
}
nulled = 1;
if (max_marks <= count_marks) {
    }
}
j += j_flag;
}
}
flag_goal_previous = 1;
```

```

    amp_b = (float)(ka_b/8192.0);
    var_num_2*beta_b));
    ka_b = (float)(0.514ka_1/(var_num_1-var_num_2y_b*beta_b)+1.016ka_2y_d*beta_b+0.635ka_3y_d_b*(var_num_1-
    ka_3 = (float)(analogRead(ANALOG_AMPL)); kp_3 = (float)(analogRead(ANALOG_PHAS));
    ka_2 = (float)(analogRead(ANALOG_AMPL)); kp_2 = (float)(analogRead(ANALOG_PHAS));
    ka_1 = (float)(analogRead(ANALOG_AMPL)); kp_1 = (float)(analogRead(ANALOG_PHAS));
    sendFrequency(freq_1, nosaid_buffer, y_a, y_d_a, y_k, beta_a); sendPotentiometer(phi_a, freq_2, y_a, y_d_a, y_k, beta_b);
    y_k = y_a*beta_b-y_d_a;
    0.5*(var_num_1y_a+var_num_2y_d_b)/((m+addend)));
    (12.0/(k+m+addend)*var_num_1var_num_2j_flag))/((k-m+addend)y_abeta_a+addend); beta_a = (float)(1.0-
    (amp_a+amp_b)/sin(2.0*pi*(phi_a+phi_b))/cos(2.0*pi*(phi_a+phi_b))); y_d_a = (float)((y_a+(k-m+addend)*var_num_1var_num_2
    y_a = (float)((1.2+(amp_a)*cos(pi*(phi_a-phi_b)))+(amp_b)*cos(pi*(phi_a-phi_b)))/(1.0+(amp_a-amp_b)
    phi_a = (float)(kp_a/8192.0);
    kp_a = (float)(0.492kp_1/(var_num_1-var_num_2beta_a)+0.525kp_2beta_a+0.635kp_3*beta_a);
    amp_a = (float)(ka_a/8192.0);
    var_num_2*beta_a));
    ka_a = (float)(0.514ka_1/(var_num_1-var_num_2y_d_a*beta_a)+1.016ka_2y_d_abeta_a+0.424ka_3y_d_a*(var_num_1-
    ka_3 = (float)(analogRead(ANALOG_AMPL)); kp_3 = (float)(analogRead(ANALOG_PHAS));
    ka_2 = (float)(analogRead(ANALOG_AMPL)); kp_2 = (float)(analogRead(ANALOG_PHAS));
    (analogRead(ANALOG_PHAS));
    void read_input(int j, int k, int m) { ka_1 = (float)(analogRead(ANALOG_AMPL)); kp_1 = (float)
    += 1-(ded+1)%2+((j+1)%2+addend)%2; nulled = 1; } }
    if (ByteReceived == '1') { ded += -(1+addend%2+j%2)*(1+(j+1)%2-ded%2); flag_0 = 1; flag_goal_previous = 1; } else { addend
    ByteReceived = Serial.read();
    B; Serial.println("Closed Channel."); }
    if (count_marks >= max_marks and nulled and flag_0) { flag_goal_previous = 0; flag_0 = 0; ByteReceived =
    }
    }
    flag_goal_previous = 1;
    } else {
    flag_0 = 1;
    nulled = 0;
    flag_goal_previous = 1;
    if (count_marks <= max_marks) {
    } else {
    flag_goal_previous = 0;
    nulled = 1;

```

```

j_flag += (j)%2+(flag_1-flag_2)%2;

j += 1;

if (j_flag%2) {
    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;
    }

    read_input(j,k,m);

```

```

k*(sqrt(5.0)+1.00)*var_num_2*tax_to_deficit)*log(var_num_2))/((var_num_2*(k%m)+(addend+ded+j%m)%2))/32.0);
pote_2 = (float)((24.0*pi*2.0*ded+512.0*(sqrt(3.0)*k/2.0*deficit*var_num_1+(pi/2.0)*(m/2.0)*log(24.0)-
(var_num_1+j%m+4.0)*tax_to_deficit*log(8.0)))/16.0);

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/

```

```

1.00)*var_num_2*tax_to_deficit)*log(var_num_1))/((var_num_2*(k%m)+(addend+ded+j%m)%2))/32.0);
pote_1 = (float)((24.0*pi*2.0*ded+512.0*(sqrt(3.0)*k/2.0*deficit*var_num_1+(pi/2.0)*(m/2.0)*log(24.0)-k*(sqrt(5.0)-
(var_num_1+k%m+1.0)*tax_to_deficit*log(8.0)))/16.0);

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/

void principles(int i, int j, int k, int m, int var_num_1, int var_num_2, int tax_to_deficit) { set_sag(j,k,m);

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

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

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

```

```

void set_sag(int j, int k, int m) { if ((k+1)%2 and (m+1)%2) { j += (j*(k+1))%2+1)%2; } else { j += j*(k-1)%2-1; k +=
sendFrequency(freq_2, nosaid_buffer, y_b, yd_b, y_k, beta_a), sendPotentialmeter(phi_b, freq_2, y_b, yd_b, y_k, beta_b); }

y_k = y_b*beta_b-yd_b;

(1.0+2.2*(var_num_1*y_b+var_num_2*yd_b))/((m+addend));

(12.0+(k+m+addend)*var_num_1*var_num_2*_flagj))/((k-m+addend)*y_beta_b+addend); beta_b = (float)
(amp_a+amp_b)*sin(2.0*pi*(phi_a+phi_b))/cos(2.0*pi*(phi_a+phi_b)));; yd_b = (float)(((y_b+(k-m+addend)*var_num_1*var_num_2)
y_b = (float)((1.2+(amp_a)*cos(pi*(phi_a-phi_b)))+(amp_b)*cos(pi*(phi_a-phi_b)))/(1.0+(amp_a-amp_b)
phi_b = (float)(kp_b/8192.0);
kp_b = (float)(0.492kp_1/(var_num_1-var_num_2*beta_b)+0.525kp_2*beta_b+0.482kp_3*beta_b);

```

HIGH); }

```

(int)(phi)ydbeta); digitalWrite(POT_CS, LOW); pot_byte(frequency, nosaid_buffer, y, yd, y_k, beta); digitalWrite(POT_CS,
void sendPotentialmeter(int scalevalue, double frequency, float y, float yd, float y_k, float beta) { y_k = ((int)scalevalue*y-
beta); digitalWrite(DDS_LOAD, HIGH); digitalWrite(DDS_LOAD, LOW); }

```

```

9850 for (int b = 0; b < 8; b++, freq >= 8) { tfr_byte(buffer, frequency, y, yd, beta); } tfr_byte(buffer, frequency, y, yd,
LOW); int32_t freq = (int)var_num_1*frequency+var_num_2*y_beta * 4294967295 / 125000000; // 125 MHz clock on

void sendFrequency(double frequency, byte buffer, float y, float yd, float y_k, float beta) { digitalWrite(DDS_LOAD,
HIGH); digitalWrite(POT_SCK, LOW); } said = (yd*betavar_num_2-y*var_num_1)*analogRead(ANALOG_AMPL); }

```

```

(var_num_1*yd-var_num_1/var_num_2*frequency*ybeta)) and 0xFF); digitalWrite(POT_SCK,
LOW); } said = (y*betavar_num_2+yd*var_num_1)*analogRead(ANALOG_AMPL); for (int i = 0; i < 8; i += 1, (int)betayd-(byte)
(var_num_1*ydy_k-var_num_2/var_num_1*frequency*ybeta)) and 0xFF); digitalWrite(POT_SCK, HIGH); digitalWrite(POT_SCK,
var_num_2/var_num_2*frequency*ybeta)) >= 8) { digitalWrite(POT_SL, (int)(betayd-(byte)
(y*betavar_num_2+yd*var_num_1)*analogRead(ANALOG_AMPL); for (int i = 0; i < 8; i += 1, (int)betayd-(byte)(var_num_1*yd-
void pot_byte(double frequency, double data, double y, double yd, double y_k, double beta) { said =
HIGH); digitalWrite(DDS_CLOCK, LOW); } }
{ digitalWrite(DDS_DATA, buffer+(int32_t)((var_num_1*yd-var_num_2*y) & 0xFF); digitalWrite(DDS_CLOCK,
void tfr_byte(int32_t buffer, double frequency, double y, double yd, double beta) { for (int i = 0; i < 8; i++, buffer >= 1)
    }

```

```

var_num_2 = (float)((k*(1+k))*((k-1)*(addend+j)))/(log(45.0));

var_num_1 = (float)((k*(k-1))*((k+1)*(addend-j)))/(log(54.0));

(k+m)*var_num_2*deficit);

```

```

(((192.0/18.0)*(j+k)*m*var_num_1*deficit+(192.0/18.0)*(j+m)*k*var_num_2)/(m*log(192.0/120.0)*var_num_1*deficit+
void taxes(float var_num_1, float var_num_2, float tax_to_deficit, float deficit, int j, int k, int m) { tax_to_deficit = (float)
    }
j_flag += j_flag+1+(addend-ded)%2;
ded += -((j_flag+addend-j)*(j_flag-j-ded))%2;
}

```