01.09.2008

Henne's Sites

AN014: Dimmen von Leuchtdioden

Inhalt

- Einleitung
- Nutzungsbedingungen
- C-Code

Einleitung

Gewöhnlich wird die Intensität von Leuchtdioden mittels Pulsweitenmodulation gesteuert. Dies hat gegenüber einer analogen Stromregelung den Vorteil, dass der Farbpunkt der LED konstant bleibt und im Leistungsschalter nur wenig Abwärme anfällt.

Soll die maximale Intensität von High Power LEDs ohne Verringerung der nominellen Lebensdauer erreicht werden, ist der Betrieb an einer Konstantstromquelle notwendig. Für den Aufbau von linearen oder getakteten Konstantstromquellen soll an dieser Stelle auf die Application Notes und Datenblätter diverser Halbleiterhersteller verwiesen werden.

Diese AN befasst sich mit Verfahren zur notwendigen Umgehung eines Patentes von Colorkinetics / Philips, welches Geräte mit folgenden Eigenschaften abdeckt:

- Es werden insgesamt mindestens zwei Kanäle mittels PWM gedimmt.
- Die Leuchtdioden werden an Konstantstromquellen betrieben.
- Für das Dimmen ist ein Mikrocontroller, FPGA, etc. zuständig, der über einen Bus mit anderen Geräten kommuniziert.

Da dieses anscheinende Trivialpatent vor wenigen Jahren von amerikanischen Gerichten bestätigt wurde, hat Artistic License die Bitangle-Modulation (BAM) eingeführt. Bei diesem Verfahren wird jedem Bit eine feste Pulslänge entsprechend seiner Wertigkeit zugeordnet. Ist nun ein Bit im Helligkeitswert eines Kanals gesetzt, so wird der Ausgang dieses Kanals für die dem Bit entsprechende Zeitdauer aktiviert.

BAM ist einerseits royalty-free – stellt aber hohe Anforderungen an den Zeitdeterminismus der Firmware und benötigt für ein homogenes Fading eine sehr hohe refresh rate.

Abb. 1: BAM-Pattern bei einer Intensität von 21

Alternativ dazu wurde von EZ Semiconductor eine stochastische Modulation zur Intensitätsregelung von LEDs patentiert. Hierbei werden die Intensitätswerte mit dem Ergebnis eines Zufallsgenerators verglichen. Ist der Intensitätswert größer oder gleich dem Zufallswert, wird der Ausgang des jeweiligen Kanals aktiviert.

Abgesehen von evtl. anfallenden Lizenzkosten sieht der Autor hier die Gefahr von einer Schwebung der Intensität, falls der der Zufallsgenerator bevorzugt für einen kurzen Zeitabschnitt sehr kleine oder sehr große Werte ausgibt.

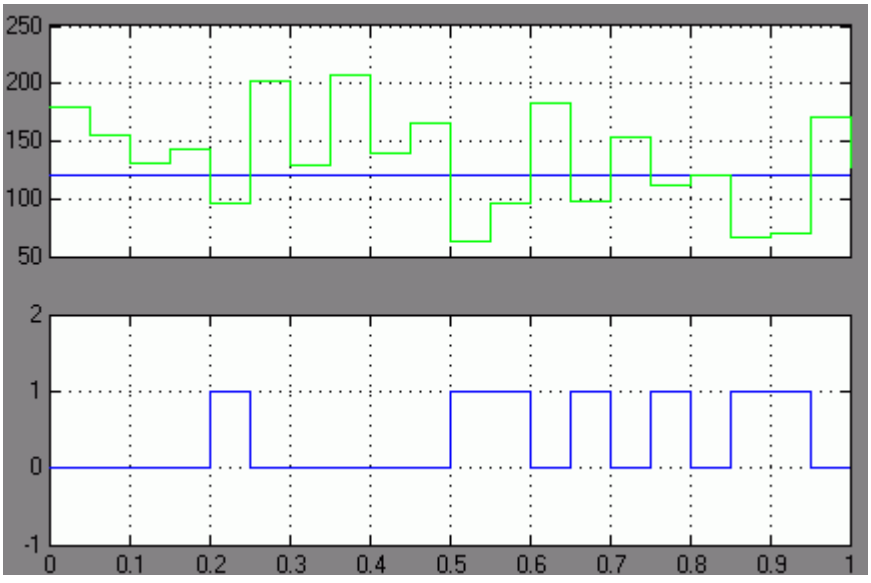

Abb. 2: SSTM-Pattern bei einer Intensität von 50% (blau)

Die Nachteile der genannten Verfahren sollen durch die im Folgenden beschriebene Pulsdichtemodulation (PDM) behoben werden. Eine oberflächliche Patentrecherche ergab, dass dieses Verfahren bislang noch nicht im Zusammenhang mit dem Dimmen von LEDs erwähnt wurde. (Sollte dennoch ein Patent bestehen, würde der Autor sich über eine Nachricht freuen.)

Bei der PDM wird der Intensitätswert eines Kanals mit einem „springenden Vergleichswert“ verglichen. Ist der Intensitätswert größer oder gleich dem Vergleichswert, wird der Ausgang des jeweiligen Kanals aktiviert. Der Vergleichswert wird gebildet, indem bei jedem zweiten Vergleichsdurchlauf das most significant bit (MSB) getoggelt wird und in den anderen Fällen der Vergleichswert inkrementiert wird. Für eine PDM mit 3bit ergibt sich somit folgende Sequenz:

0b100
0b001
0b101
0b010
0b110
0b011
0b111

Dieses Verfahren hat zusätzlich den Vorteil einer virtuell sehr viel höheren refresh rate als die einer gewöhnlichen PWM bei sonst vergleichbarem Verhalten.
Mit PDM lassen sich auch andere Lasten wie Laser, CCFL oder Motoren steuern. Die beschriebene Sequenz ist nur beispielhaft: Es können durchaus auch andere Bits zu anderen Zeitpunkten getoggelt werden.

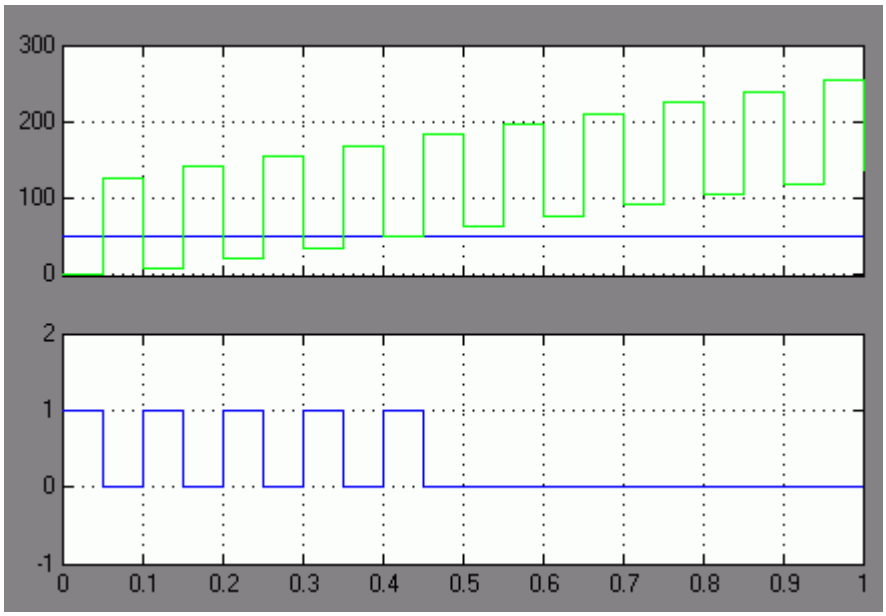

Abb. 3: PDM-Pattern bei einer Intensität von 20%

Nutzungsbedingungen

Der Code (bzw. das Verfahren) darf gemäß der gnu general public license (GPL) genutzt werden.

Falls eine GPL-konforme Nutzung auf Grund einer kommerziellen Verwendung nicht möglich ist, kann natürlich auch eine geringe „Lizenzgebühr“ vereinbart werden.

C-Code

Der folgende Code wurde für den DMX-Transceiver auf Basis eines ATmega8515 von ATMEL geschrieben und sollte mit möglichst hoher Frequenz in einer Timer-ISR ausgeführt werden.

```
uint8_t PdmState= 0;
if (PdmField[0] >= PdmCompare) PdmState|= (1<<CH1);       //compare channels
if (PdmField[1] >= PdmCompare) PdmState|= (1<<CH2);
if (PdmField[2] >= PdmCompare) PdmState|= (1<<CH3);
if (PdmField[3] >= PdmCompare) PdmState|= (1<<CH4);
if (PdmField[4] >= PdmCompare) PdmState|= (1<<CH5);
if (PdmField[5] >= PdmCompare) PdmState|= (1<<CH6);
if (PdmField[6] >= PdmCompare) PdmState|= (1<<CH7);
if (PdmField[7] >= PdmCompare) PdmState|= (1<<CH8);
OUTPORT= PdmState;

if (PdmCompare &(1<<7))
      {
      PdmCompare ++;                                      //increment compare register
      PdmCompare &= ~(1<<7);                              //clear MSB
      if (PdmCompare == 0) PdmCompare = 0x80;
      }
else PdmCompare |= (1<<7);                                //set MSB
```