# 8-fach 230V-Dimmerschaltung

Martin Pitt
Home: http://www.piware.de
eMail: martin@piware.de

3. Oktober 2002

**Zusammenfassung**

Dieses Dokument beschreibt die Hard- und Software einer Dimmerschaltung mit 8 230V-Ausgängen. Die Steuerung erfolgt über eine RS232-Schnittstelle, kann also mit vielen Geräte (PCs, PDAs, Micro-Controller) angesteuert werden. Die Hardware wurde darauf ausgelegt, möglichst flexibel, aber preisgünstig zu sein.

## 1 Entstehung und Motivation

Die Aufgabe war, eine Bühnenbeleuchtung für 6 300W-Strahler zu entwerfen, die mit einem Notebook sehr flexibel steuerbar ist. Dabei sollten die Lampen einzeln dimmbar und der Einsatz als VU-Meter ebenfalls möglich sein. Da handelsübliche Anlagen viel zu teuer sind (ab etwa 500 € ist man dabei), fiel die Entscheidung auf einen Selbstbau, der mit etwa 25 € zu Buche schlägt und unsere Bedürfnisse übererfüllte.

Die Controller-Hardware ist dabei auf maximal 16 Ausgänge zugeschnitten, die stufenlos gedimmt werden können. Die vorliegende Software unterstützt momentan jedoch nur 8 Ausgänge mit 32 Helligkeitsstufen, um das Steuerprotokoll so einfach wie möglich zu halten.

## 2 Allgemeiner Aufbau

Der Dimmer wurde auf zwei Platinen verteilt. Die Controller-Platine kommuniziert über RS232 mit dem Steuergerät (PC, Notebook, PDA, Mikrocontroller, etc.) und ist für die Steuerung der Ausgänge durch Phasenanschnitt verantwortlich. Auf ihr befinden sich außer dem Trafo-Primäranschluss nur 5V-Niederspannungskomponenten. Für die Platine existiert ein fertiges Layout (einseitige Leiterplatte, ca. 90 × 60 mm).

Über ein neunadriges Flachbandkabel (acht Steuerleitungen und Masse) erfolgt die Verbindung zu den Ausgangstreibern. Die Nieder- und Hochvoltkreise sind durch Optokoppler galvanisch voneinander getrennt, um das Risiko teurer und gefährlicher Schäden zu minimieren. Für diese Platine existiert kein Layout, ich habe sie auf einer Lochrasterkarte handverdrahtet. Da in unserem Fall bis zu 8 A durch die Leitungen fliessen, ist eine Kupferbahnleiterplatte auch nicht mehr angebracht.

Die Schaltpläne und das Platinendesign des Controllers wurden mit EAGLE (www.cadsoft.de) entworfen. Sie von meiner Homepage (www.piware.de/hardware.shtml) heruntergeladen und beliebig verwendet werden. In dem Archiv ist auch die nötige Software für den Atmel vorhanden.

# 3 Der Controller

Abbildung 1 (S. 7) zeigt den Schaltplan des Controllers.

Die Platine gliedert sich in Stromversorgung (im Schaltbild der Block links oben), Mikrocontroller und dessen Beschaltung (Mitte bis rechts unten) und RS232-Schnittstelle (links unten).

## 3.1 Stromversorgung

Weil für den Dimmer sowieso Netzspannung zur Verfügung steht und ausserdem deren Nulldurchgänge detektiert werden müssen, wird die Platine über einen Trafo und Gleichrichter mit Strom versorgt.

Die Gleichrichtung und Stabilisierung auf 5 V erfolgt über D1, C1, den 5V-Festspannungsregler IC1, C2 und C9 filtern dann noch hoch- und niederfrequente Störungen aus. Die entstehende Gleichspannung ist nicht sehr glatt, reicht aber für den Betrieb der Platine völlig aus.

R4 und D2 sind optional und stellen nur eine Betriebsanzeige dar.

Zur Nulldurchgangserkennung wird der im Mikrocontroller integrierte Operationsverstärker (dort „Analog comparator“ genannt) verwendet. Um dessen Eingänge nicht mit negativen oder hohen Spannungen zu belasten, reduziert der Spannungsteiler R1/R3 über die Diode D3 die 9V-Wechselspannung auf erträgliche 4.1 VAC (im Schaltbild V+) und filtert negative Halbwellen weg. Der Spannungsteiler R5/R6 dient dann als Referenzspannung, im Schaltbind V-.

## 3.2 Mikrocontroller

Als Mikrokontroller wurde ein Atmel AT90S4414 verwendet, weil er preisgünstig und einfach zu beschalten ist und sehr viel integrierte Peripherie mitbringt. Für unsere Zwecke brauchen wir den Analogkomparator zur Nulldurchgangserkennung und den RS232-UART.

Der Takt wurde auf 4 MHz festgelegt, dabei bleibt der Atmel sehr stabil und es ermöglicht ein sehr feines Timing. Herausgeführt wurde Port A, welcher die Phasenanschnitts-Steuersignale an die Ausgangstreiber übermittelt und Port C für eventuelle Erweiterungen.

Zur Programmierung wurde auch der SPI-Bus (JP9) als Pinleiste herausgeführt, so kann der Atmel mit geeigneten Programmen direkt in der Schaltung (in circuit) programmiert werden. Ich verwende die Open Source Software `uisp` (`www.amalek.gda.pl/avr/uisp/`) unter Linux; es gibt auch eine etwas umständlich zu bedienende, aber ebenfalls freie Portierung auf Windows, die ich auch kurz erfolgreich getestet habe.

## 3.3 RS232-Schnittstelle

Da der Atmel nur TTL-Signale liefert und empfängt, müssen diese mit Hilfe des MAX232 (IC3) noch in die RS232-Pegel (+9V und -9V) transformiert werden. An JP2 erfolgt dann der Anschluss eines seriellen Kabels, vorzugsweise über eine 9-polige Sub-D-Buchse (wenn ein serielles Anschlusskabel verwendet werden soll) oder eines 9-poligen Sub-D-Steckers (bei Verwendung eines Nullmodemkabels). Pin 1 ist Masse, Pin 2 der Receive-Eingang und Pin3 der Transmit-Ausgang. Die vorliegende Software sendet nichts an das Kontrollgerät, so dass die Transmit-Leitung nur der Vollständigkeit halber vorhanden ist.

### 3.4 Benötigte Bauelemente

| Anzahl | Bauelement | Bezeichnung |
|---|---|---|
| | *Widerstände* | |
| 1 | 150 Ω | R4 |
| 2 | 2,2 kΩ | R1, R2 |
| 1 | 5,6 kΩ | R6 |
| 2 | 10 kΩ | R3, R5 |
| | *Kondensatoren* | |
| 1 | 470 $\mu$F Elko 7.5 mm | C1 |
| 1 | 100 nF | C2 |
| 2 | 15 pF | C3, C4 |
| 4 | 1 $\mu$F Elko 5 mm | C5 – C8 |
| 1 | 1 $\mu$F Elko 2.5 mm | C9 |
| | *Halbleiter* | |
| 2 | Diode 1N4001 bis 4004 | D1, D3 |
| 1 | LED 5 mm (20mA) | D2 |
| 1 | Quarz 4 MHz | Q1 |
| | *ICs* | |
| 1 | 5V-Festspannungsregler 7805 (TO220) | IC1 |
| 1 | Atmel AT90S4414 (DIL40) | IC2 |
| 1 | MAX232 (DIL16) | IC3 |
| | *Stecker und Fassungen* | |
| 1 | Fassung DIL40 | für Atmel |
| 1 | Fassung DIL16 | für MAX232 |
| 1 | Pinleiste 3x1 | SERIAL |
| 1 | Pinleiste 3x2 | JP9 |
| 2 | Pinleisten 8x1 | PORTA, PORTC |
| | *Sonstiges* | |
| 1 | Trafo 220VAC - 9VAC | TR1 |
| 1 | Flachbandkabel 9-adrig | für PORTA |

# 4 Die Ausgangs-Treiber

Das Dimmen erfolgt über je eine Triac-Stufe pro Ausgang durch Phasenanschnitt. Die Steuerung der Gates erfolgt über einen Optokoppler durch die Controller-Platine, die Treiberstufe selbst enthält keinerlei Logik.

Abbildung 2 (S. 8) zeigt den Schaltplan der Ausgangstreiberstufe in 8-facher Realisierung. JP1 ist eine 8-polige Pinleiste, die über ein Flachbandkabel mit Port A der Controllerplatine verbunden ist und die Steuersignale für die Triacs überträgt. JP2 ist ein weiterer Pin, der mit der Controller-Masse verbunden werden muss, also an Pin 1 von JP9. Natürlich kann ein 9-adriges Flachbandkabel verwendet werden.

Werden andere Triacs verwendet, muss die Dimensionierung der Gate-Widerstände R1 bis R8 angepasst werden. Sie müssen so gross sein, dass durch die Optokoppler ein maximaler (Peak-) Strom von 1 A fliesst. Andererseits sollten sie auch nicht zu gross sein, damit der Triac möglichst früh nach dem Nulldurchgang zünden kann. Deshalb wurde hier der TIC 225 vorgeschlagen, da er schon ab 5 bis 10 mA, also schon ab etwa 18 V der Netzspannung zündet.

Bei anderen Optokopplertypen (z. B. MOC 3020 bis 3022) müssen die Dioden-Vorwiderstände R9 bis R16 angepasst werden, um den Optokoppler-LEDS genügend Strom zu geben. Der benötigte Schaltstrom („forward current“) ist in den Datenblättern nachzulesen.

Werden an die Ausgänge Verbraucher sehr hoher Leistung angeschlossen, dürfen diese unter

Umständen nicht mehr an einen gemeinsamen Null-Leiter angeschlossen werden; übliche Haushaltssicherungen sind gegen 10 A abgesichert, was einen längeren Betrieb von etwa 1,5 kW pro Null-Leiter gestattet. Es ist aber *unbedingt* darauf zu achten, dass sämtliche verwendete Steckdosen für die Ausgangstreiber und die Steuerplatine an *der gleichen* Phase angeschlossen sind, da sonst unterschiedliche Nulldurchgangszeitpunkte vorliegen und kein Phasenanschnitt mehr möglich ist.

Werden hohe Leistungen geschaltet, so müssen die Triacs passiv gekühlt werden. Beim Aufbau der Leiterplatte sollte darauf geachtet werden, die Triacs möglichst nahe und parallel zueinander zu plazieren, dann kann ein großer gemeinsamer Kühlkörper verwendet werden.

Für $n$ Treiberstufen werden folgende Bauteile benötigt:

| **Anzahl** | **Bauelement** | **Bezeichnung** |
|---|---|---|
| $n$ | Optokoppler MOC 3023 (o. ä.) | OK1... OK$n$ |
| $n$ | Triac TIC 225 (o. ä.) | T1... T$n$ |
| $n$ | Widerstand 1,8 kΩ, Metallschicht | R1... R8 |
| $n$ | Widerstand 350 Ω | R9... R16 |
| $n+1$ | Lüsterklemmen | PAD1... PAD$n$, PE |
| 1 | 9-polige Stiftleiste | JP1, JP2 |
| 1... 2 | Kühlkörper | — |

# 5 Software

## 5.1 Anforderungen

Die Steuersoftware muss die folgenden Funktionen erfüllen:

- Kommunikation mit dem Steuergerät über den RS232- oder SPI-Bus
- Abfrage des AC-IRQs; dieser wird bei jedem Nulldurchgang der Netzspannung ausgelöst und muss die Zünd-Timer synchronisieren
- Zünden der Triacs zum entsprechenden Zeitpunkt nach dem letzten Nulldurchgang durch Setzen des der Lampe entsprechenden Bits in Port A. Das Bit sollte recht schnell wieder gelöscht werden (minimal 25 $\mu$s), um nicht unnötig lange Strom durch die Triac-Gates fliessen zu lassen.
- Die Helligkeitsstufen müssen von der linearen Skala auf das integrale Flächenverhältnis der durchgelassenen zur gesamten Halbwelle umgerechnet werden. Die Formel und deren Herleitung werden in Abschnitt 5.3 beschrieben.

## 5.2 Vorliegende Software

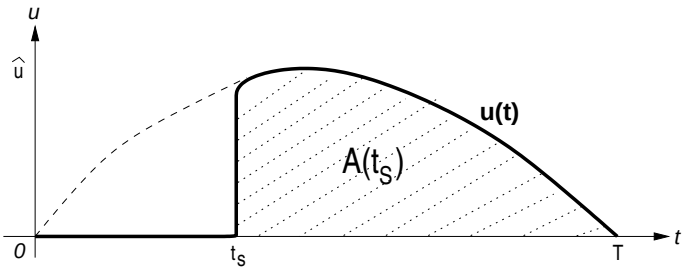
Die vorliegende Steuersoftware für den Atmel ist nur ein Vorschlag und war nur auf die aktuellen Bedürfnisse zugeschnitten. Sie unterstützt 8 angeschlossene Lampen an Port A mit 32 Helligkeitsstufen. Das Steuerprotokoll wurde dabei sehr einfach gehalten:

Der Atmel sendet keinerlei Daten an das Steuergerät. Gesteuert werden die Lampen durch das Senden eines einzelnen Bytes. Dessen drei höchstwertige Bits geben dabei die Lampennummer (0 (000b) bis 7 (111b)) an, die fünf niederwertigen Bits die Helligkeitsstufe (0 (00000b) für aus, 31 (1111b) für maximale Helligkeit). Die Helligkeitsstufen sind linear, die Steuersoftware setzt dann mit Hilfe einer Umrechnungstabelle (erzeugt in der Routine `InitConversionTable`) den entsprechenden Zündzeitpunkt des Triacs.

Die Software kann von meiner Homepage (`http://www.piware.de/hardware.shtml`) als Quellcode und als assemblierter HEX-Code heruntergeladen und beliebig verwendet werden.

## 5.3 Linearisierung der Helligkeitswerte

Hier ist eine Halbwelle der Netzspannung abgebildet. Die dickgezeichnete Funktion $u(t)$ ist phasenangeschnitten, die gestrichelte Linie stellt den nicht angeschnittenen Phasenverlauf dar:




Dabei ist $T$ die Dauer einer Halbwelle (z. B. in der vorliegenden Software 100) und $t_S$ der Zündzeitpunkt nach dem Nulldurchgang, gemessen in der Einheit von $T$. $\hat{u}$ ist die Amplitude der Netzspannung, die aber aus Linearitätsgründen nicht von Belang ist.

Eine nichtangeschnittene Halbwelle wird durch $u(t) = \hat{u} \cdot \sin(\frac{\pi}{T}t)$ beschrieben. Die Fläche $A(t_S)$ ist demnach

$$A(t_S) = \int_{t_s}^{T} u(t)dt = \frac{\hat{u}T}{\pi}\left(1 + \cos(\frac{\pi}{T}t)\right)$$

Der lineare Helligkeitswert $p \in [0, 1]$ berechnet sich zu $p = \frac{A(t_s)}{A(0)}$. So ergibt sich durch Einsetzen und Umstellen die Formel, die aus einem gegebenen Helligkeitswert $p$ den gesuchten Zündzeitpunkt $t_s$ bestimmt:

$$t_s = \frac{T}{\pi}\arccos(2p - 1)$$

In der vorliegenden Software wurde eine Halbwelle mit Hilfe eines Timers in 100 Abschnitte zerlegt, d. h. dort ist $T = 100$. Die Berechnung der Zündzeitpunkte erfolgte mit einer Tabelle, die durch ein kleines Perl-Skript erzeugt und in den Quellcode integriert wurde.

# 6 Copyright und verwendete Software

Zur Erstellung dieses Projekts kam folgende Software zum Einsatz:

- Betriebssystem Debian GNU/Linux 3.0, welches die gesamte benötigte Software außer Eagle schon mitbringt :-)
- Die Schaltplan- und Layoutsoftware Eagle 4.0 (kostenlose Demoversion von `www.cadsoft.de`)
- Der Editor Emacs zur Erstellung der Atmel-Software und dieser Dokumentation.
- LaTeX $2_\varepsilon$ zum Setzen dieser Dokumentation.
- Der Atmel-AVR-Assembler avra (`tihlde.org/~jonah/el/avra.html`) zum Assemblieren der Atmel-Software
- Der $\mu$-in-system-programmer uisp (`www.amalek.gda.pl/avr/uisp/`) zum Programmieren des Atmel
- Der Alles-Spieler xmms zur musikalischen Inspiration :-)

Das gesamte Projekt – das umfasst die Schaltpläne, das Platinendesign, die Software und diese Dokumentation – kann frei verwendet, kopiert und modifiziert werden. Ich wäre aber über Feedback an `martin@piware.de`, vor allem über Verbesserungsvorschläge und Kritik, sehr dankbar!

Martin Pitt

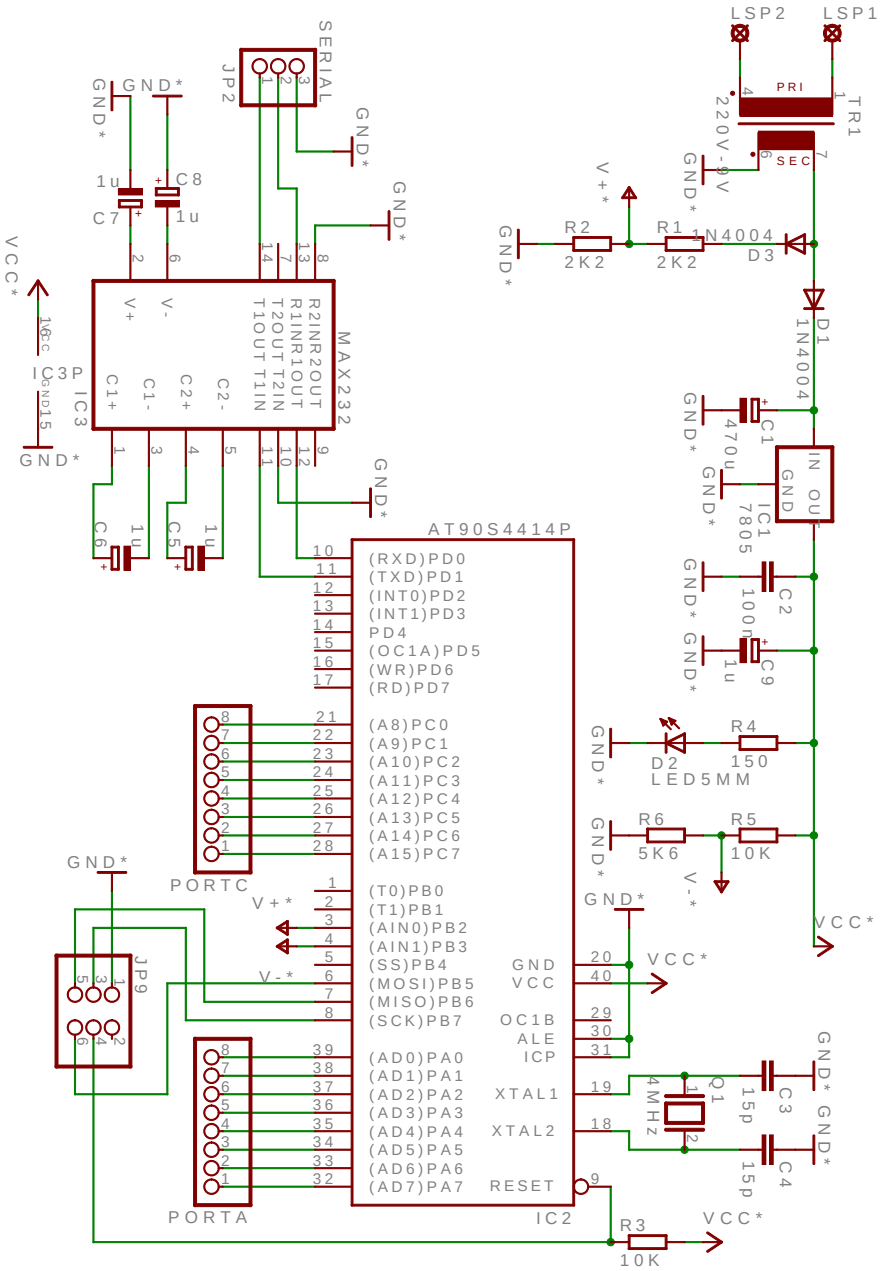



Abbildung 1: Schaltplan des Controllers

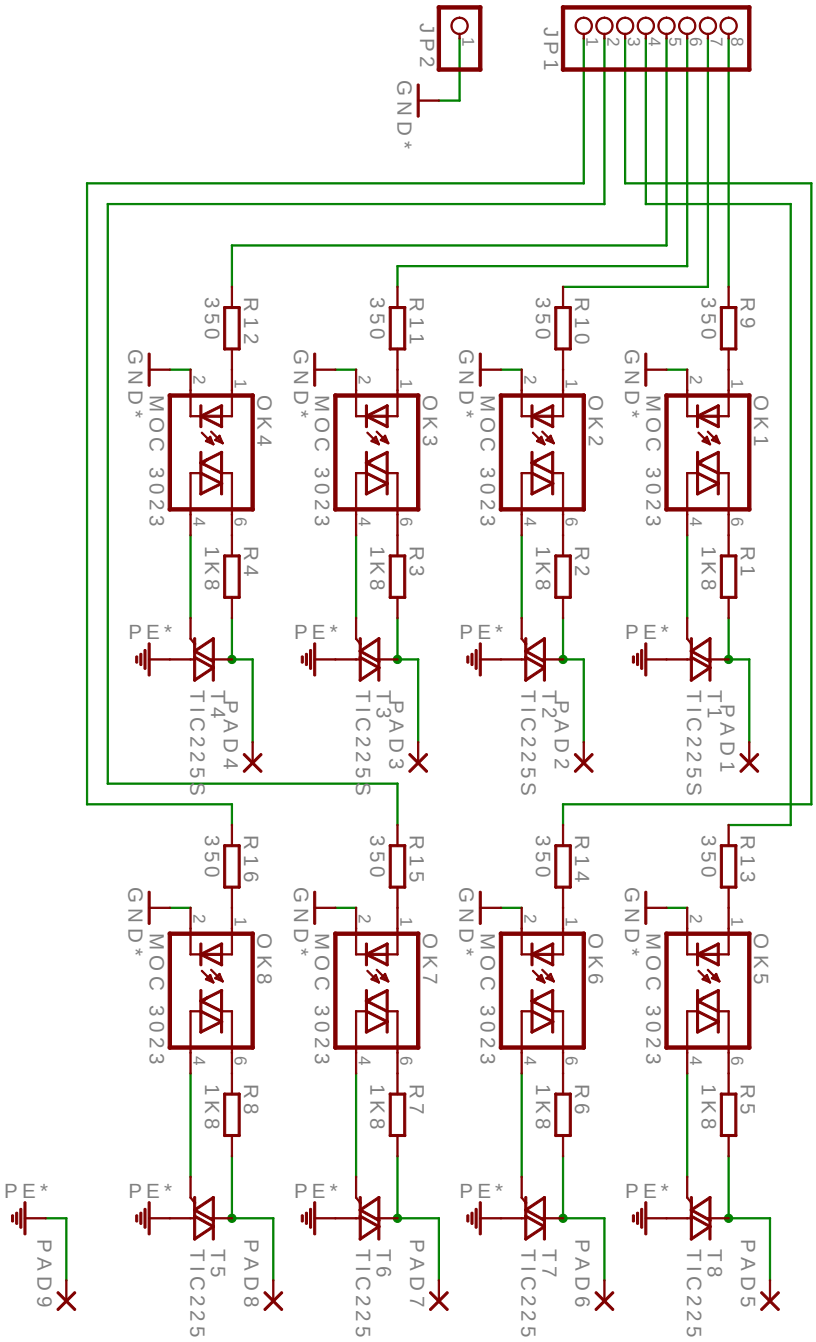



Abbildung 2: Schaltplan der Ausgangs-Treiber