```
//********************************************************************************************************
// Header: FUZZY regulator s RTX51 pre taviace pece
// File Name: FUZZY.C
// Author: Ing. Eduard Jadron, Jilemnickeho 3999/37, 03601 Martin, Slovak Republic
// Date: 29.12.2005
//********************************************************************************************************

#include <reg552.h>
#include <stdio.h>
#include <stdlib.h>
#include <rtx51.h>
#include <string.h>
#include <dsw.h>

#define REGULATOR 0
#define ACTION    1

int Error[3];     //int Error,dError,LastError;

#define SIZE  5   //Velkost inferencneho bloku !!!

int X1[SIZE],X2[SIZE],Y[SIZE],Output;

int Soll,Vert;

//typedef enum {NM,NS,ZE,PS,PM} State;

#define NM  0 // Negative medium
#define NS  1 // Negative small
#define ZE  2 // Zero equal
#define PS  3 // Positive small
#define PM  4 // Positive medium

const unsigned char InferencnaTabulka[SIZE][SIZE] =
{    /*NM,NS,ZE,PS,PM*///dError X2[]
/*NM*/{PM,PM,PM,PS,ZE},//NM  E
/*NS*/{PM,PM,PS,ZE,NS},//NS  r
/*ZE*/{PM,PS,ZE,NS,NM},//ZE  r
/*PS*/{PS,ZE,NS,NM,NM},//PS  o
/*PM*/{ZE,NS,NM,NM,NM} //PM  r
};               //    X1[]

#define K 0x18

const int VystupnaFunkcia[SIZE] = {-2*K,-K,+0x00,+K,+2*K};

unsigned int ADC(unsigned char Channel)   //Rutina trva 55us pri 18.432MHz
{
 #define ADCS 0x08
 #define ADCI 0x10
 #define ADEX 0x20
 unsigned int x;
 ADCON=Channel;                             //Vyber kanalu pre prevod 0..7
 ADCON|=ADEX+ADCS;                          //Spustim prevod, bolo tu ADCON|=ADEX+ADCS;
 while((ADCON&ADCI)==0x00);                 //Cakam na ukoncenie prevodu
 ADCON^=ADCI;                               //Nulujem priznak AD prevodnika
 x=ADCH*0x04;                               //Normovanie na (int)
 switch(ADCON&(0x80+0x40))
  {
   case 0x40  : x+=0x01; break;
   case 0x80  : x+=0x02; break;
   case 0xC0  : x+=0x03; break;
  }
 return(x);
}

typedef enum {SKY=0x0400,SK0=+0x0000,SK1=+0x0400,SK2=+0x0800} Osi;

void Fuzzyfikator(int Value, int *Data) //reentrant
{
 unsigned char i;
 for(i=0x00; i<SIZE; i++) Data[i]=0x00;
 if(Value<-SK2) Data[NM]=SKY;
```

```c
 else if(Value<-SK1)
 {
  Data[NM]=SKY-(Value+SK2);
  Data[NS]=Value+SK2;
 }
 else if(Value<+SK0)
 {
  Data[NS]=SKY-(Value+SK1);
  Data[ZE]=Value+SK1;
 }
 else if(Value<+SK1)
 {
  Data[ZE]=SKY-Value;
  Data[PS]=Value;
 }
 else if(Value<+SK2)
 {
  Data[PS]=SKY-(Value-SK1);
  Data[PM]=Value-SK1;
 }
 else Data[PM]=SKY;
}

void FuzzyInferencnyModul(int *X1, int *X2, int *Y) //reentrant
{
 int min[SIZE];
 int max,Temp;              //Tu musi byt v pripade RTX51 a LARGE modelu "data int max" s optimalizaciou Loop rotation - 6
 unsigned char i,j,maxpos;
// for(i=0x00; i<SIZE; i++) Y[i]=0x00;  //Vymazem vystupny vektor Y[], je to neefektivnejsie ako memset()
 memset(Y,0x00,2*SIZE);         //Vymazem vystupny vektor Y[], je to rychlejsie
 for(i=0x00; i<SIZE; i++)
  {
   for(j=0x00; j<SIZE; j++) if(X1[i]<X2[j]) min[j]=X1[i]; else min[j]=X2[j];
   max=min[0];
   maxpos=0x00;
   for(j=0x01; j<SIZE; j++) if(max<min[j]) {max=min[j]; maxpos=j;}
   Temp=Y[InferencnaTabulka[i][maxpos]];
   if(Temp<max) Y[InferencnaTabulka[i][maxpos]]+=max;
   if(Temp>SKY) Y[InferencnaTabulka[i][maxpos]]=SKY;
  }
}

int Defuzzyfikator(int *Y) //reentrant
{
 unsigned char i;
 int ReturnValue=0,SumY=0;   //COG algoritmus - Centrum Of Gravity
 for(i=0x00; i<SIZE; i++)
  {
   SumY+=Y[i];
   ReturnValue+=Y[i]*VystupnaFunkcia[i];
  }
 return((long)ReturnValue<<2)/SumY; //Skalujem - nasobim cislom 4
}

void Regulator(void) _task_ REGULATOR _priority_ 0
{
 os_set_slice(xtal(18.432E6)/10);
 PWMP=0x20;
 Soll=Vert=0x0000;
 while(1)
  {
//   Vert=ADC(0);
   if(++Vert>=1024) Vert=0;
   Soll=ADC(2);
   Error[2]=Error[0];
   Error[0]=(Vert-Soll)<<1;
   Error[1]=(Error[0]-Error[2]);
   Fuzzyfikator(Error[0],X1);
   Fuzzyfikator(Error[1],X2);
   FuzzyInferencnyModul(X1,X2,Y);
   Output=Defuzzyfikator(Y);
//   PWM0=0x80-Output;
   PWM0=PWM1=/*0x80-*/Output;
   os_wait(K_TMO,2,NULL);
  }
}

void main(void)
{
 os_start_system(REGULATOR);
}
```