```c
#include "Functions.h"

int main(void)
{
    // define variables
    int N_step, N_delay;
    unsigned long y_max, y_min,sp_x;

    // ADC&DAC setting
    ADCpoweron(20000);  // power on ADC
    REFCON = 0x01;  // internal 2.5V reference
    DAC0CON = 0x12; // AGND-ARef range 0x12 2.5V
    ADCCP = 0x00;   // conversion on ADC0
    ADCCON = 0x6E4; // continuous conversion

    // IO setting
    GP1CON = 0x00000000;    // IO initialization
    GP1DAT = 0xFF000000;    // set group 1 as output
    GP0CON = 0x00000000;    // IO initialization
    GP0DAT = 0x00000000;    // set group 0 as input

    N_step = 1; // step size
    N_delay = 500;  // wait for certain time

    Write_Digital(0,0); //Write a low level digital value on pin P1.0 for indicating out of locking

    while(1){
        Max_Min(&y_max, &y_min); // sweeping for obtaining the maximum and minimum values

        while(Read_Digital(3) == 0){ // switch between sweep mode and locking mode;
            Max_Min(&y_max, &y_min);
            Write_Digital(0,0); //Write a low level digital value on pin P1.0 for indicating in the sweeping mode
        }

        Start_Point( y_max, y_min, &sp_x); //find start point for the locking

        Lock_Maximum_Searching(N_step, N_delay, y_max, y_min, sp_x); // Locking based on maximum_searching algorithm
        //Lock_PI_Like(N_step, N_delay, y_max, y_min, sp_x); // Locking based on PI-like algorithm
    }
}
```

```c
#include<aduc7020.h>

void ADCpoweron(int);
void Delay(int);
int Read_Digital(int n);
void Write_Digital(int n, int state);
void Delay(int);
unsigned long Bound(unsigned long N);
int ADCtoDAT(unsigned long ADC);
unsigned long DATtoADC(int DAT);
void Max_Min(unsigned long *y_max, unsigned long *y_min);
void Start_Point(unsigned long y_max, unsigned long y_min, unsigned long *sp_x);
void Lock_Maximum_Searching(int N_step, int N_delay, unsigned long y_max, unsigned long y_min, unsigned long sp_x);
void Lock_PI_Like(int N_gain, int N_delay, unsigned long y_max, unsigned long y_min, unsigned long sp_x);

// flag for sweep direction
struct bs
{
    unsigned c0:1;
    unsigned c1:1;
    unsigned c2:1;
    unsigned c3:1;
    unsigned c4:1;
    unsigned c5:1;
    unsigned c6:1;
    unsigned c7:1;
};
struct bs flag = {0,0,0,0,0,0,0,0};

// read digital value from the pin P0.n
int Read_Digital(int n)
{
    return ((GP0DAT&0x000000FF) >> n) & 0x1;
}

// write digital value to the pin P1.n
void Write_Digital(int n, int state)
{
    if(state == 1)
      GP1DAT = (0x00000001<<(n + 16)) | GP1DAT;
    else
        GP1DAT = ~((0x00000001<<(n + 16)) | (~GP1DAT));
}

// wait for ADC to be fully powered on
void ADCpoweron(int time)
{
    ADCCON = 0x620; // power-on the ADC
    while (time >= 0)
    time--;
}

// convert ADC/DAC format to integer format
int ADCtoDAT(unsigned long ADC)
{
    return (ADC&0xFFF0000)>>16;
}

// convert integer format to ADC/DAC format
unsigned long DATtoADC(int DAT)
{
    unsigned long ADC;
    ADC = DAT;
    return ADC<<16;
}
```

```c

// wait for a certain time
void Delay(int cnt)
{
    while(cnt>0) cnt--;
}

// full scanning for finding the max and min value
void Max_Min(unsigned long *y_max, unsigned long *y_min)
{
    int i;
    unsigned long V, Ymax, Ymin;

    Ymax = 0;
    Ymin = 0xFFF0000;
    V = 0;

    if(flag.c0 == 0){   // make a ramp sweeping
        for(i = 10; i <= 4080; i += 10){
            DAC0DAT = DATtoADC(i);
            Delay(100);  // wait for PZT
            V = ADCDAT;
            if(V > Ymax)    Ymax = V;
            if(V < Ymin)    Ymin=V;
        }
    }
    else {
        for(i = 4080; i >= 10; i -= 10){
            DAC0DAT = DATtoADC(i);
            Delay(100);  // wait for PZT
            V = ADCDAT;
            if(V > Ymax)    Ymax = V;
            if(V < Ymin)    Ymin = V;
        }
    }
flag.c0++;
*y_max = Ymax;
*y_min = Ymin;
}

// find start point for the locking
void Start_Point(unsigned long y_max, unsigned long y_min, unsigned long *sp_x)
{
    int i;
    unsigned long V_th, V;
    V_th = (y_max - y_min) * 3/4 + y_min; // set the threshold

    if(flag.c0 == 0){
        for(i = 10; i <= 4080; i++){
            DAC0DAT = DATtoADC(i);
            Delay(100); // wait for certain time for the responce of PZT
            V = ADCDAT;
            if(V > V_th){
                *sp_x = DATtoADC(i);
                break;
            }
        }
    }
    else{
        for(i = 4080; i >= 10; i--){
            DAC0DAT = DATtoADC(i);
            Delay(100); // wait for certain time for the responce of PZT
            V = ADCDAT;
            if(V > V_th){
                *sp_x = DATtoADC(i);
                break;
```

```
            }
        }
    }
flag.c0++;
}


// Bound the output value of DAC, when V reaches the limit of DAC range
unsigned long Bound(unsigned long V)
{
    if (V > 0xFF60000)
        return 0xFF60000;
    else if(V < 0xA0000)
        return 0xA0000;
    else
        return V;
}



// Locking based on maximum_searching algorithm
void Lock_Maximum_Searching(int N_step, int N_delay, unsigned long y_max, unsigned long y_min, unsigned long sp_x)
{
    int flag, k, sum, N;
    unsigned long V1, V2, V_th, V;

    flag = 1; // indicator for searching direction
    V_th = (y_max-y_min)/4+y_min; // set the threshold
    V = sp_x; // initialized with start ponit
    V1 = ADCDAT; // read ADC value
    N = 200; // accumulation number for each locking step


    while(1){
        if(Read_Digital(3) == 0 || V1 < V_th)   break; // lower than the threshold or enabling of sweep mode will break out of the while
        Write_Digital(0,1); // Output a high level digital output on pin P1.0 for indicating in the locking mode

        V = V + flag * DATtoADC(N_step); // calculate the voltage for next step
        V = Bound(V); // limit the range of V
        DAC0DAT = V; // output voltage on DAC0
        Delay(N_delay); //wait for a certain time for the response of PZT

        sum = 0; // initialization
        for(k = 0; k < N; k++){
        while (!ADCSTA){}       // wait for end of ADC conversion
        sum += ADCtoDAT(ADCDAT); // read voltage from ADC0
        }
        V2 = DATtoADC(sum/N);   // calculate average value for the voltage of second step

        if(V2 < V1) flag = -1 * flag; // change maximum searching direction if V2 < V1

        V1 = V2; // update the voltage of first step

    }
        Write_Digital(0,0); //Write a low level digital value on pin P1.0 for indicating out of locking
}

// Locking based on PI-like algorithm
void Lock_PI_Like(int N_gain, int N_delay, unsigned long y_max, unsigned long y_min, unsigned long sp_x)
{
    int i, N, diff, gp, gi, I, error;
    unsigned long V, V1, V2, sum, V_th;

```

```
    N = 100;
    V = sp_x;
    V_th = (y_max-y_min)/4+y_min; // set the threshold
    gp = 1;
    gi = 1;
    I = 0;
    error = 0;

    while(1){
        Write_Digital(0,1); // Output a high level digital output on pin P1.0 for indicating in the locking mode

        sum = 0; // initialization
        for(i = 1; i <= N; i++){
            while (!ADCSTA){}   // wait for end of ADC conversion
            sum += ADCtoDAT(ADCDAT);
        }
        V1=DATtoADC(sum/N); // calculte average value for the voltage of second step

        if(Read_Digital(3) == 0 || V1 < V_th) break; // lower than the threshold or enabling of sweep mode (P0.3) will break out of the while

        V=V+DATtoADC(1);
        V=Bound(V);
        DAC0DAT=V;
        Delay(N_delay);

        sum = 0;
        for(i = 1; i <= N; i++){
            while (!ADCSTA){}
            sum += ADCtoDAT(ADCDAT);
        }
        V2 = DATtoADC(sum/N);

        diff = ADCtoDAT(V2) - ADCtoDAT(V1);

        I = I + diff;
        error = I * gi / 2 + diff * gp / 5;
        V = sp_x + DATtoADC(error * N_gain);

        V = Bound(V); // limit the range of V
        DAC0DAT = V; // output voltage on DAC0
        Delay(N_delay); //wait for a certain time for the response of PZT
    }
        Write_Digital(0,0); //Write a low level digital value on pin P1.0 for indicating out of locking
}
```