```c
#include "PWMS12.h"
#include <ME218_C32.h>
#include "ADS12.H"
#include <timers12.h>
#include <stdio.h>

char ADMODE[] = "OOIIIAAA";
int MAXTIME = 600;
short SONGTIME = 200;
char BUTTONTIME = 3;
char RETURNTIME = 20;

char RETURNDUTYCYCLE[] = {85, 75};
char MAXDUTYCYCLE[] = {100, 90};
char MINDUTYCYCLE[] = {70, 60};
unsigned short MID[2] = {445, 515};
char STOPMARGIN = 50;
unsigned short MAXSHORT = 1023;

unsigned char OPENDOWNDUTY = 100, OPENUPDUTY = 100;
unsigned char CLOSEDDOWNDUTY = 100, CLOSEDUPDUTY = 100;
unsigned char UPPAUSE = 105;
unsigned short DOWNTIME = 250;

unsigned short HANDCUTOFF = 350;

static char handState = 0;

enum motor {x = 0, y = 1, open = 3, closed = 0};

void StopMusic(void);
void PressPlayPause(void);
void UpdateMotor(char pin);
unsigned char GetLoPin(char pin);
unsigned char GetHiPin(char pin);
unsigned char GetPWMChannel(char pin);
char CheckForHand(void);
char CheckForCoin(void);
char CheckSWAGState(void);
char CheckForClawButton(void);
void GoClaw(void);
char CheckPosLimit(int pin);
char CheckNegLimit(int pin);
char CheckClosedLimit(void);
char CheckOpenLimit(void);
void RespondToHand(void);
void ReturnToOrigin(void);


void main(void)
{
    PWMS12_Init();
    ADS12_Init(ADMODE);
    TMRS12_Init(TMRS12_RATE_32MS);
    DDRM = 0x00;  //Port M configured as inputs
    DDRT = 0xFF;  //Port T configured as outputs
    PTT &= BIT3LO; //Set MUX to output to x motor.
```

```
    while (1)
    {
        while (1) //Outside game play
        {
            if (CheckForCoin()) break;
            if (CheckForHand()) RespondToHand();
        }

        /* Coin found.  Start game timer (~20s). */
        TMRS12_InitTimer(1,MAXTIME);

        while(1) //Inside game play
        {
            if (TMRS12_IsTimerExpired(2)) //Check for music timer
                StopMusic();

            UpdateMotor(x);
            UpdateMotor(y);

            if (CheckForClawButton())
            {
                GoClaw();
                break;
            }
            if (TMRS12_IsTimerExpired(1))
            {
                ReturnToOrigin();
                break;
            }
        }
    }
}

void StopMusic()
{
    TMRS12_ClearTimerExpired(2);  //Signal music is no longer playing.
    PressPlayPause();
}

void PressPlayPause()
{
    PTAD |= BIT7HI; //Hit pause button.
    TMRS12_InitTimer(3,BUTTONTIME);

    while(!TMRS12_IsTimerExpired(3));
    PTAD &= BIT7LO;  //Release pause button.
}

void UpdateMotor(char pin)
{
    short input, dutyCycle;
    input = ADS12_ReadADPin(pin);

    if (input > MID[pin] + STOPMARGIN) //Positive motion

    {
```

```
        PTT |= GetHiPin(pin+4);
        dutyCycle = MINDUTYCYCLE[pin] + ((long) (input -
MID[pin]))*(MAXDUTYCYCLE[pin] - MINDUTYCYCLE[pin])/(MAXSHORT -
MID[pin]);

          if (CheckPosLimit(pin))
              dutyCycle = 0;
    }
    else if (input < MID[pin] - STOPMARGIN) //Negative motion
    {
        PTT &= GetLoPin(pin+4);
        dutyCycle = MAXDUTYCYCLE[pin]-((long)
input)*(MAXDUTYCYCLE[pin]-MINDUTYCYCLE[pin])/MID[pin];

        if (CheckNegLimit(pin))
            dutyCycle = 0;
    }
    else dutyCycle = 0;

    PWMS12_SetDuty(dutyCycle, GetPWMChannel(pin));
}

char CheckForHand(void)
{
    short input;
    input = ADS12_ReadADPin(2);

    if (TMRS12_IsTimerExpired(2))
        StopMusic();

    if (input < HANDCUTOFF && handState == 0)
        return(handState = 1);

    else if (input >= HANDCUTOFF && handState == 1)
        handState = 0;

    return 0;
}

char CheckForCoin(void)
{
    if (!(PTIAD & BIT3HI))
        return 1;
    else return 0;
}

char CheckForClawButton(void)
{
    if (PTIAD & BIT4HI)
        return 1;
    else return 0;
}

void GoClaw(void)
{
    //Turn off lateral motion
    PWMS12_SetDuty(0, GetPWMChannel(x));
```

```
    PWMS12_SetDuty(0, GetPWMChannel(y));
    PTT |= BIT3HI; //Switch to closed motor

    //Lower (open) claw
    TMRS12_InitTimer(0,DOWNTIME);
    PTT &= BIT6LO;
    PWMS12_SetDuty(OPENDOWNDUTY, GetPWMChannel(open));

    PTT &= BIT7LO;
    PWMS12_SetDuty(CLOSEDDOWNDUTY, GetPWMChannel(closed));

    while(!TMRS12_IsTimerExpired(0));

    PWMS12_SetDuty(0, GetPWMChannel(open));
    PWMS12_SetDuty(0, GetPWMChannel(closed));

    //Start closed motor to make its tether limiting (close claw)
    TMRS12_InitTimer(0,UPPAUSE);
    PTT |= BIT6HI;
    PWMS12_SetDuty(OPENUPDUTY, GetPWMChannel(open));
    while(!TMRS12_IsTimerExpired(0));

    //Start open motor so both tethers are wound
    PTT |= BIT7HI;
    PWMS12_SetDuty(CLOSEDUPDUTY, GetPWMChannel(closed));

    while (!CheckClosedLimit());

    PWMS12_SetDuty(0, GetPWMChannel(closed));
    PWMS12_SetDuty(0, GetPWMChannel(open));

    //Stop vertical motors
    PWMS12_SetDuty(0, GetPWMChannel(closed));
    PWMS12_SetDuty(0, GetPWMChannel(open));

    ReturnToOrigin();

    //Unwind closed tether to make open tether limiting (open claw)
    PTT &= BIT6LO;
    PWMS12_SetDuty(OPENDOWNDUTY, GetPWMChannel(open));
    while(!CheckOpenLimit());
    PWMS12_SetDuty(0,  GetPWMChannel(open));

}

char CheckNegLimit(int pin)
{
    if (pin == x && PTM & BIT0HI)
        return 1;
    else if (pin == y && PTM & BIT2HI)
        return 1;
    else return 0;
}

char CheckPosLimit(int pin)
{
    if (pin == x && PTM & BIT1HI)
```

```
        return 1;
    else if (pin == y && PTM & BIT3HI)
        return 1;
    else return 0;
}

char CheckClosedLimit(void)
{
    if (PTM & BIT5HI)
    return 1;
    else return 0;
}

char CheckOpenLimit(void)
{
    if (PTM & BIT4HI)
    return 1;
    else return 0;
}

void RespondToHand(void)
{
  if (!TMRS12_IsTimerActive(2))
  {
    PTAD |= BIT7HI;
    TMRS12_InitTimer(2,SONGTIME);
    TMRS12_InitTimer(3,BUTTONTIME);

    while(!TMRS12_IsTimerExpired(3));
    PTAD &= BIT7LO;
  }

}

unsigned char GetLoPin(char pin)
{
    switch (pin)
    {
    case 0:
        return BIT0LO;
    case 1:
        return BIT1LO;
    case 2:
        return BIT2LO;
    case 3:
        return BIT3LO;
    case 4:
        return BIT4LO;
    case 5:
        return BIT5LO;
    case 6:
        return BIT6LO;
    case 7:
        return BIT7LO;
    default:
        return 0xFF;
    }
```

```
}

unsigned char GetHiPin(char pin)
{
    switch (pin)
    {
    case 0:
        return BIT0HI;
    case 1:
        return BIT1HI;
    case 2:
        return BIT2HI;
    case 3:
        return BIT3HI;
    case 4:
        return BIT4HI;
    case 5:
        return BIT5HI;
    case 6:
        return BIT6HI;
    case 7:
        return BIT7HI;
    default:
        return 0x00;
    }
}

unsigned char GetPWMChannel(char pin)
{
    switch (pin)
    {
    case 0:
        return PWMS12_CHAN0;
    case 1:
        return PWMS12_CHAN1;
    default:
        return PWMS12_CHAN2;
    }
}

void ReturnToOrigin(void)
{

    PTT &= BIT3LO; //Switch to x motor

    /* If positioned against left wall, move right slightly
     * to prevent getting stuck in back right corner. */
    if(CheckNegLimit(x))
    {
        PTT |= BIT4HI;
        PWMS12_SetDuty(RETURNDUTYCYCLE[x],GetPWMChannel(x));
        TMRS12_InitTimer(0,RETURNTIME);
        while(!TMRS12_IsTimerExpired(0));
    }
    PWMS12_SetDuty(0, GetPWMChannel(x));

    //Start horizontal motors to return to negative corner
```

```
    PTT &= BIT4LO & BIT5LO;
    PWMS12_SetDuty(RETURNDUTYCYCLE[y], GetPWMChannel(y));
    while(!CheckNegLimit(y));

    PWMS12_SetDuty(0, GetPWMChannel(y));
    PWMS12_SetDuty(RETURNDUTYCYCLE[x], GetPWMChannel(x));
    while(!CheckNegLimit(x));

    PWMS12_SetDuty(0, GetPWMChannel(x));
}
```