

V15.1.15

# LED-BASIC

KOMPONENTEN, EDITOR, BEFEHLSSATZ

© 2017-2018 DIAMEX GMBH

Erwin Reuß und Folker Stange



# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

### Inhaltsverzeichnis

LED-Basic? .....	4
Installation und Programmstart .....	5
Windows 10 und die Sicherheit.....	5
Automatische und manuelle Updates.....	5
LED-Basic Editor.....	6
Editor-Funktionen.....	6
Info-Fenster .....	7
Konfiguration.....	7
System .....	7
Komponenten.....	8
LED-Basic Grundbefehlssatz .....	10
Grundsätzliche Regeln für LED-Basic.....	10
Hinweise zur Benutzung von Variablen.....	10
Eingabe von Zahlwerten:.....	11
Arithmetische Operatoren: .....	11
Bit-Operatoren: .....	11
Vergleichs-Operatoren: .....	11
Logische-Operatoren:.....	11
Syntaxbeschreibung .....	12
REM .....	12
END.....	12
LET .....	12
FOR-NEXT-SCHLEIFE .....	12
IF-THEN-ELSE .....	13
GOTO .....	13
GOSUB/RETURN .....	13
RANDOM .....	13
DELAY.....	13
PRINT .....	13
DATA/READ .....	14
Hardware-Konfiguration.....	15
### KONFIGURATIONSZEILE .....	15
LED-Zusatzbefehle für LED-Basic-Komponenten.....	16
LED-Befehle für serielle RGB-LEDs mit PWM (z.B. WS2812, APA102, SK6812) .....	16
LED.SHOW .....	16
LED.LRGB .....	16
LED.LHSV.....	16
LED.IRGB.....	16
LED.IHSV .....	16
LED.ILED.....	17
LED.IALL .....	17
LED.IRANGE .....	17
LED.RAINBOW .....	17
LED.COPY .....	17

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

LED.REPEAT .....	17
LED.SHIFT.....	18
LED.MIRROR .....	18
LED.BLACKOUT .....	18
LED-Befehle für RGB-LEDs ohne PWM und Einzel-LEDs.....	19
LED.SETLED .....	19
LED.SETALL .....	19
LED-Befehle für mehrere LED-Typen.....	19
LED.BRIGHT.....	19
LED-Befehle für 7-SEGMENT-DISPLAY .....	20
LED.CLEAR.....	20
LED.PCHAR.....	20
LED.PRAW .....	20
LED.ADP .....	20
LED.PHEX .....	21
LED.PDEZ .....	21
LED-Befehle nur für 4-stelliges 7-SEGMENT-DISPLAY .....	21
LED.ACHAR .....	21
LED.ARAW .....	21
LED-Befehle für Matrix-Anzeigen .....	21
IO-Zusatzbefehle für LED-Basic-Komponenten .....	22
IO.WAITKEY .....	22
IO.GETKEY.....	22
IO.KEYSTATE .....	23
IO.SETPORT.....	23
IO.CLRPORT .....	23
IO.GETRTC.....	23
IO.SETRTC .....	23
IO.GETLDR.....	24
IO.GETIR.....	24
IO.GETTEMP .....	24
IO.XTEMPCNT .....	25
IO.XTEMPVAL.....	25
IO.BEEP .....	25
IO.GETENC.....	26
IO.SETENC.....	26
IO.GETPOTI .....	26
IO.GETADC.....	26
IO.EEREAD .....	26
IO.EEWRITE.....	26
IO.SYS.....	27
Laufzeit-Fehlermeldungen .....	29
Led-Basic-Komponenten .....	30
Treiberinstallation .....	30
Unterschiedliche Komponenten.....	30

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

Liste der von LED-Basic unterstützten Komponenten.....	30
LED-Badge (12 LEDs).....	31
LED-Badge (16 LEDs).....	31
Basic-Pentagon-Board .....	32
Basic-Budget-Board .....	33
Cronios 1.....	34
Cronios-Segmenta .....	35
Basic-Booster.....	36
Cortex-Clock .....	37
Temperatur-Sensor Interface.....	38
Lauflicht mit 16 LEDs .....	39
All-In-One Power-M4-Board.....	40
LED-Box.....	42
LED-Matrix 10x10 .....	43
APA-Booster .....	44
Touch-Stick (Nano) .....	45
Touch-Lamp.....	46
VFD-Clock .....	47
6-O-Clock .....	49
LED-Tube-Clock.....	50
LED-Nixie-4 .....	51
Zubehör, Erweiterungen .....	52
Prog-SB, Seriell-Basic Programmier- und Terminaladapter .....	52
DCF77-Zeitmodul.....	53
GPS -> DCF-Zeitmodul .....	54
WLAN -> DCF-Zeitmodul.....	54
Versionen .....	55
V15.1.15 .....	55
V15.1.14 .....	55
V15.1.13 .....	55
V15.1.12 .....	55
V15.1.11 .....	55
V15.1.10 .....	55
V15.1.9.....	55
V15.1.8.....	56
V15.1.7.....	56
V15.1.6.....	56
V15.1.5.....	56
Fehler, Bugs .....	57
Hinweise .....	58
Links.....	58

# LED-BASIC

Komponenten, Editor, Befehlssatz

---

## LED-Basic?

Hinter der Idee von LED-Basic stecken langjährige Erfahrungen im Umgang mit Leuchtdioden und deren Ansteuerung mit dem PC oder als selbstständige Module wie die verschiedensten LED-Player oder LED-Controller. Alle hatten das Problem, dass die Daten für die LEDs von irgendeinem Programm erzeugt werden mussten. Meist wurde hierfür eines der Programme JINX! oder GLEDiator benutzt, die jedoch hauptsächlich für LED-Matrizen ausgelegt sind und teilweise recht umständlich zu konfigurieren und zu bedienen sind. Um die erzeugten Daten auf dem LED-Player benutzen zu können, mussten diese auf eine SD-Karte oder USB-Stick kopiert und in den Player gesteckt werden. Für kleine, einfache Projekte ist dieser Vorgang zu aufwändig und man ist auf die Effekte der benutzten Programme angewiesen. Eigene Kreationen sind nur im begrenzten Maße durch Änderungen der Effektparameter möglich.

LED-Basic ist hauptsächlich für die Ansteuerung von Einzel-LEDs oder LED-Stripes ausgelegt. Die maximale Anzahl der anzusteuern LEDs hängt weitgehend von der Leistung des Microcontrollers und der Größe des Speichers der verwendeten LED-Basic-Komponenten ab.

LED-Basic muss nicht für Anwendungen mit LEDs benutzt werden. Steuerungen, die z.B. Sensoren abfragen und abhängig von den gemessenen Werten Signale auf IO-Ports ausgeben sind ebenso möglich (siehe Temperatur-Sensor-Interface).

### Erweiterungen?

LED-Basic soll einfach zu benutzen sein und die Befehle soll sich jeder Anwender leicht merken können. Deswegen wurde auf eine aufwändige Benutzeroberfläche und Basic-Befehlen mit umständlicher Syntax absichtlich verzichtet. Sinnvolle Erweiterungen, die von den Anwendern vorgeschlagen werden, können sicher in zukünftige Versionen integriert werden. Ihre Mitarbeit ist also erwünscht. Senden Sie uns Ihre Vorschläge und auch eventuelle Fehlermeldungen an [feedback@led-basic.de](mailto:feedback@led-basic.de).

# LED-BASIC

Komponenten, Editor, Befehlssatz

---

## Installation und Programmstart

Laden Sie das Installationspaket „LedBasic\_aa.b.c.exe“ von unserem Server herunter und installieren Sie dieses auf ihrem PC (aa.b.c = aktuelle Version).

Im Unterverzeichnis „SAMPLES“ der Installation befinden sich einige Beispieldateien für verschiedene LED-Basic Komponenten. Sie sollten Ihre eigenen Kreationen jedoch nicht in diesem Verzeichnis speichern, da dessen Inhalt bei der Deinstallation gelöscht oder bei einem Update eventuell überschrieben werden kann.

Im Installationsverzeichnis befinden sich auch die USB-Treiberdatei und diese Bedienungsanleitung im PDF-Format.

## Windows 10 und die Sicherheit

Sollte es unter Windows 10 Fehlermeldungen wie „Zugriff verweigert“ geben, starten Sie das Programm bitte mit Administrator-Rechten (mit der rechten Maustaste auf das LED-Basic-Icon klicken und „Als Administrator ausführen“ auswählen).

## Automatische und manuelle Updates

Der LED-Basic Editor testet automatisch bei Programmstart, ob eine neue Version vorhanden ist (wenn diese Option in der Konfiguration aktiviert ist). Sie können dies auch jederzeit über das Hilfe-Menü -> Teste auf Aktualisierung überprüfen. Sollte eine aktualisierte Version vorhanden sein, wird diese automatisch installiert und gestartet. Voraussetzung hierfür ist natürlich, dass Ihr PC mit dem Internet verbunden ist. Sie können die aktuelle Version natürlich auch von unserem Server herunterladen und manuell installieren.

Zusammen mit der aktuellen Software wird auch diese PDF-Anleitung aktualisiert. Diese wird ebenfalls bei einem Update heruntergeladen und ersetzt die alte Version.

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

### LED-Basic Editor

Der LED-Basic-Editor ist gleichzeitig auch ein sogenannter „Tokenizer“. Er übersetzt den Basic-Code aus dem Klartext in „Tokens“, dies spart Speicherplatz auf der LED-Basic-Hardware und bringt einen enormen Geschwindigkeitsvorteil bei der Ausführung des Programmes. Der „Tokenizer“ prüft die Syntax des eingegebenen Basic-Codes, wandelt Sprungbefehle und Labels in absolute Adressen um und checkt die korrekte Verwendung von Variablen und die Anzahl der Parameter bei LED- und IO-Befehlen.

Es wird vorausgesetzt, dass sich der Anwender mit der Programmiersprache BASIC auskennt und die Unterschiede zwischen Variablen, Konstanten und Ausdrücken versteht. Wenn nicht, es gibt im Internet etliche Literatur über diese recht einfache Programmiersprache. Die Syntax und einige Besonderheiten des LED-Basic befinden sich in der Beschreibung des Grundbefehlssatzes und der Zusatzbefehle. Außerdem wird es mit der Zeit jede Menge Beispielprogramme geben, wo man sich die Programmier-techniken anschauen kann.

Die Version des LED-Basic Editor (z.B. 15.1.9) ergibt sich aus der aktuellen Basic-Version (v15) und der Version des Editor (v1.9). Sollte die Version des Editors nicht mit der Basic-Version der verwendeten LED-Basic-Komponente übereinstimmen, wird automatisch eine Aktualisierung des Bios durchgeführt.

### Editor-Funktionen





Die Bedienung eines Texteditors sollte allgemein bekannt sein. Der LED-Basic-Editor unterstützt die Standard-Funktionen Ausschneiden, Kopieren und Einfügen sowie eine Rückgängig- und Wiederherstellen-Funktion (Undo/Redo) mit bis zu 100 Schritten. Beim Speichern einer Quelldatei wird immer eine Sicherheitskopie der vorhandenen Datei mit der Endung .bak angelegt.

Die Sprache der Benutzeroberfläche des Editors kann über den Menüpunkt „Language“ gewählt werden. Beachten Sie, dass die Sprache der Datei- oder Suchen-Dialoge von Ihrer installierten Windows-Version abhängig ist.

Da LED-Basic zurzeit noch keine Include-Dateien unterstützt, kann der Editor nur eine Datei gleichzeitig öffnen und bearbeiten.

Über das Menü, die Icon-Leiste oder über Tasten-Kurzbefehle können die wichtigsten Funktionen des Editors schnell aufgerufen werden.



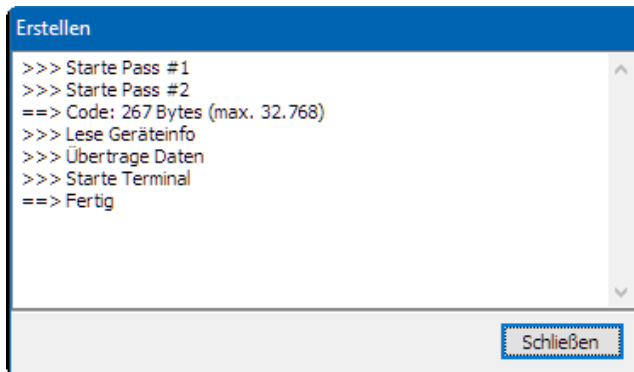
	F9	Übersetzt den Basic-Code und zeigt eventuelle Fehler im Info-Fenster an.
	Umsch + F9 oder F11	Übersetzt den Basic-Code und zeigt eventuelle Fehler Info-Fenster an. Nach erfolgreicher Übersetzung wird der Code zur LED-Basic-Komponente übertragen.
	F12	Löst einen Neustart des laufenden Basic-Programmes auf der LED-Basic-Komponente aus.
	Umsch + F11	Stellen der Echtzeituhr (falls von der LED-Basic-Komponente unterstützt).

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

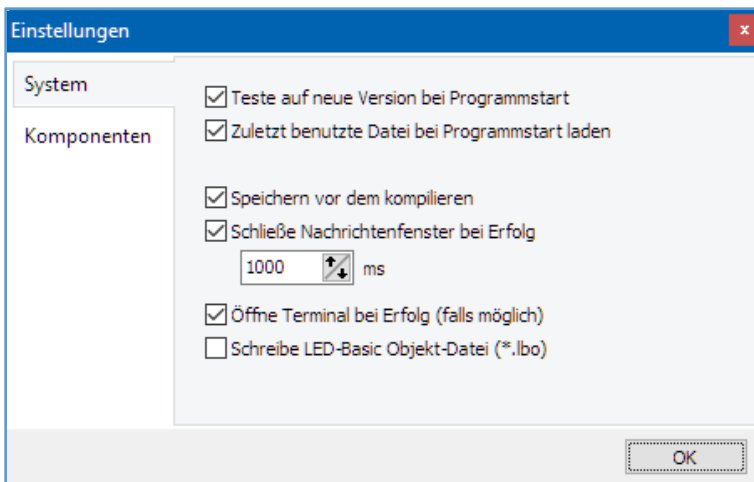
### Info-Fenster



Im Info-Fenster werden eventuelle Fehler während des Übersetzens des Basic-Quellcodes oder der Datenübertragung zur LED-Basic-Komponente angezeigt. Wenn der Vorgang fehlerfrei durchgeführt wurde, schließt sich das Fenster nach der in der Konfiguration eingestellten Zeit von selber oder muss manuell geschlossen werden. Sollte ein Fehler aufgetreten sein, bleibt das Fenster immer geöffnet und muss manuell geschlossen werden.

### Konfiguration

#### System



#### Teste auf neue Version bei Programmstart:

Wenn eine Internet-Verbindung existiert, wird bei Start des LED-Basic Editors automatisch nachgeschaut, ob eine neue Version existiert. Diese kann dann heruntergeladen und installiert werden. Eine manuelle Abfrage kann jederzeit über das Hilfe-Menü vorgenommen werden.

#### Zuletzt benutzte Datei bei Programmstart laden:

Die zuletzt bearbeitete Datei wird bei Programmstart wieder eingeladen. Wenn dieser Menüpunkt nicht angewählt ist, wird der LED-Basic Editor mit einem leeren Dokument geöffnet.

#### Speichern vor dem kompilieren:

Vor Erstellen/Kompilieren des Basic-Codes wird die aktuelle Datei automatisch gespeichert. Zusätzlich wird eine Sicherung der vorherigen Datei angelegt (\*.bak).



# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

### SchlieÙe Nachrichtenfenster bei Erfolg:

Bei einer fehlerfreien Übersetzung des Basic-Codes wird das Nachrichtenfenster nach der eingestellten Zeit automatisch geschlossen. Wenn dieser Punkt nicht gewählt ist, bleibt das Nachrichtenfenster so lange geöffnet, bis es manuell geschlossen wird.

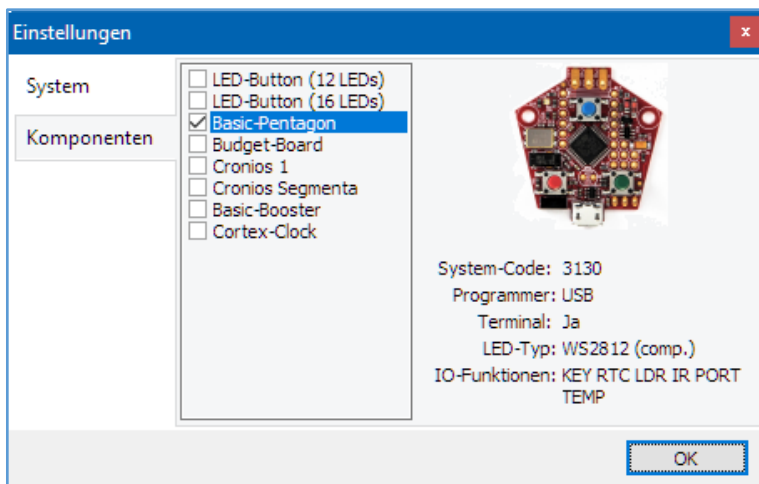
### Öffne Terminal bei Erfolg:

Wenn die Übersetzung des Basic-Codes und die Übertragung in den Microcontroller fehlerfrei ist, kann das Terminal-Fenster zur Print- und Fehlerausgabe geöffnet werden, wenn dies von der LED-Basic-Komponente unterstützt wird.

### Schreibe LED-Basic Objekt-Datei (\*.lbo):

Die vom Tokenizer erzeugte Objekt-Datei wird im selben Verzeichnis wie der Basic-Code abgelegt. Diese kann mit einem externen Programmierwerkzeug zur LedBasic-Komponente übertragen werden (in Planung).

## Komponenten



Wählen Sie die passende LED-Basic-Komponente aus der Liste aus. Sie können sich alle Komponenten zur Auswahl anschauen. Wichtig! Nur wenn die Komponente mit einem Haken markiert ist, wird es auch benutzt.

Unter dem Foto der Komponente finden Sie dessen Spezifikationen:

### System-Code:

Dieser Code wird zur Identifikation der LED-Basic-Komponente benutzt. Wird eine Komponente angeschlossen, die nicht zum eingestellten System-Code passt, kann der erzeugte Basic-Code nicht hochgeladen werden.

### Programmer:

Hier wird angezeigt, ob ein Programmieradapter (z.B. Prog\_SB) benutzt werden muss, um das den Basic-Code in die Komponente hochzuladen.

### Terminal:

Ist eine Print- und Fehlerausgabe über das Terminal möglich?

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

### LED-Typ:

Welche LED-Typen werden von der Komponente unterstützt?

RGB	Einfache RGB-LEDs ohne PWM, 7 verschiedene Farben sind möglich.
WS2812	Serielle LEDs mit integriertem PWM-Controller des Typs WS2811 (RGB oder RGBW), auch SK6812 oder APA104/106. Merkmal: 3 Anschlussleitungen (+5V, DATA, GND)
APA102	Serielle LEDs mit integriertem PWM-Controller des Typs WS2801 (RGB). Merkmal: 4 Anschlussleitungen (+5V, DATA, CLOCK, GND)
7-SEGMENT	7-Segment-Anzeige, z.B. bei Cortex-Clock
SIMPLE-LED	Einzel-Leuchtdioden, z.B. bei Lauflicht mit 16 LEDs

### IO-Funktionen:

Welche IO-Funktionen werden von der Komponente unterstützt?

KEY	Tasten und Eingangsports (KEY_x)
PORT	Ausgangsports (PORT_x)
RTC	Echtzeituhr
LDR	Helligkeitssensor
TEMP	Temperatursensor (1 x DS18B20), nur Abfrage der Temperatur
XTEMP	Temperatursensor (max. 8 x DS18B20), Temperatur- und Parameterabfrage
IR	Sensor für Infrarot-Fernbedienung
BEEP	Tonausgabe über Lautsprecher
ENC	Drehimpulgeber
POTI	Analoge Drehregler (z.B. Potentiometer)
ADC	Analogwerte abfragen (z.B. Batteriespannung)
EED	EEPROM, speichert Daten, die auch nach entfernen der Stromversorgung erhalten bleiben.
SYS	System-Funktionen aufrufen, System-Variablen lesen oder schreiben

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

### LED-Basic Grundbefehlssatz

Der Sprachumfang des LED-Basic beschränkt sich auf wenige Funktionen, die von der BASIC-Programmiersprache her bekannt sind. Spezielle LED- und IO-Befehle erweitern das Basic. Durch das Umwandeln der Befehle in „Tokens“ wird für eine hohe Abarbeitungsgeschwindigkeit von über 10000 Zeilen pro Sekunde (abhängig vom verwendeten Microcontroller) gesorgt.

#### Grundsätzliche Regeln für LED-Basic

- Groß- und Kleinschreibung wird nicht unterschieden
- Es gibt 26 globale Variablen a...z bzw. A...Z (a ist identisch mit A)
- LED-Basic rechnet mit maximal 16 Bit, -32768...+32767
- Es sind keine Fließkommawerte möglich
- Es gibt keine String-Variablen
- Maximal 4 verschachtelte GOSUB sind möglich
- Maximal 4 verschachtelte FOR-NEXT-Schleifen sind möglich
- Alle Befehle nach END werden ignoriert
- Bei einem ' (Hochkomma) am Anfang einer Zeile wird diese komplett ignoriert (wie bei REM)
- Über ein Terminalprogramm (z.B. das aus dem LED-Basic-Editor) werden PRINT-Ausgaben sowie Fehlermeldungen des LED-Basic-Players angezeigt, wenn dies von der LED-Basic-Komponente unterstützt wird

Es gibt in LED-Basic keine Zeilennummern, wie es von anderen Basic-Varianten bekannt ist. Zeilennummern werden dennoch als Sprungmarken für GOTO, GOSUB oder als Index auf DATA-Werte eingesetzt. Eine Sprungmarke besteht aus einer Zahl zwischen 0 und 32766 mit nachfolgendem Doppelpunkt

#### Beispiele

```
1000:
  i = 123
  ...
  Return
30000: a = 0
  ...
  Return
```

Die Reihenfolge der Nummern ist beliebig, muss jedoch im gesamten Quelltext einmalig sein. Der nachfolgende Befehl kann in derselben oder der darauffolgenden Zeile stehen.

#### Hinweise zur Benutzung von Variablen

Die 26 Variablen a..z (A..Z) sind global gültig. Es gibt keine lokalen Variablen, wenn Sie Variablen und Unterrouinen benutzen, sollten Sie darauf achten, dass Sie nicht dieselben Variablen wie in der aufrufenden Routine verwenden. Machen Sie Sich eine Liste oder schreiben die verwendeten Variablen als Kommentar in den Code.

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

### Eingabe von Zahlwerten:

Dezimalzahlen:	0 1234 -2000
Hexadezimalzahlen:	0x1234 0xAB
Binärzahlen:	0b10101010 0b10

### Arithmetische Operatoren:

+	Addition, 3 + 4
-	Subtraktion, 5 - 2
/	Division, 9 / 3
*	Multiplikation, 10 * 5
%	Modulo (Divisionsrest), 6 % 5

### Bit-Operatoren:

	bitweise OR-Verknüpfung, 0x05   0x80
&	bitweise AND-Verknüpfung, 0xAE & 0x07

### Vergleichs-Operatoren:

<	kleiner als
>	größer als
=	gleich
<>	ungleich
<=	kleiner gleich
>=	größer gleich

### Logische Operatoren:

or	logische OR-Verknüpfung, if a=1 or a=3 then ...
and	logische AND-Verknüpfung, if a=2 and b=3 then ...

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

### Syntaxbeschreibung

Für die folgende Syntaxbeschreibung der LED-Basic-Befehl gilt:

- *VAL* = numerischer Wert (123, 0x100, 0b101010)
- *STR* = String, muss in Anführungszeichen eingeschlossen sein ("HALLO WELT")  
(nur im PRINT-Befehl möglich)
- *EXPR* = Ein Ausdruck mit numerischem Ergebnis (1 + 2, a \* 3)
- *VAR* = Variable (a, A, z, Z)
- *REL* = Vergleichsoperation (a > 4, x = y)
- *LABEL* = Sprungziel oder Datenbasis (1234:)
- ... = beliebige Anweisung
- <xxx> = Eingabe erforderlich
- xxx|yyy = xxx oder yyy sind möglich
- <VAL|EXPR|VAR> = hier kann ein numerischer Wert, ein Ausdruck oder eine Variable stehen
- [xxx] = kann optional hinzugefügt werden

### REM

```
rem ...  
` ...
```

Remark, Anmerkung, alle Anweisungen hinter **rem** oder ` (Hochkomma) werden ignoriert.

### END

```
end
```

Alle Befehle hinter **end** werden ignoriert

### LET

```
[let] <var>=<VAL|EXPR|VAR>  
<var>=<VAL|EXPR|VAR>
```

Zuweisung von Werten an eine Variable.

**let** ist optional und kann auch weggelassen werden.

### FOR-NEXT-SCHLEIFE

```
for <VAR>=<VAL|EXPR|VAR> to|downto <VAL|EXPR|VAR> [step  
<VAL|EXPR|VAR>]  
...  
next VAR
```

Bitte darauf achten, dass der rechte Wert nach **to** niemals kleiner wird als der Wert vor dem **to** und dass der rechte Wert nach **downto** niemals größer wird als der Wert vor dem **downto**.

Der Wert für **step** ist immer positiv, auch beim **downto**.

Ohne **step** ist die Schrittweite immer 1

Eine Verschachtelung von maximal 4 FOR-NEXT-Schleifen ist möglich. Dabei immer darauf achten dass sich der zur FOR-Variable passende NEXT-Befehl innerhalb derselben Ebene befindet.

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

### IF-THEN-ELSE

```
if <VAL|EXPR|VAR> <REL> <VAL|EXPR|VAR> [then] ... [else ...]
```

**then** ist nicht unbedingt erforderlich, muss aber vorhanden sein, wenn auch **else** verwendet wird.

**else** kann optional benutzt werden.

Nach **then** oder **else** ist nur ein Befehl oder Ausdruck möglich, sollen mehrere Befehle ausgeführt werden, kann das über eine Unterroutine mit **gosub/return** gelöst werden.

### GOTO

```
goto <VAL>
```

Es ist kein Ausdruck möglich, nur ein numerischer Wert.

### GOSUB/RETURN

```
gosub <VAL>  
...  
<LABEL>  
Return
```

Es ist kein Ausdruck möglich, nur ein numerischer Wert.

Eine Verschachtelungstiefe von maximal 4 **gosub**-Anweisungen sind möglich.

### RANDOM

```
random
```

Erzeugt eine positive Zufallszahl zwischen 0 und 32767

Eine Initialisierung des Random-Generators wie in anderen Basic-Varianten mit „randomize“ ist nicht erforderlich.

### DELAY

```
delay <VAL|EXPR|VAR>
```

Der Programmablauf wird um xx Millisekunden angehalten.

Der Wert für **delay** darf nicht 0 enthalten, Maximalwert ist 32767 (entspricht ca. 32,8 Sekunden)

### PRINT

```
print <VAL|EXPR|VAR|STR> [;] [, ] [<VAL|EXPR|VAR|STR>]
```

Dient zur Status- oder Debug-Ausgabe über den Terminalausgang (nicht bei jeder LED-Basic-Komponente verfügbar).

Mehrere Werte oder Texte können hintereinander ausgegeben werden. Bei einem Komma wird ein Leerzeichen eingefügt, bei einem Semikolon wird kein Leerzeichen eingefügt. Die maximale Länge des mit **print** ausgegebenen Textes ist 256 Zeichen. Längere Texte werden abgeschnitten.

Mit diesem Befehl sollten nur wenige Daten ausgegeben werden, da dieser die Ausführungsgeschwindigkeit des LED-Basic reduziert. Über den Konfigurationsparameter PO kann die PRINT-Funktion global deaktiviert werden. Werden die Daten zu schnell hintereinander gesendet, kann es zur Blockade des Terminals aufgrund eines Speicherüberlaufes oder zu einer fehlerhaften Anzeige kommen. Sollte sich das LED-Basic-Programm hierdurch nicht mehr bedienen lassen, ziehen Sie kurz den USB-Stecker von der Komponente oder Prog\_SB ab.

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

### DATA/READ

```
<LABEL>
data <VAL> [, <VAL>] [, <VAL>]
...
read <VAL>, <VAL | EXPR | VAR>
```

Achtung! Die hier verwendete Syntax ist nicht kompatibel mit anderen Basic-Varianten und speziell für LED-Basic optimiert.

Hinter einem Label können maximal 127 Tabellenwerte definiert werden. Diese dürfen nur numerische Werte sein und können nachträglich nicht verändert werden. Es können auch mehrere data-Zeilen hintereinander folgen, insgesamt jedoch nur 126 Werte. Alle Werte dürfen maximal 16-Bit groß sein (-32768...32767).

Beispiel:

```
100:
data 10, 20, 30, 40
data 100, 200, 400, 800

200:
data 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80
```

Data-Zeilen müssen immer am Anfang des Programmes, spätestens jedoch vor ihrer ersten Benutzung definiert werden.

Mit dem **read**-Befehl wird gezielt auf die einzelnen **data**-Werte zugegriffen.

Beispiel:

```
[1] a = read 100,0
[2] x = read 200,i
```

[1] Liest den ersten Data-Wert hinter Label 100 (Zählung beginnt immer mit 0)

[2] Liest den Data-Wert hinter Label 200 mit dem Index in der Variablen i

Wenn versucht wird, hinter den letzten Data-Wert zuzugreifen, wird kein Fehler ausgegeben und als Wert 0 gelesen.

Hinter der Data-Zeile darf kein Kommentar stehen. Schreiben Sie den Kommentar einfach vor die Zeile.

# LED-BASIC

Komponenten, Editor, Befehlssatz

## Hardware-Konfiguration

Über die Konfigurationszeile können verschiedene Einstellungen global geändert werden. Beachten Sie bitte, dass nicht jede LED-Basic-Komponente alle Parameter unterstützt. Nicht unterstützte Parameter werden ignoriert.

### ### KONFIGURATIONSZEILE

```
### L64 CRGB M100 P1 S3 T0 A3 F25
```

Diese Zeile muss zwingend direkt am Anfang des BASIC-Codes eingefügt werden, wenn die Standardwerte nicht benutzt werden sollen. Beginnend mit **###** sind mehrere Konfigurationsparameter möglich:

Parameter	Beschreibung	Standardwert
Lxxx	Anzahl der angeschlossenen LEDs (MAX_LED) Gültige Werte: 8..x (x = abhängig von der verwendeten Komponente)	L256, L128, L64
Cxxyy	Farbanordnung der angeschlossenen LEDs GRB = WS2812, RGB = SK6812, APA102, APA106 Für RGBW-Leds, muss hier CRGBW eingetragen werden	CGRB
Mxxx	Master-Helligkeit in % Gültige Werte: 5..100	M100
Px	Printausgabe global ein/ausschalten Gültige Werte: 0,1 (0 = aus, 1 = ein) Hinweis: Laufzeit-Fehlermeldungen werden immer ausgegeben und können nicht ausgeschaltet werden.	P1
Sx	System-Leds ein/ausschalten Gültige Werte: 0..3 (0 = aus, 1 = Ausgabe-LED ein, 2 = Warte-LED ein, 3 = Alle LEDs ein)	S3
Tx	LED-Typ auswählen (bei LED-Basic-Komponenten, die mehrere LED-Typen unterstützen) Gültige Werte: Siehe Hinweis bei den Komponenten	T0
Ax	Bitrate für die LED-Typen APA102 auswählen (SPI-Takt) Je höher der Wert, desto niedriger die Bitrate. Gültige Werte: 0..7 (Die Bitraten sind bei den Komponenten aufgelistet)	A3
Fxx	Framerate bei seriellen LEDs, die den LED.show() – Befehl nutzen. Bei zu niedrigen Werte erfolgt die Ausgabe ruckelnd und flackernd. Bei zu hohen Werten kann es zu Störungen der LED-Anzeige kommen und es kann sich die Ausführungsgeschwindigkeit des LED-Basic reduzieren. Gültige Werte: 1..100	F25



# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

### LED-Zusatzbefehle für LED-Basic-Komponenten

Anzahl und Parameter der Zusatzbefehle sind abhängig von der verwendeten LED-Basic-Komponente und können sich in jeder Version ändern. Bitte deshalb unbedingt die Hinweise bei den Updates beachten.

Die Anzahl der Parameter muss unbedingt stimmen. Ist kein Parameter erforderlich, müssen jedoch wie in der C-Syntax die Klammern dem Befehl folgen. Die Wertebereiche der Parameter werden nicht überprüft. Werden falsche Werte übermittelt, kann es zu fehlerhaften LED-Anzeigen kommen.

*Als Parameter können neben numerischen Werten natürlich auch Variablen oder berechnete Ausdrücke eingesetzt werden.*

Die Funktion einiger Befehle kann am besten durch ausprobieren herausgefunden werden. Beispiele befinden sich auch im Installationspaket und auf der Support-Homepage (Link am Ende).

In der Beschreibung der LED-Basic-Komponenten finden Sie die Liste der unterstützten LED-Befehle.

### LED-Befehle für serielle RGB-LEDs mit PWM (z.B. WS2812, APA102, SK6812)

#### LED.SHOW

```
LED.show()
```

Die Ausgabe zu den LEDs wird gestartet. Aufgrund der seriellen Übertragung zu den LEDs ist dies nur alle 40ms möglich (25 Frames / Sekunde) und wird automatisch auf diese Geschwindigkeit begrenzt. Ohne diesen Befehl ist keine Anzeige auf den LEDs möglich.

#### LED.LRGB

```
LED.lrgb(led, r, g, b)
```

LED an Position **led** wird mit den Werten in **r**, **g** und **b** gesetzt.

Werte: led=[0..MAX\_LED-1], r/g/b=[0..255]

#### LED.LHSV

```
LED.lhsv(led, h, s, v)
```

LED an Position **led** wird mit den Werten in **h** (HUE, Farbwert), **s** (SAT, Sättigung) und **v** (VOL, Helligkeit) gesetzt.

Werte: led=[0..MAX\_LED-1], h=[0..359], s=[0..255], v=[0..255]

#### LED.IRGB

```
LED.irgb(idx, r, g, b)
```

Bis zu 10 LED-Farbwerte können in Index-Registern gespeichert werden. Farbindex **idx** wird auf die Werte in **r**, **g** und **b** gesetzt.

Werte: idx=[0..9], r/g/b=[0..255]

#### LED.IHSV

```
LED.ihsv(idx, h, s, v)
```

Bis zu 10 LED-Farbwerte können in Index-Registern gespeichert werden. Farbindex **idx** wird auf die Werte in **h**, **s** und **v** gesetzt.

Werte: idx=[0..9], h=[0..359], s=[0..255], v=[0..255]

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

### LED.ILED

```
LED.iled(idx, led)
```

LED an Position **led** wird mit dem Wert im Farbindex **idx** gesetzt.

Werte: led=[0..MAX\_LED-1], idx=[0..9]

### LED.IALL

```
LED.iall(idx)
```

Alle LEDs werden mit dem Wert im Farbindex **idx** gesetzt.

Werte: idx=[0..9]

### LED.IRANGE

```
LED.irange(idx, beg, end)
```

Die LEDs im Bereich von **beg** bis **end** werden mit dem Wert im Farbindex **idx** gesetzt.

Werte: idx=[0..9], beg=[0..MAX\_LEDS-1], end=[beg..MAX\_LEDS-1]

### LED.RAINBOW

```
LED.rainbow(h, s, v, beg, end, inc)
```

Ein Regenbogen-Effekt wird über einen definierten LED-Bereich (**beg..end**) erzeugt. Der Start-Farbwert wird über **h**, **s**, und **v** festgelegt, die Stärke des Farbverlaufes über den Wert in **inc**. Zu einem bewegten Farbverlauf kommt es, wenn der Startwert **h** ständig verändert wird (siehe Beispiel auf der Support-Homepage).

Werte: h=[0..359], s=[0..255], v=[0..255], beg=[0..MAX\_LEDS-1], end=[beg..MAX\_LEDS-1], inc=[0..100]

### LED.COPY

```
LED.copy(from, to)
```

Eine einzelne LED wird von Position **from** an Position **to** kopiert. Die LED an Position **from** bleibt dabei unverändert.

Werte: from=[0..MAX\_LEDS-1], to=[0..MAX\_LEDS-1]

### LED.REPEAT

```
LED.repeat(beg, end, count)
```

Der LED-Bereich **beg** bis **end** wird um die Anzahl in **count** wiederholt.

Werte: beg=[0..MAX\_LEDS-1], end=[beg..MAX\_LEDS-1], count=[1..x]

Beispiel:

```
LED.repeat(0, 5, 3)
```

Led-Anordnung vor Aufruf:

```
0 1 2 3 4 5
```

Led-Anordnung nach Aufruf (3-fache Wiederholung):

```
0 1 2 3 4 5 0 1 2 3 4 5 0 1 2 3 4 5 0 1 2 3 4 5
```

Es erfolgt automatisch eine obere Begrenzung bei MAX\_LEDS.

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

### LED.SHIFT

```
LED.shift(beg, end, to)
```

Der LED-Bereich **beg** bis **end** wird nach **to** verschoben.

Werte: beg=[0..MAX\_LEDS-1], end=[beg..MAX\_LEDS-1], to=[0..MAX\_LEDS-1]

Beispiele:

```
LED.shift(0,4,2)
Led-Anordnung vor Aufruf:
0 1 2 3 4 x x
Led-Anordnung nach Aufruf:
0 1 0 1 2 3 4
```

```
LED.shift(6,9,3)
Led-Anordnung vor Aufruf:
x x x x x x 6 7 8 9
Led-Anordnung nach Aufruf:
x x x 6 7 8 9 7 8 9
```

Die LEDs im frei gewordenen Bereich behalten ihren Ursprungswert und müssen bei Bedarf auf neue Werte gesetzt werden.

### LED.MIRROR

```
LED.mirror(beg, end, to)
```

Der LED-Bereich **beg** bis **end** nach **to** gespiegelt. Um unerwünschte Effekte zu vermeiden, sollte darauf geachtet werden, dass sich die Bereiche **beg...end** und **to** nicht überschneiden.

Werte: beg=[0..MAX\_LEDS-1], end=[beg..MAX\_LEDS-1], to=[0..MAX\_LEDS-1]

Beispiel:

```
LED.mirror(0,5,6)
Led-Anordnung vor Aufruf:
0 1 2 3 4 5
Led-Anordnung nach Aufruf:
0 1 2 3 4 5 5 4 3 2 1 0
```

### LED.BLACKOUT

```
LED.blackout()
```

Alle LEDs werden ausgeschaltet. Ein LED.show() ist nicht zusätzlich erforderlich.

# LED-BASIC

Komponenten, Editor, Befehlssatz

---

## LED-Befehle für RGB-LEDs ohne PWM und Einzel-LEDs

### LED.SETLED

```
LED.setled(led, color)
```

LED **led** wird auf **color** gesetzt









Werte: led=0...max (abhängig von der Komponente), color=0...7

### LED.SETALL

```
LED.setall(color)
```

Alle LEDs werden auf **color** gesetzt

Werte: color=0...7

COLOR	FARBE	
0	Aus	
1	Rot	
2	Grün	
3	Gelb	
4	Blau	
5	Magenta	
6	Cyan	
7	Weiß	

## LED-Befehle für mehrere LED-Typen

### LED.BRIGHT

```
LED.bright(value)
```

Helligkeit der LEDs oder des Displays einstellen. Hinweis: Helligkeitsänderungen im oberen Bereich sind vom menschlichen Auge kaum wahrnehmbar, verändern jedoch die Stromaufnahme von LEDs enorm.

Werte: value = [0..x], x ist abhängig von der verwendeten Komponente und ist in der Liste der LED-Befehle angegeben.

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

### LED-Befehle für 7-SEGMENT-DISPLAY

#### LED.CLEAR

```
LED.clear()
```

Die Anzeige wird gelöscht, alle Segmente aus. Um ein Flackern der Anzeige zu vermeiden, sollte dieser Befehl nicht in einer Ausgabeschleife stehen.

#### LED.PCHAR

```
LED.pchar(pos, char)
```

Gibt das vordefinierte Zeichen **char** an Position **pos** (+1) aus. Pos gibt die 7-Segment-Stelle an, an der das Zeichen ausgegeben werden soll, von links nach rechts. Die Anzeige der Dezimalpunkte bleibt unbeeinflusst.

Werte: pos=0...x, char=0...29 (x = Anzahl der Stellen – 1)

char	Zeichen	char	Zeichen	char	Zeichen
0	0	10	A	20	-
1	1	11	b	21	SPACE
2	2	12	C	22	i
3	3	13	d	23	n
4	4	14	E	24	r
5	5	15	F	25	N
6	6	16	H	26	t
7	7	17	L	27	o
8	8	18	P	28	G
9	9	19	U	29	Y

#### LED.PRAW

```
LED.praw(pos, raw)
```

Gibt das Zeichen in **raw** an Position **pos** (+1) aus. Pos gibt die 7-Segment-Stelle an, an der das Zeichen ausgegeben werden soll, von links nach rechts. Die Anzeige der Dezimalpunkte bleibt unbeeinflusst.

Werte (x = abhängig von der Komponente):

pos=0...x, raw=0...127 (0x7F)



Der Wert in raw gibt die leuchtenden Segmente bitkodiert an.

Bit 0 = A (0x01), Bit 1 = B (0x02), Bit 2 = C (0x04), Bit 3 = D (0x08),

Bit 4 = E (0x10), Bit 5 = F (0x20), Bit 6 = G (0x40)

Der Dezimalpunkt kann nur über den Befehl LED.ADP angesteuert werden.

#### LED.ADP

```
LED.adp(dp)
```

Setzt die Dezimalpunkte bitkodiert. Bit 0 = Position 0, Bit 1 = Position 1, usw. Die Anzeige aller anderen Segmente bleibt unberührt.

Werte: dp=0...255 (0xFF) (Abhängig von der Anzahl der Anzeigen)

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

### LED.PHEX

```
LED.phex(pos, value, width)
```

Gibt den Wert in **value** als Hexzahl auf dem Display an Position **pos** (+1) aus. Der Wert in **width** (+1) gibt die Breite der Ausgabe an. Die Anzeige der Hexzahl geschieht immer mit Vornullen.

Beispiel: Eine 2-stellige Hexzahl (0x00..0xFF) soll auf den beiden rechten Anzeigen ausgegeben werden: LED.phex(2, value, 1).

Werte (x = abhängig von der Komponente):

pos=0...x, value=0...65536 (0xFFFF), width=0..4

### LED.PDEZ

```
LED.pdez(pos, value, width, lzero)
```

Gibt den Wert in **value** als Dezimalzahl auf dem Display an Position **pos** (+1) aus. Der Wert in **width** (+1) gibt die Breite der Ausgabe an. Der Wert in **lzero** gibt an, ob die Vornullen unterdrückt werden sollen (0 = JA, 1 = NEIN). Wird ein Wert größer 9999 übergeben, erfolgt keine Anzeige.

Beispiel: Eine 3-stellige Dezimalzahl (0..999) soll rechtsbündig mit unterdrückten Vornullen ausgegeben werden: LED.pdez(1, value, 2, 0).

Werte (x = abhängig von der Komponente):

pos=0...x, value=0...9999, width=0..3, lzero=0..1

pos=0...x, value=0...65535, width=0..4, lzero=0..1

## LED-Befehle nur für 4-stelliges 7-SEGMENT-DISPLAY

### LED.ACHAR

```
LED.achar(ch1, ch2, ch3, ch4)
```

Gibt alle 4 Zeichen mit einem einzigen Befehl gleichzeitig aus. Es müssen die Werte aller 4 Zeichen übergeben werden. Die Werte sind identisch mit denen in der Tabelle bei LED.PCHAR. Die Anzeige der Dezimalpunkte bleibt unbeeinflusst.

Werte: ch1, ch2, ch3, ch4=0...29

### LED.ARAW

```
LED.araw(raw1, raw2, raw3, raw4)
```

Gibt alle 4 Zeichen mit einem einzigen Befehl gleichzeitig aus. Es müssen die Werte aller 4 Zeichen übergeben werden. Die Werte sind identisch mit denen bei LED.PRAW. Die Anzeige der Dezimalpunkte bleibt unbeeinflusst.

Werte: raw1, raw2, raw3, raw4=0...127 (0x7F)

## LED-Befehle für Matrix-Anzeigen

Dieser Bereich wird derzeit für zukünftige Komponenten überarbeitet.

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

### IO-Zusatzbefehle für LED-Basic-Komponenten

Anzahl und Parameter der Zusatzbefehle können sich in jeder Version ändern. Bitte deshalb unbedingt die Hinweise bei den Updates beachten. Beachten Sie bitte, dass die benutzte LED-Basic-Komponente eventuell nicht alle Funktionen ermöglicht.

Die Anzahl der Parameter muss unbedingt stimmen. Ist kein Parameter erforderlich, müssen jedoch wie in der C-Syntax die Klammern dem Befehl folgen.

*Als Parameter können neben numerischen Werten natürlich auch Variablen oder berechnete Ausdrücke eingesetzt werden.*

In der Beschreibung der LED-Basic-Komponenten finden Sie die Liste der unterstützten IO-Befehle.

#### IO.WAITKEY

```
IO.waitkey()
```

Es wird auf einen beliebigen Tastendruck gewartet. Die Warte-LED blinkt, wenn dies nicht über die Konfigurationszeile mit Sx abgeschaltet ist.

#### IO.GETKEY

```
<VAR>=IO.getkey()
```

Während des Programmablaufes kann der Status der Tasten abgefragt werden. Die Anzahl der Tasten ist von der verwendeten LED-Basic-Komponente abhängig.

Der Tastendruck bleibt so lange gespeichert, bis eine Abfrage erfolgte.

Taste	Wert
Keine	0
1	1 (Bit 0)
2	2 (Bit 1)
3	4 (Bit 2)
4	8 (Bit 3)
...	...

#### FALSCH:

```
if IO.getkey() = 1 then goto 1000
if IO.getkey() = 2 then goto 2000
if IO.getkey() = 4 then goto 3000
```

Durch die erste **getkey**-Abfrage wird der Tastaturstatus zurückgesetzt, die zweite Abfrage wird also niemals ausgeführt, auch wenn Taste 2 zuvor gedrückt war.

#### RICHTIG:

```
k = IO.getkey()
if k = 1 then goto 1000
if k = 2 then goto 2000
if k = 4 then goto 3000
```

Hier wird der Tastaturstatus in eine Variable eingelesen und kann dadurch auf mehrere verschiedene Werte abgefragt werden.

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

### IO.KEYSTATE

```
<VAR>=IO.keystate()
```

Im Gegensatz zu **getkey** kann mit dieser Funktion abgefragt werden, ob eine oder mehrere Tasten gedrückt sind. Hierbei werden die Tasten bitweise (siehe **IO.getkey**) verknüpft. Der Rückgabewert bleibt so lange auf dem Tastaturwert, wie die Taste gedrückt ist. Erst wenn die Taste losgelassen wird, ist der Rückgabewert Null. Es gelten dieselben Wertebereiche wie bei **getkey**. Diese Funktion ist besonders für Eingangsports wichtig, wenn anliegende Signale (High, Low) verarbeitet werden sollen. Alle Eingangsports besitzen integrierte Pull-Up Widerstände und sind per Software entprellt.

### IO.SETPORT

```
IO.setport(port)
```

Setzt die Ausgangsports auf High-Pegel (ca. 3.3V). Welche der Ports beeinflusst werden, bestimmt der Wert in **port**. Die Anzahl der Ports ist abhängig von der LED-Basic-Komponente.

Port 1 = Bit 0, Port 2 = Bit 1, Port 3 = Bit 2, usw.

### IO.CLRPORT

```
IO.clrport(port)
```

Setzt die Ausgangsports auf Low-Pegel (0 V). Welche der Ports beeinflusst werden, bestimmt der Wert in **port**.

Port 1 = Bit 0, Port 2 = Bit 1, Port 3 = Bit 2, usw.

### IO.GETRTC

```
<VAR>=IO.getrtc(idx)
```

Liest die Werte der Echtzeituhr (RTC) aus (falls vorhanden). Die Index-Werte ab 6 sind eventuell bei der verwendeten LED-Basic-Komponente nicht vorhanden.

Werte: idx = [0..8]

idx	Bedeutung	Bereich
0	Sekunde	0 - 59
1	Minute	0 - 59
2	Stunde	0 - 23
3	Tag	1 - 28/29/30/31
4	Monat	1 - 12
5	Jahr	2000 - 20xx
6	Tag im Jahr	1 - 365 (366)
7	Wochentag	0 - 6 (0 = Montag)
8	Schaltjahr	0 = Nein, 1 = Ja

### IO.SETRTC

```
IO.setrtc(idx, val)
```

Setzt die Werte der Echtzeituhr (falls vorhanden). Hier ist es sinnvoll zunächst den gewünschten Wert zu lesen, diesen zu verändern und dann wieder zu schreiben. Wenn mehrere Werte geschrieben werden sollen, mit den niedrigsten **idx**-Werten beginnen.

Werte: idx = [0..5] (wie bei **GETRTC**, 6..8 werden berechnet und können nicht verändert werden)



# LED-BASIC

## Komponenten, Editor, Befehlssatz

### IO.GETLDR

```
<VAR>=IO.getldr()
```

Liest den Wert des Helligkeitssensors (LDR) aus (falls vorhanden). Die Ergebniswerte liegen je nach Helligkeit zwischen 0 und 255, wobei technisch bedingt die oberen und unteren Werte nie erreicht werden.

### IO.GETIR

```
<VAR>=IO.getir()
```

Liest die Wert des Infrarot-Empfängers aus (falls vorhanden). Ein Wert 0 bedeutet, dass keine Daten vorhanden sind. Der gelesene Wert sollte in eine Variable geschrieben werden (wie bei der GETKEY-Funktion). Es wird ein 16-Bit-Wert ausgelesen, der eigentliche Tastenwert steht dabei in den unteren 8 Bits des Wertes, in den oberen 8 Bit ist die Anzahl der Wiederholungen zu finden, wenn eine Taste länger gedrückt wird.

Beispiel:

```
z = IO.getir()
k = z & 0xFF      ` Tastenwert ausmaskieren
n = z / 256      ` In n steht der Wiederholungswert
```

Die Standard-Infrarot-Fernbedienung für alle Komponenten, die einen Infrarot-Empfänger besitzen. Die dezimalen Tastenwerte sind auf der Abbildung rechts zu finden. Die Werte sind vom Hersteller der Fernbedienung vorgegeben und können nicht verändert werden. Alle LED-Basic-Komponenten mit Infrarot-Sensor können nur Signale dieser Fernbedienung empfangen und verarbeiten.

Beachten Sie bitte, dass Farbabweichungen der LEDs gegenüber den Farbtasten möglich sind.



### IO.GETTEMP

```
<VAR>=IO.gettemp()
```

Liest den Wert des Temperatursensors DS18B20 aus (falls vorhanden). Ergebnis ist ein Wert in 0,1°C Auflösung. Aufgrund der relativ langen Messdauer sollte die Abfrage nicht in kürzeren Abständen als 1 Sekunde erfolgen. Erst ein Lesen des Wertes startet einen neuen Messvorgang. Der erste gelesene Wert sollte deswegen verworfen werden.

Beispiele: 0 = 0°C, 217 = 21.7°C, -81 = -8.1°C

Hinweis: Eine höhere Auflösung als 0,1°C ist aufgrund des verwendeten Temperatursensors nicht möglich. Die Genauigkeit wird vom Hersteller mit  $\pm 0,5^\circ\text{C}$  im Bereich  $-10^\circ\text{C}$  -  $+85^\circ\text{C}$  angegeben.

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

### IO.XTEMPCNT

```
<VAR>=IO.xtempcnt()
```

Bei Komponenten, die mehrere Temperatursensoren des Typs DS18B20 unterstützen (z.B. Temperatur-Sensor-Interface) wird mit diesem Befehl die Anzahl der erkannten Sensoren ausgegeben. Maximal 8 Sensoren können angeschlossen werden. Wenn die Sensoren mit parasitärer Stromversorgung betrieben werden, sind aufgrund der höheren Belastung der Datenleitung eventuell nur weniger Sensoren möglich (ausprobieren). Beachten Sie bitte, dass die Sensoren nur bei einem Neustart eingelesen und identifiziert werden. Um Beschädigungen an der Hardware und an den Sensoren zu vermeiden sollten Sie die Sensoren nur im stromlosen Zustand anschließen oder entfernen.

### IO.XTEMPVAL

```
<VAR>=IO.xtempval(nr, idx)
```

Bei Komponenten, die mehrere Temperatursensoren des Typs DS18B20 unterstützen, können mit diesem Befehl für jeden Sensor (**nr**) verschiedene Werte (**idx**) ausgelesen werden.

Werte: nr = [0..x] (x = Wert aus IO.xtempcnt – 1), idx = 0..10

idx	Bedeutung
0	0 = Temperaturwert nicht gültig, <> 0 = Temperaturwert gültig
1	Temperaturwert in 0.1°C Auflösung (siehe IO.gettemp)
2	0 = parasitäre Stromversorgung, 1 = externe Stromversorgung
3..10	ROM-ID des Sensors

### IO.BEEP

```
IO.beep(val)
```

Erzeugt einen Ton über den Lautsprecher (falls vorhanden).

val	Bedeutung
0	Ton aus
1..36	Note
ab 200	Frequenz = val Hz

Frequenzen unter 200 Hz können systembedingt nicht erzeugt werden.

Folgende Notenwerte sind fest gespeichert:

val	Note	val	Note	val	Note
1	C2	13	C3	25	C4
2	C2#	14	C3#	26	C4#
3	D2	15	D3	27	D4
4	D2#	16	D3#	28	D4#
5	E2	17	E3	29	E4
6	F2	18	F3	30	F4
7	F2#	19	F3#	31	F4#
8	G2	20	G3	32	G4
9	G2#	21	G3#	33	G4#
10	A2	22	A3	34	A4
11	A2#	23	A3#	35	A4#
12	H2	24	H3	36	H4

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

### IO.GETENC

```
<VAR>=IO.getenc()
```

Liest den Wert des Drehimpulsgebers aus (falls vorhanden). Der Wertebereich wird über den folgenden Befehl **IO.SETENC** festgelegt.

### IO.SETENC

```
IO.setenc(pos, max, stop)
```

Setzt die Werte-Parameter für den Drehimpulsgeber (falls vorhanden).

Werte: pos=[0..max], max=[1..65535], stop=[0/1]

Der Drehimpulsgeber liefert Werte zwischen 0 und **max**. Der aktuelle Startwert wird mit **pos** festgelegt, dieser muss immer  $\leq$  **max** sein. Bei **stop** = 1 bleibt der Maximalwert bestehen, wenn der Impulsgeber weiter gedreht wird. Wenn **stop** = 0 beginnt der Zähler wieder bei 0 wenn der Maximalwert überschritten ist und wieder bei **max** wenn 0 unterschritten werden soll.

### IO.GETPOTI

```
<var>=IO.getpoti(idx)
```

Liest den Wert eines Potentiometers oder dessen umgewandelten Wert aus (falls vorhanden). Welche und wie viele Werte ausgelesen werden können, hängt von der Komponente ab.

Werte: idx=[0..x], x ist abhängig von der verwendeten Komponente

### IO.GETADC

```
<var>=IO.getadc(idx)
```

Liest einen Analogwert. Welche Werte ausgelesen werden können, hängt von der Komponente ab.

Werte: idx=[0..x], x ist abhängig von der verwendeten Komponente

### IO.EEREAD

```
<VAR>=IO.eeread(adr)
```

Liest den Inhalt des EEPROM an Adresse **adr** aus (falls vorhanden). Rückgabe ist ein 16-Bit-Wert. Im Auslieferungszustand sind alle Speicherstellen des EEPROM auf 0xFFFF (-1). Ein ungültiger Wert von **adr** liefert immer den Wert 0.

Werte: adr=[0..x], x ist abhängig von der verwendeten Komponente und ist in der Liste der IO-Befehle angegeben.

### IO.EEWRITE

```
IO.eewrite(adr, data)
```

Schreibt das EEPROM an Adresse **adr** mit dem 16-Bit Wert in **data** (falls vorhanden). Beachten Sie bitte, dass der Schreibvorgang einer Speicherstelle eventuell mehrere Millisekunden benötigt und damit den Programmablauf verzögert. Um die Lebensdauer des EEPROM zu verlängern, sollten Schreibvorgänge vermieden werden und nur geänderte Werte gespeichert werden.

Werte: adr=[0..x], x ist abhängig von der verwendeten Komponente und ist in der Liste der IO-Befehle angegeben. data=[-32768..32767]

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

### IO.SYS

```
<VAR>=IO.sys(par1, par2)
```

Universeller Befehl zum Setzen oder Lesen von Systemparametern (falls vorhanden). Es müssen immer beide Parameter übergeben werden, auch wenn sie nicht benötigt werden.

Die gängigen Werte für **par1** sind hier aufgeführt. Sollte eine Komponente noch über andere Werte verfügen, sind diese dort extra beschrieben.

#### [COM] Communication

Über diese Befehle können einzelne oder mehrere Zeichen an die Komponente gesendet werden. Diese kann per LED-Basic Befehle die Daten auswerten und eventuell Ergebnisse über den Print-Befehl zurückgeben.

**<var> = IO.sys(0, 0)**

Liest die Anzahl der über den virtuellen COM-Port empfangenen Zeichen aus.

0 = keine Zeichen empfangen

1..64 = Zeichen befinden sich im Empfangsspeicher.

**IO.sys(0, 1)**

Setzt die Anzahl der empfangenen Zeichen auf null.

**<var> = IO.sys(n, 0)**

Liest den Empfangsspeicher an Position n aus.

Gültige Werte für n: 1..x (x = Anzahl der empfangenen Zeichen, max. 64).

Über das Terminal (ab Version 15.1.14) können die ASCII-Werte der gedrückten Taste übermittelt werden. Es wird dabei jeweils 1 Zeichen in den Empfangsspeicher geschrieben.

Beispiel: Taste A wird gedrückt. IO.sys(0,0) liefert den Wert 1 = 1 Wert im Speicher. IO.sys(1, 0) liefert den Wert 65 (0x41) = ASCII-Code für „A“. Danach sollte der Empfangszähler mit IO.sys(0, 1) auf null gesetzt werden.

#### [CALIB] Calibration

**IO.sys(99, c)**

Kalibrierung der Echtzeituhr. Sollte die Integrierte Echtzeituhr vor- oder nachgehen, kann dies hier ausgeglichen werden.

Gültige Werte für c: -510 bis 510

Ein negativer Wert lässt die Uhr langsamer, ein positiver Wert schneller laufen.

Ändern Sie die Werte in 10er, 20er oder 50er Schritten und synchronisieren dann die Uhr mit dem PC. Da die Änderungen nur eine geringe Auswirkung auf die Geschwindigkeit der Uhr haben, kann es mehrere Stunden oder sogar Tage dauern, bis man eine Änderung der Geschwindigkeit erkennt.

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

### [DFP] DFPlayer Mini

Der DFPlayer Mini spielt MP3-Dateien von einer Micro-SD-Karte ab.

<b>&lt;val&gt;=IO.sys(1000, 0)</b>	Status lesen, <b>val</b> : 0 = busy
<b>IO.sys(1000, 1)</b>	Status zurücksetzen
<b>IO.sys(1000 + cmd, par)</b>	<b>cmd</b> : entspricht CMD 0x01 bis 0x4D des DFPlayer Mini <b>par</b> : 16 Bit Parameter (0 wenn nicht benötigt)
<b>IO.sys(1100, 0)</b>	Warteschlange (Queue) löschen
<b>IO.sys(1100, 1)</b>	Warteschlange ausführen
<b>IO.sys(1101, par)</b>	Datei zur Warteschlange hinzufügen (max. 16 Einträge)

*Beispiele:*

IO.sys(1012, 0)                      DF-Player Reset (knackt zweimal)  
IO.sys(1006, 20)                    Stelle Lautstärke auf 20 (gültige Werte 0..30)  
Nach diesen Befehlen muss der Busy-Status abgefragt werden.

IO.sys(1015, (2 \* 256) + 45)      Spiele MP3-Datei [045.mp3] in Verzeichnis [02] ab  
Busy kann abgefragt werden um das Ende der Datei abzuwarten.  
Die laufende Datei wird sofort beendet, wenn ein neuer Befehl gesendet wird.

IO.sys(1100, 0)                      Warteschlange löschen  
IO.sys(1101, (2 \* 256) + 1)        MP3-Datei [001.mp3] in Verzeichnis [02] hinzufügen  
IO.sys(1101, (2 \* 256) + 3)        MP3-Datei [003.mp3] in Verzeichnis [02] hinzufügen  
IO.sys(1101, (2 \* 256) + 24)       MP3-Datei [024.mp3] in Verzeichnis [02] hinzufügen  
IO.sys(1101, (2 \* 256) + 31)       MP3-Datei [031.mp3] in Verzeichnis [02] hinzufügen  
IO.sys(1101, (1 \* 256) + 1)        MP3-Datei [001.mp3] in Verzeichnis [01] hinzufügen  
IO.sys(1100, 1)                      Warteschlange ausführen

*Beispiel für Busy-Status lesen:*

1000:  
    s = IO.sys(1000, 0)              Status lesen  
    if s = 0 goto 1000                Wenn busy = 0, status lesen wiederholen  
    IO.sys(1000, 1)                  Status zurücksetzen  
    return                              und zurück

Link zur DF-Player Anleitung (englisch):

[https://www.dfrobot.com/wiki/index.php/DFPlayer\\_Mini\\_SKU:DFR0299](https://www.dfrobot.com/wiki/index.php/DFPlayer_Mini_SKU:DFR0299)

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

### Laufzeit-Fehlermeldungen

Über das Terminal (wenn es von der LED-Basic-Komponente unterstützt wird) werden Laufzeitfehler ausgegeben.

```
?ERROR xx IN LINE zzzzz
```

Fehler	Bedeutung
11	Unbekannter Token (sollte nicht vorkommen)
12	Falsche Adresse (sollte nicht vorkommen)
13	Zu viele verschachtelte GOSUB-Befehle
14	RETURN ohne GOSUB
15	Wert darf nicht 0 sein
16	Zu viele verschachtelte FOR-NEXT-Schleifen
17	Falsche Werte bei TO/DOWNT0
18	Next-Variable ungültig
19	Falscher Wert bei LED-Befehl
20	Falscher Wert bei IO-Befehl

Bei LED-Basic-Komponenten ohne Terminal (z.B. LED-Button 12) wird der Fehlercode durch die Anzahl der rot blinkenden LEDs (+9) angezeigt (6 blinkende LEDs bedeutet Fehler 15).

# LED-BASIC

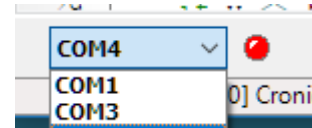
Komponenten, Editor, Befehlssatz

---

## Led-Basic-Komponenten

### Treiberinstallation

Jede LED-Basic-Komponente mit USB-Anschluss und auch der Programmieradapter „Prog-SB“ werden über einen virtuellen COM-Port angesprochen. Unter Windows 7 und 8.x muss hierzu die Treiberdatei „LedBasic.inf“ installiert werden. Sollten Sie noch Windows 8.x benutzen, müssen Sie eventuell das „Erzwingen der Treibersignatur deaktivieren“. Wie das funktioniert, können sie leicht mit einer Google-Suche herausfinden. Unter Windows 10 ist keine Installation eines Treibers erforderlich. Hier meldet sich jede Komponente als „Seriell USB-Gerät (COMx)“ an. Welcher COM-Port für Ihre verwendete Komponente gültig ist, können Sie am besten testen, indem Sie im LED-Basic-Editor die Liste der COM-Ports anklicken bevor und nachdem Sie die Komponente angesteckt haben. Der COM-Port der hinzugekommen ist, sollte ausgewählt werden.



### Unterschiedliche Komponenten

Für LED-Basic sind verschiedene Komponenten verfügbar. Diese unterscheiden sich in der Regel durch die Möglichkeiten, unterschiedliche Typen und Mengen von LEDs anzusteuern sowie der Anschluss von Sensoren, Tastern und der internen Peripherie wie Echtzeituhr oder IO-Ports.

Jede LED-Basic-Komponente benutzt denselben BASIC-Grundbefehlssatz. Es unterscheiden sich nur die LED- und IO-Zusatzbefehle. Welche Befehle Ihre LED-Basic -Komponente versteht, entnehmen Sie bitte der weiter unten folgenden Beschreibung. LED-Basic ist universell, es muss auch keine LED-Ansteuerung vorhanden sein, um LED-Basic zu verwenden. LED-Basic kann auch in Komponenten verwendet werden, die nur Sensoren abfragt und Schaltvorgänge auslöst.

Es gibt eine Reihe von LED-Basic Komponenten, die über einen USB-Anschluss verfügen. Dieser wird dann in der Regel gleichzeitig zur Programmierung und als Terminal-Ausgang verwendet. Sollte die Komponente über keinen USB-Anschluss verfügen, ist ein USB-Programmieradapter „Prog-SB“ erforderlich. Dieser verbindet den PC mit der 6-poligen Programmierleiste der LED-Basic -Komponente.

### Liste der von LED-Basic unterstützten Komponenten

Es folgt eine Beschreibung der von LED-Basic-Editor unterstützten Komponenten. Beachten Sie bitte, dass es teilweise identische Komponenten in unterschiedlichen Bauformen gibt. Neben der Anschlussbelegung zu jeder LED-Basic-Komponente folgt eine Liste der möglichen LED- und IO-Befehle sowie der Parameter in der Konfigurationszeile.

Hinweise: Alle Input/Output-Pins haben einen Pegel von maximal 3,3 Volt. Um eine Zerstörung der Hardware vorzubeugen, sollten Sie Vorwiderstände oder Pegelwandler benutzen, wenn höhere Spannungen angelegt werden. Input-Pins haben integrierte Pullup-Widerstände, ein Taster wird einfach vom Input-Pin gegen GND angeschlossen. Output-Pins liegen im Ruhezustand auf Low-Pegel (0 Volt). Sie können im High-Pegel nur wenige Milliampere treiben, eine LED kann direkt über einen passenden Vorwiderstand angeschlossen werden. Sollten Sie einen höheren Strom benötigen (z.B. zum Ansteuern eines Relais), sind unbedingt Transistoren oder IC-Treiber erforderlich.

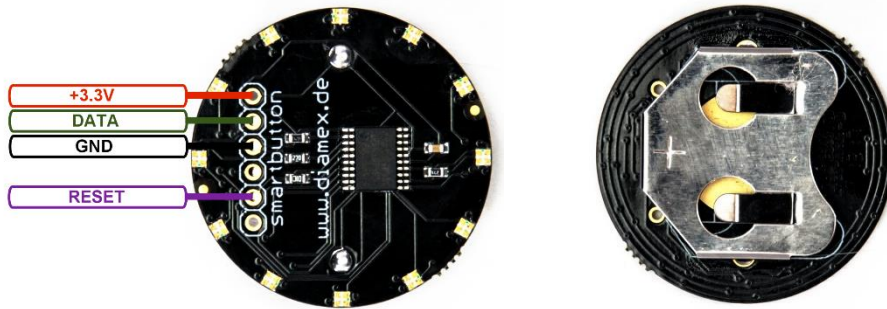
LED-Ausgänge für serielle RGB-LEDs (WS2812, APA102 o.A.) haben immer einen Pegel von 5 Volt.

Alle nicht bezeichneten Pins sind unbenutzt oder dienen zur Programmierung und zum Test bei der Herstellung.

# LED-BASIC

## Komponenten, Editor, Befehlssatz

### LED-Badge (12 LEDs)



BADGE = Abzeichen, Sticker, Plakette

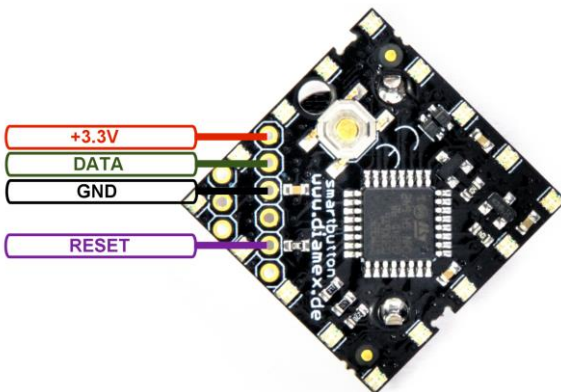
12 RGB-LEDs ohne PWM. Ein- und Ausschalten durch Einsetzen und Herausnehmen der Batterie möglich.

Programmierung über Prog-SB (+3.3V, DATA, GND, RESET).

System-Code: 3110

Konfigurationszeile	LED-Befehle	IO-Befehle
KEINE	SETLED, SETALL	KEINE

### LED-Badge (16 LEDs)



Unterschiedliche Bauformen sind erhältlich. Die Pinbelegung für den Programmieradapter ist bei allen Bauformen identisch.

BADGE = Abzeichen, Sticker, Plakette

16 RGB-LEDs ohne PWM. Einschalten durch kurzen Tastendruck, Ausschalten durch Tastendruck länger als 2 Sekunden. Eine Tastenabfrage ist per Basic-Befehl möglich. Terminal Print- und Fehler-Ausgabe über Prog-SB.

Programmierung über Prog-SB (+3.3V, DATA, GND, RESET, TERMINAL).

System-Code: 3120

Konfigurationszeile	LED-Befehle	IO-Befehle
P...	SETLED, SETALL	WAITKEY, GETKEY CLRPORT

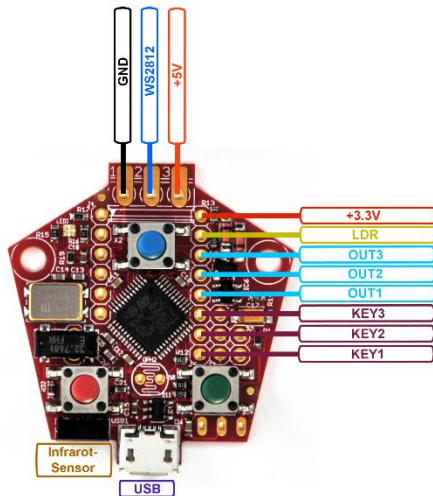
Durch `IO.clrport(0)` wird der LED-Button abgeschaltet. Hiermit kann z.B. nach einer per Basic vorgegebenen Anzahl von Schleifendurchläufen die Komponente abgeschaltet und damit Energie gespart werden.



# LED-BASIC

## Komponenten, Editor, Befehlssatz

### Basic-Pentagon-Board



Die Basis-Schaltung für LED-Basic. Für maximal 256 LEDs mit integrierter PWM (WS2812, SK6812 und kompatibel). Auch für RGBW-LEDs geeignet. Optional mit Infrarot-Fernbedienung steuerbar. Terminal Print- und Fehler-Ausgabe über USB. 3 Tasten oder Eingangspins, 3 programmierbare Ausgangspins.

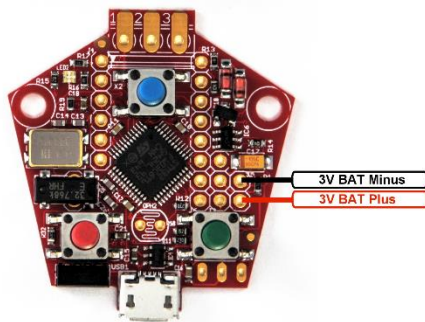
Stromversorgung über USB- oder LED-Anschluss möglich.

Ein LDR (Fotowiderstand) kann an direkt auf die Platine (OPH2) aufgelötet oder an die Pins LDR und +3,3V angeschlossen werden.

Programmierung über USB-Anschluss.

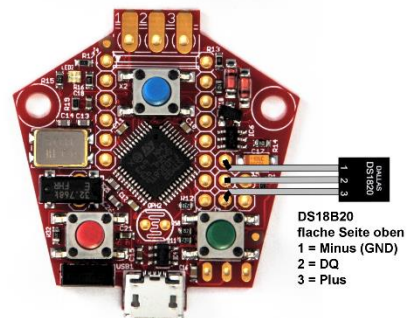
System-Code: 3130

Konfigurationszeile	LED-Befehle	IO-Befehle
L... C... M... P... S... F... Lxxx: Maximum = 256	Alle Befehle für RGB-LEDs mit PWM	WAITKEY, GETKEY, KEYSTATE GETRTC, SETRTC GETLDR GETIR SETPORT, CLRPORT GETTEMP



Wenn die Echtzeituhr benutzt werden soll, kann eine 3 Volt Stützbatterie (z.B. CR2032 oder 2 x AAA-Zellen) an die bezeichneten Pins angeschlossen werden.

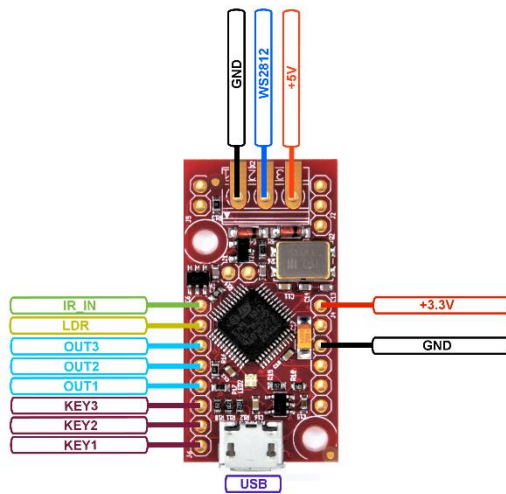
Anschluss eines DS18B20 Temperatursensors an das Pentagon-Board.



# LED-BASIC

## Komponenten, Editor, Befehlssatz

### Basic-Budget-Board



Die Basis-Schaltung für LED-Basic in besonders leichter Ausführung (z.B. für Flugmodelle geeignet). Softwarekompatibel mit dem Basic-Pentagon-Board. Für maximal 256 LEDs mit integrierter PWM (WS2812, SK6812 und kompatible). Auch für RGBW-LEDs geeignet. Terminal Print- und Fehler-Ausgabe über USB. 3 Tasten oder Eingangspins, 3 programmierbare Ausgangspins.

Stromversorgung über USB- oder LED-Anschluss möglich.

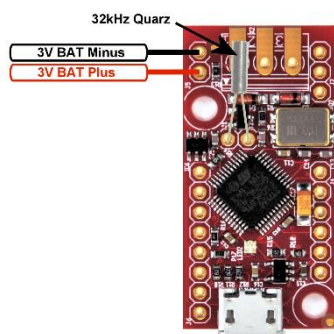
Ein LDR (Fotowiderstand) kann an die Pins LDR und +3,3V angeschlossen werden.

Ein Infrarot-Empfänger kann an den Pin IR\_IN angeschlossen werden (siehe Anschlussplan beim Basic-Booster).

Programmierung über USB-Anschluss.

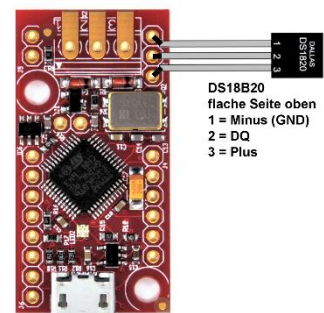
System-Code: 3130

Konfigurationszeile	LED-Befehle	IO-Befehle
L... C... M... P... S... F... Lxxx: Maximum = 256	Alle Befehle für RGB-LEDs mit PWM	WAITKEY, GETKEY, KEYSTATE GETRTC, SETRTC GETLDR GETIR SETPORT, CLRPORT GETTEMP



Wenn die Echtzeituhr benutzt werden soll, kann eine 3 Volt Stützbatterie (z.B. CR2032 oder 2 x AAA-Zellen) an die bezeichneten Pins angeschlossen werden. Zusätzlich muss ein 32,768 kHz Quarz aufgelötet werden.

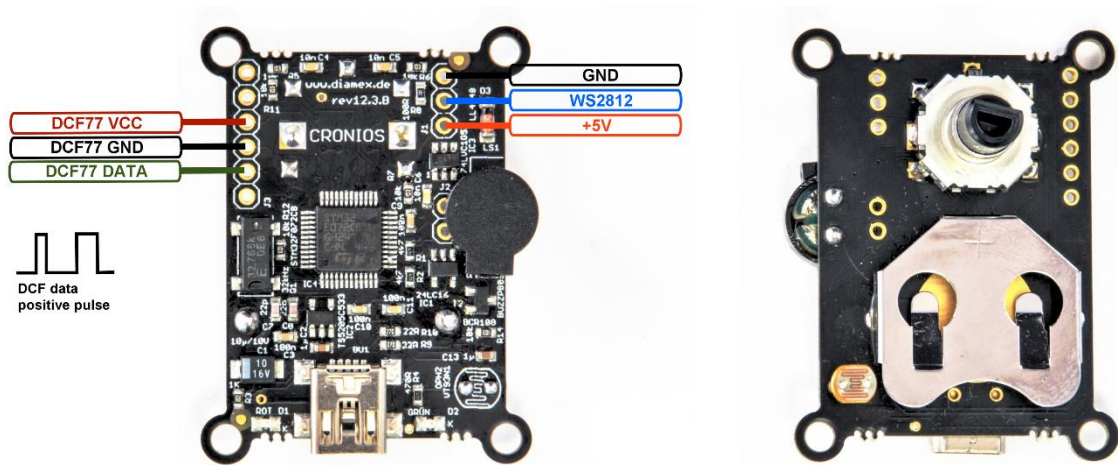
Anschluss eines DS18B20 Temperatursensors an das Budget-Board.



# LED-BASIC

## Komponenten, Editor, Befehlssatz

### Cronios 1



Die Cronios 1 Komponente mit LED-Basic-Software. Für maximal 256 LEDs mit integrierter PWM (WS2812, SK6812 und kompatible). Auch für RGBW-LEDs geeignet. Terminal Print- und Fehler-Ausgabe über USB. Integrierte Echtzeituhr mit Stützbatterie, Drehimpulsgeber mit Tastenfunktion, LDR und Lautsprecher sind auf dem Board vorhanden. EEPROM mit 1024 Speicherstellen. Stromversorgung über USB- oder LED-Anschluss möglich. Ab LED-Basic-Version 15.1.12 kann zur Zeitsynchronisation ein DCF77-Modul angeschlossen werden.

Programmierung über USB-Anschluss.

System-Code: 3140

Konfigurationszeile	LED-Befehle	IO-Befehle
L... C... M... P... S... F... Lxxx: Maximum = 256	Alle Befehle Für RGB-LEDs mit PWM	WAITKEY, GETKEY, KEYSTATE GETRTC, SETRTC GETLDR BEEP GETENC, SETENC EEREAD, EEWRITE [0..1023] SYS [COM, CALIB]

# LED-BASIC

## Komponenten, Editor, Befehlssatz

### Cronios-Segmenta



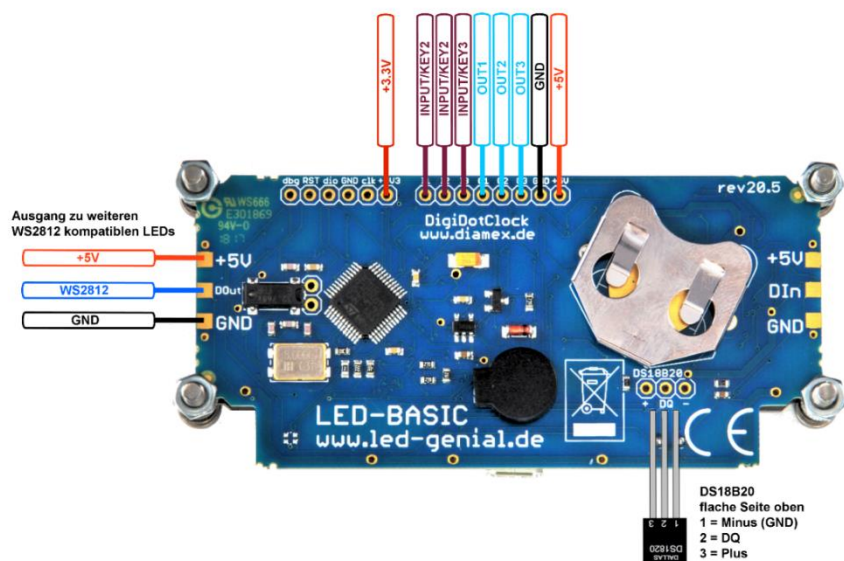
Bausatz für eine 4-stellige 7-Segment-Uhr mit farbigen Ziffern, bestehend aus SK6812-Mini-LEDs. Terminal Print- und Fehler-Ausgabe über USB. Integrierte Echtzeituhr mit Stützbatterie, 3 Tasten oder Eingangspins, 3 Ausgangspins für Schaltsignale, LDR und Lautsprecher sind auf dem Board vorhanden. Anschluss für optionalen Temperatursensor DS18B20. Stromversorgung über USB-Anschluss. Zusätzlich zu fest verdrahteten 30 LEDs können über den Ausgang (Dout) bis zu insgesamt 256 LEDs angesteuert werden.

Programmierung über USB-Anschluss.

System-Code: 3150

Konfigurationszeile	LED-Befehle	IO-Befehle
L... M... P... S... F... Lxxx: Maximum = 256	Alle Befehle für RGB-LEDs mit PWM	WAITKEY, GETKEY, KEYSTATE GETRTC, SETRTC GETLDR SETPORT, CLRPORT GETTEMP BEEP EEREAD, EEWRITE [0..15]

Hinweis: Aufgrund der recht hohen Stromaufnahme kann es passieren, dass die Anzeige nicht sichtbar ist. Versuchen Sie es dann an einem anderen USB-Anschluss oder verwenden ein USB-Hub mit eigenem Netzteil.

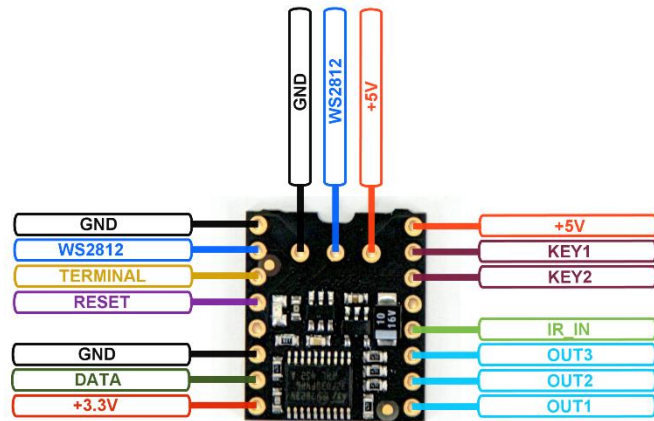


# LED-BASIC

## Komponenten, Editor, Befehlssatz

### Basic-Booster

Mini-Interface für LED-Basic. Für maximal 64 LEDs mit integrierter PWM (WS2812, SK6812 und kompatibel). Auch für RGBW-LEDs geeignet. Terminal Print- und Fehler-Ausgabe über Prog-SB. Über 2 Eingangspins für Taster (KEY1,2) können Funktionen ausgelöst werden, 3 Ausgangspins (OUT1,2,3) sind für Schaltvorgänge vorhanden. Optional kann ein Infrarot-Empfänger an IR\_IN angeschlossen werden, die hierfür erforderlichen Bauteile sind im Set mit der Fernbedienung erhältlich. Die Stromversorgung geschieht über einen der +5V-Anschlüsse.



Programmierung über Prog-SB (+3.3V, DATA, GND, RESET, TERMINAL).

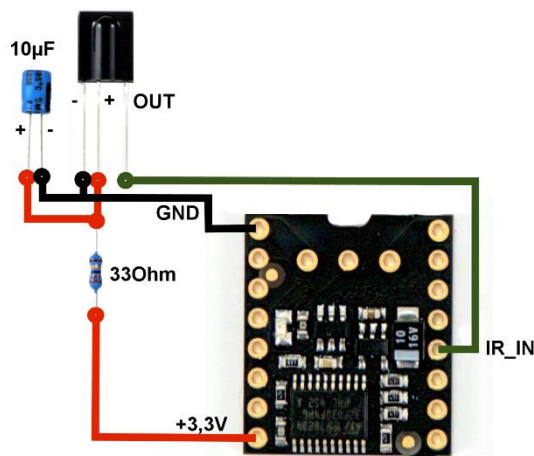
System-Code: 3160

Konfigurationszeile	LED-Befehle	IO-Befehle
L... C... M... P... S... F... Lxxx: Maximum = 64	Alle Befehle für RGB-LEDs mit PWM	WAITKEY, GETKEY, KEYSTATE SETPORT, CLRPORT GETIR

### Anschluss eines Infrarot-Empfängers an den Basic-Booster

Zum Lieferumfang der Infrarot-Erweiterung gehören neben der Infrarot-Fernbedienung auch der passende Infrarot-Empfänger TSOP31436, ein Widerstand 33 Ohm und ein Elko 10  $\mu$ F. Widerstand und Elko dienen zur Störunterdrückung aus der Betriebsspannung und damit zu sicherem Empfang der Infrarot-Signale. Bitte nicht die Polarität des Elkos vertauschen, das kürzere Beinchen (schwarze Markierung auf dem Gehäuse) kommt an GND. Schließen Sie die Bauteile nach der folgenden Zeichnung an den Basic-Booster an.

Tipp: Die Basic-Booster Platine lässt sich mit passenden Stiftleisten auf einem Steckbrett (Breadboard) platzieren. Die Bauteile für den Infrarot-Empfänger lassen sich damit bequem ohne Lötverbindungen verdrahten.

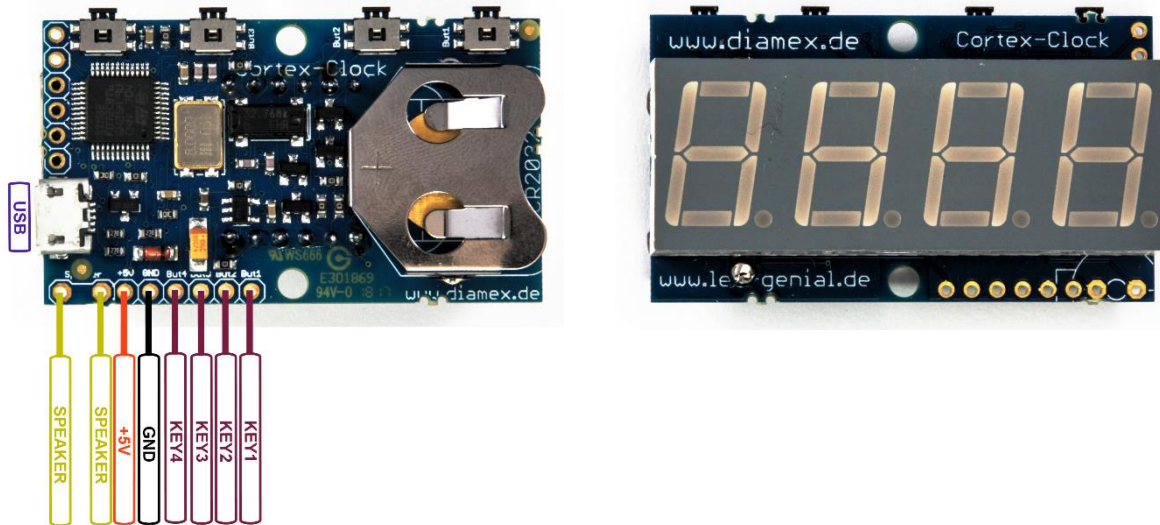




# LED-BASIC

## Komponenten, Editor, Befehlssatz

### Cortex-Clock



4-stelliges 7-Segment-Uhr Modul. Terminal Print- und Fehler-Ausgabe über USB. Integrierte Echtzeituhr mit Stützbatterie. 4 Taster auch über Lötunkte als Eingangspins zu verwenden. Anschluss für Miniatur-Lautsprecher zur Signalausgabe ist auf dem Board vorhanden. Stromversorgung über USB-Anschluss oder über die Lötunkte +5V und GND.

Die Anwendung dieses Moduls ist nicht auf die Uhrzeit-Anzeige beschränkt. Durch Programmierung sind unter anderem auch solche Anwendungen wie Kurzzeitwecker oder Eieruhr, Stoppuhr, Schrittzähler und mehr möglich.

Programmierung über USB-Anschluss.

System-Code: 3170

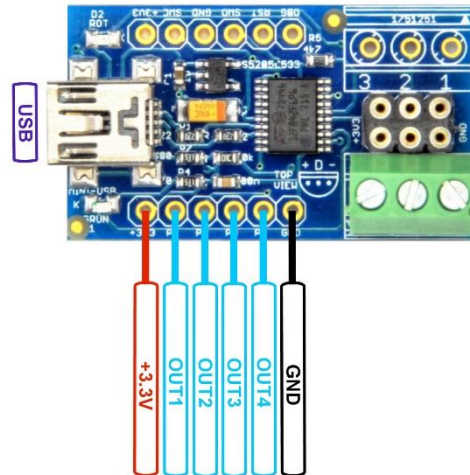
Konfigurationszeile	LED-Befehle	IO-Befehle
P...	Befehle für 7-SEGMENT-DISPLAY, BRIGHT [0..15]	WAITKEY, GETKEY, KEYSTATE GETRTC, SETRTC BEEP EEREAD, EEWRITE [0..15]

# LED-BASIC

## Komponenten, Editor, Befehlssatz

### Temperatur-Sensor Interface

Interface für maximal 8 Temperatur-Sensoren des Typs DS18B20 von Dallas. Über 4 Ausgangsports können Schaltvorgänge ausgelöst werden. Terminal Print- und Fehler-Ausgabe über USB. Die Stromversorgung geschieht über den USB-Anschluss. Steckbrett-geeignet mit passenden Stiftleisten.



Programmierung über USB-Anschluss.

System-Code: 3180

Konfigurationszeile	LED-Befehle	IO-Befehle
P... S...	Keine	XTEMPCNT, XTEMPVAL SETPORT, CLRPORT

Hinweise zur Benutzung:

Keine Sensoren bei angeschlossener Stromversorgung anschließen oder entfernen.

Die Sensoren nicht verpolen, sie werden hierdurch zerstört.

Bei Anschluss mit parasitärer Stromversorgung über nur 2 Leitungen ist eventuell nicht die maximale Anzahl von Sensoren möglich (ausprobieren).

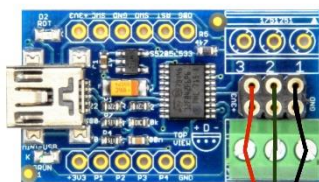
Alle Sensoren werden nacheinander im Abstand von ca. 1 Sekunde abgefragt, es kann also bis zu 8 Sekunden dauern, bis der gewünschte Temperaturwert aktualisiert wird (bei 8 angeschlossenen Sensoren).

Die Reihenfolge der erkannten Sensoren kann sich beim Austausch von Sensoren verändern.

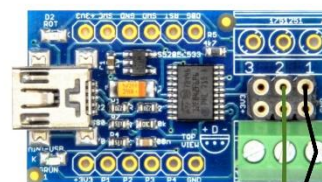
Alle Sensor-Anschlüsse auf der Platine sind parallel geschaltet, die Reihenfolge der Sensoren ist somit unabhängig davon, an welchen Pins sie angeschlossen sind.

Die grüne LED blinkt beim Lesen eines Sensors kurz auf, dies kann durch S0 in der Konfigurationszeile abgeschaltet werden.

Die rote LED zeigt durch Blinken an, dass ein Laufzeitfehler aufgetreten ist. Die genaue Fehlermeldung wird über das Terminal ausgegeben.



Externe Stromversorgung  
DS18B20  
flache Seite oben  
1 = Minus (GND)  
2 = DQ  
3 = mit Minus verbunden

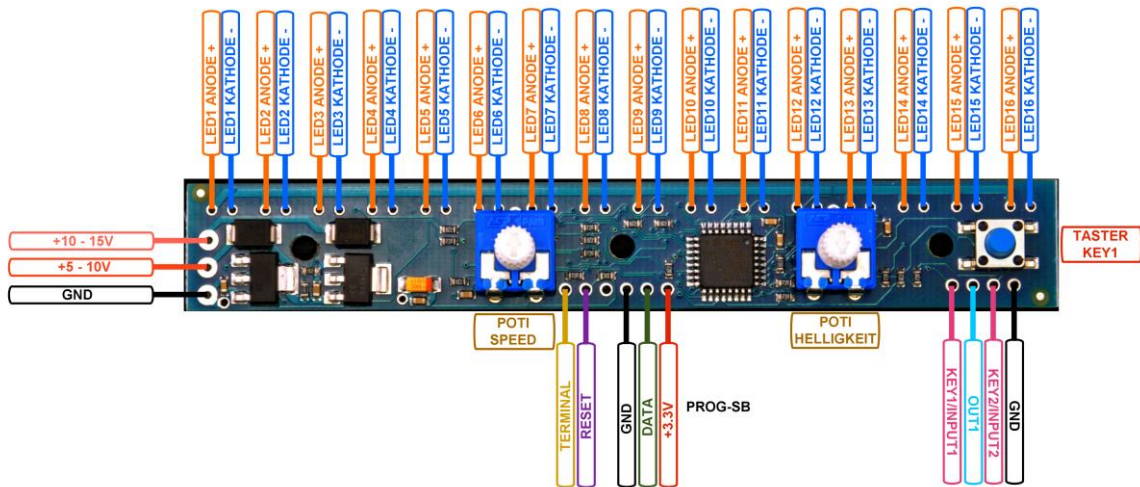


Parasitäre Stromversorgung  
DS18B20  
flache Seite oben  
1 = Minus (GND)  
2 = DQ  
3 = Plus

# LED-BASIC

## Komponenten, Editor, Befehlssatz

### Lauflicht mit 16 LEDs



Lauflicht für 16 einfarbige LEDs. 2 Potentiometer zur Geschwindigkeit- und Helligkeitsregelung (oder andere Funktion programmierbar). 1 Taster und ein zusätzlicher Eingangsport vorhanden. Über einen Ausgangsport können Schaltvorgänge ausgelöst werden. Terminal Print- und Fehler-Ausgabe über Prog-SB. Die Stromversorgung (5-10V / 10-15V) geschieht über Lötanschlüsse.

Programmierung über Prog-SB (+3.3V, DATA, GND, RESET, TERMINAL).

System-Code: 3190

Konfigurationszeile	LED-Befehle	IO-Befehle
P...	SETLED, SETALL BRIGHT	GETKEY, WAITKEY, KEYSTATE SETPORT, CLRPORT GETPOTI EEREAD, EEWRITE [0..15]

LED.setled(pos, val)

val: 0 = LED an Position **pos** aus, 1 = LED an Position **pos** ein

pos: 0..15

LED.setall(val)

val: 0 = alle LEDs aus, 1 = alle LEDs ein

LED.bright(val)

val: 0..255 = alle LED auf Helligkeit **val** setzen, >255 = Helligkeit wird mit Poti geregelt (Standard)

IO.getpoti(idx)

Liest die Werte der Potentiometer (Poti) oder deren umgewandelten Werte aus einer Tabelle aus.

Gültige idx-Werte:

0 = Speed-Poti-Wert (0..255) direkt lesen

1 = Umgewandelter Speed-Poti-Wert aus Tabelle (0..15) lesen

2 = Helligkeits-Poti-Wert (0..255) direkt lesen

3 = Umgewandelter Helligkeits-Poti-Wert aus logarithmischer Tabelle (0..255) lesen

Bei einem ungültigen **idx**-Wert wird Null gelesen.

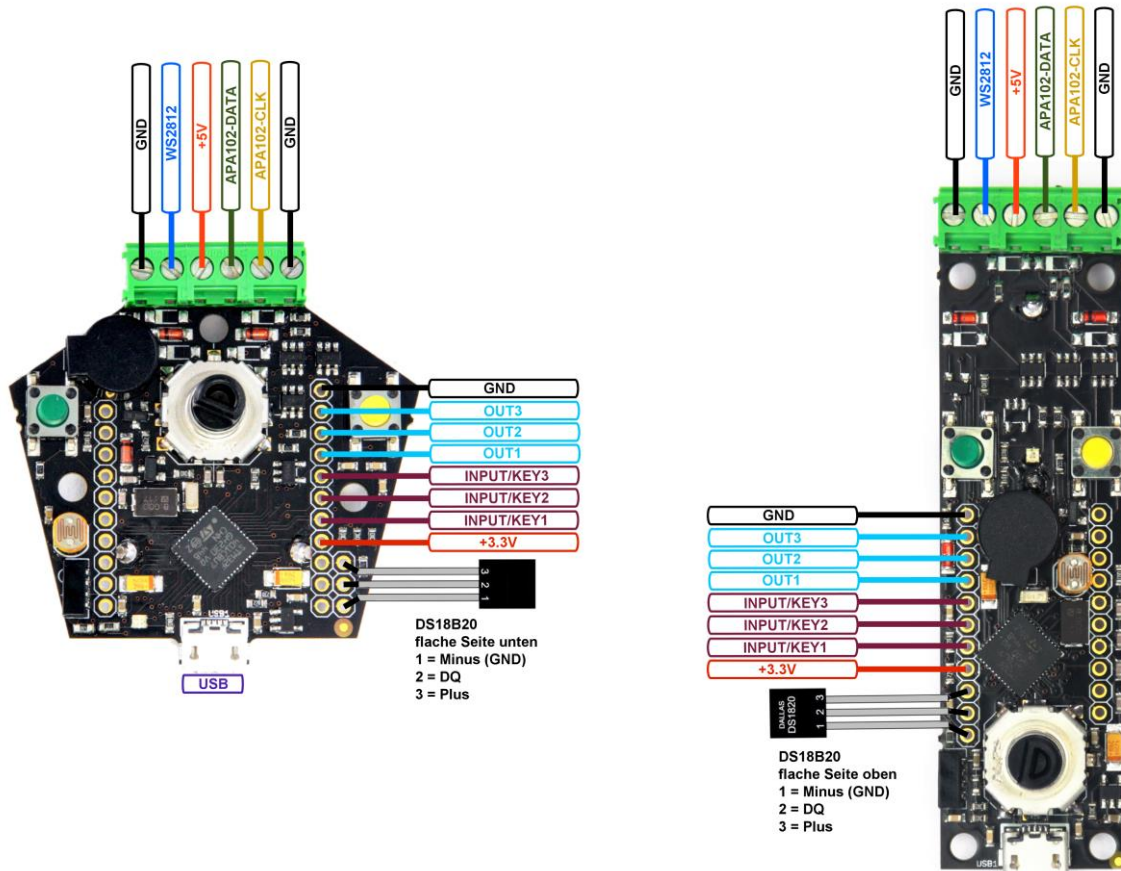
Das Auslesen der Potentiometer-Werte geschieht über einen Analog-Digital-Wandler im Microcontroller des Lauflichtes. Um einen gleichmäßigen Wert zu ermitteln, werden mehrere Werte hintereinander gemessen und miteinander verglichen. Der erste mit der **getpoti**-Funktion gelesene Wert kann deshalb noch ungültig sein (Wert = 0).



# LED-BASIC

## Komponenten, Editor, Befehlssatz

### All-In-One Power-M4-Board



Die Power-Schaltung mit vielen Funktionen auf Cortex-M4-Basis. Für maximal 1024 LEDs mit integrierter PWM (WS2812, SK6812 und kompatible sowie APA102 und kompatible). Auch für RGBW-LEDs geeignet. Optional mit Infrarot-Fernbedienung steuerbar. Terminal Print- und Fehler-Ausgabe über USB. 2 Tasten, 3 Eingangspins, 3 programmierbare Ausgangspins. Drehimpulsgeber mit Taste, Lautsprecher und LDR (Fotowiderstand), Echtzeituhr (RTC) mit Batterie, Anschlussmöglichkeit für Temperatursensor DS18B20, Beeper. EEPROM mit 1024 Speicherstellen. Stromversorgung über USB- oder LED-Anschluss möglich.

Programmierung über USB-Anschluss.

System-Code: 3210

Konfigurationszeile	LED-Befehle	IO-Befehle
L... C... M... P... S... T... A... F... Lxxx: Maximum = 1024 Tx: 0 = WS2812, 1 = APA102	Alle Befehle für RGB-LEDs mit PWM	WAITKEY, GETKEY, KEYSTATE GETRTC, SETRTC GETLDR GETIR SETENC, GETENC SETPORT, CLRPORT GETTEMP BEEP EEREAD, EEWRITE [0..1023]

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

In der Konfigurationszeile kann mit dem Wert **Ax** die Bitrate für die Ansteuerung der APA102-LEDs eingestellt werden. Je höher der Wert, desto niedriger die Bitrate. Bei längeren Verbindungsleitungen zu den LEDs kann es erforderlich sein, eine geringere Bitrate einzustellen, wenn Störungen auftreten. Bei Verwendung von WS2812-LEDs hat dieser Wert keine Bedeutung.

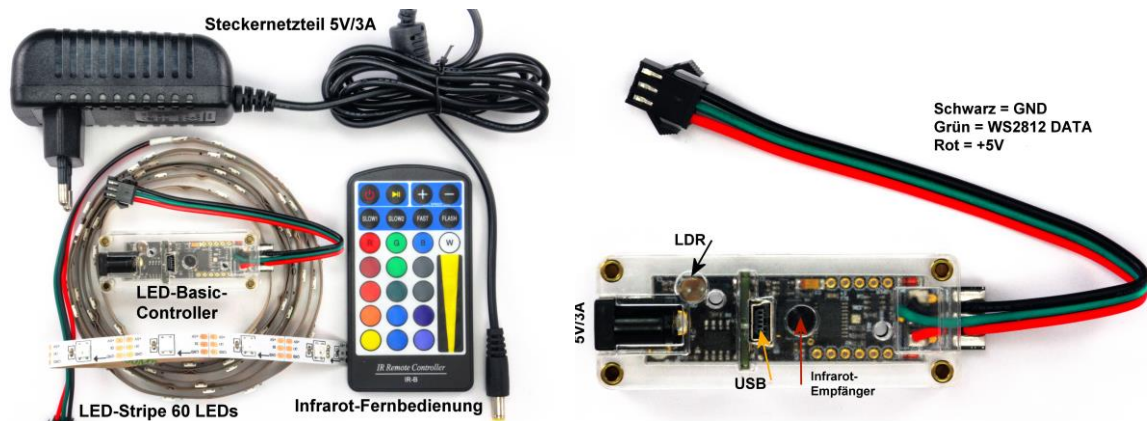
A0 = 42 Mbit	A1 = 21 Mbit	A2 = 10,5 Mbit	A3 = 5,25 Mbit
A4 = 2,6 Mbit (Std.)	A5 = 1,3 Mbit	A6 = 656 kBit	A7 = 328 kBit

Bei Werten oberhalb von 10 Mbit kann es zum Flackern der LEDs kommen.

# LED-BASIC

## Komponenten, Editor, Befehlssatz

### LED-Box



Spezierschaltung zur Steuerung von WS2812-Stripes mit Infrarot-Fernbedienung. Die LED-Box beinhaltet den LED-Basic-Controller, einen LED-Stripe mit 60 LEDs, Infrarot-Fernbedienung und Netzteil. Ideal für Effektbeleuchtungen in der Wohnung. Über den LDR sind Helligkeitsanpassungen des Stripes abhängig von der Umgebungshelligkeit möglich. EEPROM mit 8 Speicherstellen. Terminal Print- und Fehler-Ausgabe über USB. Stromversorgung über 5V/3A Steckernetzteil.

Programmierung über USB-Anschluss.

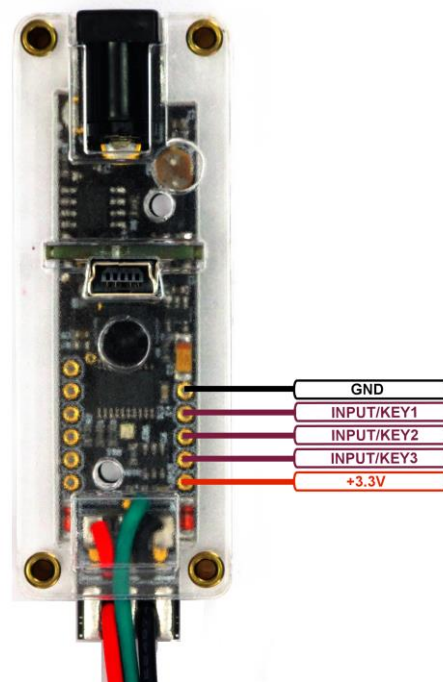
System-Code: 3220

Konfigurationszeile	LED-Befehle	IO-Befehle
L... M... P... S... F... Lxxx: Maximum = 64	Alle Befehle für RGB-LEDs mit PWM	WAITKEY, GETKEY, KEYSTATE SETPORT, CLRPORT GETIR GETLDR EEREAD, EEWRITE [0..7]

Da die LEDs auch im ausgeschalteten Zustand einen Ruhestrom aufnehmen, kann die Stromversorgung der LEDs über den Befehl **IO.setport(1)** ein- und über den Befehl **IO.clrport(1)** ausgeschaltet werden.

Es kann passieren, dass die LEDs nach dem Einschalten kurz flackern oder sogar ständig leuchten. Es sollte deshalb sofort nach dem Einschalten ein Befehl zum Löschen aller LEDs geschickt werden (z.B. **LED.blackout()**)

Drei Eingangs-pins (Input/Key) können im LED-Basic abgefragt werden um z.B. bestimmte LED-Sequenzen über ein externes Signal auszulösen.



# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

LED-Matrix 10x10

**DEMNÄCHST LIEFERBAR**

*Status: Aufgrund von technischen Schwierigkeiten bei der Produktion, verzögert sich die Veröffentlichung dieser Komponente.*

### BILDER FOLGEN

Mini 10x10 Matrix mit RGB-Leuchtdioden. Echtzeituhr mit Batterie, 2 Taster. Terminal Print- und Fehler-Ausgabe über USB. Stromversorgung und Programmierung über USB-Anschluss.

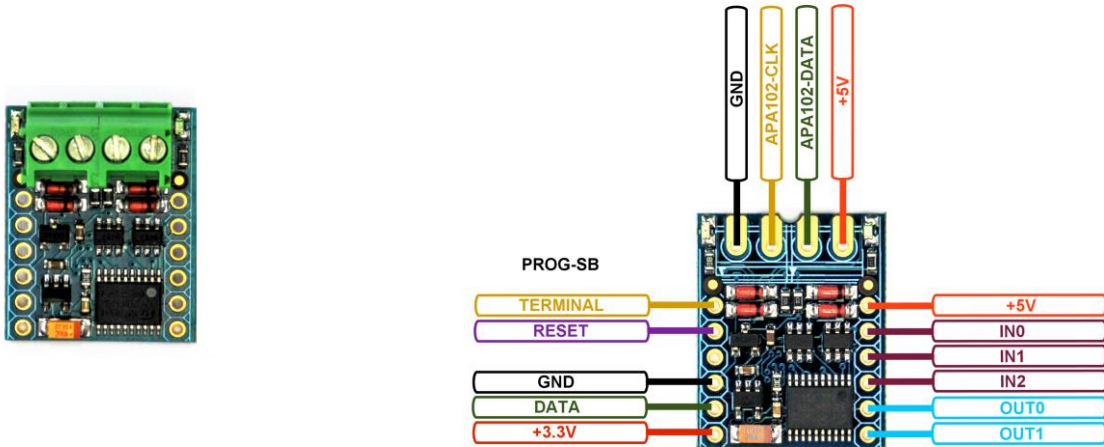
System-Code: 3200

Konfigurationszeile	LED-Befehle	IO-Befehle
P...	MATRIX-Befehle (noch in der Entwicklung)	WAITKEY, GETKEY, KEYSTATE GETRTC, SETRTC

# LED-BASIC

## Komponenten, Editor, Befehlssatz

### APA-Booster



Spezial-Mini-Interface für LED-Basic. Für maximal 256 RGB-LEDs mit integrierter PWM vom Typ APA102 (DATA- und CLOCK-Leitung). Terminal Print- und Fehler-Ausgabe über Prog-SB. Über 3 Eingangspins (IN0..3) können Funktionen ausgelöst werden, 2 Ausgangspins (OUT0..1) sind für Schaltvorgänge vorhanden. Die Stromversorgung geschieht über den +5V-LED-Anschluss.

**Diese Komponente ist nicht für WS2812/SK6812-LEDs geeignet!**

Programmierung über Prog-SB (+3.3V, DATA, GND, RESET, TERMINAL).

System-Code: 3230

Konfigurationszeile	LED-Befehle	IO-Befehle
L... CRGB P... S... A... F... Lxxx: Maximum = 256	Alle Befehle für RGB-LEDs mit PWM	WAITKEY, GETKEY, KEYSTATE SETPORT, CLRPORT

In der Konfigurationszeile kann mit dem Wert **Ax** die Bitrate für die Ansteuerung der LEDs eingestellt werden. Je höher der Wert, desto niedriger die Bitrate. Bei längeren Verbindungsleitungen zu den LEDs kann es erforderlich sein, eine geringere Bitrate einzustellen, wenn Störungen auftreten.

A0 = 24 Mbit                      A1 = 12 Mbit                      A2 = 6 Mbit                      A3 = 3 Mbit  
 A4 = 1,5 Mbit (Std.)            A5 = 750 kBit                      A6 = 375 kBit                      A7 = 187,5 kBit

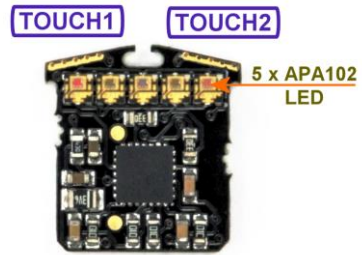
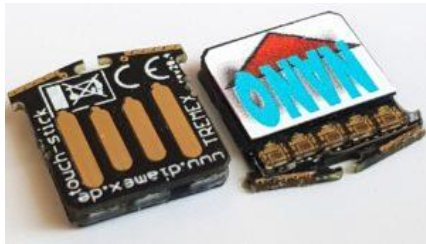
Bei Werten oberhalb von 10 Mbit kann es zum Flackern der LEDs kommen.

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

### Touch-Stick (Nano)



Mini-USB-Stick mit LED-Basic-Intelligenz. 5 Mini-APA102-LEDs, 2 Touch-Sensoren. EEPROM mit 16 Speicherstellen. Ein- und Ausgabe über Terminal. Print- und Fehler-Ausgabe über USB. Stromversorgung über USB.

Programmierung über USB-Anschluss.

System-Code: 3260

Konfigurationszeile	LED-Befehle	IO-Befehle
M... P... CRGB	Alle Befehle für RGB-LEDs mit PWM	WAITKEY, GETKEY, KEYSTATE EEREAD, EEWRITE [0..15] SYS [COM]

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

Touch-Lamp

**DEMNÄCHST LIEFERBAR**

*Status: Entwicklung abgeschlossen, warten auf Produktion.*

### BILDER FOLGEN

Effektlampe mit Sensor-Tasten und Fernbedienung. 8 Integrierte WS2812-LEDs sind in Farbe und Helligkeit veränderbar und beleuchten ein Plexiglas-Display. EEPROM mit 8 Speicherstellen. Terminal Print- und Fehler-Ausgabe über USB. Stromversorgung über USB.

Programmierung über USB-Anschluss.

System-Code: 3270

Konfigurationszeile	LED-Befehle	IO-Befehle
M... P...	Alle Befehle für RGB-LEDs mit PWM	WAITKEY, GETKEY, KEYSTATE SETPORT, CLRPORT GETIR EEREAD, EEWRITE [0..7]

Da die LEDs auch im ausgeschalteten Zustand einen Ruhestrom aufnehmen, kann die Stromversorgung der LEDs über den Befehl **IO.setport(1)** ein- und über den Befehl **IO.clrport(1)** ausgeschaltet werden.

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

VFD-Clock

**DEMNÄCHST LIEFERBAR**

*Status: Entwicklung fast abgeschlossen.*

### BILDER FOLGEN

8-stellige Uhr mit 7-Segment VFD-Röhren IV-6. Terminal Print- und Fehler-Ausgabe über USB. Integrierte Echtzeituhr mit Stützbatterie. DCF-77 Anschluss mit Klinkenbuchse. 4 Taster, LDR, Beeper, Temperatur-Sensor DS18B20. Stromversorgung über USB-Anschluss, Stromaufnahme ca. 450mA.

Programmierung über USB-Anschluss.

System-Code: 3280

Konfigurationszeile	LED-Befehle	IO-Befehle
P... M... F...	Befehle für 7-SEGMENT-DISPLAY BRIGHT [0..256]* Alle Befehle für RGB-LEDs mit PWM	WAITKEY, GETKEY, KEYSTATE GETRTC, SETRTC BEEP GETLDR GETTEMP EEREAD, EEWRITE [0..999]* SYS [COM, CALIB, DFP]

\*LED.bright(val)

val: 0..256 = VFD-Display auf Helligkeit **val** setzen.

Bei 0 sind Heiz- und Hochspannung für die Röhren abgeschaltet.

\*IO.eeread(adr), IO.eewrite(adr, data)

**adr:** [0..999]

Die EEPROM-Adressen 0..7 sind für die Helligkeitssteuerung der VFD-Röhren reserviert, die herstellungsbedingt sehr große Unterschiede aufweisen können. Bitte benutzen Sie diese Adressen nicht für das Anwenderprogramm. Für jede VFD-Röhre ist ein Helligkeits-Korrekturwert vorhanden. 0 = keine Korrektur, positiver Wert = heller (Maximum 4000), negativer Wert = dunkler (Minimum -1900). Bei zu hohen positiven Werten kann es zum Flackern der Anzeige kommen. Nach Änderung der Werte im EEPROM muss ein Neustart (F12 drücken) durchgeführt werden um die Werte zu übernehmen. Einzelne Segmente können hierdurch natürlich nicht ausgeglichen werden, hier hilft nur ein Austausch der Röhre.

\*IO.eeread(adr), IO.eewrite(adr, data)

**adr:** [1000..1007]

Diese „virtuellen Adressen“ sind für die temporäre Helligkeitseinstellung der VFD-Röhren vorgesehen. Ein Schreiben auf eine dieser Adressen bewirkt eine sofortige Änderung der Helligkeit der zugehörigen Röhre. Diese Adressen gehen jedoch beim Entfernen der Stromversorgung verloren. Zur permanenten Speicherung müssen sie in die EEPROM-Adressen 0..7 kopiert werden.

Anzahl, Typ und Farbanordnung der LEDs in der Konfigurationszeile werden ignoriert. Die großen zentralen LEDs werden über die Nummern 0 bis 7 angesprochen, beginnend von links. Die kleinen LEDs werden über die Nummern 8 bis 39 angesprochen, beginnend von links (4 unter jeder Röhre).

Eine Wärmeentwicklung der Schaltung ist bedingt durch die Heiz- und Hochspannungserzeugung normal.

Der Hochspannungsgenerator und die PWM der LEDs können den Empfang von DCF77-Signalen stören. Tipp: Schalten Sie in den Nachtstunden die VFD-Anzeigen und die LED-Beleuchtung für einige Minuten ab, so dass das DCF77-Signal sauber empfangen werden und die Uhr sich synchronisieren kann.



# LED-BASIC

**Komponenten, Editor, Befehlssatz**

---

Vorsicht! Die Hochspannung in der Schaltung kann bis zu 60 Volt betragen. Bitte berühren Sie die Komponenten in der Schaltung nur im stromlosen Zustand.

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

6-O-Clock

**DEMNÄCHST LIEFERBAR**

*Status: Entwicklung abgeschlossen, warten auf Produktion.*

### BILDER FOLGEN

6-stellige Uhr mit 7-Segment-Anzeige. Terminal Print- und Fehler-Ausgabe über USB. Integrierte Echtzeituhr. 2 Taster. Stromversorgung über USB-Anschluss oder 3V Batterie (nur zum Kurzzeitbetrieb auf Knopfdruck).

Programmierung über USB-Anschluss.

System-Code: 3290

Konfigurationszeile	LED-Befehle	IO-Befehle
P...	Befehle für 7-SEGMENT-DISPLAY BRIGHT [0..9]*	WAITKEY, GETKEY, KEYSTATE GETRTC, SETRTC EEREAD, EEWRITE [0..1023] GETADC SYS [COM, CALIB]

\*LED.bright(val)

val: 0..9 = Display-Helligkeit einstellen. Bei Batteriebetrieb hier nur Werte bis 5 benutzen um die Batterie nicht zu stark zu belasten.

An Position 1 und 3 befinden sich Doppelpunkte und an Position 4 ein Dezimalpunkt.

IO.getadc(0)

Abfragen der Batteriespannung. Rückgabe ist ein Spannungswert in 0.01 Volt Auflösung. 290 entspricht 2.90V. Hier kann z.B. eine Warnmeldung auf dem Display angezeigt werden, wenn die Batterie nahezu erschöpft ist. Beachten Sie, dass dieser Wert aufgrund der unterschiedlichen Anzahl der leuchtenden Segmente schwankt und eventuell mehrfach gelesen werden muss.

<VAR> = IO.sys(100, 0)

Abfragen, ob Betrieb mit Batterie (VAR = 0) oder über den USB-Anschluss (VAR = 1) erfolgt

IO.sys(101, 0)

Versetzt die Uhr in den Schlafmodus. Der Microcontroller wird in einen extremen Stromsparmodus versetzt (Stromaufnahme ca. 3µA), die Anzeige wird abgeschaltet, die Ausführung des Programmes wird beendet. Nur der Quarzoszillator für die Uhrzeit ist noch in Betrieb. Aufwecken ist nur durch Druck auf Taste 1 möglich.

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

LED-Tube-Clock

**DEMNÄCHST LIEFERBAR**

*Status: Entwicklung abgeschlossen, warten auf Produktion.*

### BILDER FOLGEN

8-stellige Uhr mit 7-Segment LED-Anzeigen im VFD-Tube-Design. Terminal Print- und Fehler-Ausgabe über USB. Integrierte Echtzeituhr mit Stützbatterie. DCF-77 Anschluss mit Klinkebuchse. 4 Taster, Beeper, Stromversorgung über USB-Anschluss, Stromaufnahme ca. 30mA.

Programmierung über USB-Anschluss.

System-Code: 3300

Konfigurationszeile	LED-Befehle	IO-Befehle
P...	Befehle für 7-SEGMENT-DISPLAY BRIGHT [0..15]	WAITKEY, GETKEY, KEYSTATE GETRTC, SETRTC BEEP EEREAD, EEWRITE [0..15] SYS [COM, CALIB]

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

LED-Nixie-4

**DEMNÄCHST LIEFERBAR**

*Status: In der Entwicklung.*

### BILDER FOLGEN

4-stellige LED-Nixie Uhr. Terminal Print- und Fehler-Ausgabe über USB. Integrierte Echtzeituhr mit Stützbatterie. DCF-77 Anschluss mit Klinkebuchse. Drehimpulsgeber, 1 Taster, Beeper, LDR und Anschluss für DS18B20 Temperatursensor. Stromversorgung über USB-Anschluss.

Programmierung über USB-Anschluss.

System-Code: 3320

Konfigurationszeile	LED-Befehle	IO-Befehle
P... M... S... L... F... Lxxx: Maximum = 64	Alle Befehle für RGB-LEDs mit PWM	WAITKEY, GETKEY, KEYSTATE GETRTC, SETRTC BEEP GETLDR GETTEMP EEREAD, EEWRITE [0..1023[ SYS [COM, CALIB, DFP]

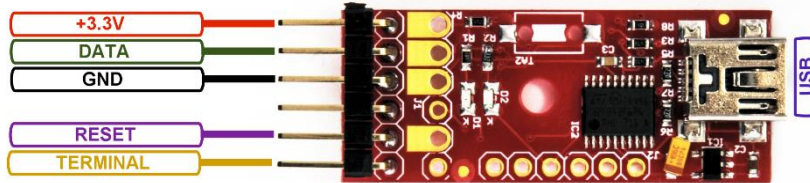
# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

### Zubehör, Erweiterungen

Prog-SB, Seriell-Basic Programmier- und Terminaladapter



LED-Basic Komponenten (z.B. Button12, Button16, Lauflicht oder Basic-Booster), die über keinen eigenen USB-Port verfügen, benötigen diesen Programmieradapter. Mit diesem sind dieselben Funktionen möglich, die auch bei Komponenten mit eigenem USB-Port möglich sind: Aufspielen des LED-Basic-Codes, eventuelles Bios-Update sowie Print- und Fehlerausgaben über das Terminal (nicht bei Button12).

Prog-SB gibt es als einzelnen Programmieradapter mit Stiftleisten zum Verbinden der LED-Basic -Komponente mit Jumperkabeln oder als Programmierzange, mit der die LED-Basic-Komponente mittels Federkontakten verbunden wird.

Bei der Verbindung mit Jumperkabeln benötigen Sie 4 oder 5 Verbindungen. Zum hochladen des LED-Basic-Codes und zum Bios-Update werden 4 Leitungen benötigt:

- **(PIN1) +3,3V**
- **(PIN2) DATA**
- **(PIN3) GND**
- **(PIN5) RESET**

(Pin 4 und 6 sind nicht beschaltet)

Wenn die Print- und Fehlerausgabe über das Terminal benutzt werden soll, muss zusätzlich noch PIN6 angeschlossen werden (bei Button12 ohne Funktion).

- **(PIN6) TERMINAL**

(Pin 4 ist nicht beschaltet)

Die LED-Basic Komponenten werden über den Programmieradapter mit Strom versorgt, bitte entfernen Sie vor der Programmierung die eventuell eingesteckte Batterie.

Hinweise:

- TA2 wird auf der Prog-SB-Platine für die Seriell-Basic-Funktion nicht benötigt und ist aus diesem Grund nicht vorhanden.
- Bei der Programmierung können auch alle 6 Leitungen verbunden werden, die unbenutzten Leitungen beeinflussen die Programmierung nicht.

Installieren Sie den Treiber, der sich im LED-Basic Installationspaket befindet, unter Windows 7 und 8.x, wie im Kapitel „[Treiberinstallation](#)“ beschrieben. Für Windows 10 ist keine Treiberinstallation erforderlich.

# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

DCF77-Zeitmodul

**DEMNÄCHST LIEFERBAR**

*Status: Warten auf Produktion.*

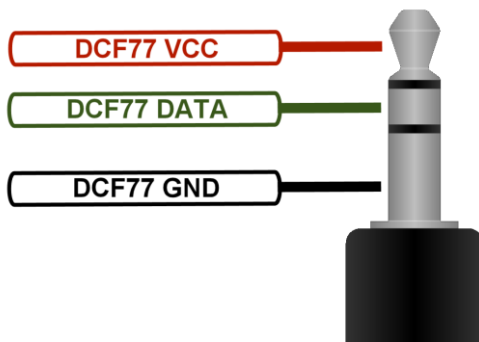
### BILDER FOLGEN

Dieses DCF77-Modul ist für alle LED-Basic-Komponenten mit Uhr-Funktion geeignet, die eine Bios-Version besitzen, in der DCF77 unterstützt wird.

LEDs mit integrierter PWM (z.B. WS2812, APA102) stören den DCF77-Empfang. Platzieren Sie den Empfänger möglichst weit entfernt von diesen LEDs. Die LED im DCF77-Empfängermodul muss im Sekundenrhythmus blinken und darf nicht flackern.

Wenn das Empfangssignal sauber ist, sollte sich die LED-Basic-Komponente nach maximal 3 Minuten auf die aktuelle Zeit synchronisieren.

- Ausgangssignal: positive Pulse in Höhe der Betriebsspannung (kein Pullup-Widerstand erforderlich)
- LED-Anzeige zur Empfangskontrolle
- Stromversorgung und Daten über 3,5mm Stereo-Klinkenbuchse
- Spannungsversorgung: 1,5 - 3,5V (ca. 50 $\mu$ A)



# LED-BASIC

## Komponenten, Editor, Befehlssatz

---

GPS -> DCF-Zeitmodul

**DEMNÄCHST LIEFERBAR**

*Status: Entwicklung abgeschlossen, warten auf Produktion*

### BILDER FOLGEN

Wenn ein DCF77-Empfänger kein Signal empfängt, weil z.B. die nähere Umgebung mit zu viel Störstrahlung verseucht ist, kann dieses Modul helfen. Sobald freie Sicht zu den GPS-Satelliten existiert, empfängt dieses Modul die genaue über GPS übertragene GMT-Zeit und wandelt diese in ein DCF77-kompatibles Signal um.

- Ausgangssignal: positive oder negative Pulse in Höhe der Betriebsspannung (umschaltbar)
- Zeitdifferenz zu GMT maximal plus/minus 7 Stunden einstellbar
- Automatische Sommer- und Winterzeiterkennung zuschaltbar
- 2 LEDs zur Empfangs- und Datenkontrolle
- Stromversorgung und Daten über 3,5mm Stereo-Klinkenbuchse (wie beim DCF77-Modul)
- Spannungsversorgung: 3,3V (ca. 60mA)

Beachten Sie bitte, dass es 3 bis 5 Minuten dauern kann, bis das Modul einen sauberen GPS-Empfang hat und die Empfangs-LED konstant leuchtet.

WLAN -> DCF-Zeitmodul

**DEMNÄCHST LIEFERBAR**

*Status: In Entwicklung*

### BILDER FOLGEN

Wenn DCF77 und GPS keine Chance haben, dann vielleicht dieses Modul. Über WLAN wird die aktuelle Zeit vom NTP-Server empfangen und in ein kompatibles DCF77-Signal umgewandelt. Zur Konfiguration des WLAN-Zuganges ist ein Smartphone mit WLAN und Web-Browser erforderlich.

- Ausgangssignal: positive oder negative Pulse in Höhe der Betriebsspannung (umschaltbar)
- LED zur Datenkontrolle
- Stromversorgung und Daten über 3,5mm Stereo-Klinkenbuchse (wie beim DCF77-Modul)
- Spannungsversorgung: 3,3V (ca. ??? mA)

# LED-BASIC

Komponenten, Editor, Befehlssatz

---

## Versionen

Hier finden Sie eine Auflistung aller erschienenen LED-Basic-Versionen. Diese Dokumentation wird bei jeder Version aktualisiert und wird deshalb nicht extra erwähnt.

### V15.1.15

Neue Komponenten:

LED-Tube, die LED-Uhr im VFD-Tube-Design.

LED-Nixie-4, 4-stellige LED-Nixie-Uhr

Update: Echtzeituhr-Kalibrierung über IO.sys-Befehl bei Cronios-Controller.

Fehler bei IO.setenc beseitigt.

### V15.1.14

Update: 16-Kanal-Lauflicht: Bug im Bios beseitigt. Anschlusspins für Prog-SB hinzugefügt.

APA-Booster: Bilder hinzugefügt und aktualisiert.

Neue Komponenten:

VFD-Clock, die Nostalgie-Uhr mit VFD-Röhren.

Touch-Lamp, die Effekt-Lampe mit Sensor-Tasten und Fernbedienung.

6-O-Clock, die 6-stellige LED-Uhr.

Nano-Stick, der Mini-USB-Stick mit 5 RGB-LEDs und programmierbarer Intelligenz.

Neuer Befehl: IO.sys(a, b) zum Lesen und Setzen von Systemparametern, bei einigen neuen Komponenten vorhanden.

### V15.1.13

Nicht veröffentlichte Testversion.

### V15.1.12

Update: Neue Version des 16-Kanal Lauflicht. Neue und aktualisierte Beispieldateien Cortex-Clock, Cronixie, Lauflicht

Neu: Unterstützung des DCF77-Moduls zur Zeitsynchronisierung bei Chronios1 (andere Komponenten folgen).

### V15.1.11

Update: Cortex-Clock und Cronios-Segmenta besitzen jetzt eine EEPROM-Funktion. Damit können Helligkeiten, Farben und Alarmzeiten fest abgespeichert werden.

Neu und Korrektur: Anschlussbild LED-Box hinzugefügt. Das EEPROM in der LED-Box hat 8 Speicherstellen (0..7).

Änderungen: Button 12 und Button 16 heißen jetzt Badge 12 und Badge 16 (Badge = Abzeichen, Sticker, Plakette)

### V15.1.10

Bugfix: In Data-Zeilen konnten keine negativen Werte benutzt werden.

### V15.1.9

Bugfix: REM nach einem LABEL erzeugte einen Laufzeitfehler

Neu: APA-Booster hinzugefügt.

Update: Neuer Konfigurationswert **Ax** zum Einstellen der Bitrate bei APA102-Leds.



# LED-BASIC

Komponenten, Editor, Befehlssatz

---

## V15.1.8

Neu: LED-Matrix 10x10, All-In-One Cortex-M4-Power-Board und Led-Box hinzugefügt.

Update: Cronios1: EEPROM-Lesen/Schreiben hinzugefügt (1024 Adressen).

## V15.1.7

Bugfix: Fehler im Lauflicht-Bios beseitigt.

## V15.1.6

Neu: Budget-Board, Cortex-Clock, Temperatur-Sensor-Interface und Lauflicht hinzugefügt.

## V15.1.5

Erste veröffentlichte Version

# LED-BASIC

Komponenten, Editor, Befehlssatz

---

## Fehler, Bugs

Hier folgt eine Auflistung bekannter Fehler.

### V15.1.12

Fehler im neuen Bios des 16-Kanal Lauflichtes. Bitte auf V15.1.14 updaten.

### V15.1.11

Wird in einem if-Statement eine IO-Funktion mit Parameter verwendet, erzeugt dieser einen Laufzeitfehler 11.

#### **Beispiel:**

```
if IO.eeread(0) <> 10 then ...
```

#### **Lösung 1:**

```
t = IO.eeread(0)
if t <> 10 then ...
```

#### **Lösung 2:**

```
if (IO.eeread(0)) <> 10 then ...
```

# LED-BASIC

Komponenten, Editor, Befehlssatz

---

## Hinweise

© Erwin Reuß, Folker Stange. Nutzung und Weitergabe dieser Informationen auch auszugsweise nur mit Erlaubnis der Copyright-Inhaber. Alle Markennamen, Warenzeichen und eingetragenen Warenzeichen sind Eigentum Ihrer rechtmäßigen Eigentümer und dienen hier nur der Beschreibung.

Als Anregung für LED-Basic diente der µBasic-Interpreter von Adam Dunkel. Aufgrund der Auslagerung des Tokenizers auf den PC mit einem selbst entwickelten „Token-Code“ sowie der Hinzufügung von LED- und IO-Routinen ist daraus ein nahezu vollständiger eigener Code entstanden.

Obwohl diese Anleitung mehrfach überarbeitet und zur Kontrolle gelesen wurde, kann sie immer noch sachliche oder grammatikalische Fehler beinhalten. Über Hinweise zu Fehlern oder Verbesserungsvorschlägen sind wir Ihnen als Anwender sehr dankbar. E-Mail: [feedback@led-basic.de](mailto:feedback@led-basic.de)

## Links

LED-Basic Homepage

<http://www.led-basic.de>

LED-Genial Online-Shop

<http://www.led-genial.de>

Diamex-Shop

<http://www.diamex.de>

µBasic von Adam Dunkel

<http://dunkels.com/adam/ubasic/>

### VERTRIEB



#### DIAMEX Produktion und Handel GmbH

Innovationspark Wuhlheide  
Köpenicker Straße 325, Haus 41  
12555 Berlin

Telefon: 030-65762631

E-Mail: [info@diamex.de](mailto:info@diamex.de)

Homepage: <http://www.diamex.de>

### HERSTELLUNG



[www.tremex.de](http://www.tremex.de)

Köpenicker Str. 325 12555 Berlin  
Tel. 030-65762631

Hersteller: Tremex GmbH  
DIAMEX × OBD-DIAG × TREMEX  
WEE-Reg.Nr. DE 51673403