**NEC infrared tranmission protocol**

Bart Huyskens – april 2013

In tegenstelling tot wat veel beweerd wordt werken de meeste RC die ik getest heb (afstandbedieningen voor TV’s, DVD, …) niet volgens het RC5 protocol, maar wel volgens het veel mminder goed gedocumenteerde NEC protocol (of een variant er van…). (test RC best via logic analyser!!!)

Info van: <http://wiki.altium.com/display/ADOH/NEC+Infrared+Transmission+Protocol>

The NEC IR transmission protocol uses pulse distance encoding of the message bits. Each pulse burst (mark – RC transmitter ON) is 562.5µs in length, at a carrier frequency of 38kHz (26.3µs). Logical bits are transmitted as follows:

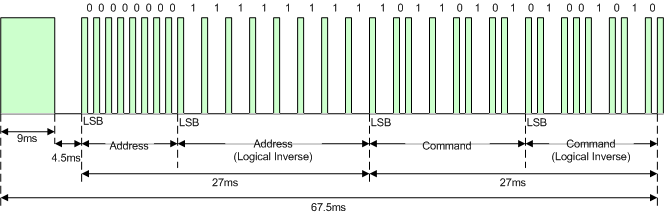
* Logical '0' – a 562.5µs pulse burst followed by a 562.5µs space, with a total transmit time of 1.125ms
* Logical '1' – a 562.5µs pulse burst followed by a 1.6875ms space, with a total transmit time of 2.25ms

When transmitting or receiving remote control codes using the NEC IR transmission protocol, the WB\_IRRC performs optimally when the carrier frequency (used for modulation/demodulation) is set to 38.222kHz.

When a key is pressed on the remote controller, the message transmitted consists of the following, in order:

* a 9ms leading pulse burst (16 times the pulse burst length used for a logical data bit)
* a 4.5ms space
* the 8-bit address for the receiving device
* the 8-bit logical inverse of the address
* the 8-bit command
* the 8-bit logical inverse of the command
* a final 562.5µs pulse burst to signify the end of message transmission.

The four bytes of data bits are each sent least significant bit first. Figure 1 illustrates the format of an NEC IR transmission frame, for an address of 00h (00000000b) and a command of ADh (10101101b).

  
Figure 1. Example message frame using the NEC IR transmission protocol.

Notice from Figure 1 that it takes:

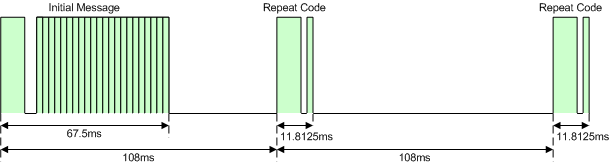
* 27ms to transmit both the 16 bits for the address (address + inverse) and the 16 bits for the command (command + inverse). This comes from each of the 16 bit blocks ultimately containing eight '0's and eight '1's - giving (8 \* 1.125ms) + (8 \* 2.25ms).
* 67.5ms to fully transmit the message frame (discounting the final 562.5µs pulse burst that signifies the end of message).

**Repeat Codes**

If the key on the remote controller is kept depressed, a repeat code will be issued, typically around 40ms after the pulse burst that signified the end of the message. A repeat code will continue to be sent out at 108ms intervals, until the key is finally released. The repeat code consists of the following, in order:

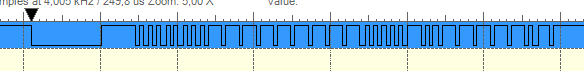
* a 9ms leading pulse burst
* a 2.25ms space
* a 562.5µs pulse burst to mark the end of the space (and hence end of the transmitted repeat code).

Figure 2 illustrates the transmission of two repeat codes after an initial message frame is sent.

  
Figure 2. Example repeat codes sent for a key held down on the transmitting remote controller.

In bijlage ziet u een aantal screenshots van de meting op de uitgang van de TSOP1236 sensor.

* Merk op dat de sensor zijn signaal geinverteerd uitgeeft en dat we dus de tijd zullen moeten meten tussen 2 dalende flanken.
* Merk ook op dat onze TSOP1236 sensor demoduleerd op 36Khz, maar dat onze NEC signalen die op 38Khz gemoduleerd zijn toch ook mooi gedemoduleerd worden.



U ziet mooi de start – 00000000; 11111111; 10000010; 01111101



…

Programma:

Dit programma decodeerd het adres en de data uit een NEC signaal – gedemoduleerd via een TSOP1236 sensor.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <htc.h>

#include "HITECH\_LCD.h" // include LCD library

#define \_XTAL\_FREQ 19660800

\_\_CONFIG**(**0x2FF2**);** // external HS Xtal 19660800

\_\_CONFIG**(**0x3FFF**);**

// global variables - for use in all functions

unsigned int TMR1HCOPY**,** TMR1LCOPY**,** TMR1X**;**

unsigned char situatie**,** START **=** 0**;**

unsigned char x**,**BIT**,** ADRES**,**ADRESINV**,**DATA**,** DATAINV **=** 0**;**

void main**()**

**{**

TRISD **=** 0x00**;** // leds aan PORTD output

TRISB **=** 0x01**;** // RB0 = 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

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

LCD\_Start**();**

LCD\_Clear**();**

LCD\_PrintString**(**"NEC IR DECODER"**);**

**for** **(**char s **=** 0**;** s**<**6**;** s**++)**\_\_delay\_ms**(**250**);** // show start message

**while(**1**)**

**{**

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

**}**

**}**

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

void interrupt isr**(**void**)**

**{**

**if** **(**TMR1IF**)** // int op TMR1 overflow

**{** // 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**)**// interrupt op RB0 (dalende flank op RB0)

**{**

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**++;**

**}**

**}**

**}**

**}**