

# ***LightStack***

**- Technische Beschreibung -**  
V2.0



# INHALTSVERZEICHNIS

1	Einführung	3
2	Sprachbeschreibung	3
	2.1 Zeitsteuerung	3
	2.2 Helligkeitsstufen	3
	2.3 Beispiel für einen Befehl	4
3	Befehlsbeschreibung	5
	3.1 STANDARD Befehlssatz	5
	3.2 EXTENDED Befehlssatz	5
4	Einfaches Programmbeispiel	6
5	Schnittstellen zum <i>LightStack</i>	6
6	Befehlsbeschreibung	7
7	Anhang – Beispiele	10
8	Kontakt	10

# 1 Einführung

Leuchtdioden werden immer heller. Durch den Einsatz von farbigen und weißen LEDs bieten sich völlig neue Licht-Gestaltungsmöglichkeiten. Die Industrie bietet inzwischen eine Vielzahl von Treibern an, um Leuchtdioden unterschiedlicher Stromstärke zu betreiben. Dabei gerät leider die Software in den Hintergrund, die notwendig ist, um die LEDs zu steuern. Lichtsteuerungen wie Farbwechsel, Dimmen, Erzeugung von Zufallsfarben oder langsame, kontinuierliche Farbübergänge werden heute üblicherweise durch hardwareabhängige, fest programmierte Algorithmen realisiert. Viel einfacher ist es aber, wenn man eine simple, hardwareunabhängige Programmiersprache nutzen könnte. Um die Entwicklung solcher Lichtsequenzen zu vereinfachen, wurde die Sprache *LightStack* entwickelt, die speziell auf die Bedürfnisse von Lichtsteuerungen zugeschnitten ist. Diese Sprache bietet einfache Beschreibungsmittel um (fast) beliebige Lichtsteuer-Sequenzen zu erzeugen.

Folgende Features sind implementiert:

- wenige, einfache Standard Befehle, die den allgemeinen Steuerungsbedarf abdecken
- Skalierbarkeit (Anzahl der zu steuernden Lichter / LED's)
- Hardwareunabhängigkeit
- wahlweise bis zu 65535 Helligkeitsstufen pro LED, je nach verfügbarer Rechenleistung bzw. PWM-Modus (Pulsweitenmodulation)
- Erweiterbarkeit des Befehlssatzes zur Implementierung von komplexen Steuerungen

## 2 Sprachbeschreibung

### 2.1 Zeitsteuerung

*LightStack* implementiert eine zeitgesteuerte Lichtsteuerung mit frei wählbarer Basisfrequenz (üblicherweise 100 Hz = 10 ms). Zu jedem Clock-Tick wird der *LightStack* aufgerufen, dieser bearbeitet entweder den aktuellen Befehl, oder holt sich, wenn der letzte Befehl fertig abgearbeitet ist, den nächsten auszuführenden Befehl über eine Callback-Funktion. Diese Befehle beinhalten üblicherweise 2 Zeitangaben, nämlich Minimal- und Maximal-Zeit für die Befehls-Ausführung. Es wird dann ein zufälliger Wert zwischen Min- und Max-Zeit ermittelt. Um feste Zeiten vorzugeben (z.B. eine Sekunde), werden einfach beide Werte gleichgesetzt (100,100). Zeitwerte sind 16 Bit Werte, als Zeitbasis dient die eingestellte Frequenz (typisch 100 Hz), maximaler Zeitwert für einen Befehl ist dann also 655,35 Sekunden.

### 2.2 Helligkeitsstufen

Für die Standard-Befehle wurden 16 Helligkeitsstufen pro LED gewählt (4 Bit). Auch hier werden wieder immer 2 Werte angegeben, Minimal- und Maximal-Wert, der echte Wert wird dann per Zufall aus diesem Bereich ermittelt. Für fest-vorgegebene Werte müssen beide Werte gleichgesetzt werden. Intern wird aber nicht mit 4-Bit Werten, sondern mit 16-Bit Werten gerechnet (65536 Helligkeitsstufen). Somit können bei Farbwechsel-Steuerungen etc. immer sanfte Übergänge erreicht werden.

Da die Helligkeitszunahme von LEDs nicht linear zur angesteuerten Pulsbreite zunimmt und die Lichtempfindlichkeit des Auges nicht linear ist, können diese 16-Bit Werte nach der Berechnung über eine Formel oder eine Tabelle auf die implementierungsabhängig erreichbaren Helligkeitsstufen abgebildet werden, z.B. 1024 Stufen bei 10 Bit PWM. Bei den zeitgesteuerten Übergängen wird der aktuelle Helligkeitswert jeweils durch Interpolation zwischen Start- und Endwert errechnet. Bei Steuerung einer RGB-Leuchtdiode ergeben sich bei 16 Stufen pro Farbe insgesamt  $16 \times 16 \times 16 = 4096$  verschiedene Farben. Wenn das nicht ausreicht, können immer noch alle verfügbaren Helligkeitsstufen über erweiterte (nicht so effiziente) Befehle erreicht werden. Die Helligkeitsstufen werden wie die Zeitwerte als Minimal- und Maximal-Werte angegeben, also ein Byte pro LED. Die Anzahl der möglichen echten Helligkeitsstufen hängt zum einen von der verfügbaren Rechenleistung ab (wenn man z.B. 1024 Stufen per Software-Pulsweiten-Modulation implementiert, muss die CPU 102 400 Interrupts pro Sekunde abarbeiten können). Zum anderen ist das von den verwendeten LED-Treibern abhängig, viele Hochstrom-LED-Treiber können z.B. nur 100 Helligkeitsstufen differenzieren.

### 2.3 Beispiel für einen Befehl

Ansteuerung einer RGB-LED (3 Lichter):

**SET (TIME(100, 500), VAL(0xff, 0x0f, 0x77))**

bedeutet: innerhalb einer Zufallszeit zwischen **1** und **5** Sekunden (**TIME(100,500)**)  
Wechseln in den Zustand

- **ROT: 100 % (0xff)**
- **GRÜN: Zufallswert zwischen 0 und 100 % (0x0f)**
- **BLAU 50 % (0x77).**

Für die Farbwerte wird zur Vereinfachung die hexadezimale Schreibweise verwendet, Werte zwischen 0 und f bedeuten zwischen 0 und 15, also ff bedeutet fest auf den maximalen Wert 15 setzen. Bei den Werten sollte immer erst der kleinere und dann der größere aufgeführt werden (z.B. 1e für Wert zwischen 1 und 14). Somit können die restlichen Werte für Spezial-Parameter genutzt werden, beispielsweise 21, um einen Wert unverändert zu lassen.

### 3 Befehlsbeschreibung

#### 3.1 STANDARD Befehlssatz

<b>SET</b> (TIME(min, max), VAL(data4[..]))	Wechsel in Zustand data4[..] innerhalb einer Zeit zwischen min und max
<b>IDLE</b> (TIME(min, max))	Nichts ändern in der Zeit zwischen min und max
<b>JUMP</b> (destination)	Sprung auf <destination>
<b>LOOP</b> (destination, COUNT(min, max))	Schleife, zwischen min und max nach Destination, dann weiter
<b>MOV</b> (R<x>, VAL(data4[..]))	Internes Register setzen
<b>MIX</b> (TIME(min, max), VAL(data4[..]))	Wie SET, die Werte data4[..] werden in zufälliger Reihenfolge den Lichtern 1..n zugewiesen.
<b>DONE</b>	Ende eines Programms. Über diesen Befehl kann man von der Hauptschleife aus ein Nachfolge-Programm starten, falls gewünscht

Mit Hilfe dieses Befehlssatzes können bereits einfache Lichtsteuerungen implementiert werden. Die benötigte Code-Größe für diesen Befehlssatz liegt bei ca. 4 KByte. Damit können Lichtsteuerungen bereits mit kleinen Microcontrollern implementiert werden.

#### 3.2 EXTENDED Befehlssatz

<b>EXCHANGE</b> (RND(min, max), VAL(data8[..]))	Das Feld data8[..] beschreibt, von welchen Lichtern die Werte übernommen werden sollen
<b>CALL</b> (destination)	Unterprogramm Aufruf
<b>RET</b>	Return aus Unterprogramm
<b>JE, JNE, JG, JGE, JL, JLE</b>	Bedingte Sprünge

Zusätzlich sind implementiert:

- 4 Bit Datenregister für Helligkeitsstufen (R0..Rn)
- 16 Bit Kontrollregister für Zeiten / Zähler ... (ER0 .. ERm)
- Arithmetisch / logische Befehle
- Erweiterte Dateneingänge, z.B. Userdaten wie AD-Wandler, FFT Ergebnisse, Digitale Inputs etc.

Außerdem stehen genügend freie Opcodes für benutzerspezifische Erweiterungen zur Verfügung.

## 4 Einfaches Beispiel Programm

Aufgabenstellung: flackerndes Kaminfeuer (für Theateraufführungen).  
Implementierung mit zwei Hochleistungs-LEDs in rot und gelb.

**Label:**

**SET** (TIME(1, 3), VAL(0x0f, 0x0f))

**JUMP** (Label)

In Zeitschritten zwischen jeweils 10 und 30 ms werden die beiden LEDs auf Zufallswerte zwischen 0 und 100 % gesetzt. Dies ergibt ein sehr realistisches zufälliges Flackern.

## 5 Schnittstellen zum Stack

Zum Betrieb des *LightStack* sind zwei Funktionen erforderlich.

Eine Funktion, die die Helligkeit der einzelnen LEDs setzt:

**void LSSetLight(unsigned char ix, unsigned short value);**

ix ist dabei die Nummer der LED, value ein 16 Bit-Wert für die Helligkeit.

Die zweite Funktion liefert den nächsten Opcode:

**unsigned char LSGetInstr(unsigned short pc);**

pc ist dabei der aktuelle Programm-Counter (Index) in ein Daten-Array, das den *LightStack* enthält.

Zur Initialisierung von *LightStack* wird folgende Funktion aufgerufen:

**void\* LSInitLight(unsigned char numOfLights, U8FUNCU16\_t pGetInstr, voidFUNCU8U16\_t pSetLight);**

Sie wird mit folgenden Parametern versorgt.

- Anzahl der LEDs, die in einer Gruppe gesteuert werden sollen
- Zeiger auf die Funktion, die jeweils den nächsten Opcode des *LightStack* Programms liefert (in diesem Beispiel „LSGetInstr()“)
- Zeiger auf eine Funktion, die die Helligkeit der LEDs setzt („LSSetLight()“)

Die Funktion reserviert Speicher für die internen Datenstrukturen und liefert einen Zeiger darauf zurück. Diese Funktion kann auch mehrfach gerufen werden, wenn verschiedene LED-Gruppen unabhängig voneinander gesteuert werden sollen.

Zum Betrieb von LIGHTSTACK wird nun die Funktion

**unsigned char LSDoState(void\* ptr)** zyklisch (typisch mit 100 Hz) aufgerufen. Versorgt wird die Funktion mit dem oben beschriebenen Zeiger.

Danach holt sich *LightStack* über die oben angegebene Callback-Funktion („LSGetInstr()“) jeweils den nächsten Opcode des *LightStack* Programms und setzt bei Bedarf über die zweite Callback-Funktion („LSSetLight()“) die jeweils berechnete Helligkeit der LEDs.

Der Benutzer ist verantwortlich für die Umsetzung dieser Werte in die reale Helligkeit der LEDs, das kann durch Hardware- oder Software-PWM erfolgen. Für die Implementierung der Software-PWM sind C-Code Beispiele vorhanden.

Die Funktion **LSDoState()** liefert normalerweise LsOK zurück, bei Fehlern (z.B. undefiniertem Opcode) LsError bzw. LsDONE, wenn das Programm über den DONE Befehl beendet wurde.

## 6 Befehlsbeschreibung

LightStack Programme können auf verschiedene Arten implementiert werden:

1. Als C-Makros in C-Datenstrukturen. Auf diese Methode wird hier nicht weiter eingegangen, da die Implementierung von Sprungbefehlen schwierig ist.
2. Als GNU Assembler Programme, dafür stehen geeignete Makros zur Verfügung
3. Über einen mitgelieferten Assembler (LSasm). Dieser Assembler erzeugt ein C-Daten-Array, das mit der Applikation zusammen kompiliert und gelinkt wird. Dadurch wird Unabhängigkeit von der Entwicklungsumgebung (z.B. gCC) erreicht.

### Standard Befehlssatz:

Befehl	GnuAsm Syntax	Code	Beschreibung
SET	SET (RND(t1, t2), VAL(vv,... vv))	0x01	Verändern aller Lichter in der Zeit (t1 .. t2) auf die Werte (vv .. vv)
MIX	MIX (RND(t1,t2), VAL(vv, ... vv))	0xfb	Verändern aller Lichter innerhalb (t1 .. t2) auf die Werte (vv .. vv), wobei die Reihenfolge der LEDs beliebig gemischt wird
LOOP	LOOP(label, RND(c1,c2))	0xfd	Schleife (c1 .. c2) mal zum Label
IDLE	DELAY (RND(t1,t2))	0xfe	Wartezeit zwischen (t1 .. t2)
JUMP	JP label	0xfc	Sprung auf Label
DONE		0xff	Programm Ende
MARKE	Label:		Sprungziel x

### Beschreibung der Werte:

v: HEX-Ziffer zwischen 0 und 9 bzw. a und f, 4 Bit Wert für Helligkeit

c: 16 Bit wert (dezimal), Schleifenzähler.

t: 16 Bit wert (dezimal), Zeitangaben.

Helligkeitswerte (vv) bestehen immer aus zwei 4-Bit Werten. Es wird jeweils ein Zufallswert zwischen dem 1. und 2. Wert gebildet, für einen festen Wert werden beide gleichgesetzt. Es ist darauf zu achten, dass der 1. Wert immer kleiner/gleich dem 2. Wert gesetzt wird. Hex-Werte mit 1. Wert größer 2. Wert werden als Spezial-Parameter verwendet.

10: Zufalls-Wert kleiner gleich dem letzten Wert

20: Wert aus Register

21: Wert unverändert

Weitere Werte sind für den Extended Instruction Set reserviert.

Zeitwerte (t, t) bestehen auch immer aus einem Paar von 16 Bit-Werten (0 ... 65535). Es wird immer ein zufälliger Zeitwert zwischen dem 1. und dem 2. Wert ermittelt. Zeitbasis sind per Default 100 Hz, d.h. 100 entspricht einer Sekunde. Für feste Werte wiederum 1. Wert gleich 2. Wert setzen, auch hier gilt: 1. Wert kleiner/gleich 2. Wert. Die anderen Werte sind für Spezial-Parameter reserviert: Entsprechendes gilt für die Schleifenzähler.

Für den LIGHSTACK-Assembler „LSasm“ gilt folgende Syntax:

### Anweisungen:

Anweisung	Beschreibung	Beispiel	Erklärung
<b>LIGHTS</b>	Anzahl der zu steuernden Lichter	LIGHTS 12	Die nächsten Programme sind für 12 LEDs
<b>TIME</b>	Zeitwert für die nächsten Licht-Steuer Befehle setzen	TIME 100,200	Zufallszeit zwischen 1 und 2 Sekunden
<b>DELAY</b>	Zeitwert für die nächsten Warte-Befehle setzen	DELAY 100	Wartezeit genau 1 Sekunde
<b>COUNT</b>	Schleifenzähler für die nächsten LOOP-Befehle setzen	COUNT 1,3	Schleifenzähler zwischen 1 und 3

### Befehle:

Befehl	Beschreibung	Beispiel	Erklärung
<b>VALUE</b>	Werte für die Lichter	TIME 100,200 VALUE 00,0f,ff	Innerhalb 1-2 Sekunden 1. LED aus, 2. zufällig zw. 0 und 15, 3. auf 15
<b>MIX</b>	Werte in beliebiger Reihenfolge für die Lichter	TIME 50 MIX 00,0f,ff	Innerhalb von 0.5 Sekunden eine LED aus, eine zufällig zw. 0 und 15, und eine auf 15
<b>IDLE</b>	Wartet die mit der letzten DELAY Anweisung gegebene Zeit	DELAY 100 IDLE	Wartet 1 Sekunde
<b>LOOP</b>	Schleifen-Befehl	COUNT 5 Label: <xxx> LOOP label	5 x Befehl(e) <xxx> ausführen
<b>JUMP</b>	Sprung-Befehl	JUMP <xxx>	Auf Marke <xxx> springen
<b>MOV</b>	Register Laden	MOV R0,05	Register R0 mit Wert zwischen 0 und 5 laden

## Parameter:

Parameter	Beschreibung	Beispiel	Erklärung
<xx>	Wert für jeweils eine LED	VALUE 1a	Zufallswert zwischen 1 und 10
XX	LED unverändert lassen	VALUE XX	Letzten Wert unverändert lassen
LT	Wert kleiner (less-than) als letzter Wert	VALUE 0f, LT	z. B. rote LED auf Wert zw. 0 und 15, grüne LED auf Wert kleiner als die rote
R	Wert aus Register übernehmen	VALUE R	Für jede LED gibt es ein eigenes Register, das von außen gesetzt werden kann
R<x>	Wert aus Register übernehmen	VALUE R0,R0,R0	Alle LEDs auf den Wert von Register 0 setzen

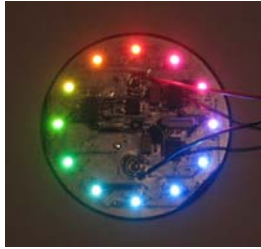
## Beispiel-Programm für weißes Licht mit einer RGB-LED, das sich langsam verändert:

```

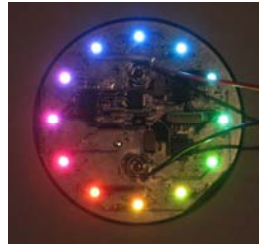
LIGHTS 3                # 3 LEDs (RGB)
PROGRAM Prog           # Programm Name "Prog"
    TIME 500,1000      # 5-10 Sekunden
White:                 # Marke für Sprung-Befehl
    VALUE df,df,df    # Wert für alle LEDs zufällig zwischen 13 und 15
    JUMP White         # und wieder von vorne
ENDP                   # Ende das Programms

```

## 7 Anhang - Beispiel



Hier ein Beispiel für kontinuierliche Farbwechsel mit 12 RGB Leuchtdioden.



## 8 Kontakt



**ARS Software GmbH**  
Starnberger Str. 22  
D-82131 GAUTING/München  
Telefon: 089-893 41 30  
Telefax: 089-893 41 310  
Email: [info@ars2000.com](mailto:info@ars2000.com)  
[www.ars2000.com](http://www.ars2000.com)