```
/*************************************************************
TOTAAL TESTPROGRAMMA PIC DEVELOPMENT SYSTEM V1.2
Bart Huyskens - barthuyskens@telenet.be - juni 2013

Connections:

-8 leds aan POORTD
-Buzzer aan RB5
-LCD (B7-A3)(B6-A2)(B5-A1)(B4-A0)(EN-A4)(RS-A5)

*************************************************************/

#include <htc.h>
#include "HITECH_LCD.h"     // include LCD library

__CONFIG(0x23F2);           //extern Xtal HS 19660800
__CONFIG(0x3FFF);           //

#define _XTAL_FREQ  19660800
#define MAXMENU  6
#define TMR1H_FREQ (_XTAL_FREQ/(4*1*256))   // RC5- Xtal freq / 4 / prescaler 1:1 / TMR1L
(256) = 19200Hz = 52.08usec
#define BITFREQ  751                        // RC5- 75% of bittime = 889 + (889/2) =
1,331msec = 751,31Hz
#define RELBIT (TMR1H_FREQ/BITFREQ)         // RC5- 25 - 1,331msec / 52.08usec = 25 - TMR1H
zal op 25 staan na 1.331msec

//*****************Variables
*****************************************************************************/
unsigned char SWB1, SWB2, PROGRAM, PROGRAMOLD, LEFT, RIGHT = 0;
// global variables - for use in all functions of NEC code
unsigned int TMR1HCOPY, TMR1LCOPY, TMR1X;
unsigned char situatie, START, NEC, RC5X = 0;
unsigned char x,BIT, ADRES,ADRESINV,DATA, DATAINV = 0;
// global variables - for use in all functions of RC5 code
unsigned char ADRES , DATA = 0;
unsigned char BITCOUNTER = 0;
unsigned int RC5CODE_16BIT, RC5CODE_DONE = 0;

//*************Function prototypes
*********************************************************************/

void BEEPH (void);
void BEEPL (void);
void BEEPSHORT (void);
void LEDS_SOUND();
void SWITCH_POTM();
void AUDIO();
void RC_NEC();
void RC_RC5();
void DC_MOTOR();

//************* Main Program
*************************************************************************/
void main()
{
TRISD = 0x00;               // leds aan poortD = output
PORTD = 0x00;
```

```
TRISC = 0x00;               // leds aan poortD = output
PORTC = 0x00;
ANSEL = 0x00;               // Alle Analoge pins als digitale pin
ANSELH = 0X00;
TRISB = 0b00011111;         //RB0 externe interrupt + 4xswitch
LCD_Start();                // Initialiseer LCD

//*************** START MENU ROUTINE
************************************************************/
while(1)
{
    while (!RB2)            // zolang er niet op knop RB2 gedrukt wordt
        {
        if (RB1)            // als er op knop RB1 gedrukt wordt
            {
            if (PROGRAM >= MAXMENU) PROGRAM = 0;    // MAXMENU is het aantal menu's dat er
            zijn - zie #define MAXMENU
            else PROGRAM = PROGRAM + 1;             // hiermee wordt het menu geslecteeerd
            BEEPH();                                // hoge beeptoon
            LCD_Clear();
            __delay_ms(100); // om te vermijden dat er met één druk op de knop meerdere
            stappen genomen worden
            }
//**** MENU routine om tekst op het LCD te
zetten*************************************************/
    switch(PROGRAM)   // afhankelijk van welke waarde de variabele PROGRAM heeft
    {
    case 0:
    {
    LCD_PrintNumber(PROGRAM);
    LCD_PrintString(" START DEMO");
    LCD_Cursor(5, 1);
    LCD_PrintString("      B1=>>");
    break;
    }

    case 1:
    {
    LCD_PrintNumber(PROGRAM);
    LCD_PrintString(" LEDS+SOUND");
    LCD_Cursor(5, 1);
    LCD_PrintString("B2=OK B1=>>");
    break;
    }

    case 2:
    {
    LCD_PrintNumber(PROGRAM);
    LCD_PrintString(" SWITCH+POTM");
    LCD_Cursor(5, 1);
    LCD_PrintString("B2=OK B1=>>");
    break;
    }

    case 3:
    {
    LCD_PrintNumber(PROGRAM);
    LCD_PrintString(" AUDIO");
```

```
    LCD_Cursor(5, 1);
    LCD_PrintString("B2=OK B1=>>");
    break;
    }

    case 4:
    {
    LCD_PrintNumber(PROGRAM);
    LCD_PrintString(" RC NEC");
    LCD_Cursor(5, 1);
    LCD_PrintString("B2=OK B1=>>");
    break;
    }

    case 5:
    {
    LCD_PrintNumber(PROGRAM);
    LCD_PrintString(" RC RC5");
    LCD_Cursor(5, 1);
    LCD_PrintString("B2=OK B1=>>");
    break;
    }

    case 6:
    {
    LCD_PrintNumber(PROGRAM);
    LCD_PrintString(" DC MOTOR");
    LCD_Cursor(5, 1);
    LCD_PrintString("B2=OK B1=>>");
    break;
    }

    default:
    {
    LCD_PrintNumber(PROGRAM);
    break;
    }
    }
    } // einde loop "zolang er niet op knop RB2 gedrukt wordt

// Als er dus wel op RB2 gedrukt wordt - komen we in deze code
    BEEPL(); // lage beep
    switch (PROGRAM)  // hier worden de eigenlijke subroutines met de respectievelijke
    subprogramma's aangeroepen
    {
    case 1:
        {
        LEDS_SOUND();
        break;
        }
    case 2:
        {
        SWITCH_POTM();
        break;
        }
    case 3:
        {
        AUDIO();
```

```
        break;
        }
    case 4:
        {
        RC_NEC();
        break;
        }
    case 5:
        {
        RC_RC5();
        break;
        }
    case 6: DC_MOTOR();

    }
__delay_ms(250);  // Delay die er voor zorgt dat er niet meteen naar de volgende loop
gesprongen wordt

    }// einde while(1) loop
}   // einde void main ftie

//*****************************************************************/
//  SUBPROCEDURES                                                  //
//*****************************************************************//

//**** BEEP FUNCTIES ****************************************//
void BEEPH (void) // Hoge beeptoon
{
for (int x = 0; x < 500; x++)
    {
    RB5 = ~RB5;
    __delay_us(200);
    }
}

void BEEPL (void)  // lage beeptoon
{
for (int x = 0; x < 300; x++)
    {
    RB5 = ~RB5;
    __delay_us(400);
    }
}

void BEEPSHORT (void)  // korte beeptoon
{
for (int x = 0; x < 200; x++)
    {
    RB5 = ~RB5;
    __delay_us(200);
    }
}

//****SUB PROGRAMMA'S*********************************************/
// Looplicht op Leds - PORTD + Beep op Buzzer aan RB5
// Looplicht staat ook op POORTC - testen uitgangen SUBD
void LEDS_SOUND()
{
```

```
char x = 1;
LCD_Cursor(0, 1);
LCD_PrintString("      EXIT B1=<<");
while (RB1 ==0)
    {
        while ((x < 128)&&(RB1 ==0))  // Met RB1 kan loop direct onderbroken worden
        {
        x = x << 1;
        PORTD = x;
        PORTC = x;
        BEEPSHORT();
        }
        while ((x > 1)&&(RB1 ==0))
        {
        x = x >> 1;
        PORTD = x;
        PORTC = x;
        BEEPSHORT();
        }
    }
PORTD = 0;
PORTC = 0;
LCD_Clear();
BEEPH();
}
//*****************************************************************//
// SWB4 aan PIN B4 en SWB3 aan pin B3
// POTM hangt aan PIN RE0 - 3 waarden worden weergegeven op LCD

void SWITCH_POTM()
{
char x = 1;
while (RB1 ==0)
    {
    LCD_Cursor(0, 0);
    LCD_PrintString("SWB4=");
    LCD_PrintNumber(RB4);
    LCD_PrintString("  SWB3=");
    LCD_PrintNumber(RB3);

    TRISE = 0b00000001;     // RE0 = input
    ANSEL = 0b00100000;     // PINA1= AD input - alle andere normal IO
    ADCON0 = 0b00010101;    //ADCS1 ADCS0 CHS3 CHS2 CHS1 CHS0 GO/DONE ADON
    ADCON1 = 0b00000000;    //Meting tussen VDD en VSS- Left justified

    __delay_us(20);         //Acquisition time om C op te laden
    ADCON0 = ADCON0 | 0b00000010;   // GO/DONE = 1 met masker
    while (ADCON0 & 0b00000010)     // zolang AD omzetting bezig is
    {
    }
    PORTD = ADRESH;                 //toon 8 bit resultaat van AD omzetting op leds

    LCD_Cursor(0, 1);

    LCD_PrintString("POTM:");
    LCD_PrintNumber(ADRESH);
    LCD_PrintString("     ");
    LCD_Cursor(11, 1);
```

```
    LCD_PrintString("B1=<<");
    }
PORTD = 0x00;                   // alle leds terug uit
LCD_Clear();
BEEPH();
}

//*****************************************************************//
// Via Jack connector wordt audio (L+R)ingelezen op AN6 en AN7
// resultaat wordt als VU meter L + R weergegeven op LCD
void AUDIO()
{
unsigned char LCD_RIGHT, LCD_LEFT  = 0;
do
{
LCD_Cursor(0, 0);
LCD_PrintString("CONN L:AN6-R:AN7");
LCD_Cursor(0, 1);
LCD_PrintString("     OK-PRESS B2");
__delay_ms(250);

}
while (RB2 == 0);  // zoalng niet op B2 gedrukt wordt
    BEEPL();

    TRISE = 0b00000110;     // RE1, RE2 = input
    ANSEL = 0b11000000;     // PINA6 en AN7= AD input - alle andere normal IO
                            // ANSEL :  ANS7(2) ANS6(2) ANS5(2) ANS4 ANS3 ANS2 ANS1 ANS0
    ADCON1 = 0b00000000;    //Meting tussen VDD en VSS- Left justified

while (RB1 ==0)
    {
    ADCON0 = 0b00011001; //ACTIVATE AN6 - ADCON0: ADCS1 ADCS0 CHS3 CHS2 CHS1 CHS0 GO/DONE ADON
    __delay_us(20);         //Acquisition time om C op te laden
    ADCON0 = ADCON0 | 0b00000010;  // GO/DONE = 1 met masker
    while (ADCON0 & 0b00000010) // zolang AD omzetting bezig is
    {
    }
    LEFT = ADRESH; //toon 8 bit resultaat van AD omzetting op leds

    ADCON0 = 0b00011101; //ACTIVATE AN7 - ADCON0: ADCS1 ADCS0 CHS3 CHS2 CHS1 CHS0 GO/DONE ADON
    __delay_us(20);         //Acquisition time om C op te laden
    ADCON0 = ADCON0 | 0b00000010;  // GO/DONE = 1 met masker
    while (ADCON0 & 0b00000010) // zolang AD omzetting bezig is
    {
    }
    RIGHT = ADRESH; //toon 8 bit resultaat van AD omzetting op leds

    LCD_RIGHT = RIGHT / 16;     // schaal waarde tss 0 en 255 naar 0-16
    LCD_LEFT = LEFT / 16;       // schaal waarde tss 0 en 255 naar 0-16
    LCD_Clear();
    LCD_Cursor(0, 0);
    for (char x = 0; x < LCD_RIGHT; x++) // Aantal tekens op LCD is evenredig met geluid
    {
    LCD_PrintASCII(0x52); // ascii karakter voor 'R'
    }
    LCD_Cursor(0, 1);
    for (char x = 0; x < LCD_LEFT; x++) // Aantal tekens op LCD is evenredig met geluid
```

```
    {
    LCD_PrintASCII(0x4C); // ascii karakter voor 'L'
    }
    __delay_ms(10);
    }
LCD_Clear();
BEEPH();
}

/************************************************************
NEC IR decoder programma - Bart Huyskens - SJI Schoten - 23/04/2013

Sensor: TSOP1236 - aangesloten aan PIN RB0 (Externe interrupt)
deze sensor inverteert wel het resultaat... rekening mee houden - reactie op dalende flank!!

Timer 1 wordt gebruikt om de tijd tussen twee dalende flanken te meten

- Xtal: 19660800
- Prescaler: 8 - increment periode= 1.627usec

situaties:
0 - TMR1 overflow (interrupt) - NO KEY PRESSED +/-== 65536
1 - 13.5 msec tussen 2 neg flanken - START      (8294 x 1.627usec)  7500<time<9000
2 - 11.25 msec tussen 2 neg flanken - REPEAT    (6912 x 1.627usec)  5000<time<7499
3 - 2.25 msec tussen 2 neg flanken - "1"        (1382 x 1.627usec)  1050<time<1700
4 - 1.125 msec tussen 2 neg flanken - "0"       (691 x 1.627usec)   300<time<1000

!!!  DEZE ROUTINE GEBRUIKT 2 INTERRUPT ACTIES  - zie void interrupt isr(void)

************************************************************/
void RC_NEC()
{
    do
    {
    LCD_Cursor(0, 0);
    LCD_PrintString("CONN IR-RC5>RB0");
    LCD_Cursor(0, 1);
    LCD_PrintString("NEC  OK-PRESS B2");
    __delay_ms(250);
    }
    while (RB2 == 0);
    BEEPL();

        NEC = 1;                    // indicatie vr interrupt routine om NEC interrupts te
        nemen
        TRISD = 0x00;               // leds aan PORTD output
        TRISB = 0b00011111;         // RB0 (+ RB1-RB4) = input (interrupt)
        ANSEL = 0x00;               // all pins digital (not analog)
        ANSELH = 0x00;              // all pins digital (not analog)

        T1CON = 0b00110001;         //TMR1 - prescaler /8 - internal CLK - Start - timer
        loopt tegen 614.400Hz
        TMR1IE = 1;                 //TMR1 interrupt enable
        TMR1L = 0;
        TMR1H = 0;
        PEIE = 1;                   // Peripheral int enable

        OPTION_REG = 0b00000000;    // INT on BO on Falling edge (Bit 6) (voor oudere
```

```
        versies is dit OPTION) OPTION_REG = 0b01000000;
        INTE = 1;                   // enable External int on B0
        GIE = 1;                    // enable global interrupts

while (RB1 ==0)
    {


        LCD_Clear();
        LCD_PrintString("ADR:");
        LCD_PrintNumber(ADRES);
        LCD_PrintString(" ");
        LCD_Cursor(8, 0);
        LCD_PrintString("/ADR:");
        LCD_PrintNumber(ADRESINV);


        LCD_Cursor(0, 1);
        LCD_PrintString("DAT:");
        LCD_PrintNumber(DATA);
        LCD_PrintString(" ");
        LCD_Cursor(8, 1);
        LCD_PrintString("/DAT:");
        LCD_PrintNumber(DATAINV);

        __delay_ms(50);     // flikkering LCD wegwerken
     }
NEC = 0;            // NEC interrupts terug afzetten
GIE = 0;            // Disable global interrupt
LCD_Clear();
BEEPH();
}

/************************************************************
RC5 decoder programma - Bart Huyskens - SJI Schoten - 16/06/2013

Sensor: TSOP1236 - aangesloten aan PIN RB0 (Externe interrupt)
Deze Sensor levert een geinverteerd signaal aan pin RB0
Connecteer:
TSOP1236 aan RB0
leds aan POORTD
LCD (B7-A3)(B6-A2)(B5-A1)(B4-A0)(EN-A4)(RS-A5)

Principe:
- Via externe int aan RB0 wordt de eerste (dalende) flank gedetecteerd
- 1.331msec later = 889 + (889/2) wordt gekeken of het signaal hoog of laag is. hoog = 0,
laag = 1
- die 1.331 msec wordt gemeten via het TMR1H register van TMR1 - DIt TMRH register staat op
25 als 1.331msec gepasseerd is.
- dit wordt herhaald voor alle 14 bits - deze 14 bits schuiven één voor één in het
RC5CODE_16BIT register.
- onvangen bericht wordt zichtbaar gemaakt op de LCD en op de leds aan poort D

!!!  DEZE ROUTINE GEBRUIKT 2 INTERRUPT ACTIES  - zie void interrupt isr(void)

************************************************************/
void RC_RC5()
{
```

```
    do
    {
    LCD_Cursor(0, 0);
    LCD_PrintString("CONN IR-RC5>RB0");
    LCD_Cursor(0, 1);
    LCD_PrintString("RC5  OK-PRESS B2");
    __delay_ms(250);
    }
    while (RB2 == 0);
BEEPL();

        RC5X = 1; // om juiste interrupt rourtine op te starten

        TRISD = 0x00;               // leds aan poortD = output
        PORTD = 0x00;
        ANSEL = 0x00;
        ANSELH = 0X00;
        TRISB = 0b00011111;         // RB0 (+ RB1-RB4) = input (interrupt)


        // TMR1 settings
        TMR1L = 0;
        TMR1H = 0;
        T1CON = 0x00;
        TMR1IE=1;                   // Enable TMR1 overflow interrupt - geeft aan dat er
        13msec gepasseerd zijn

        // External Interrupt setting - RB0
        INTEDG=0;                   //falling edge - vermits de TSOP sensor een inver
        signaal geeft - zoeken we de eerste dalende flank.
        INTE = 1;                   // Enable externe interrupt
        PEIE = 1;                   // Enable peripheral interrupt
        GIE = 1;                    // Enable global interrupt

while (RB1 ==0)
    {
        DATA = RC5CODE_DONE & 0b0000000000111111;               // 6 LSB's zijn de data
        ADRES = (RC5CODE_DONE>>6)&0b0000000000011111;           // Bit 6-10 zijn het adres

        LCD_Clear();
        LCD_PrintString("RC5 DECODER");
        LCD_Cursor(0, 1);
        LCD_PrintString("ADR:");
        LCD_PrintNumber(ADRES);
        LCD_Cursor(8, 1);
        LCD_PrintString("DAT:");
        LCD_PrintNumber(DATA);
        __delay_ms(50);     // flikkering LCD wegwerken

    }

RC5X = 0; // RC5 interrupts terug uitschakelen
GIE = 0; // schakel interrupts terug uit
LCD_Clear();
BEEPH();
}

//*****************************************************************//
```

```
// DC motor - SLuit als volgt aan:
// L293D input  : 1A>C0, 2A>C1, 3A>C2, 4A>C3
// L293 output  : Motor1 tussen 1Y en 2Y, Motor2 tussen 3Y en 4Y

void DC_MOTOR()
{
    do
    {
    LCD_Cursor(0, 0);
    LCD_PrintString("1A>RC0 2A>RC1");
    LCD_Cursor(0, 1);
    LCD_PrintString("DC1 OK-PRESS B2");
    __delay_ms(250);

    }
    while (RB2 == 0);
    BEEPL();

    do
    {
    LCD_Cursor(0, 0);
    LCD_PrintString("3A>RC2 4A>RC3");
    LCD_Cursor(0, 1);
    LCD_PrintString("DC2 OK-PRESS B2");
    __delay_ms(250);

    }
    while (RB2 == 0);
    BEEPL();

unsigned char Duty_Cycle1,Duty_Cycle2  = 0;  // variabele voor Duty Cycle
TRISC = 0x00;   // leds aan PORTC - Output
CCP1CON = 0b00001100;   //CCP1 in PWM mode zetten
CCP2CON = 0b00001100;   //CCP1 in PWM mode zetten
T2CON = 0b00000100;  // Timer2 aan - prescaler op 1 = hoogste frequentie via TMR2
PR2 = 255;  // periode register op maximum = lage PWM freq

while (RB1 ==0)
    {

        RC0 = RB3;
        RC3 = RB4;
        CCPR1L = Duty_Cycle1;   // copieer inhoud van var Duty_Cycle naar CCPR1L buffer
        register
        CCPR2L = Duty_Cycle2;
        Duty_Cycle1++;      // increment Duty_Cycle
        Duty_Cycle2--;
        __delay_ms(20);     // wacht 20 msec - totale loop = 255 x 20msec = 5.1 seconden

        LCD_Clear();
        LCD_PrintString("PWM");
        LCD_Cursor(0, 1);
        LCD_PrintString("CCP1:");
        LCD_PrintNumber(Duty_Cycle1);
        LCD_Cursor(8, 1);
        LCD_PrintString("CCP2:");
        LCD_PrintNumber(Duty_Cycle2);
        __delay_ms(50);     // flikkering LCD wegwerken
```

```
    }
LCD_Clear();
RC0 = 0;    // stop all motors
RC3 = 0;
CCPR1L = 0;
CCPR2L = 0;
BEEPH();
}


/* Demo for a good loop - just copy -paste
void XXX()
{
    do
    {
    LCD_Cursor(0, 0);
    LCD_PrintString("CONN IR-RC5>RB0");
    LCD_Cursor(0, 1);
    LCD_PrintString("RC5  OK-PRESS B2");
    __delay_ms(250);

    }
    while (RB2 == 0);


while (RB1 ==0)
    {

        LCD_Clear();
        LCD_PrintString("RC5 DECODER");
        LCD_Cursor(0, 1);
        LCD_PrintString("ADR:");
        LCD_PrintNumber(ADRES);
        LCD_Cursor(8, 1);
        LCD_PrintString("DAT:");
        LCD_PrintNumber(DATA);
        __delay_ms(50);     // flikkering LCD wegwerken

    }
LCD_Clear();
}
*/

//*************************************************************************************************
*******************************//
//*************************************************************************************************
*******************************//
/***** INTERRUPT SERVICE ROUTINE *****/

void interrupt isr(void)
{

if((TMR1IF)&&(RC5X))                    //check the timer overflow voor RC5 routine - elke
+/-13msec
    {
        TMR1IF=0;                       // reset de interrupt flag
        TMR1ON=0;                       // stop de timer
```

```
        INTEDG=0;                       // een nieuw RC5 signaal begint steeds met een
        dalende flank
        RC5CODE_DONE = RC5CODE_16BIT;   // Als er 13msec geen flank gedetecteerd is, dan
        nemen we aan dat de data binnen is
                                        // Deze data wordt gecopieerd naar het 16 bit
                                        RC5CODE_DONE register
        PORTD =RC5CODE_DONE & 0b00111111;           // Zet 6 lsb's (data)op de leds
    }


else if ((TMR1IF) && (NEC))             // check the timer overflow voor NEC routine
    {                                   // geeft aan dat er geen knop is ingedrukt
    TMR1IF = 0;                         // clear TMR1 int flag
    situatie = 0;                       // situatie 0 - zie boven
    TMR1H = 0x00;                       // clear TMR1L
    TMR1L = 0x00;                       // clear TMR1H
    }

else if ((INTF)&&(RC5X))                // controleer of er een flank is op RB0 voor de RC5
routine
    {
        RD2 = 1;
        INTF=0;                         // reset the interrupt
        INTEDG = !INTEDG;               // verander de flankgevoeligheid van hoog naar laag
        of omgekeerd

        if(!TMR1ON)                     // ALs de timer nog niet aan staat, dan betekent dit
        dat de eerste bit van het RC5 signaal wordt binnengehaald
        {
            BITCOUNTER=13;              // zet de bitcounter op 13 om alle 14 bits binnen te
            halen
            TMR1H=0;                    // reset the timer
            TMR1L=0;                    //
            TMR1ON=1;                   // start the timer
        }

        else                            // doe dit als het niet de eerste bit is die je
        binnen haalt
        {
        if(TMR1H>RELBIT)                // controleer of er reeds 1331 uS gepasseerd zijn -
        als dit zo is, haal dan deze nieuwe bitwaarde binnen
            {
                RC5CODE_16BIT = RC5CODE_16BIT <<1;          // Schuif alle bits van de 16
                bit RC5CODE_16BIT één plaats naar links
                if(!RB0)                                    // als RB0 op dit moment 0 is,
                dan moet er een 1 in het RC5CODE_16BIT register geladen worden
                    {
                    RC5CODE_16BIT = RC5CODE_16BIT | 0b0000000000000001;     //  // laad een
                    1 in de LSB van het RC5CODE_16BIT gerister
                    }
                BITCOUNTER--;           // decrement de BITCOUNTER
                TMR1ON=0;               // off timer
                TMR1H=0;                // reset the timer
                TMR1L=0;                //
                TMR1ON=1;               // start the timer
                PORTD = 0;              // alle leds uit
            }
        }
```

```
    }

else if ((INTF) && (NEC))// interrupt op RB0 (dalende flank op RB0) - voor de NEC routine
    {
    RD3 = 1;
    INTF = 0;               // Clear RB0 INT flag
    TMR1LCOPY = TMR1L;      // copieer TMR1L
    TMR1HCOPY = TMR1H;      // copieer TMR1H
    TMR1H = 0x00;           // clear TMR1L
    TMR1L = 0x00;           // clear TMR1H
    TMR1X = ((TMR1HCOPY << 8)+(TMR1LCOPY)); // 16 bit samenstelling

        if ((TMR1X > 7500)&(TMR1X < 9000))  // detect situatie 1: START
        {
        START = 1;
        x = 0;
        situatie = 1;
        ADRES = 0;
        ADRESINV = 0;
        DATA = 0;
        DATAINV = 0;
        }

        else if ((TMR1X > 5000)&(TMR1X < 7499)) // detect situatie 2: REPEAT
        {
        situatie = 2; // repeat
        RD0 = ~RD0;
        }

        else if ((TMR1X > 1050)&(TMR1X < 1700)) // detect situatie 3: "1"
        {
        situatie = 3; //"1"
        BIT = 1;
        }

        else if ((TMR1X > 300)&(TMR1X < 1000)) // detect situatie 4: "0"
        {
        situatie = 4; //"0"
        BIT = 0;
        }

        else situatie = 5;  // alle andere situaties (zou niet mogen voorkomen)


        if ((situatie == 3)|||(situatie == 4))  // als er een "1" of een "0" binnenkomt
        {
            if (x<8) // Bit by Bit laden van Byte ADRES
                    {
                    ADRES = ADRES | BIT;
                    if (x < 7) ADRES = ADRES << 1;  // laatste bit mag niet geshift worden
                    x++;
                    }
            else if (x<16)// Bit by Bit laden van Byte ADRESINV
                    {
                    ADRESINV = ADRESINV | BIT;
                    if (x< 15) ADRESINV = ADRESINV << 1; // laatste bit mag niet geshift
                    worden
                    x++;
```

```
                }
            else if (x<24)// Bit by Bit laden van Byte DATA
                {
                DATA = DATA | BIT;
                if (x< 23) DATA = DATA << 1; // laatste bit mag niet geshift worden
                x++;
                }
            else if (x<32)// Bit by Bit laden van Byte DATAINV
                {
                DATAINV = DATAINV | BIT;
                if (x< 31) DATAINV = DATAINV << 1; // laatste bit mag niet geshift worden
                x++;
                }
        }
        }
} // einde void interrupt
```