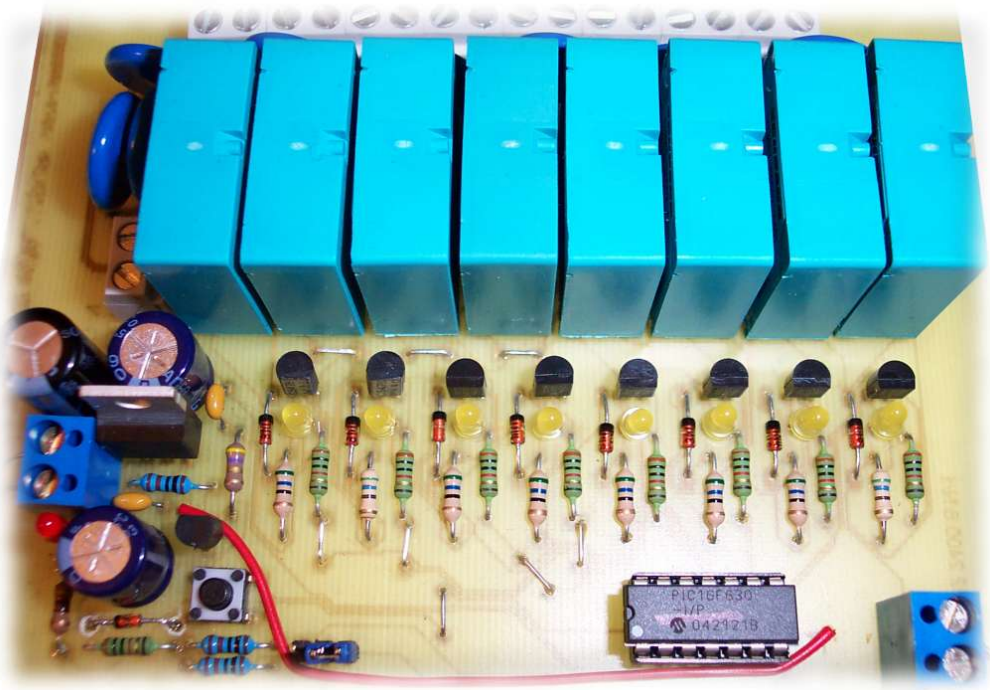
Implementierung einer erweiterbaren seriellen Relaiskarte



Projektarbeit Microprozessor-Technik

NTA FH Isny
12. Info

David Mayr

Inhalt

1 Einleitung

1.1 Was ist eine serielle Relaiskarte?

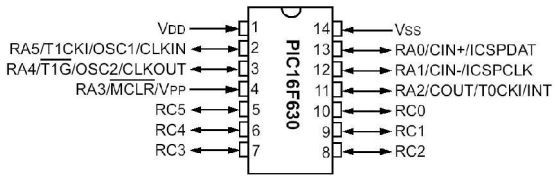
Mit einer seriellen Relaiskarte lassen sich mehrere Geräte/Verbraucher – hier maximal acht Stück pro Karte - über eine serielle Schnittstelle z.B. von einem Computer ein- und ausschalten. Mit entsprechend eingerichteter Software ist es dann möglich, z.B. bei ankommenden Telefonanrufen oder neuen eMails bestimmte Signal-Lampen automatisch anschalten zu lassen oder sonstige elektrische Geräte zu steuern[1].

1.2 Wozu?

Die Idee dazu entstand dadurch, dass meine schwerhörige Freundin beim Fernsehen mit Kopfhörer weder Telefon noch Türklingel hört. Durch meinen mit dem ISDN-Netz verbundenen Server[2] könnte man so Anrufe[3] auch in mehreren Räumen signalisieren. Ausserdem ließe sich so z.B. der Laserdrucker automatisch mit Strom versorgen, sobald ein Druckauftrag am Server ansteht. Nach erfolgtem Ausdruck kann dann der Stromfresser wieder ganz vom Netz getrennt werden.

2 Grundlagen

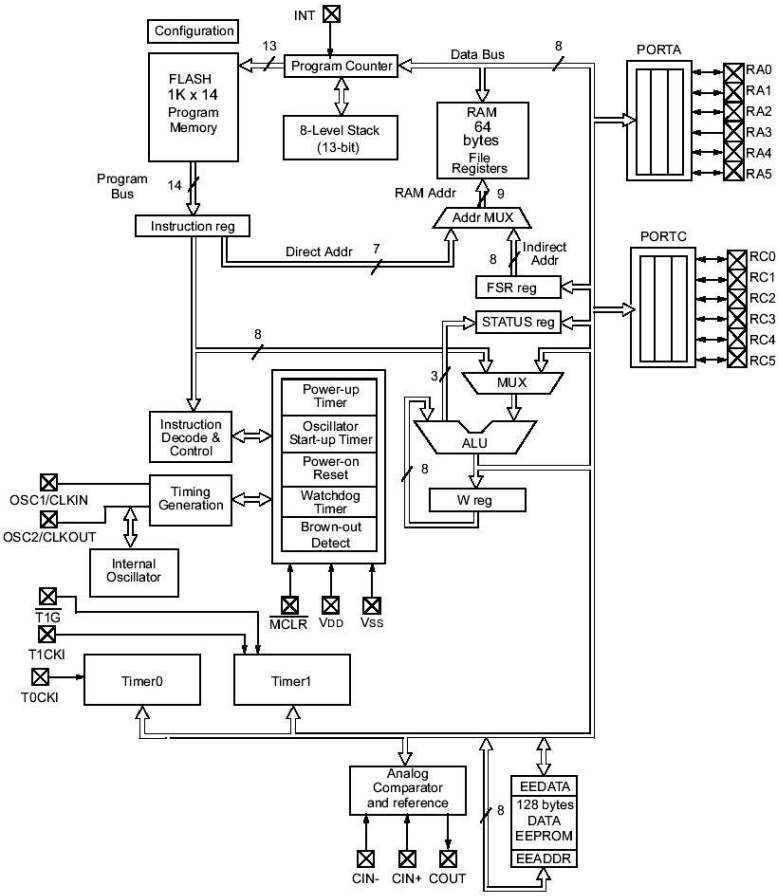
2.1 Der IC: PIC16F630



Kernbestandteil der Schaltung ist ein programmierter Mircocontroller PIC 16F630 von der Firma Microchip[4], der die Umsetzung des seriellen RS232-Signals in die Steuersignale der 8 Relais vornimmt.
Dieser 14-polige DIL-IC ist ein Flash-basierter 8-Bit RISC CMOS Microcontroller, der mit bis zu 20MHz betrieben werden könnte. Da er jedoch durch die sehr niedrige Baudrate am seriellen Anschluss[5] eine wesentlich niedrigerer Betriebstakt nötig ist, arbeitet er in diesem Fall auch sehr stromsparend. Als RISC-Prozessor reichen ihm 35 Befehlsworte, die Wortlänge beträgt 14Bit. Ausserdem beherbergt der 19x6mm kleine Chip einen Flash-Programmspeicher für 1024 14Bit-Worte (=1792 Bytes), 128 Byte EEPROM Datenspeicher, 64 Byte RAM, einen Vergleicher und 12 programmierbare Ein- oder Ausgänge.

1 bzw. die Geräte mit dem Stromnetz zu verbinden oder sie von diesem zu trennen.
2 der mit dem äusserst anpassungsfähigem OpensSource-Betriebssystem Linux arbeitet
3 durch unterschiedliche Kombinationen von mehreren Signallampen könnten sogar verschiedene Anrufer optisch angezeigt werden (z.B. drei Lampen für 7 vorgegebene Anrufer und alle anderen)
4 http://www.microchip.com, weitere direkt-Links in den Quellenangaben
5 2400 Baud – somit sind auch bei schlechten Umgebungsbedingungen relativ sichere/zuverlässige Verbindungen möglich.

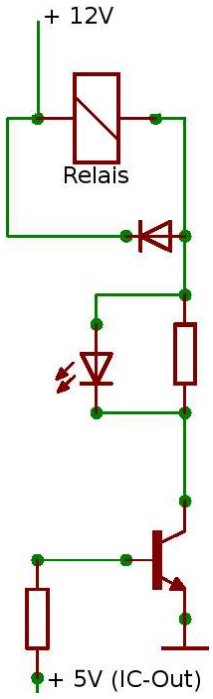
Im folgenden ein Blockschaltbild des PIC16F630:



In diesem Projekt wurde ein bereits fertig programmierter IC von der Firma velleman verwendet[6]. Alles weitere, insbesondere die Ansteuerung und die Software beziehen sich nur auf diese Implementierung.

2.2 Relais-Ansteuerung

Obwohl der PIC an seinen Ausgängen mit relativ hohen Strömen bis immerhin 25mA belastet werden kann, reicht das nicht für die direkte Ansteuerung der Relais aus – abgesehen davon dass die hier verwendeten Relais mit 12V und der IC mit 5V arbeiten. Aus diesem Grund müssen den Relais Transistoren vorgeschalten werden, die die Masse-Pins der Relais mit Masse verschalten. Da die Magnetspulen der Relais beim Abschalten einen kurzen aber starken Spannungsanstieg erzeugen, die den jeweiligen vorgeschaltenen Transistor zerstören würden, müssen diese mit Dioden gegen diesen Spannungsanstieg geschützt werden. Ausserdem soll durch eine LED der Status des jeweiligen Relais angezeigt werden.
Rechts der entsprechende Schaltbildausschnitt:



6 Da das Programm des PIC lese-/kopiergeschützt ist, kann ich darauf hier nicht näher eingehen.

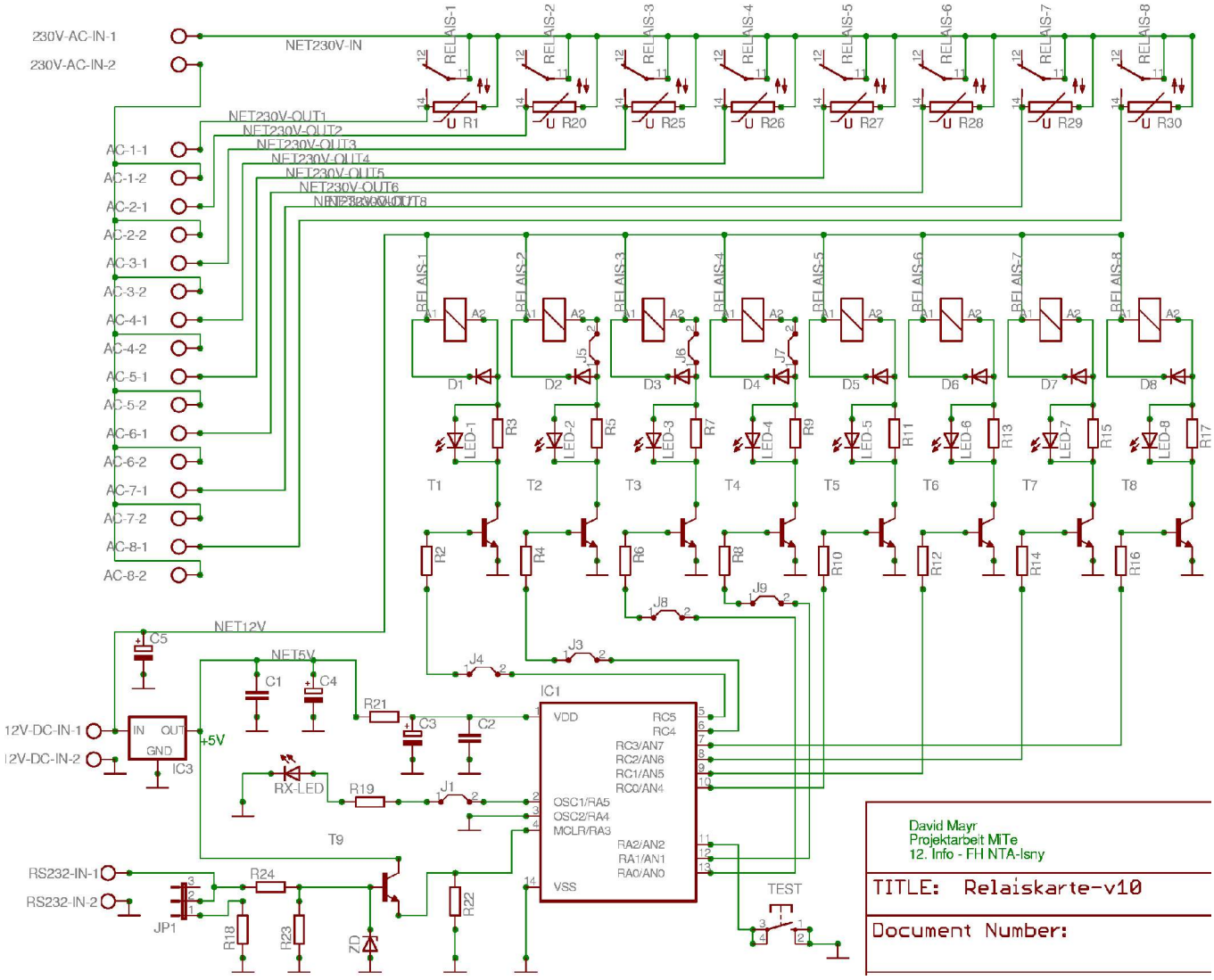
2.3 Verschleiß-Schutz für die Relais

Damit an den Last-Kontakten der Relais beim Schalten nicht Funken gezogen werden, die neben dem schnelleren Verschleiß der Kontakte auch ein hörbares Knacken z.B. in Radios verursachen, wurden die 230V-Lastausgänge zusätzlich mit Varistoren bestückt.

3 Hardware

Der Schaltplan und das Platinenlayout wurden mit der kostenlosen Light-Version von Eagle[7] erstellt. Das Ätzen der Platine erfolgte an der NTA.

3.1 Schaltplan

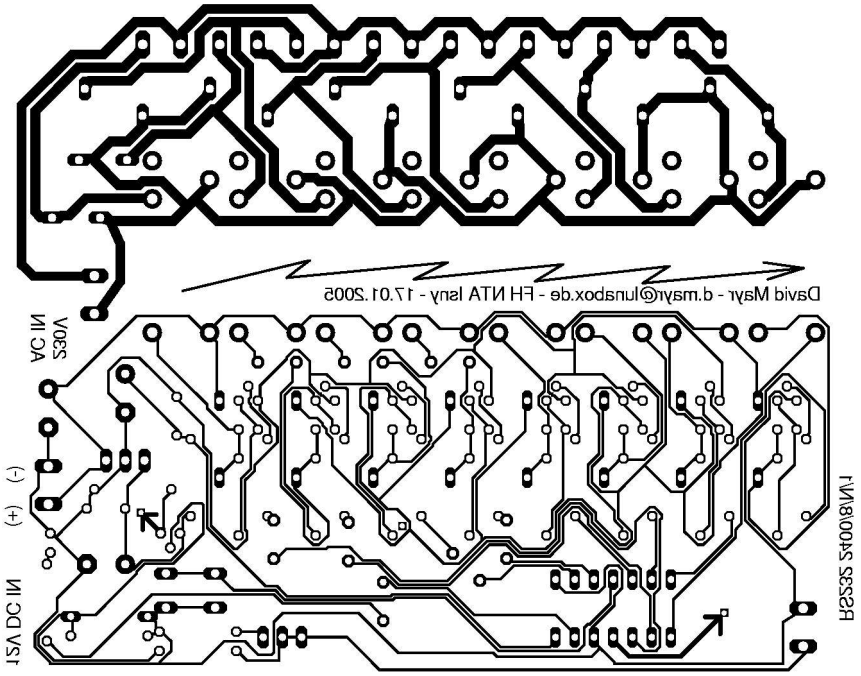


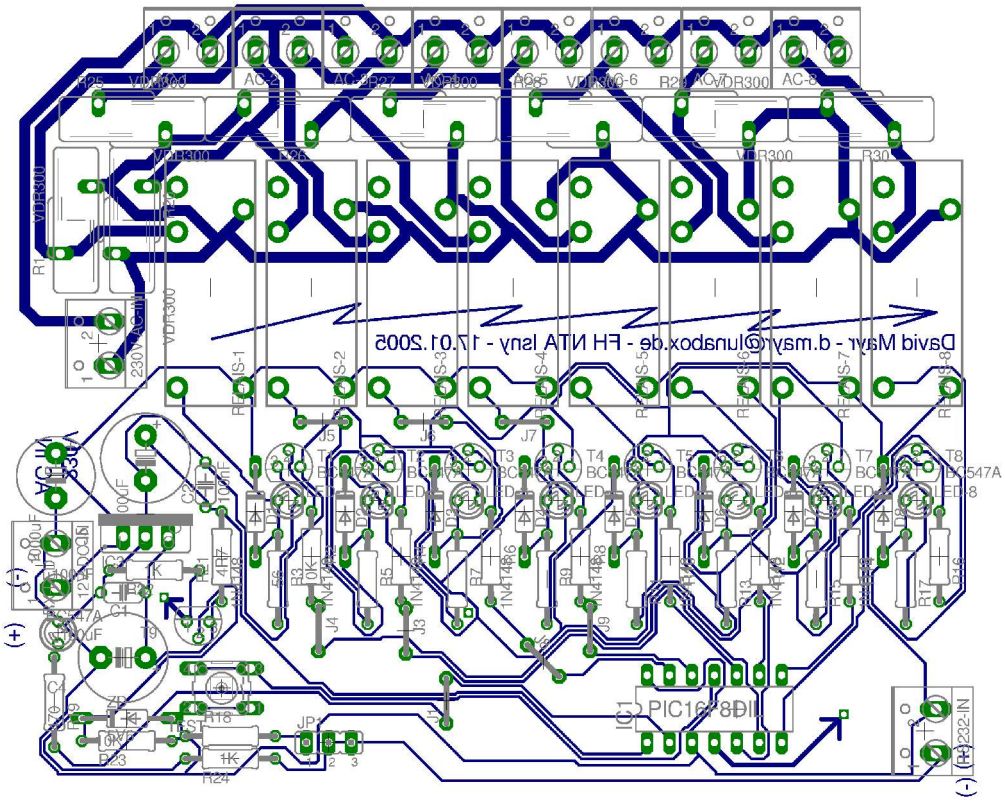
Der Schaltplan wurde in Anlehnung an den Bausatz K8056 von velleman erstellt.

7 http://www.cadsoft.de – freie Light-Version für Linux, Mac und Windows für Platinen bis 100x80mm

3.2 Platinenlayout

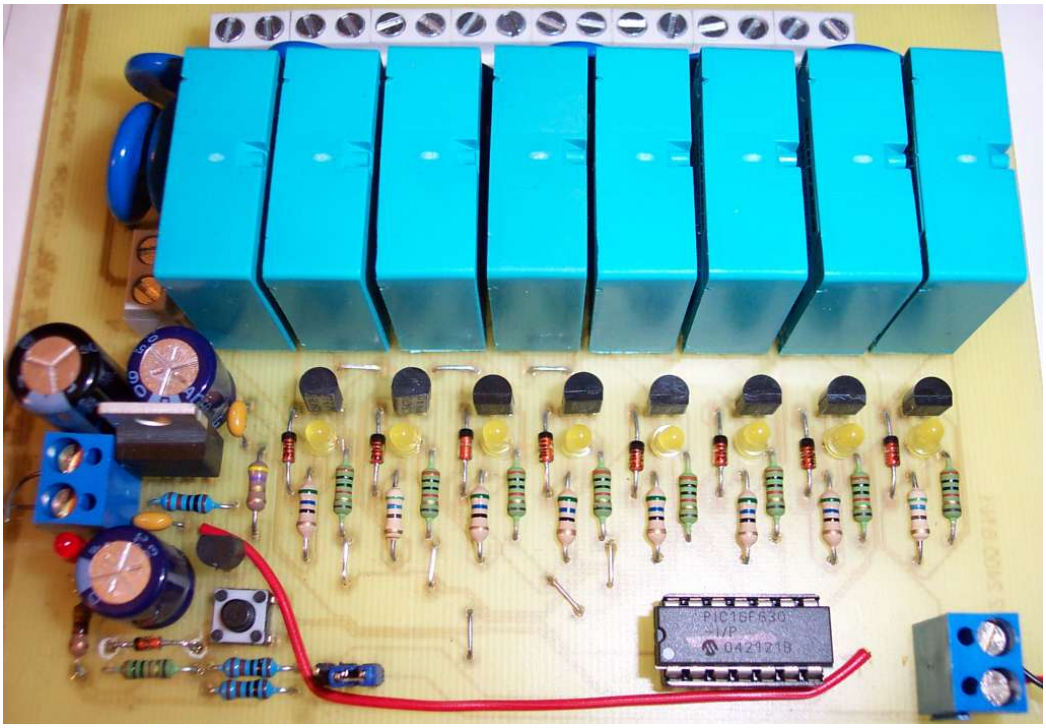
Die Leiterbahnen wurden von Eagle's Autorouter verlegt – auch wenn dafür durch den begrenzten Raum viel Geduld und Nacharbeit nötig war. Hier das Platinenlayout (ohne und mit Bauteile):



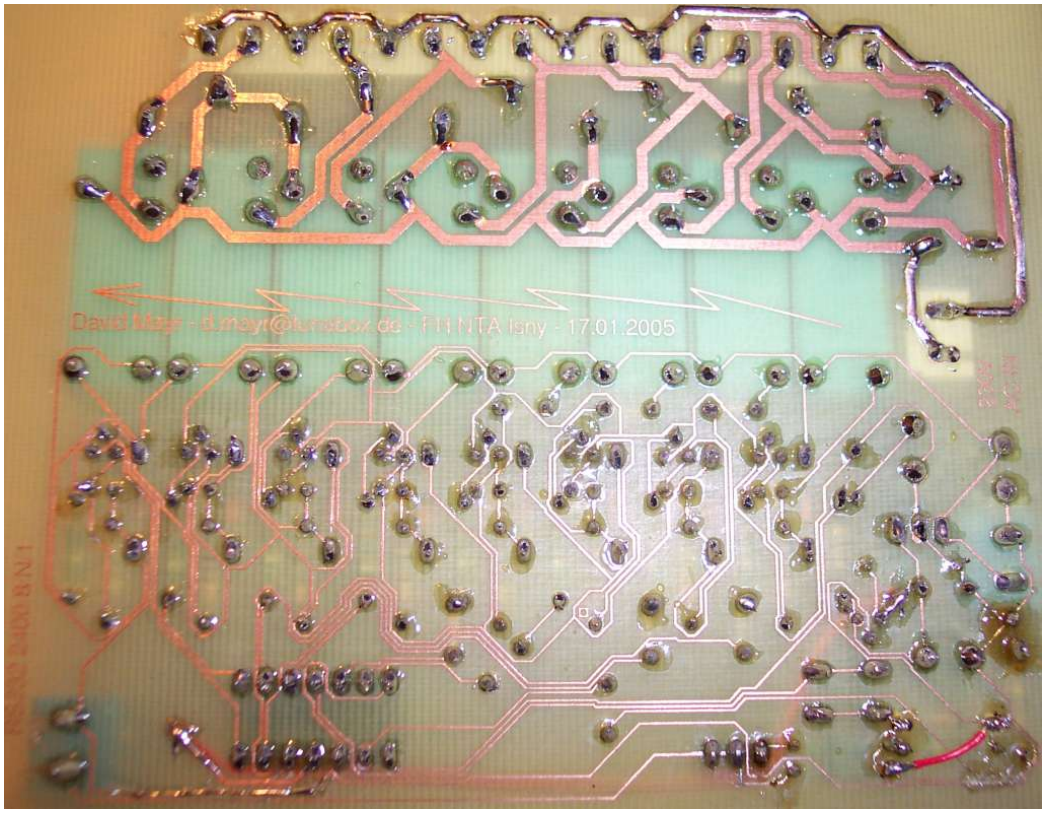


3.3 Aufbau der Platine

Platinenoberseite:



Platinenunterseite:



4 Software

Da der Server, an den diese Karte später angeschlossen wird, unter Linux läuft und das ganze möglichst flexibel sein soll, ist ein einfaches Kommandozeilenprogramm mit den Grundbefehlen zur Steuerung der Relaiskarte völlig ausreichend. Dieses Kommandozeilenprogramm soll über die Aufrufparameter Befehle und die zugehörigen Werte entgegennehmen und über einen seriellen Anschluss an die Karte schicken.

4.1 Schnittstellen-Parameter

Der IC erwartet über die serielle Schnittstelle Steuersequenzen mit 2400 Baud bei 8 Datenbits, ohne Parität und mit einem Stoppbit. Zusätzlich zur Steuerung per RS232 bietet ein Taster an Pin11 des ICs die Möglichkeit, die grundlegende Funktion des Chips und aller Relais zu testen.

4.2 Aufbau der Befehlssequenz

Der Aufbau einer Steuersequenz ist recht einfach und besteht aus 4 Bytes:

1. Byte: CarriageReturn[8]
2. Byte: Kartenadresse
3. Byte: **Befehl**
4. Byte: Relaisadresse oder Wert
5. Byte: Checksumme (256-(Byte1+Byte2+Byte3+Byte4))

Die möglichen Befehle gliedern sich in zwei Gruppen und bestehen aus dem jeweils angegebenen ASCII-Zeichen:

Steuerbefehle:

S – **S**et:	Relais[9] AN
C – **C**lear:	Relais AUS
T – **T**oggle:	Relais toggeln
B – **B**yte:	Alle Relais mit einem Byte (Bitweise) setzen
E – **E**mergency:	Alle Relais auf allen Karten AUS

Adressierungsbefehle:

F – **F**orce:	Alle Relaiskarten mit Kartenadresse 1 behandeln[10]
A – **A**dresswechsel:	Ändern der Kartenadresse, Wert=neue Adresse
D – **D**isplay:	Anzeige der aktuellen Kartenadresse (durch Relais)

Laut Herstellerangaben ist es zur zuverlässigen Erkennung des jeweiligen Befehls nötig, jede Steuersequenz mindestens zweimal direkt hintereinander zu schicken. Desweiteren hat sich in den praktischen Tests herausgestellt, dass zwischen zwei unterschiedlichen Befehlen ein Pause von wenigstens einigen hundert Millisekunden nötig ist, damit auch während z.B. 2000 Schaltfolgen kein Befehl verloren geht.

8 ASCII-Steuerzeichen 13 (dezimal)
9 Relaisadresse (ASCII) 1 - 8 ; 9=alle
10 Damit wird die Kartenadressierung deaktiviert - alle Karten haben Adresse 1. Das ist die Standardeinstellung.

Die Adressierung der Karten erfolgt recht einfach. Jede Karte kann mit einer Adresse fest belegt werden (A-Befehl). Diese Adresse behält der Chip in seinem EEPROM-Speicher, der auch ohne Spannungsversorgung erhalten bleibt. Alle Karten werden parallel an die selbe serielle Schnittstelle angeschlossen und empfangen alle die selben Befehle. Nur wenn die eigene Adresse mit der in der Befehlssequenz übereinstimmt (2. Byte), fühlt sich der jeweilige Chip angesprochen. Mit dem D-Befehl lässt sich die aktuelle Adresse anzeigen, dazu muss die Stellung der Relais nach dem Befehl abgelesen werden – sie repräsentiert das Bitmuster der Kartenadresse.

4.3 Ansteuerung unter Linux

Mit dem folgenden C++-Programm lässt sich eine Relaiskarte unter Linux steuern:

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <termios.h>
#include <string.h>
#include <unistd.h>
#include <iostream>
#define BAUDRATE B2400
#define _POSIX_SOURCE 1
#define FALSE 0
#define TRUE 1
#define dont_show 0x00
volatile int STOP=FALSE;

using namespace std;

void show_help(void)
{
    printf("\nProgrammaufruf:\t  prorammname  <GERAETEDATEI>  <KARTENNUMMER>  <BEFEHL: "\
                        "-e(mergency)|-s(et)|-c(lear)|-t(oggle)|-b(yte)|-h(elp)>  <RELAIS-Nr/WERT> \n"\
          "\tBeispiel: prorammname /dev/ttyS0 1 -s 9   -->  alle 8 Relais [ein] auf 1. Karte an COM1\n"\
          "\tBeispiel: prorammname /dev/ttyS1 7 -c 5   -->  Relais 5 [aus] auf 7. Karte an COM2\n\n");
}

int main( int argc, char *argv[])
{
    int fd;
    char cmd[5];
    struct termios oldtio, newtio;

    if ( argc < 4 ) { show_help(); exit(0); }
    // descriptor oeffnen
    if ( ( fd = open( argv[1], O_RDWR | O_NOCTTY | O_NONBLOCK ) ) == -1 ) {
        printf("Konnte Geraedatei nicht oeffnen!!\n"); exit(1);
    }
    // aktuelle port-einstellungen sichern
    tcgetattr( fd, &oldtio );
    bzero( &newtio, sizeof(newtio) );
    // einstellungen und eingabe-modus bestimmen
    newtio.c_cflag     = BAUDRATE | CS8 | CLOCAL | CSTOPB | CREAD;
    newtio.c_iflag     = IGNPAR;
    newtio.c_oflag     = 0;
    newtio.c_lflag     = 0;
    newtio.c_cc[VTIME] = 0;
    newtio.c_cc[VMIN]  = 1;
    // einstellungen setzen
    tcflush( fd, TCIFLUSH );
    tcsetattr( fd, TCSANOW, &newtio );
    //  zu versendende befehlsfolge  { CR, Karte, Befehl, Relais/Adr., Checksum } ;
    cmd[0] = 13;
    cmd[1] = *argv[2];
    cmd[2] = argv[3][1];
    cmd[3] = *argv[4];
    // Checksumme berechnen
    cmd[4] = 256-(cmd[0]+cmd[1]+cmd[2]+cmd[3]);
    // Debug - Ausgabe
    cout << "Befehl senden:  [ Karte: " << cmd[1] << "  Befehl: " << cmd[2] << "/" << (int)cmd[2]
         << "  Relais/Adresse: " << cmd[3] << "/" << (int)cmd[3] << " \t(Checksumme: " << cmd[4] << "/" << (int)cmd[4] << ") ] \n" ;
    // mehrmals hintereinander senden
    for ( int tmp=0; tmp<=5; tmp++ )
    {  // 5 bytes pro befehl
        for ( int b=0; b<=5; b++ )
        {
            if ( (write( fd, &cmd[b], 1 )) < 0 ) { fputs("write() fehlgeschlagen!\n", stderr); }
        }
    }
    // alte porteinstellungen zuruecksetzen
    tcsetattr( fd, TCSANOW, &oldtio );
    // Pause zwischen verschiedenen Befehlen erzwingen
    usleep( 1000000 );
    return 0;
}
```

Der Programmaufruf gestaltet sich wie folgt:

```
programm <GERAETEDATEI> <KARTENNUMMER> <BEFEHL> <RELAIS-Nr/WERT>
```

Beispiele:

```
programm /dev/ttyS0 1 -S 9    -> alle 8 Relais [ein] auf 1. Karte an
programm /dev/ttyS1 7 -C 5    -> Relais 5 [aus] auf 7. Karte an COM2
programm /dev/ttyS0 1 -T 2    -> Relais 2 auf 1. Karte toggeln (COM1)
programm /dev/ttyS0 0 -E 0    -> alle Relais auf allen Karten [aus]
```

Durch geschickte Verknüpfung mit anderen Linux-Programmen wie z.B. Cron zur Zeitsteuerung, ISDNLog zum Befehlsaufruf bei ankommenden Anrufen oder Apache/PHP für dynamische Webseiten[11] lassen sich die eingangs beschriebenen Ziele dann einfach verwirklichen.

5 Bauteile-Liste

(Exported from Relaiskarte-v10.brd at 1/22/2005 23:47:48 / EAGLE Version 4.13r1 Copyright (c) 1988-2004 CadSoft)

Part	Value	Package	Library
12V-DC-IN		W237-102	con-wago-500
230V-AC-IN		W237-102	con-wago-500
AC-1 – AC-8		W237-102	con-wago-500
C1, C2	100nF	C050-024X044	rcl
C3	100uF	E5-10,5	rcl
C4	100uF	E5-10,5	rcl
C5	1000uF	E5-8,5	rcl
D1 - D8	1N4148	DO35-10	diode
IC1	PIC16F630	DIL14	pic16f6xx_14pin
IC3		78XXS	v-reg
J1 - J9	J5MM	05	jumper
JP1		JP2	jumper
LED-1 – LED-8		LED3MM	led
R1	VDR300	S14K300	varistor
R2	10K	0207/10	rcl
R3	56	0207/10	rcl
R4	10K	0207/10	rcl
R5	56	0207/10	rcl
R6	10K	0207/10	rcl
R7	56	0207/10	rcl
R8	10K	0207/10	rcl
R9	56	0207/10	rcl
R10	10K	0207/10	rcl
R11	56	0207/10	rcl
R12	10K	0207/10	rcl
R13	56	0207/10	rcl
R14	10K	0207/10	rcl
R15	56	0207/10	rcl
R16	10K	0207/10	rcl
R17	56	0207/10	rcl
R18		0207/10	rcl
R19	470	0207/10	rcl
R20	VDR300	S14K300	varistor
R21	4R7	0207/10	rcl
R22	1K	0207/10	rcl
R23	10K	0207/10	rcl
R24	1K	0207/10	rcl
R25	VDR300	S14K300	varistor
R26	VDR300	S14K300	varistor
R27	VDR300	S14K300	varistor
R28	VDR300	S14K300	varistor
R29	VDR300	S14K300	varistor
R30	VDR300	S14K300	varistor
RELAIS-1 - RELAIS-8		RP51	relay
RS232-IN		W237-102	con-wago-500
RX-LED		LED3MM	led
T1 - T9	BC547A	TO92	transistor-npn
TEST		B3F-10XX	switch-omron
ZD	5V6	DO35Z10	diode

11 um z.B. auch per Internet Geräte ein- und ausschalten zu können

6 Quellenangaben

Tipps zum Umgang und Programmieren von PICs:
http://home.t-online.de/home/holger.klabunde/pichelp.htm#rccal

Viele Informationen rund um die PIC Microcontroller:
http://home.arcor.de/kay.galinsky/PICs/pics.html

Microchip-Homepage – Hersteller des PIC 16F630:

Homepage:
http://www.microchip.com/
Chipbeschreibung PIC 16F630:
/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1335&dDocName=en010211
Datenblatt PIC 16F6F630/676:
http://ww1.microchip.com/downloads/en/DeviceDoc/40039c.pdf

Velleman Homepage

Homepage
http://www.velleman.be/
Datenblatt des Bausatzes K8056 mit Schaltplan:
http://www.velleman.be/Downloads/0/Manual_K8056.pdf

Anregungen zur Software:
http://www.relaiskarte.thomas-dohl.de/
http://www.netzmafia.de/skripten/hardware/relais/relais.html