```
// Dit programma kan geprogrammeerd worden in een PIC 12F683
// en heeft als doel: het testen van modelbouwservo's. De (twee)
// servo's kunnen worden ingesteld met behulp van twee trimmers.
//
// Ruben Buysschaert, 4 februari 2011.
//
// www.rubu.be

#include "int16CXX.h"           //Deze header is nodig om enkele standaard routines
                                //voor tijdens de interrupt te kunnen gebruiken.
                                //Bijvoorbeeld: int_save_registers
#include "math16.h"

//Configuration word instellingen
#pragma config |= 0x00D4        // 0000.0000.1101.0100 = 0x00D4 gebruik
                                // interne oscillator van 8Mhz

//Definities van de in- en uitgangen
#pragma bit ch1     @ GPIO.5    //output servo 1
#pragma bit ch2     @ GPIO.4    //output servo 2
                                //trimmers op AN1 en AN2
#pragma bit testpin @ GPIO.0    //toont het wachten van ongeveer 100ns,
                                //vóór het starten van de AD-conversie


//Prototypes van de functies
void init(void);
void initTimer1(void);          //Let op: deze functie werkt met Timer 0!
void Timer1On(unsigned char pwm);
void Timer1Off(void);
void initTimer2(void);          //Let op: deze functie werkt met Timer 1!
void Timer2On(void);
void Timer2Off(void);
void initAd(void);
void startAd(unsigned char tempKanaal);
```

```
void wait100ns(void);


//Declaraties van de variabelen
unsigned char kanaal, adresult;

//Interrupt definitie
#pragma origin 4
interrupt interruptSubroutine(void)
{
    int_save_registers          //zie int16CXX.h

    //Interrupt van timer 1 (iedere 7,5ms)
    if((TMR1IF == 1) && (TMR1IE == 1))
    {
        TMR1IF = 0;
        Timer2On();             //her instellen op 7,5ms

        startAd(kanaal);        //AD conversie starten

        if(kanaal == 1)         //kanaal wisselen
            kanaal = 2;
        else
            kanaal = 1;
    }

    //Interrupt van AD conversie
    if(ADIF == 1)
    {
        ADIF = 0;

        adresult = ADRESH/2;        //256/2 = 128 => ongeveer de maximum variatie om timer1 1ms te
laten variëren
                                    //ADRES is left justified!
```

```
        if(kanaal == 2)
        {
            Timer1On(adresult);     //timer starten met waarde tussen 1ms en 2ms
            ch1 = 1;                //ch1 hoog maken
        }

        if(kanaal == 1)
        {
            Timer1On(adresult);     //timer starten met waarde tussen 1ms en 2ms
            ch2 = 1;                //ch2 hoog maken
        }
    }


    //Interrupt van timer 0
    if((T0IF == 1) && (T0IE == 1))
    {
        T0IF = 0;
        Timer1Off();

        ch1 = 0;                    //ch1 en ch2 laag maken
        nop();
        ch2 = 0;
    }

    int_restore_registers
}


//Start van de main functie
void main( void)
{
    init();

    //initialisaties
```

```
    kanaal = 1;
    ch1 = 0;
    nop();
    ch2 = 0;

    while(1)
    {
        nop();              //Leeg, cool! Leve de interrupt...!  ;-)
    }
}

void init(void)
{
    OSCCON.6 = 1;           //kies 8MHz
    OSCCON.5 = 1;
    OSCCON.4 = 1;

    CM2 = 1;                //comparatoren uitschakelen
    CM1 = 1;
    CM0 = 1;

    ANSEL = 0;              //Analoge ingangen uitschakelen

    TRISIO = 0b00001110;    //input-output instellen

    GIE = 1;                //interrupts toelaten
    GPIE = 0;               //interrupts van GPIO's niet toelaten
    PEIE = 1;               //interrupts van pheriperals toelaten

    initTimer1();
    initTimer2();
    initAd();

    Timer2On();             //PWM opstarten (zorgt voor puls van 7,5ms)
}
```

```
void initTimer1()
{
    T0CS = 0;               //interne klok selectern
    PSA = 0;                //prescaler gebruiken
    PS0 = 1;                //instellen op 1:16
    PS1 = 1;
    PS2 = 0;
    TMR0 = 0;               //teller resetten
    T0IF = 0;               //IF clearen
    T0IE = 0;               //interrupt nog niet toelaten
}

void Timer1On(unsigned char pwm)
{
    // Een kloksignaal van 8MHz/4 = 2MHz => 0,5µs per tel,
    // met prescaler van 16 => één tel van Timer 0 = 0,5µs * 16 = 8µs

    // 1ms => 1ms/8µs = 125 pulsen => instellen op 255 - 125 = 130 als init waarde
    // 2ms => 2ms/8µs = 250 pulsen => instellen op 255 - 250 = 5 als init waarde
    // dus 125 pulsen verschil tussen 1ms en 2ms, dus pwm signaal mag variëren tussen 0 en 125

    unsigned char temp;

    temp = 130;     //minimum 1ms pulsbreedte

    if(pwm > 0)
    {
        if(pwm < 130)
            temp = temp - pwm;
        else
            temp = 0;       //maximum pulsbreedte als temp = 0
    }

    TMR0 = temp;            //teller resetten
```

```
    T0IF = 0;               //IF clearen
    T0IE = 1;               //interrupt toelaten
}

void Timer1Off()
{
    T0IE = 0;               //interrupt niet meer toelaten
    T0IF = 0;               //IF clearen
    TMR0 = 0;               //teller resetten
    //Let op: De timer is niet uitgeschakeld! Vandaar dat in de interrupt
    //subroutine, ook gecontrolleerd moet worden of T0IE == 1 ...
}


void initTimer2()
{
    TMR1CS = 0;             //interne klok selecteren
    T1CKPS1 = 1;            //prescaler op 1:8
    T1CKPS0 = 1;
    T1GE = 0;
    TMR1H = 0;              //teller resetten
    TMR1L = 0;
    TMR1ON = 1;             //timer inschakelen
    TMR1IF = 0;             //interrupt flag resetten
    TMR1IE = 0;             //interrupt nog niet toelaten
}

void Timer2On()
{
    // Je wil 7,5ms meten. Aan een kloksignaal van 8MHz/4 = 2MHz => 0,5µs per tel
    // met prescaler van 8 => één tel van Timer 0 = 0,5µs * 8 = 4µs
    // 7,5ms/4µs = 1875 pulsen
    // 65536 - 1875 = 63661 als init waarde
    // = 11111000.10101101
    TMR1L = 0b10101101;
```

```
    TMR1H = 0b11111000;
    TMR1IF = 0;
    TMR1IE = 1;             //interrupt toelaten
}

void Timer2Off()
{
    TMR1IE = 0;             //interrupt niet meer toelaten
    TMR1IF = 0;
    TMR1H = 0;
    TMR1L = 0;
    //Let op: De timer is niet uitgeschakeld! Vandaar dat in de interrupt
    //subroutine, ook gecontrolleerd moet worden of TMR1IE == 1 ...
}

void initAd(void)
{
    ADFM = 0;       //LEFT justified!!  8 MSB's in ADRESH
    VCFG = 0;       //Vref = VDD

    ADCS2 = 0;      //Fosc/32 als AD klok
    ADCS1 = 1;
    ADCS0 = 0;

    ANS0 = 0;       //digitale I/O
    ANS1 = 1;       //analoge I/O
    ANS2 = 1;       //analoge I/O
    ANS3 = 0;       //digitale I/O

    CHS0 = 1;       //selecteer AN1 als ingang
    CHS1 = 0;

    ADON = 1;       //AD inschakelen
    ADIF = 0;       //interrupt flag resetten
    ADIE = 1;       //AD interrupt enable
```

```
    PEIE = 1;       //pheriperal interrupts toelaten
}

void startAd(unsigned char tempKanaal)
{
    if(tempKanaal == 1)
    {
        CHS0 = 1;       //selecteer AN1 als ingang
        CHS1 = 0;
        wait100ns();    //na het wijzigen van het analoog kanaal, moet je
                        //een wachttijd respecteren (typisch 100ns)
        GO = 1;         //start de AD-conversie
    }

    if(tempKanaal == 2)
    {
        CHS0 = 0;       //selecteer AN2 als ingang
        CHS1 = 1;
        wait100ns();    //na het wijzigen van het analoog kanaal, moet je
                        //een wachttijd respecteren (typisch 100ns)
        GO = 1;         //start de AD-conversie
    }
}

void wait100ns(void)
{
    //experimenteel vastgesteld (met scoop)
    char teller = 0;

    testpin = 1;

    for(teller = 0; teller < 25; teller++)
        nop();

    testpin = 0;
```

```
}
```