ภาคผนวก

ภาคผนวก ก
ส่วนของโปรแกรมไมโครคอนโทรลเลอร์ (Microcontroller)

โปรแกรมไมโครคอนโทรลเลอร์

```
#include <LPC2103.H>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

/*+++ Global Var. +++*/
unsigned char hold,feed;
unsigned int freq_xyz,freq_a;
unsigned long periodxyz,perioda;
unsigned long counta,count,countx,county,countz,position_x,position_y,position_z,position_a;

int Ts_T = 20;
int T_T = 0;
int Told_T = 0;
int D_T;
int E_T;
float N_T;

int Ts_AX = 20;
int T_AX = 0;
int Told_AX = 0;
int D_AX;
int E_AX;
float N_AX;

int Ts_AY = 20;
int T_AY = 0;
int Told_AY = 0;
int D_AY;
```

```
int E_AY;
float N_AY;

#define TE_T 10
int Lower, OK, Upper, Down, Constant, Up;
#define TD_T 10
unsigned int H1_T,H2_T,H3_T,M1_T,M2_T,M3_T,L1_T,L2_T,L3_T;
#define TN_T 20



#define TE_AX 10
int N , Z , P , n , z , p;
#define TD_AX 10
unsigned int H1_AX,H2_AX,H3_AX,M1_AX,M2_AX,M3_AX,L1_AX,L2_AX,L3_AX;
#define TN_AX 20

#define TE_AY 10
int N , Z , P , n , z , p;
#define TD_AY 10
unsigned int H1_AY,H2_AY,H3_AY,M1_AY,M2_AY,M3_AY,L1_AY,L2_AY,L3_AY;
#define TN_AY 20

unsigned long dN_T;
unsigned long dN_AX;
unsigned long dN_AY;

unsigned char Check = 1;

/*+++ Motor +++*/
```

```
void uart0_init(unsigned int baudrate)
{
unsigned short u0dl;
u0dl = 29491200/(16*baudrate);
PINSEL0 &= 0xFFFFFFF0;
PINSEL0 |= 0x00000005;
U0LCR = 0x83;
U0DLL = u0dl & 0xFF;
U0DLM = (u0dl>>8);
U0LCR &= 0x7F;
}

int putchar(int ch)
{
if (ch == '\n')
{
while (!(U0LSR & 0x20));
U0THR = 0x0D;
}
while (!(U0LSR & 0x20));

return (U0THR = ch);
}

int getchar(void)
{
while (!(U0LSR & 0x01));
return (U0RBR);
}
```

```
float getnum(void)
{
unsigned char i,buffer,string[100];

for (i=0;i<100;i++) string[i]=0;
buffer = getchar();
if (buffer == '[')
{
for (buffer=getchar(),i=0;buffer!=']';buffer=getchar(),i++) string[i] = buffer;
}
else return 0;

return atoi(string);
}
void directionmotor_init(void)
{
PINSEL0 &= 0xFF00FFFF;         // Set P0.8,P0.9 P0.10,P0.11 GPIO
IODIR   |= 0x00000F00;  // Enable GPIO for direction motor
IOCLR   |= 0x00000F00; // Clear all bit to Initial
}

void motor_init(unsigned int freq)
{
PINSEL0 &= 0xF0FFFF3F;         // 1111 0000 1111 1111 1111 1111 0011 1111
PINSEL1 &= 0xFFFFFF3F;
PINSEL0 |= 0x0A000080;            // 0000 1010 0000 0000 0000 0000 1000 0000
PINSEL1 |= 0x00000080;
periodxyz = (10000000/(33.90842*freq))*100;
perioda = (10000000/(33.90842*freq))*100;
T0TCR   &= 0xFFFFFFFC;
```

```
T0TC     = 0x00000000;
 T0PR     = 0x00000000;
 T0PC     = 0x00000000;

// Initial Timer Operate
T1TCR   &= 0xFFFFFFFC;          // Timer1 = Timer Mode Count By Rising PCLK Edge
T1TC     = 0x00000000;  // Timer Start = 0
T1PR     = 0x00000000;  // Prescale = 0
T1PC     = 0x00000000;  // Prescale Count = 0

T0MCR&= 0xFF7F;
T0MCR   |= 0x00C0;

// Initial Timer1 Match Operate
T1MCR&= 0xF8FF;                 // Disable Interrupt on Match-3    (1111 1011 1111 1111)
T1MCR   |= 0x0600;              // Enable Reset TC on Match-3    (0000 0100 0000 0000)

T0PWMCON |= 0x01;

// Initial PWM Function
// Set   Output By Timer Match
// Reset Output By TC Reset
//  T1PWMCON |= 0x07;           // Enable PWM
T0IR &= 0xF8;

 // Initial Timer1 Interrupt
 T1IR &= 0xF0;              // Disable MAT0 Interrupt (1111 0000)

T0MR0 = (perioda/2);
T0MR2 =          perioda;
```

```
T1MR0 = (periodxyz/2);
T1MR1 = (periodxyz/2);
T1MR2 = (periodxyz/2);
T1MR3 = periodxyz;
}

void motorxyz_refresh(void)
{
T1TCR   &= 0xFFFFFFFC;
T1TC     = 0x00000000;
T1PR     = 0x00000000;
T1PC     = 0x00000000;
T1MR0 = (periodxyz/2);
T1MR1 = (periodxyz/2);
T1MR2 = (periodxyz/2);
T1MR3 = periodxyz;
}

void motora_refresh(void)
{
T0TCR   &= 0xFFFFFFFC;
T0TC     = 0x00000000;
T0PR     = 0x00000000;
T0PC     = 0x00000000;
T0MR0 = (perioda/2);
T0MR2 = perioda;
}

void motorxyz_freq(unsigned int freq)
```

```
{
periodxyz = (10000000/(33.90842*freq))*100;
motorxyz_refresh();
}

void motora_freq(unsigned int freq)
{
perioda = (10000000/(33.90842*freq))*100;
motora_refresh();
}

void pulsemotor_XR(void)
{
IOCLR |= 0x00000100;
T1PWMCON |= 0x01;
}

void pulsemotor_XL(void)
{
IOSET |= 0x00000100;
T1PWMCON |= 0x01;
}

void pulsemotor_YR(void)
{
IOCLR |= 0x00000200;
T1PWMCON |= 0x02;
}

void pulsemotor_YL(void)
```

```
{
IOSET |= 0x00000200;
T1PWMCON |= 0x02;
}

void pulsemotor_ZR(void)
{
IOCLR |= 0x00000400;
T1PWMCON |= 0x04;
}

void pulsemotor_ZL(void)
{
IOSET |= 0x00000400;
T1PWMCON |= 0x04;
}

void pulsemotor_AR(void)
{
IOCLR |= 0x00000800;
T0PWMCON |= 0x01;
}

void pulsemotor_AL(void)
{
IOSET |= 0x00000800;
T0PWMCON |= 0x01;
}
void comehome(void)
{
```

```
county = position_y;
countx = position_x;
if (county > 0)pulsemotor_YR();
if (countx > 0)pulsemotor_XR();
T1TCR |= 0x01;

while(countx > 0 || county > 0);
position_y = county;
position_x = countx;

countz = position_z;
if (countz > 0) pulsemotor_ZL();
T1TCR |= 0x01;
while(countz >0);
position_z = countz;

}

void countoutput_xyz(void) __irq
{
if (countx > 0) countx--;
else T1PWMCON &= 0xFE;
if (county > 0)    county--;
else T1PWMCON &= 0xFD;
if (countz > 0) countz--;
else T1PWMCON &= 0xFB;
if ((T1PWMCON & 0x07) == 0) T1TCR &= 0xFC;
 T1IR |= 0x08;
T1IR &= 0xF7;
```

```
VICVectAddr = 0x00000000;
}
void countoutput_a(void) __irq
{
if(Check == 1){
if (counta > 0) counta--;
else {T0PWMCON &= 0xFE; T0TCR &= 0xFC;}}
if(Check == 0){
if (dN_T > 0) dN_T--;
else {
T0PWMCON &= 0xFE; T0TCR &= 0xFC;}}
T0IR |= 0x08;
T0IR &= 0xF7;
VICVectAddr = 0x00000000;
}
void interrupt_init(void)
{
VICVectCntl0 = 0x00000024;
VICVectAddr0 = (unsigned)countoutput_a;
VICVectCntl1 = 0x00000025;
VICVectAddr1 = (unsigned)countoutput_xyz;
VICIntEnable = 0x00000030;
}
void Last_Seq(void)
{
long  pnt_x = 0;
long  pnt_y = 0;
long  pnt_z = 0;

pnt_x = (500*100);
```

```
pnt_y = (500*100);
pnt_z = (333*80);
pnt_x -= position_x;
pnt_y -= position_y;
pnt_z -= position_z;



if (pnt_z != 0){
if (pnt_z > 0){countz = pnt_z; pulsemotor_ZR();}
else
{
countz = (0-pnt_z); pulsemotor_ZL();}
}
position_z = pnt_z;
T1TCR |= 0x01;

while (countz > 0);
if (pnt_x != 0)
{
if (pnt_x > 0){countx = pnt_x; pulsemotor_XL();}
else
{
countx = (0-pnt_x); pulsemotor_XR();
}
}
position_x = pnt_x;
if (pnt_y != 0){
if (pnt_y > 0){county = pnt_y; pulsemotor_YL();
}
```

```
else
{
county = (0-pnt_y); pulsemotor_YR();
}
}
position_y = pnt_y;
T1TCR |= 0x01;
while (countx > 0 || county > 0);
printf("Success");

}

/*+++++++++++++++++++ A/D ++++++++++++++++++++*/

void adc_init(void)
{
PINSEL1 |= 0x00003000;
ADINTEN = 0;
ADCR &= 0x00000000;
ADCR |= 0x00210600;
}

/*+++ Feedback +++*/

int feedback(void)
{
  while(1)
{
unsigned int val,volt;
```

```
ADCR &= 0xFFFFFF00;
ADCR |= 0x01000001;
do
{
val = ADDR0;
}
while((val&0x80000000)==0);
val = (val>>6)& 0x03FF;
volt = val*100/1023;
return volt;

}
}

void delay (long int ms)
{
unsigned int i,j;
for(i=0;i<ms;i++)
for(j=0;j<6666;j++);
}

/*+++ Fuzzy T +++*/

int SubtractE(unsigned int N, unsigned int M)
{
unsigned short N16,M16;
short Result16;
N16=N;
M16=M;
Result16 = N16 - M16;
```

```
if(Result16 < -65) Result16 = -65;
if(Result16 >  35) Result16 = 35;
return (Result16);
}

int SubtractD(unsigned int N, unsigned int M)
{
unsigned short N16,M16;
short Result16;
N16=N;
M16=M;
Result16 = N16 - M16;
if(Result16 < -65) Result16 = -65;
if(Result16 >  35) Result16 = 35;
return (Result16);
}

void crispinput(void)
{
E_T = SubtractE(Ts_T,T_T);
D_T = SubtractD(T_T,Told_T);
Told_T = T_T;
}
void inputmembership(void)
{
if(E_T <= -TE_T){
Lower = 0;
OK = 0;
Upper = 1000;}
else
```

```
if(E_T < 0){
Lower = 0;
OK = 1000 - Upper;
Upper = (1000*(-E_T))/TE_T;}
else
if(E_T < TE_T)
{
Lower = (1000*E_T)/TE_T;
Upper = 0;
OK = 1000 - Lower;}
else
{
Lower = 1000;
OK = 0;
Upper = 0;
}
if(D_T <= (-TD_T)){
Down = 0;
Constant = 0;
Up = 1000;
}
else
if(D_T < 0){
Down = 0;
Constant = 1000 - Up;
Up = (1000*(-D_T))/TD_T;}
else
if(D_T < TD_T)
{
Down = (1000*D_T)/TD_T;
```

```
Up =  0;
Constant = 1000 - Down;}
else{
Down = 1000;
Constant = 0;
Up = 0;}
}
unsigned int min(unsigned int u1,unsigned int u2)
{
if(u1>u2) return(u2);
else return(u1);
}
unsigned int max(unsigned int u1,unsigned int u2)
{
if(u1<u2) return(u2);
else return(u1);
}
void OutputMembership(void)
{
L1_T    = min(Lower,Down);
L2_T    = min(Lower,Constant);
L3_T    = min(OK,Down);

M1_T    = min(Upper,Down);
M2_T    = min(OK,Constant);
M3_T    = min(Lower,Up);

H1_T    = min(Upper,Up);
H2_T    = min(Upper,Constant);
H3_T    = min(OK,Up);
```

```
}

void CrispOutput(void)
{
dN_T =
((14000*(H1_T+H2_T+H3_T))+(8000*(M1_T+M2_T+M3_T))+(2000*(L1_T+L2_T+L3_T)))/((
H1_T+H2_T+H3_T+M1_T+M2_T+M3_T+L1_T+L2_T+L3_T));
}
void Fuzzy(void)
{
T_T = feedback();
crispinput();
inputmembership();
OutputMembership();
CrispOutput();
}

/*+++ Fuzzy AX +++*/

int SubtractE_AX(unsigned int N, unsigned int M)
{
unsigned short N16,M16;
short Result16;
N16=N;
M16=M;
Result16 = N16 - M16;
if(Result16 < -10) Result16 = -10;
if(Result16 >  10) Result16 = 10;
return (Result16);
```

```
}

int SubtractD_AX(unsigned int N, unsigned int M)
{
unsigned short N16,M16;
short Result16;
N16=N;
M16=M;
Result16 = N16 - M16;
if(Result16 < -10) Result16 = -10;
if(Result16 >  10) Result16 = 10;
return (Result16);
}

void crispinput_AX(void)
{
E_AX = SubtractE_AX(Ts_AX,T_AX);
D_AX = SubtractD_AX(T_AX,Told_AX);
Told_AX = T_AX;
}
void inputmembership_AX(void)
{
if(E_AX <= -TE_AX){
N = 0;
Z = 0;
P = 1000;}
else
if(E_AX < 0){
N = 0;
Z = 1000 - P;
```

```
P = (1000*(-E_AX))/TE_AX;}
else
if(E_AX < TE_AX){
N = (1000*E_AX)/TE_AX;
P = 0;
Z = 1000 - N;}
else{
N = 1000;
Z = 0;
P = 0;}
if(D_AX <= (-TD_AX)){
n = 0;
z = 0;
p = 1000;}
else
if(D_AX < 0){
n = 0;
z = 1000 - p;
p = (1000*(-D_AX))/TD_AX;}
else
if(D_AX < TD_AX){
n = (1000*D_AX)/TD_AX;
p =  0;
z = 1000 - n;}
else{
n = 1000;
z = 0;
p = 0;}

}
```

```
unsigned int min_AX(unsigned int u1,unsigned int u2)
{
if(u1>u2) return(u2);
else return(u1);
}
unsigned int max_AX(unsigned int u1,unsigned int u2)
{
if(u1<u2) return(u2);
else return(u1);
}
void OutputMembership_AX(void)
{
L1_AX = min_AX(N,n);
L2_AX = min_AX(N,z);
L3_AX = min_AX(Z,n);
M1_AX= min_AX(P,n);
M2_AX        = min_AX(Z,z);
M3_AX        = min_AX(N,p);
H1_AX        = min_AX(P,p);
H2_AX        = min_AX(P,z);
H3_AX        = min_AX(Z,p);
}

void CrispOutput_AX(void)
{
dN_AX                                                                    =
((750*(H1_AX+H2_AX+H3_AX))+(500*(M1_AX+M2_AX+M3_AX))+(250*(L1_AX+L2_AX
+L3_AX)))/((H1_AX+H2_AX+H3_AX+M1_AX+M2_AX+M3_AX+L1_AX+L2_AX+L3_AX)
);
```

```
}

void Fuzzy_AX(void)
{
T_AX = feedback();
crispinput_AX();
inputmembership_AX();
OutputMembership_AX();
CrispOutput_AX();
}

/*+++ Fuzzy AY +++*/
int SubtractE_AY(unsigned int N, unsigned int M)
{
unsigned short N16,M16;
short Result16;
N16=N;
M16=M;
Result16 = N16 - M16;
if(Result16 < -10) Result16 = -10;
if(Result16 >  10) Result16 = 10;
return (Result16);
}
int SubtractD_AY(unsigned int N, unsigned int M)
{
unsigned short N16,M16;
short Result16;
N16=N;
M16=M;
Result16 = N16 - M16;
```

```
if(Result16 < -10) Result16 = -10;
if(Result16 >  10) Result16 = 10;
return (Result16);
}

void crispinput_AY(void)
{
E_AY = SubtractE_AY(Ts_AY,T_AY);
D_AY = SubtractD_AY(T_AY,Told_AY);
Told_AY = T_AY;
}

void inputmembership_AY(void)
{
if(E_AY <= -TE_AY){
N = 0;
Z = 0;
P = 1000;}
else
if(E_AY < 0){
N = 0;
Z = 1000 - P;
P = (1000*(-E_AY))/TE_AY;}
else
if(E_AY < TE_AY){
N = (1000*E_AY)/TE_AY;
P = 0;
Z = 1000 - N;}
else{
N = 1000;
```

```
Z = 0;
P = 0;}
if(D_AY <= (-TD_AY)){
n = 0;
z = 0;
p = 1000;}
else
if(D_AY < 0){
n = 0;
z = 1000 - p;
p = (1000*(-D_AY))/TD_AY;}
else
if(D_AY < TD_AY){
n = (1000*D_AY)/TD_AY;
p =  0;
z = 1000 - n;}
else{
n = 1000;
z = 0;
p = 0;}
}
unsigned int min_AY(unsigned int u1,unsigned int u2)
{
if(u1>u2) return(u2);
else return(u1);
}
unsigned int max_AY(unsigned int u1,unsigned int u2)
{
if(u1<u2) return(u2);
else return(u1);
```

```
}

void OutputMembership_AY(void)
{
L1_AY = min_AY(N,n);
L2_AY = min_AY(N,z);
L3_AY = min_AX(Z,n);
M1_AY= min_AY(P,n);
M2_AY = min_AY(Z,z);
M3_AY = min_AY(N,p);
H1_AY = min_AY(P,p);
H2_AY = min_AY(P,z);
H3_AY = min_AY(Z,p);
}

void CrispOutput_AY(void)
{
dN_AY = ((750*(H1_AY + H2_AY + H3_AY))+(500*(M1_AY + M2_AY +
M3_AY))+(250*(L1_AY + L2_AY + L3_AY)))/((H1_AY + H2_AY + H3_AY + M1_AY +
M2_AY + M3_AY + L1_AY + L2_AY + L3_AY));
}
void Fuzzy_AY(void)
{
T_AY = feedback();
crispinput_AY();
inputmembership_AY();
OutputMembership_AY();
CrispOutput_AY();
}
```

```
/*+++ Main +++*/
void main(void)
{
unsigned int code,dir,freq_xyz,freq_a;
unsigned long point_x,point_y,point_z;
unsigned char image_x,image_y,hold,feed;

count = 0;
point_x = 0;
point_y = 0;
point_z = 0;
position_x = 0;
position_y = 0;
position_z = 0;

uart0_init(9600);
adc_init();
directionmotor_init();
interrupt_init();
motor_init(5000);

freq_a = 5000;
freq_xyz = 5000;

while(1)
{
code = getchar();
switch(code)
{
case 'F':
```

```
code = getchar();
if (code == 'a'){freq_xyz = getnum(); motorxyz_freq(freq_xyz); printf("frequency = %d\n",freq_xyz);}
if (code == 'b'){freq_a = getnum();  motora_freq(freq_a);  printf("frequency = %d\n",freq_a);}
break;
case 'H':
code = getchar();
if (code == 'a'){
position_x = 0;
position_y = 0;
position_z = 0;
printf("Add home\n");
break;}
if (code == 'b'){
motorxyz_freq(15000);
comehome();
printf("1");
break;}
case 'P':
code = getchar();
dir = getchar();
count = getnum();
if (code == 'x'){
countx = count;
if (dir == 'l'){position_x += (countx*=500); pulsemotor_XL(); T1TCR |= 0x01;}
if (dir == 'r'){position_x -= (countx*=500); pulsemotor_XR(); T1TCR |= 0x01;}


printf("x %d\n",position_x/500);
break;}
```

```
if (code == 'y'){
county = count;
if (dir == 'l'){position_y += (county*=500); pulsemotor_YL(); T1TCR |= 0x01;}
if (dir == 'r'){position_y -= (county*=500); pulsemotor_YR(); T1TCR |= 0x01;}
printf("y %d\n",position_y/500);
break;}
if (code == 'z'){
countz = count;
if (dir == 'l'){position_z += (countz*=333); pulsemotor_ZL(); T1TCR |= 0x01;}
if (dir == 'r'){position_z -= (countz*=333); pulsemotor_ZR(); T1TCR |= 0x01;}
printf("z %d\n",position_z/333);
break;}
if (code == 'a'){
counta = count;
if (dir == 'l'){position_a += (counta*=4000); pulsemotor_AL(); T0TCR |= 0x01;}
if (dir == 'r'){position_a -= (counta*=4000); pulsemotor_AR(); T0TCR |= 0x01;}
printf("a %d\n",position_a);
break;}
case 'A':
code = getchar();
if (code == 't'){
point_x = (500*getnum());
point_y = (500*getnum());
point_z = (333*getnum());
point_x -= position_x;
point_y -= position_y;
point_z -= position_z;
if (point_z != 0){
if (point_z > 0){countz = point_z; pulsemotor_ZR();}
else                    {countz = (0-point_z); pulsemotor_ZL();}
```

```
}
position_z = point_z;
T1TCR |= 0x01;

while (countz > 0);
if (point_x != 0){
if (point_x > 0){countx = point_x; pulsemotor_XL();}
else                    {countx = (0-point_x); pulsemotor_XR();}
}
position_x = point_x;
if (point_y != 0)
{
if (point_y > 0)
{
county = point_y; pulsemotor_YL();}
else
{
county = (0-point_y); pulsemotor_YR();}
}
position_y = point_y;
T1TCR |= 0x01;
while (countx > 0 || county > 0);
break;
}

if (code == 'i'){
code = getchar();
if (code == 'x'){
image_x = getnum();
```

```
if (image_x > 92){countx = dN_AX; pulsemotor_XR(); T1TCR |= 0x01; while(countx > 0);
position_x -= 250;}
if (image_x < 88){countx = dN_AX; pulsemotor_XL(); T1TCR |= 0x01; while(countx > 0);
position_x += 250;}
break;}
if (code == 'y'){
image_y = getnum();
if (image_y > 92){county = dN_AY; pulsemotor_YR(); T1TCR |= 0x01; while(county > 0);
position_y -= 250;}
if (image_y < 88){county = dN_AY; pulsemotor_YL(); T1TCR |= 0x01; while(county > 0);
position_y += 250;}
break;}
}
if (code == 'z'){
hold = getnum();
feed = getnum();
motorxyz_freq((hold*210)/feed);
motora_freq(((hold*333)/feed)*8);
countz = (333*hold);
counta = (4000*hold);

while(counta > 0)
{
printf("%d\n",feedback());
delay(80);
if(feedback() > 35)
{
Fuzzy();
pulsemotor_AL();
pulsemotor_ZR();
```

```
Check = 0;

T0TCR |= 0x01;
T1TCR |= 0x01;
while(dN_T > 0){}
}
 else
{

pulsemotor_AR();
pulsemotor_ZL();

Check = 1;

T0TCR |= 0x01;
T1TCR |= 0x01;
}
}
while(countz > 0 || counta > 0);
position_z += (hold*333);
motorxyz_freq(freq_xyz);
motora_freq(freq_a);
Last_Seq();
printf("A");
break;
}
case 's':
T0PWMCON &= 0xFE;
T1PWMCON &= 0xF8;
printf("Stop\n");
```

```
break;


case 't':
while(1)
{
printf("%d\n",feedback());
delay(120);
}
}
}
}
```