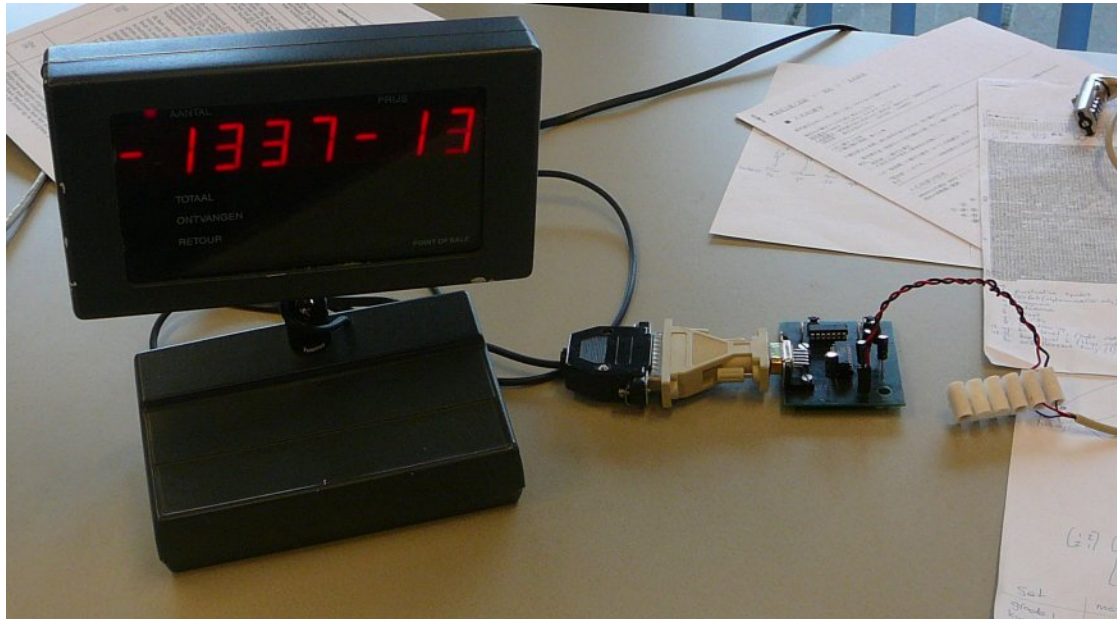


# PIC to HAMA device

**Author:** Aram Verstegen  
**Author:** Johan Otten  
**Date:** 22 Febuari 2008



# Contents

<b>Opzet van het project</b>	<b>3</b>
<b>Werkwijze</b>	<b>3</b>
De microchips . . . . .	3
Het display . . . . .	3
De printplaat . . . . .	4
<b>Hardware ontwerp</b>	<b>6</b>
Componentenlijst . . . . .	6
Referentieschema's . . . . .	6
Voeding . . . . .	6
Uiteindelijke schema . . . . .	7
Draadbrug . . . . .	9
<b>Software ontwerp</b>	<b>10</b>
IDE + Compiler . . . . .	10
PIC Programmer . . . . .	10
Referentie broncode . . . . .	11
Uiteindelijke broncode . . . . .	12
<b>Leerpunten</b>	<b>13</b>
<b>Bijlagen</b>	<b>14</b>
Bijlage A: Hardware ontwerpen . . . . .	14
Bijlage B: Software broncode . . . . .	18

## Opzet van het project

Het doel van het project is om een serieel kassadisplay aan te sturen met een programmeerbare microchip in een door onszelf ontworpen circuit.

## Werkwijze

### De microchips

We hebben gekozen voor een PIC16F636, omdat deze chip en de benodigde programmer voorhanden waren. Gezien de beperkingen van deze chip, in het bijzonder het ontbreken van een UART om communicatie via het RS-232 protocol voor seriële verbindingen aan te gaan, moest het protocol in software op deze chip worden geïmplementeerd. De PIC16F636 van Microchip is een 14 pin dual inline chip en draait standaard op 4Mhz, wat meer dan genoeg is om communicatie op 9600 baud aan te kunnen. We hebben daarnaast een RS-232 chip in ons ontwerp gebruikt om ons circuit te beveiligen tegen kortsluiting.

Na enig onderzoek zijn we er achter gekomen dat het handig is om zo'n chip te gebruiken om de RS-232 signalen te zenden, aangezien het met negatieve spanningen werkt. Het is ook mogelijk om zonder MAX232 een geldig RS-232 signaal te maken, maar we hebben gekozen voor een vertrouwde aanpak. We hebben nog overwogen om een MAX233 te kopen, zodat we geen extra condensatoren nodig hadden. Aangezien die relatief een stuk duurder is, hebben we toch gekozen voor de MAX232 en wat goedkope condensatoren.

### Het display

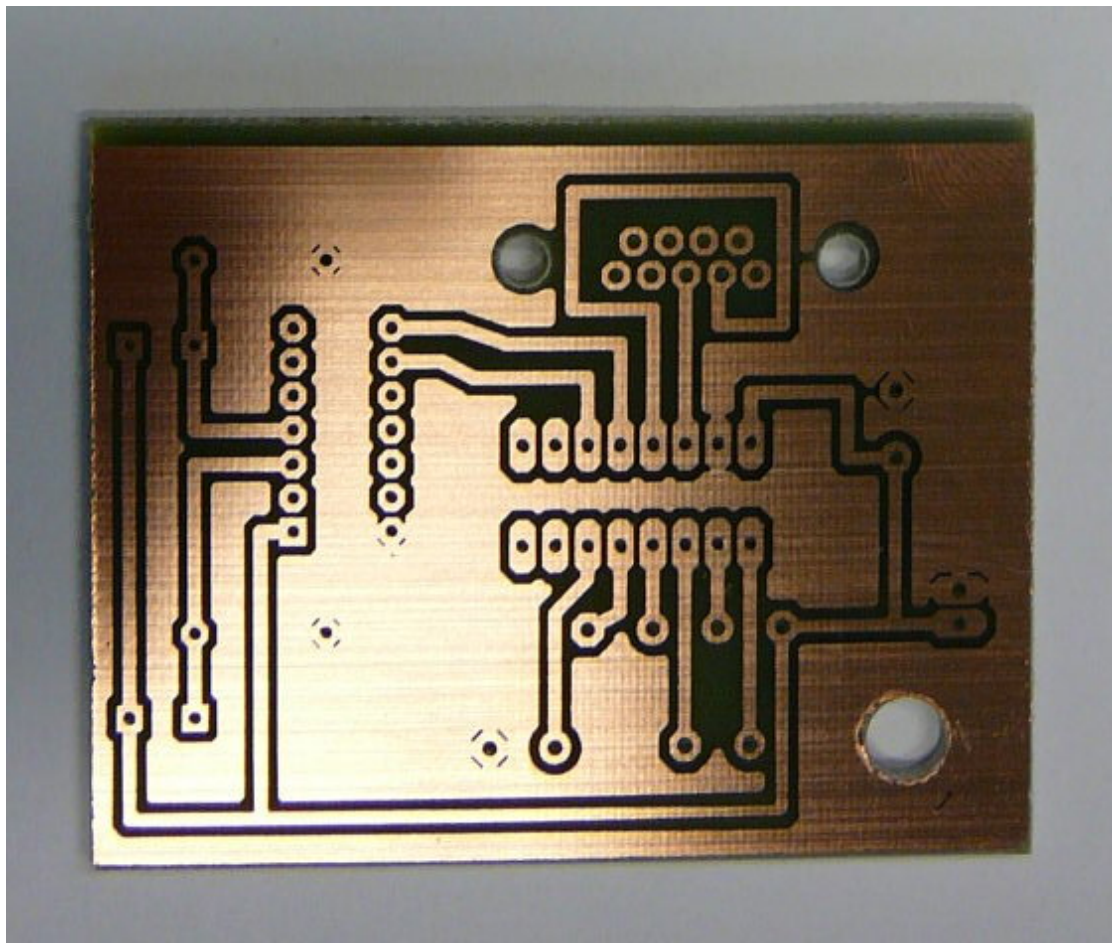
Het kassadisplay is een black box. Het bevat 8 matrices met buislampjes die numerieke en enkele alfanumerieke karakters kunnen weergeven. Er staat behalve 'HAMA' geen model of typenummer op het display. Het display is gekocht op een computerbeurs en via die koper bij ons beland. Zijn methode was om het display op een computer aan te sluiten en er random data naar te sturen. Het display lichte op, maar er gebeurde te veel op het display om er enige logica in te zien. Derhalve heeft hij een programma gemaakt dat periodiek 1 olopende byte verstuurt. Hieruit bleek dat bytes 0x0 tot 0xF respectievelijk de volgende 16 karakters opleveren: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, -, E, H, L, P, spatie.



### **De printplaat**

We hebben op basis van bestaande schema's het schema van onze schakeling in de Eagle editor voor programmeerbare circuits getekend, en op basis daarvan de layout voor een

circuit gemaakt. Dit ontwerp is geëts tot printplaat en de componenten van het circuit zijn er op gesoldeerd. Het circuit bevat sockets voor de PIC microcontroller, de RS-232 chip, 2 knopjes, een 'male' seriële poort en de nodige condensatoren en weerstanden.



## Hardware ontwerp

### Componentenlijst

- PIC16F636 + 14 pins socket
- Max232 kloon + 16 pins socket
- 2 drukknopjes
- Weerstanden 2x 10K Ohm
- Condensatoren 4x 1 en 1x 10 micro farad
- Male seriële port

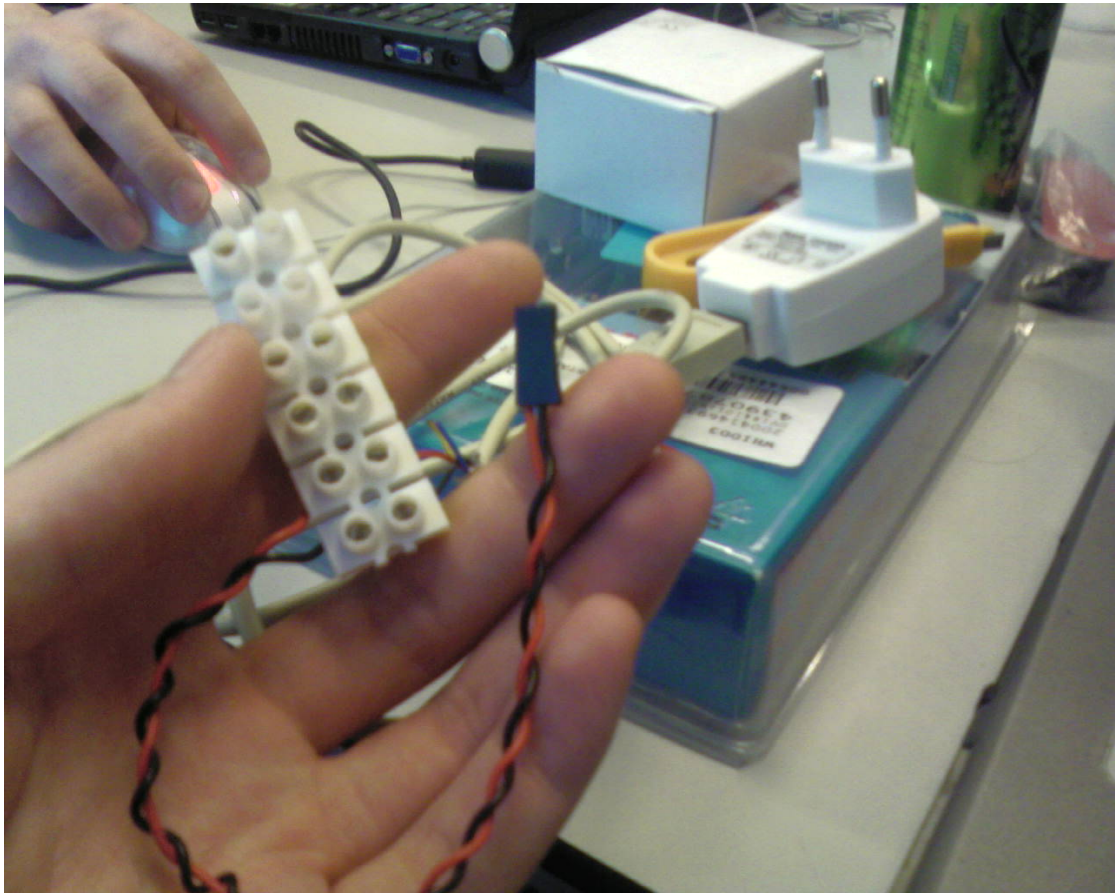
### Referentieschema's

We zijn uit gegaan van de in Bijlage A bijgevoegde schema's om de MAX232 aan te sluiten op de PIC en de seriële poort. De send en receive zijn omgewisseld omdat dit schema niet als host fungeert, maar als client. Hiervoor is pin 4 verwisseld met pin 2.

Het tweede schema is gebruikt om te zien hoe we knopjes goed aan konden sluiten op een PIC.

### Voeding

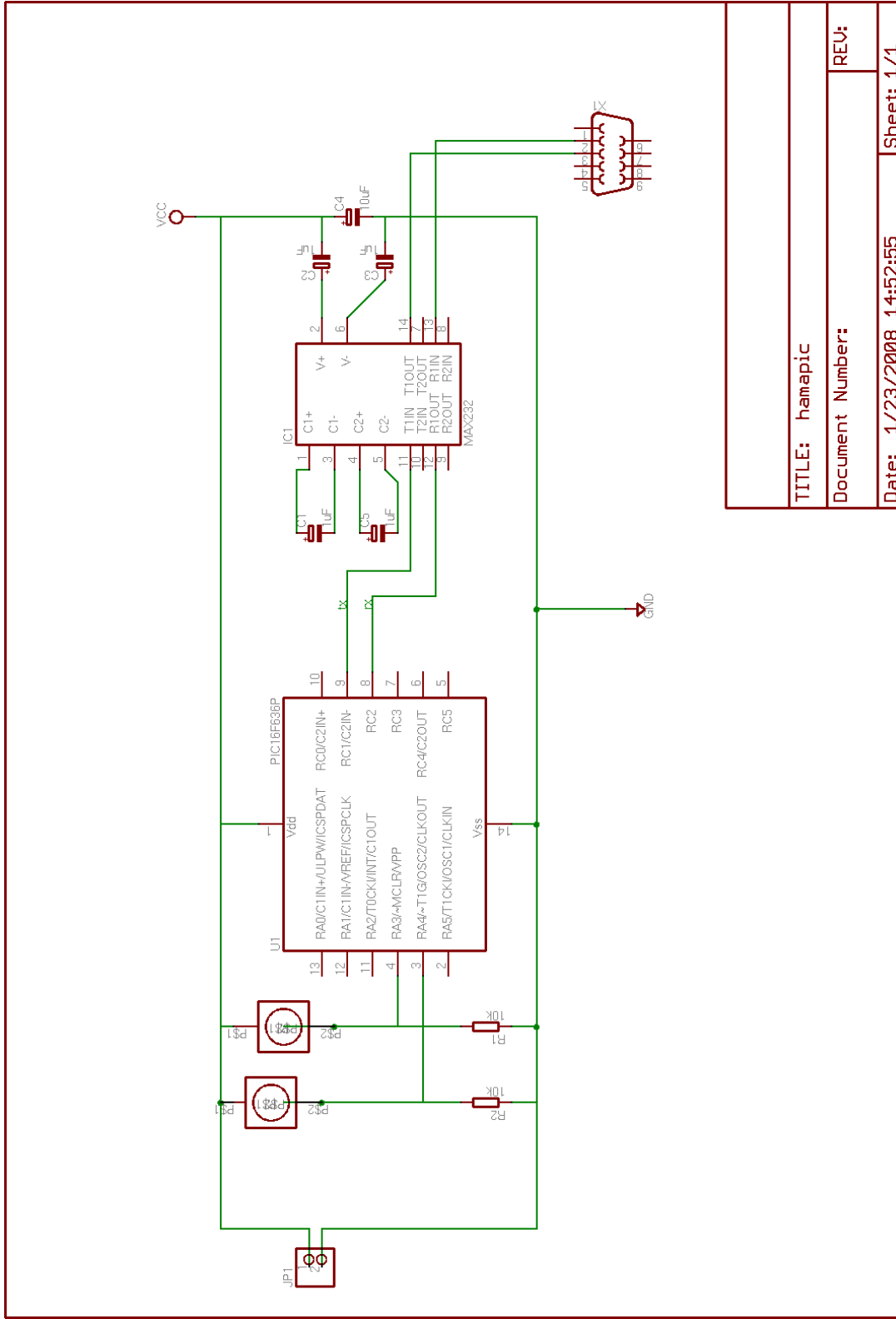
Voor het voeden van ons circuit hebben we gekozen voor een AC converter voor USB apparaten. Deze levert standaard de benodigde 5 Volt, en is met een gemodificeerde USB kabel en een simpele jumper aansluiting met ons circuit verbonden.



### **Uiteindelijke schema**

We hebben een schema gemaakt in Eagle waarin de PIC via de MAX232 met de seriële poort is verbonden. Eerst is er gebruik gemaakt van auto routing en daarna zijn een aantal paden aangepast om het beter te laten passen.

Zie Bijlage A voor afbeeldingen van alle gebruikte schema's. Voor de duidelijkheid staat hieronder het uiteindelijke schema afgebeeld.



TITLE: hamapic

Document Number:

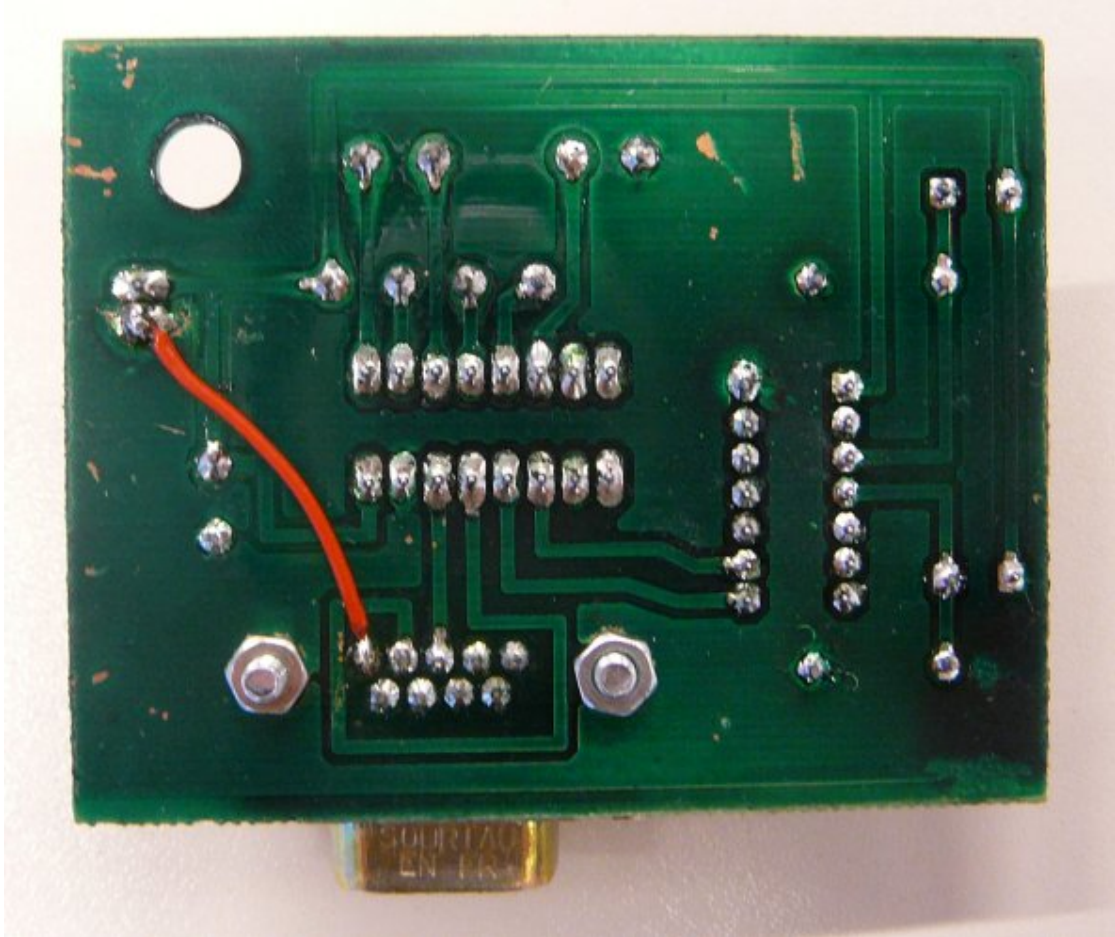
REV:

Date: 1/23/2008 14:52:55

Sheet: 1/1

## Draadbrug

Bij het testen van ons ontwerp dachten we dat het wellicht nodig was om de seriële poort te aarden. Derhalve hebben we na het solderen van de componenten nog een draadbrug gemaakt die pin 5 groundt.



## Software ontwerp

### IDE + Compiler

Als IDE hebben we gebruik gemaakt van MPLab in combinatie met de CC5X compiler. MPLab heeft onder andere een debugger met een manier om registers te inspecteren. CC5X is een compiler voor de PIC microcontrollers. De gratis versie is met enige limitaties te gebruiken voor niet-commerciële doeleinden.

### PIC Programmer

Om ons hardware ontwerp zo simpel mogelijk te houden hebben we niet gekozen voor een in-circuit-programmable oplossing, maar een losse programmer. De programmer die we hebben gebruikt is een standaard PIC starter kit, de PICKit 1 FLASH starter kit. Dit is een USB flash programmer en de gebruikte IDE (MPLab) wordt hier bij geleverd. We hebben voor dit pakket gekozen omdat het voorhanden was.



## Referentie broncode

Onze PIC heeft geen hardware UART, dus moesten we dit in software doen. We hebben het internet afgestruind om een voorbeeld van een software UART voor de PIC te vinden. We zijn uitgekomen bij hetvolgende stuk assembly.

```
; mem vars def
TEMP    EQU    8
COUNT  EQU    9
VALUE   EQU    0xA

; Port and pin definitions
PORT_B  EQU    6
Tx      EQU    4

START:
    MOVLW    8
    MOVWF    COUNT
    BCF      PORT_B, Tx ;zero start bit (clear bit 4 in reg 6)
    MOVLW    32         ;delaycount for 480 khz clock
    MOVWF    TEMP

DY1:
    DECFSZ   TEMP,1
    GOTO     DY1

LOOP:
    BCF      3,0        ;Clear carry bi, bit 0 in reg 3
    RRF      VALUE,1   ;rotate value bit to carry, LSB first
    BTFSC    3,0        ;Skip next instruction if carry
    BSF      PORT_B,Tx ;send data bit if equal to one
    BTFSS    3,0        ;skip next instruction of carry is set
    BCF      PORT_B,Tx ;send databit if equal to zero

    MOVLW    30        ;Delay count for 480khz clock
    MOVWF    TEMP

DY2:
    DECFSZ   TEMP,1
    GOTO     DY2
    DECFSZ   COUNT,1
    GOTO     LOOP

    BSF      PORT_B,Tx ;Send stop bit = 1
```

```
MOVLW      32          ;DELAY COUNT FOR 480KHZ CLOCK
MOVWF     TEMP
```

```
DY3:
DECFSZ    TEMP, 1
GOTO     DY3
```

Dit stukje assembly doorloopt een byte (VALUE) en telt zijn huidige index in deze byte met het register COUNT. Er wordt gebruik gemaakt van loops om deze byte plus de benodigde start en stop bits voor een bepaalde tijd hoog te houden. Hierbij wordt gebruik gemaakt van het feit dat iedere instructie even lang duurt. Te zien is dat een hoge bit in de byte sneller wordt verstuurd dan een lage. Dat kleine timing verschil maakt niet uit, het wordt vergeven door de RS-232 specificatie.

### **Uiteindelijke broncode**

Met behulp van de datasheet voor onze PIC chip hebben we de instructies ontcijferd en vertaald naar wat voor een geoefende assembly programmeur een overdaad aan commentaar moet zijn. Toen het principe van deze code ons eenmaal duidelijk was konden we het aangepaste voorbeeld integreren in ons C programma.

De code begint in de main loop, die de functie display aanroept met een weer te geven string. In de display functie wordt de string doorlopen. Ieder karakter wordt vertaald naar een byte in de reeks 0x0-0xF en naar de HAMA device verstuurd via de send functie. We hebben het voorbeeld gebruikt in de send functie.

Zie voor de uiteindelijke broncode Bijlage B.

## Leerpunten

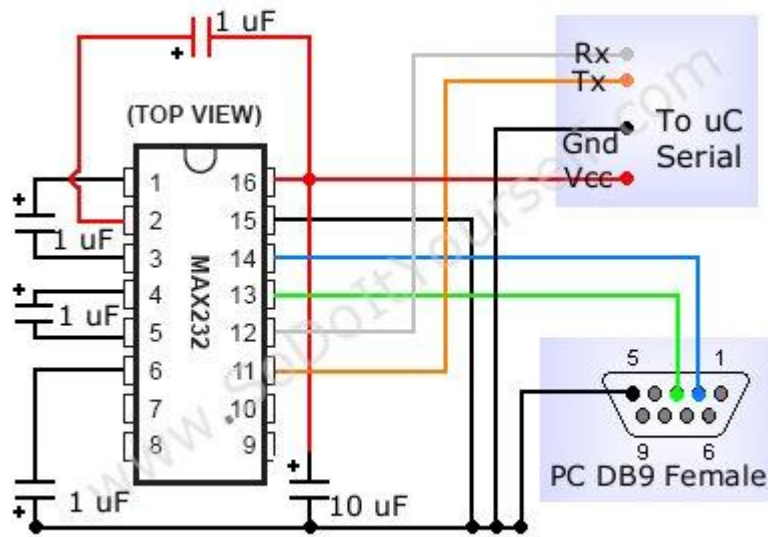
Als we nog eens RS-232 communicatie gebruiken zullen we zeker kijken naar een micro-controller met een hardware UART, om alle moeite die een software UART ons kostte te sparen, en zo goed als zeker te weten dat het werkt.

Ook zouden we de voeding beter ontwerpen, zodat het niet verkeerd om is aan te sluiten. We hebben nu nog niet de voeding verkeerd om aangesloten, maar de kans dat het ooit gebeurt bestaat. Om dit te voorkomen kan er een diode voor gezet worden. Helaas bedachten we dit pas toen het printje al geëtst was.

## Bijlagen

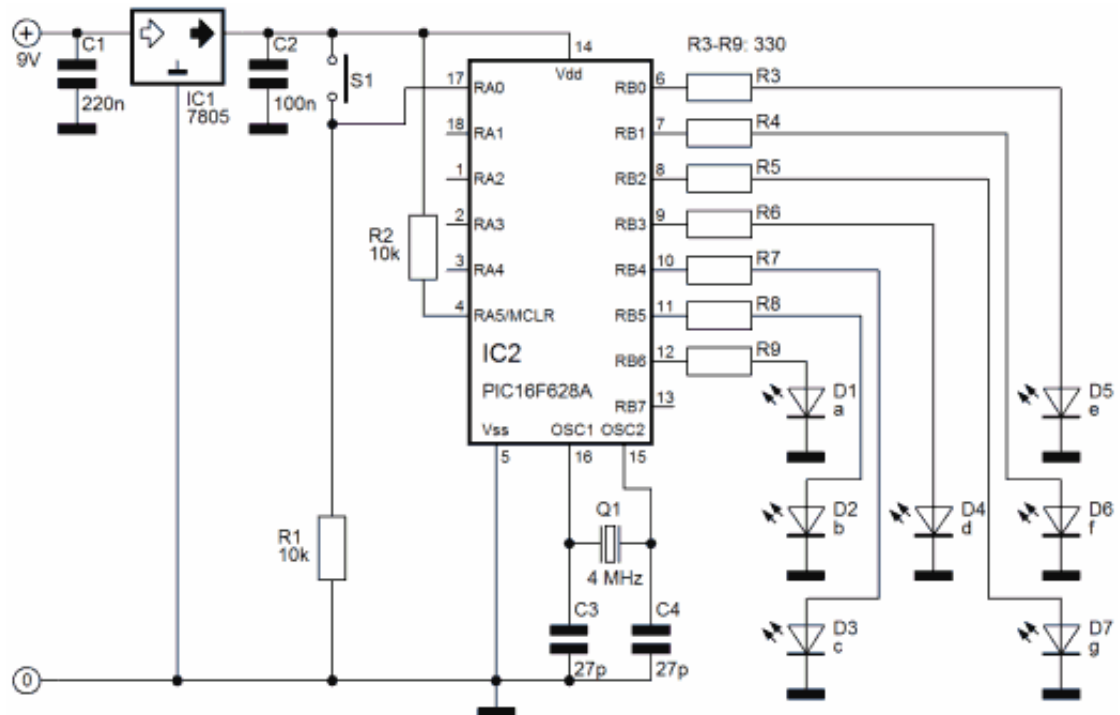
### Bijlage A: Hardware ontwerpen

Referentieschema gebruik MAX232:



[www.SoDoItYourself.com](http://www.SoDoItYourself.com)

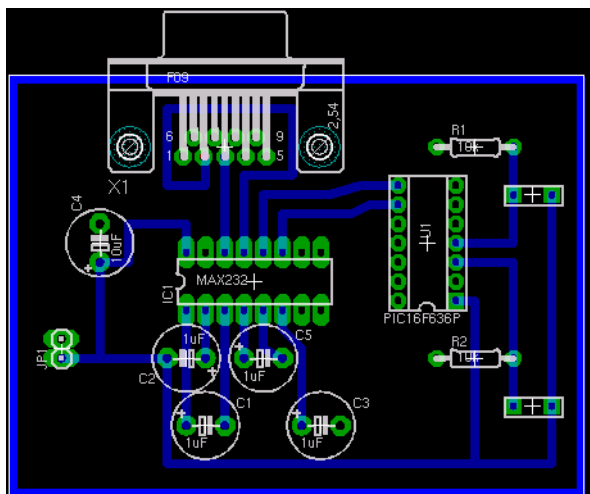
Referentieschema aansluiten knoppen:



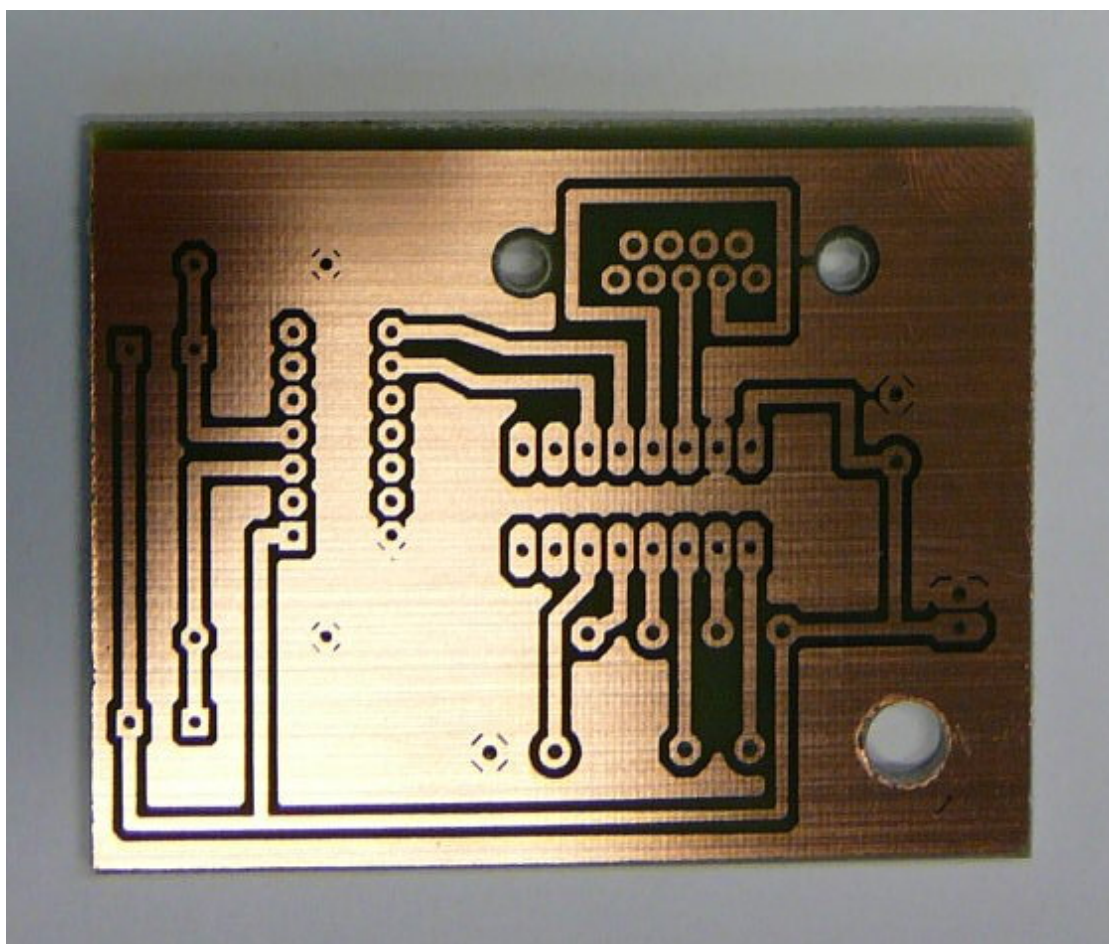
Uiteindelijke schema:



Uiteindelijke layout:



Uiteindelijke printplaat:



## Bijlage B: Software broncode

```
#include <16F636.h>
//stel de clock in zodat op pin 3 een signaal komt,
//dit moet veranderd worden zodat pin 3 als input gebruikt kan worden.
#pragma config |= 0b.0.0.0.0.0.0.0.0.0.101 // Configuratie van de WORD instellen

// Michael Dworkin
// 05.06.2003
// Compiler CC5x
// For a frequency of 4 MHz
void Delay1ms( uns16 ms) {
    while(ms) {
        OPTION = 2; // Schleife verlassen wen ms=0 ist
        TMRO = 131; // Vorteiler auf 8 einstellen
        while (TMRO); // 125 * 8 = 1000 (= 1 ms)
        ms--; // abwarten einer Milisekunde
        // "ms" mit jeder Milisekunde ernidrigen
    }
}

// characters on the HAMA display have a symbolic integer value
// this method translates a character to such an integer
unsigned int translate(char c) {
    switch (c) {
        case '-':
            return 10;
        case 'e':
            return 11;
        case 'h':
            return 12;
        case 'l':
            return 13;
        case 'p':
            return 14;
        case ' ':
            return 15;
        case '0':
        case '1':
        case '2':
        case '3':
        case '4':
        case '5':
        case '6':
        case '7':
```

```

        case '8':
        case '9':
            return c - '0';
        default:
            return 15;
    }

// Send a character as an rs232 stream to PORTC

void send(unsigned char byte) {
    unsigned char TEMP = 8;
    unsigned char COUNT = 9;
    unsigned char SBUF = 3;
    unsigned char VALUE = 5;
    W = byte;

#asm
; Transmit bit
Tx      EQU      1
MOVWF   VALUE    ;move it to register VALUE

START:
                                ;set COUNT to 8, clear bit Tx in PORTC, move 32 to TEMP
    MOVLW      8                ;literal 8 in working register
    MOVWF     COUNT            ;move it to register COUNT
    BCF       PORTC, Tx        ;zero start bit (clear bit Tx in reg PORTC)
    MOVLW     32               ;move literal 32 to working register - delaycount for 480
    MOVWF     TEMP             ;move working register contents to TEMP

DY1:
                                ;take down register TEMP to 0 in baby steps
    DECFSZ    TEMP,1           ;decrement register 8 (temp) by one. skip next instruction
    GOTO      DY1              ;return to the start of the loop

LOOP:
    BCF       STATUS,0         ;Clear carry bit, clear bit 0 in reg STATUS
    RRF       VALUE,1          ;rotate value bit to carry, LSB first (I think this means
    BTFSC     STATUS,0         ;Skip next instruction if carry (if bit 0 in register)
    BSF       PORTC,Tx        ;send data bit if equal to one (set bit Tx in register 6)
    BTFSS     STATUS,0         ;skip next instruction if carry is set (if bit 0 in register)
    BCF       PORTC,Tx        ;send databit if equal to zero (clear bit Tx in register 6)

    MOVLW     30               ;Delay count for 480khz clock (move literal 30 to working register)
    MOVWF     TEMP             ;move working register contents to TEMP

DY2:
                                ;take down register TEMP to 0 in baby steps, decrement

```

```

    DECFSZ    TEMP,1      ;decrement register TEMP by 1, skip next instruction if 0
    GOTO      DY2        ;return to the start of the loop
    DECFSZ    COUNT,1    ;decrement register COUNT by 1, skip next instruction if 0
    GOTO      LOOP      ;go to label LOOP

    BSF       PORTC,Tx   ;Send stop bit = 1 (set bit Tx in PORTC)
    MOVLW     32         ;DELAY COUNT FOR 480KHZ CLOCK (move literal 32 to working register)
    MOVWF     TEMP      ;move working register contents to TEMP

DY3:
    ;take down register TEMP to 1 in baby steps
    DECFSZ    TEMP, 1    ;decrement register TEMP by 1, skip next instruction if 0
    GOTO      DY3        ;return to the start of the loop
#endasm
}

// iterates through provided string and sends each translated character
void display(const char *string) {
    while(*string) {
        send(translate((char) *(string++)));
        // Verlengde stop bit
        Delay1ms(20);
    }
}

void main() {
    TRISC= 0b.0000.0000;
    OSCCON = 0b.0110.0001;
    while (1) {
        display("1337-");
    }
}

```