# Alle Funktionen des Programm-Generators im Detail

# Lichteffekte

## Licht

### Belebtes Haus

#### House( LED,InCh, On_Min,On_Limit, ...)

Das ist vermutlich die am häufigsten genutzte Funktion auf einer Modelleisenbahn. Mit ihr wird ein „belebtes“ Haus nachgebildet. In diesem Haus sind zufällig nur einige der Räume beleuchtet. Die Farbe und die Helligkeit der Beleuchtungen können individuell vorgegeben werden. Es lassen sich auch bestimmte Effekte wie Fernseher flackern oder ein offener Kamin für einzelne Räume konfigurieren. Außerdem kann das Einschaltverhalten angepasst werden (flackern von Neonröhren oder langsam heller werdende Gaslampen). Diese beiden Effekte eignen sich u. a. gut für Neonröhren in Ladengeschäften und für Außenbeleuchtungen am Haus. Für Straßenlaternen gibt es eine eigne Funktion.

Die in einem belebten Haus verbauten WS2812B RGB-LEDs lassen sich auch temporär als Warnleuchten für Anzeigezwecke nutzen. Dazu werden sie mit der Logic-Funktion überschrieben.

#### Minimale Anzahl der zufällig aktiven Beleuchtungen

Der Parameter „On_Min“ beschreibt wie viele Räume mindestens beleuchtet sein sollen. Nach dem Einschalten werden nach einer zufälligen Zeit so lange Lichter eingeschaltet bis die vorgegebene Anzahl erreicht ist. Dabei werden auch die Zimmer zufällig bestimmt.

#### Maximale Anzahl der zufällig aktiven Beleuchtungen

Der Parameter „On_Limit“ bestimmt wie viele Räume gleichzeitig benutzt sein sollen. Wenn entsprechend viele LEDs an sind wird zum nächsten, zufällig gewählten, Zeitpunkt eine Lampe ausgeschaltet. Wenn dieser Parameter größer als die Anzahl der Räume ist, dann sind nach einiger Zeit alle Lichter an (Das entspricht unserem Zuhause). Wenn die Häuser über einen manuell betätigten Schalter Ein- und Ausgeschaltet werde, dann soll der Benutzer ein direktes Feedback beim betätigen des Schalters erkennen. Darum wird sofort beim Einschalten des Eingangs (InCh) eine Beleuchtung aktiviert und entsprechend Eine deaktiviert, wenn der Schalter Ausgeschaltet wird.

**...**

Die drei Punkte „...“ in der Makrodefinition repräsentieren die Position an der die Liste der Raumbeleuchtungen eingetragen wird. Es können bis zu 2000 Räume angegeben werden (Schloss). Die Beleuchtung der Zimmer wird mit den folgenden Konstanten festgelegt:

### Farben/Helligkeit

| Bezeichnung | Beschreibung |
|---|---|
| ROOM_DARK | Raum mit einer sehr dunklen Beleuchtung |
| ROOM_BRIGHT | Raum mit sehr heller Beleuchtung |
| ROOM_WARM_W | Dieser Raum wird in einer warm-weißen Lichtfarbe beleuchtet. |
| ROOM_RED | Dieser Raum wird in einem Licht mit sehr hohem Rot-Anteil beleuchtet. (Sonnenstudio, Sauna) |
| ROOM_D_RED | Dieser Raum wird in einem dunklerem Licht mit sehr hohem Rot-Anteil beleuchtet. (Sonnenstudio, Sauna) |
| ROOM_COL0 | Raum welcher mit der vom Benutzer definierten Farbe 0 beleuchtet wird. |
| ROOM_COL1 | Raum welcher mit der vom Benutzer definierten Farbe 1 beleuchtet wird. |
| ROOM_COL2 | Raum welcher mit der vom Benutzer definierten Farbe 2 beleuchtet wird. |
| ROOM_COL3 | Raum welcher mit der vom Benutzer definierten Farbe 3 beleuchtet wird. |
| ROOM_COL4 | Raum welcher mit der vom Benutzer definierten Farbe 4 beleuchtet wird. |
| ROOM_COL5 | Raum welcher mit der vom Benutzer definierten Farbe 5 beleuchtet wird. |
| ROOM_COL345 | Raum welcher abwechselnd und zufällig mit einer der vom Benutzer definierten Farben 3,4 oder 5 beleuchtet wird. |

ROOM_DARK, ROOM_BRIGHT und ROOM_COL0 bis ROOM_COL5 eignen sich auch, um große Räume mit mehreren Einzel-LEDs gleichmäßig auszuleuchten.

An einen WS2811 schließt man zwei oder drei LEDs **eines** Raumes an. Bei zwei LEDs muss der dritte Ausgang frei bleiben. Diesen Raum spricht man über die Funktion ROOM_DARK oder ROOM_BRIGHT an. Da der WS2811 nun wie eine RGB-LED eingesetzt wird, gehen alle drei Einzel-LEDs gleichzeitig an oder aus.

Leuchtet nun ein Fenster heller/dunkler als die anderen, können alternativ die Funktionen ROOM_COL0 bis ROOM_COL5 genutzt werden. Mit dem Set_Col_Tab kann die Helligkeit der drei Einzel-LEDs angepasst werden. Das Set_Col_Tab dient dann nicht zur Einstellung der Farbe sondern zum Anpassen der Helligkeit.

### Animierte Effekte

| Bezeichnung | Beschreibung |
|---|---|
| FIRE | Lichtschein eines offenen Kaminfeuers |
| FIRED | Lichtschein eines offenen, dunklem Kaminfeuers |

| Bezeichnung | Beschreibung |
|---|---|
| FIREB | Lichtschein eines offenen, sehr hellem Kaminfeuers |
| ROOM_CHIMNEY | zufällig Kaminfeuer oder normale Beleuchtung |
| ROOM_CHIMNEYD | zufällig dunkles Kaminfeuer oder schwache Beleuchtung |
| ROOM_CHIMNEYB | zufällig sehr helles Kaminfeuer oder starke Beleuchtung |
| ROOM_TV0 | Lichtschein vom Fernseher Kanal 0 oder einer Deckenleuchte. Der Fernseher ist über die vorgelagerte Funktion **"Farb-TV Kanal 1 einstellen"** einstellbar. |
| ROOM_TV0_CHIMNEY | Lichtschein vom Fernseher Kanal 0, einer Deckenleuchte oder einem Kaminfeuer |
| ROOM_TV0_CHIMNEYD | dunkler Lichtschein vom Fernseher Kanal 0, einer Deckenleuchte oder einem Kaminfeuer |
| ROOM_TV0_CHIMNEYB | sehr heller Lichtschein vom Fernseher Kanal 0, einer Deckenleuchte oder einem Kaminfeuer |
| ROOM_TV1 | Lichtschein vom Fernseher Kanal 1 oder einer Deckenleuchte. Der Fernseher ist über die vorgelagerte Funktion **"Farb-TV Kanal 2 einstellen"** einstellbar. |
| ROOM_TV1_CHIMNEY | Lichtschein vom Fernseher Kanal 1, einer Deckenleuchte oder einem Kaminfeuer |
| ROOM_TV1_CHIMNEYD | dunkler Lichtschein vom Fernseher Kanal 1, einer Deckenleuchte oder einem Kaminfeuer |
| ROOM_TV1_CHIMNEYB | sehr heller Lichtschein vom Fernseher Kanal 1, einer Deckenleuchte oder einem Kaminfeuer |

### Besondere Lampen

| Bezeichnung | Beschreibung |
|---|---|
| NEON_LIGHT | Simulation einer Neon-Röhre, mit dem typischen Flackern beim Starten und der sofortigen Dunkelheit beim abschalten. Alle Kanäle werden genutzt. |
| NEON_LIGHT1 | wie oben. Es wird allerdings nur Kanal 1 (ROT) des WS2811 verwendet. |
| NEON_LIGHT2 | wie oben. Es wird allerdings nur Kanal 2 (GRÜN) des WS2811 verwendet. |
| NEON_LIGHT3 | wie oben. Es wird allerdings nur Kanal 3 (BLAU) des WS2811 verwendet. |
| NEON_LIGHTD | Simulation einer dunkler leuchtenden Neon-Röhre, mit dem typischen Flackern beim Starten und der sofortigen Dunkelheit beim abschalten. Alle Kanäle werden genutzt. |
| NEON_LIGHT1D | wie oben. Es wird allerdings nur Kanal 1 (ROT) des WS2811 verwendet. |
| NEON_LIGHT2D | wie oben. Es wird allerdings nur Kanal 2 (GRÜN) des WS2811 verwendet. |
| NEON_LIGHT3D | wie oben. Es wird allerdings nur Kanal 3 (BLAU) des WS2811 verwendet. |
| NEON_LIGHTM | Simulation einer leuchtenden Neon-Röhre mit mittlerer Helligkeit, mit dem typischen Flackern beim Starten und der sofortigen Dunkelheit beim abschalten. Alle Kanäle werden genutzt. |
| NEON_LIGHT1M | wie oben. Es wird allerdings nur Kanal 1 (ROT) des WS2811 verwendet. |
| NEON_LIGHT2M | wie oben. Es wird allerdings nur Kanal 2 (GRÜN) des WS2811 verwendet. |
| NEON_LIGHT3M | wie oben. Es wird allerdings nur Kanal 3 (BLAU) des WS2811 verwendet. |
| NEON_LIGHTL | Simulation mehrerer leuchtenden Neon-Röhre mit einer RGB-LED, mit dem typischen Flackern beim Starten und der sofortigen Dunkelheit beim abschalten. Alle Kanäle werden genutzt. |
| NEON_LIGHT1L | wie oben. Es wird allerdings nur Kanal 1 (ROT) des WS2811 verwendet. |
| NEON_LIGHT2L | wie oben. Es wird allerdings nur Kanal 2 (GRÜN) des WS2811 verwendet. |
| NEON_LIGHT2L | wie oben. Es wird allerdings nur Kanal 3 (BLAU) des WS2811 verwendet. |
| NEON_DEF_D | Simulation einer defekten Neon-Röhre mit einer RGB-LED. Alle Kanäle werden genutzt. |

| Bezeichnung | Beschreibung |
|---|---|
| NEON_DEF_D1 | wie oben. Es wird allerdings nur Kanal 1 (ROT) des WS2811 verwendet. |
| NEON_DEF_D2 | wie oben. Es wird allerdings nur Kanal 2 (GRÜN) des WS2811 verwendet. |
| NEON_DEF_D3 | wie oben. Es wird allerdings nur Kanal 3 (BLAU) des WS2811 verwendet. |
| CANDLE | Simulation einer Kerze mit einer RGB-LED. Alle Kanäle werden genutzt. Farbe und Helligkeit sind über die vorgelagerte Funktion **"Kerzen einstellen"** einstellbar. |
| CANDLE1 | wie oben. Es wird allerdings nur Kanal 1 (ROT) des WS2811 verwendet. |
| CANDLE2 | wie oben. Es wird allerdings nur Kanal 2 (GRÜN) des WS2811 verwendet. |
| CANDLE3 | wie oben. Es wird allerdings nur Kanal 3 (BLAU) des WS2811 verwendet. |
| SINGLE_LED1 | Einzel-LED ohne Start-Effekt an Kanal 1 (ROT) des WS2811. |
| SINGLE_LED2 | Einzel-LED ohne Start-Effekt an Kanal 2 (GRÜN) des WS2811. |
| SINGLE_LED3 | Einzel-LED ohne Start-Effekt an Kanal 3 (BLAU) des WS2811. |
| SINGLE_LED1D | Dunkle Einzel-LED ohne Start-Effekt an Kanal 1 (ROT) des WS2811. |
| SINGLE_LED2D | Dunkle Einzel-LED ohne Start-Effekt an Kanal 2 (GRÜN) des WS2811. |
| SINGLE_LED3D | Dunkle Einzel-LED ohne Start-Effekt an Kanal 3 (BLAU) des WS2811. |
| GAS_LIGHT | Simulation einer Straßenlaterne, welche mit Gas betrieben wurde. Alle Kanäle werden genutzt. |
| GAS_LIGHT1 | wie oben. Es wird allerdings nur Kanal 1 (ROT) des WS2811 verwendet. |
| GAS_LIGHT2 | wie oben. Es wird allerdings nur Kanal 2 (GRÜN) des WS2811 verwendet. |
| GAS_LIGHT3 | wie oben. Es wird allerdings nur Kanal 3 (BLAU) des WS2811 verwendet. |
| GAS_LIGHTD | Simulation einer dunkleren Straßenlaterne, welche mit Gas betrieben wurde. Alle Kanäle werden genutzt. |
| GAS_LIGHT1D | wie oben. Es wird allerdings nur Kanal 1 (ROT) des WS2811 verwendet. |
| GAS_LIGHT2D | wie oben. Es wird allerdings nur Kanal 2 (GRÜN) des WS2811 verwendet. |
| GAS_LIGHT3D | wie oben. Es wird allerdings nur Kanal 3 (BLAU) des WS2811 verwendet. |

#### Nicht verwendeter Raum

| Bezeichnung | Beschreibung |
|---|---|
| SKIP_ROOM | Diese LED wird nicht angesteuert sondern übersprungen. Dies kann verwendet werden um einen Shop der sich in einem Gebäude in der Mitte der LED-Kette befindet separat anzusteuern, in dem die Funktion (Next_Led()) verwendet wird. |

## Belebtes Haus mit individuellen Schaltzeiten

#### HouseT(Led, Inch, On_Min, On_Limit, T_Min, T_Max, ... )

Das belebte Haus wie zuvor beschrieben mit der Möglichkeit abweichende Schaltzeiten anzugeben. Damit ist es auch möglich ein Haus immer „komplett“ anzuschalten, indem man `On_Min` und `On_Limit` auf die Anzahl der LEDs und `T_Min` sowie `T_Max0` auf 0 setzt.

## Straßenlaternen

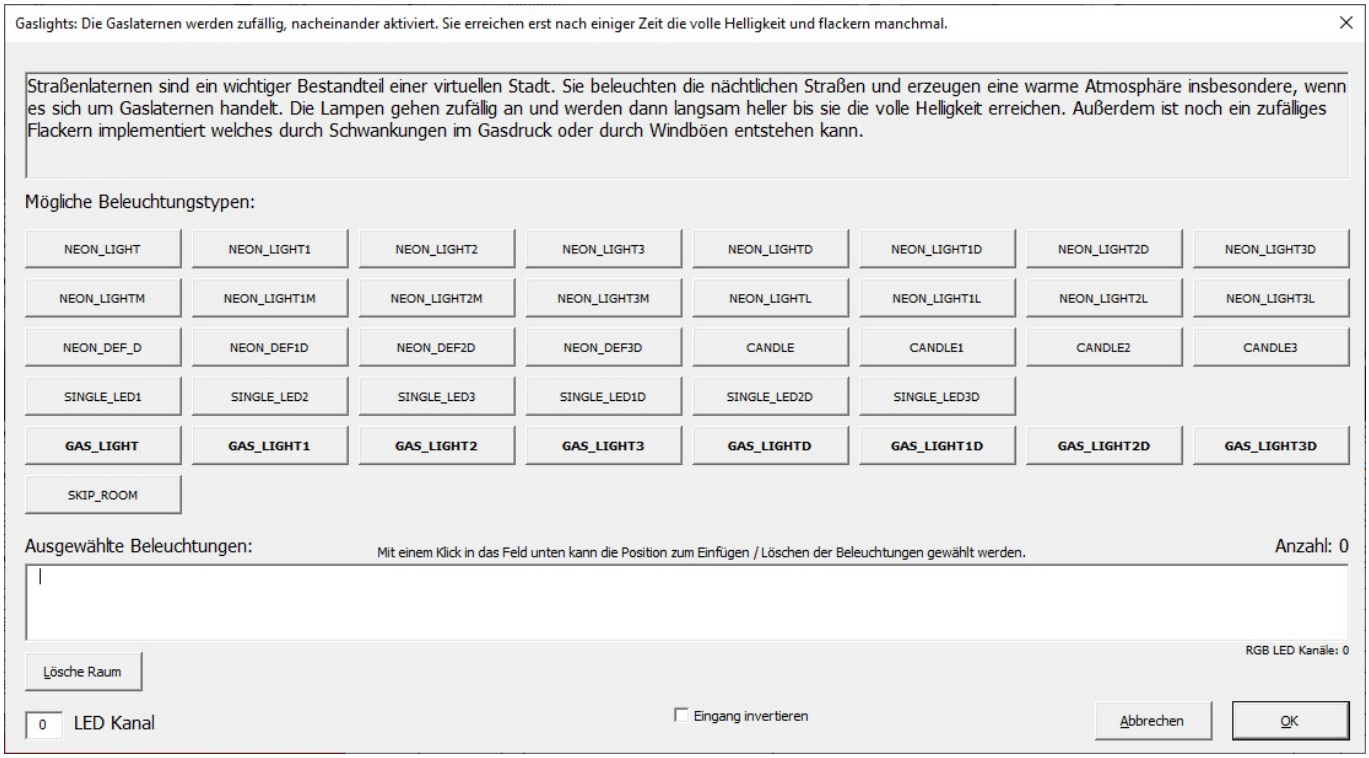
#### GasLights(Led, Inch, ...)

Die Funktion **Straßenlaternen** kann zum Ansteuern von Gaslaternen und Neonleuchten verwendet werden. Die Straßenlaternen in einem Straßenzug schalten sich nacheinander zufällig ein. Dabei wird das Einschaltverhalten gasbetriebener Lampen bzw. von Leuchtstoffröhren simuliert. Die Lampen

starten vorbildgerecht langsam.
Im Gegensatz zum „belebten Haus“ verzichtet die Funktion „Straßenlaternen“ auf minimale und maximale Anzahl der zufällig aktiven Beleuchtungen, weil innerhalb eines Straßenzuges immer alle Lampen an- und ausgehen.

### Einschalteffekte

Die Einschalteffekte sind dieselben wie bei den „besonderen Lampen“ des „belebten Hauses“.




## LED einstellbar

### Const(LED,Cx,InCh,Val0, Val1)

Definiert eine LED welche, gesteuert von “InCh”, dauerhaft An oder Aus ist.
Mit der Funktion können einzelne LEDs oder bis zu drei LEDs geschaltet werden.
Die Helligkeit im Zustand „Aus“ und „An“ kann separat eingestellt werden.

## RGB-LED einstellbar

### ConstRGB(Led, InCh, R0, G0, B0, R1, G1, B1)

Definiert eine RGB LED welche, gesteuert von “InCh”, dauerhaft An oder Aus ist.
Mit der Funktion kann eine RGB Led geschaltet werden.
Die Helligkeit und der Farbton können im Zustand „Aus“ und „An“ kann separat eingestellt werden.

Selbstverständlich kann die Funktion „RGB-LED einstellbar“

auch zum Steuern von WS2811-Chips mit angeschlossenen Einzel-LEDs genutzt werden. Das ist zum Beispiel von Vorteil, wenn man ein parkendes Auto mit Standlicht ausgerüstet hat. In dem Fall kann man die beiden weißen LEDs an die Ausgänge „Rot“ und „Grün“ eines WS2811 anschließen und die beiden roten Standlichter in Reihe an den Kanal „Blau“ (bei 5 Volt Spannungsversorgung). Diesen WS2811 kann man nun mit einer Programmzeile einstellen, obwohl die Kanäle „Rot“ und „Grün“ eine andere Helligkeit benötigen als der Kanal „Blau“.

## Heartbeat LED

### RGB_Heartbeat

Der Heartbeat ist sozusagen der Herzschlag des Programms. Er wird zur Funktionskontrolle verwendet. Auf der Hauptplatine befindet sich eine RGB Led, die normalerweise mit diesem Effekt angesteuert wird. Die Led blinkt und wechselt dabei langsam die Farbe. Daran kann man erkennen, ob die Kommunikation funktioniert. Auch die zweite Led auf der Hauptplatine sollte mit diesem Effekt angesteuert werden. Die zweite Led ist die letzte Led in der Kette. Dazu muss am Ende der Konfiguration noch einmal dieses Makro eingefügt werden. So kann man erkennen, ob die Led Kette irgendwo unterbrochen ist.

## Heartbeat LED einstellbar

### RGB_Heartbeat2

Beim RGB_Heartbeat2 können die minimale und maximale Helligkeit angegeben werden. Ansonsten verhält sich der Effekt gleich wie der RGB_Heartbeat

## Heartbeat LED einstellbare Farbe

### RGB_Heartbeat_Color

RGB LED welche als Funktionsindikator in wechselnden Regenbogenfarben blinkt. Minimale und maximale Helligkeit kann angegeben werden. Ansonsten verhält sich der Effekt gleich wie der RGB_Heartbeat

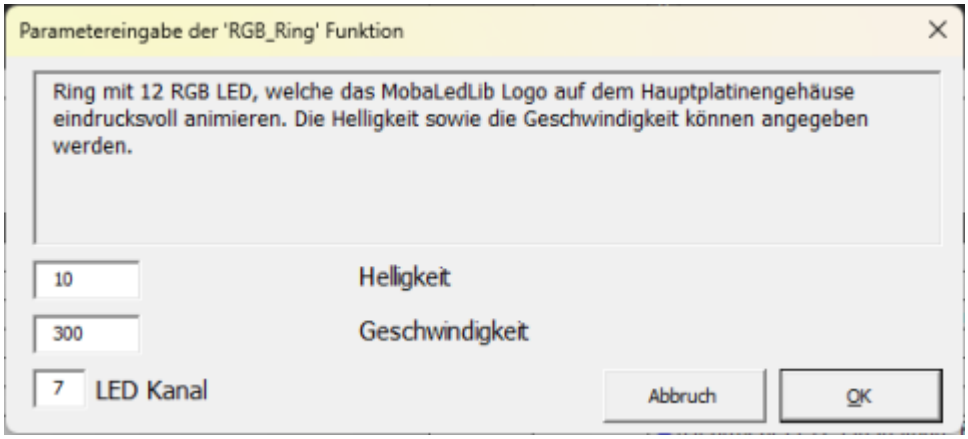
## MobaLedLib Farbring

### RGB_Ring

Ring mit 12 RGB LED, welche das MobaLedLib Logo auf dem Hauptplatinengehäuse eindrucksvoll animieren. Die Helligkeit sowie die Geschwindigkeit können angegeben werden.

Bei der LichtMaschine Pro wird der MobaLedLib-Farbring standardmäßig an Kanal 7 betrieben, dieser

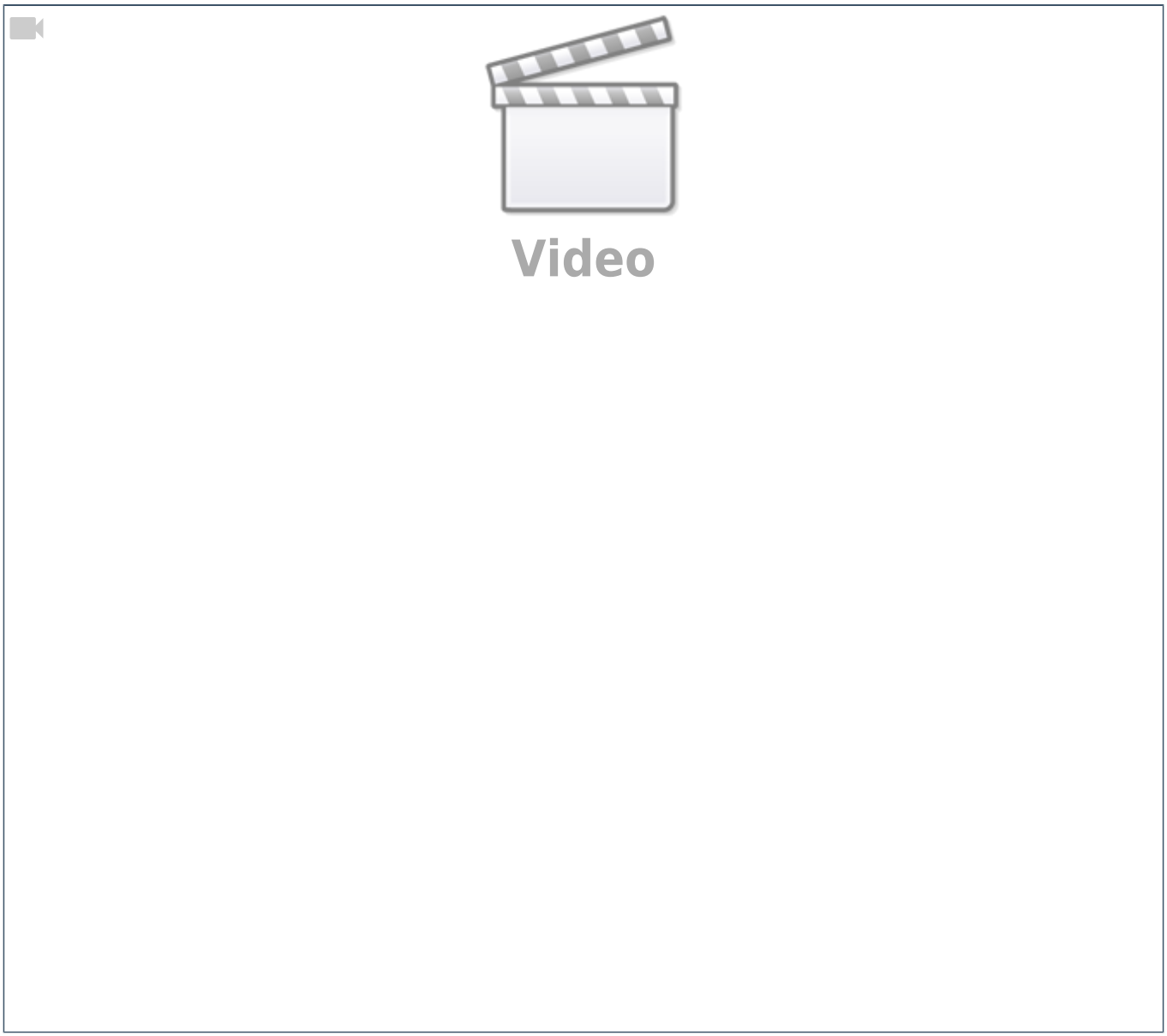
ist im Macro voreingestellt.




## Leuchtfeuer

### Leuchtfeuer

Mit dem Effekt Leuchtfeuer kann das typische Blinken eines Windrads nachempfunden werden. Das Licht ist zunächst eine Sekunde an, dann eine halbe Sekunde aus, wieder eine Sekunde an und dann 1.5 Sekunden aus. Danach fängt es wieder von vorne an.
Hier zeigt Hardi diese Funktion in einem Video:




## Blitzlicht

### Flash

Die Flash() Funktion erzeugt ein zufälliges Blitzen eines Fotografen. Über die Parameter „MinTime“ und „MaxTime“ wird bestimmt, wie häufig der Blitz ausgelöst wird. Der erste Parameter bestimmt, wie lange mindestens bis zum nächsten Blitz gewartet wird, der zweite Parameter bestimmt, wie lange maximal gewartet wird. Zwischen diesen beiden Zeiten bestimmt die Bibliothek einen zufälligen Zeitpunkt.

## Feuer

### Fire

Mit der „Fire()“ Funktion können größere Feuer simuliert werden. Dazu werden mehrere RGB LEDs verwendet welche an unterschiedlichen Stellen des „Feuers“ leuchten. Es sind maximal 100 LEDs möglich.

### Defekte Neonlampe

#### Def_Neon_Misha

Simulation eines defekten Neonlicht von Misha. Für diese Simulation wurde der Pattern_Configurator benutzt.

## Ampel

### Ampel

#### AmpelX

Mit diesem Effekt werden 2 Ampeln für eine Kreuzung simuliert. Es werden 6 einzelne LEDs verwendet, die an zwei WS2811 Modulen angeschlossen sind. Dies ist aber nur ein Beispiel einer möglichen Ampelanlage. Mit dem Pattern Configurator können beliebige Ampelanlagen erzeugt werden.

### Ampel RGB

#### RGB_AmpelX

Mit diesem Effekt werden 2 Ampeln für eine Kreuzung simuliert. Es werden 6 RGB LEDs verwendet. Dies ist aber nur ein Beispiel einer möglichen Ampelanlage. Mit dem Pattern Configurator können beliebige Ampelanlagen erzeugt werden.

### Ampel RGB soft

** RGB_AmpelXFade**

Mit diesem Effekt werden 2 Ampeln für eine Kreuzung simuliert. Es werden 6 RGB Leds verwendet. Dabei wird das langsame Auf- und Abblenden der LEDs gezeigt. Dies ist aber nur ein Beispiel einer möglichen Ampelanlage. Mit dem Pattern Configurator können beliebige Ampelanlagen erzeugt werden.

### Ampel RGB Österreich

** RGB_AmpelXA**

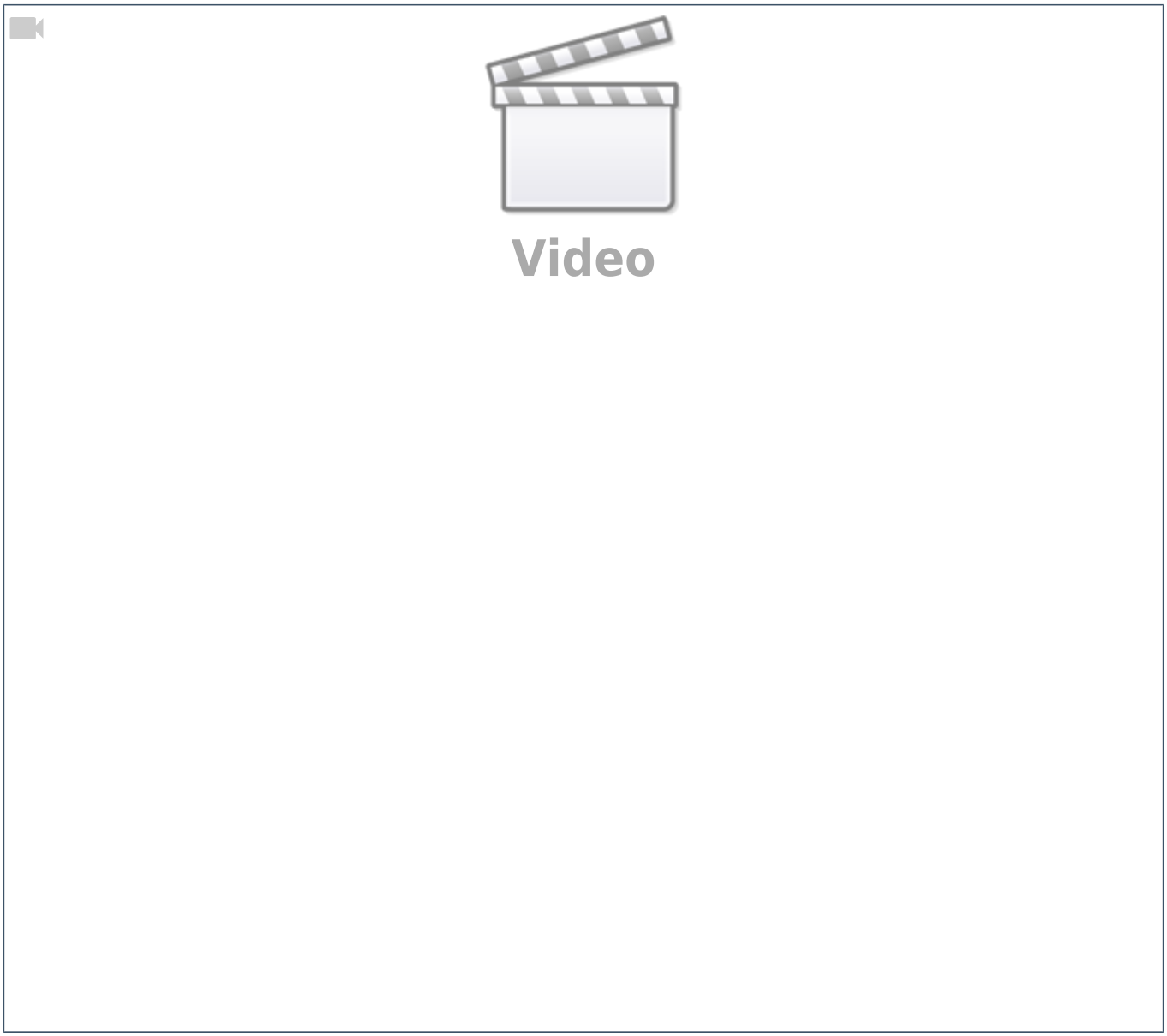
Mit diesem Effekt werden 2 Ampeln für eine Kreuzung simuliert. Es werden 6 RGB Leds verwendet. Dabei wird das langsame Auf- und Abblenden der LEDs gezeigt. Dies ist aber nur ein Beispiel einer möglichen Ampelanlage. In Österreich blinkt die grüne Lampe 4 mal an Ende. (Siehe https://www.jusline.at/gesetz/stvo/paragraf/38). Mit dem Pattern_Configurator können beliebige Ampelanlagen definiert werden.

# Andreaskreuz

## Andreaskreuz

### Andreaskreuz

Diese Funktion kann zur Ansteuerung der abwechselnd blinkenden Lampen in einem Andreaskreuz verwendet werden. Der erste Kanal kann eingestellt werden (1, 2 oder 3). Die Lampen blenden vorbildgerecht langsam auf und ab.
Hier zeigt Hardi den Effekt in einem Video:




## Andreaskreuz RGB

### AndreaskrRGB

Der Effekt Andreaskreuz RGB hat die selbe Funktion wie der Effekt Andreaskreuz, allerdings werden RGB LEDs verwendet. Das ist für Testzwecke sinnvoll. So kann man den Effekt z.B. auf einer 64 LED

Matrix testen.

### Andreaskreuz Lampentest

#### AndreaskrLT

Bei diesem besonderen Andreaskreuz werden am Anfang zunächst beide Lampen eingeschaltet. Danach blinken die Lampen abwechselnd.

### Andreaskreuz Lampentest RGB

#### AndreaskrLT_RGB

Bei diesem besonderen Andreaskreuz werden am Anfang zunächst beide Lampen eingeschaltet. Danach blinken die Lampen abwechselnd. Zu Testzwecken werden hier zwei RGB LEDs verwendet.

### Andreaskreuz Bü 1 Signal

#### AndreaskrLT3

Bei diesem besonderen Andreaskreuz werden am Anfang zunächst beide Lampen eingeschaltet. Danach blinken die Lampen abwechselnd. Eine dritte LED blinkt nach dem Lampentest, welche dem Zugführer anzeigt, dass der Bahnübergang gesichert ist.

### Andreaskreuz Bü 1 Signal RGB

#### AndreaskrLT3

Bei diesem besonderen Andreaskreuz werden am Anfang zunächst beide Lampen eingeschaltet. Danach blinken die Lampen abwechselnd. Eine dritte Led blinkt nach dem Lampentest, welche dem Zugführer anzeigt, dass der Bahnübergang gesichert ist. Zu Testzwecken werden hier 3 RGB LEDs verwendet.

## Baustellen-Lauflicht

### Baustellenlicht 6x

#### ConstrWarnLightRGB6

Diese Funktion generiert ein typisches Baustellen-Warnlicht. **Zu Testzwecken** werden hier 6 RGB LEDs verwendet.

### Baustellenlicht 3-15x

** ConstrWarnLight**

Diese Funktion generiert ein typisches Baustellen-Warnlicht. Es werden einzelne Leds an WS2811 Modulen verwendet. Die Anzahl der LEDs kann eingestellt werden.

## Blaulicht

### Blaulicht 1

#### BlueLight1

Der Effekt Blaulicht 1 simuliert das typische doppelte Blitzen eines Blaulichts bei Einsatzfahrzeugen.

### Blaulicht 2

#### BlueLight2

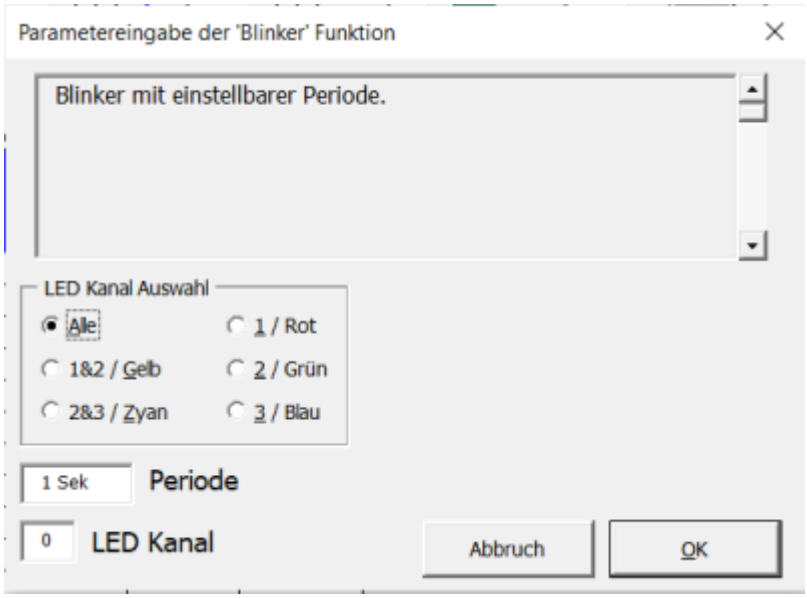
Der Effekt Blaulicht 2 simuliert wie Blaulicht 1 ein Blaulicht bei Einsatzfahrzeugen, allerdings ist die Blinkfrequenz leicht unterschiedlich. Dadurch verschieben sich die Blaulichter zueinander.

## Blinker

### Blinker

#### Blinker(Led, Cx, Inch, Period)

Diese Funktion sorgt für ein einfaches Blinken einer LED. Das Tempo kann dabei eingestellt werden. Es können entweder RGB LEDs oder normale LEDs an WS2811 Modulen verwendet werden.

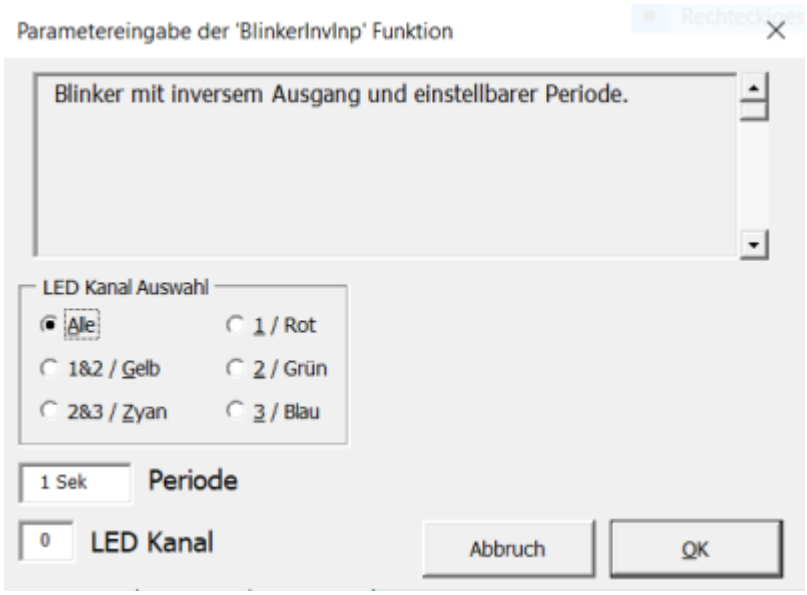



In diesem Fenster kann die Periode des Blinkers eingestellt werden. Zusätzlich kann man wählen, welche LED Kanäle verwendet werden sollen.

## Blinker invers

### BlinkerInvInp(Led, Cx, Inch, Period)

Diese Funktion sorgt für ein einfaches Blinken einer Led. Der Eingang ist dabei invertiert. Wenn man die Funktion einschaltet, ist der Blinker aus, schaltet man die Funktion aus, ist der Blinker an. Das Tempo kann dabei eingestellt werden. Es können entweder RGB Leds oder normale Leds an WS2811 Modulen verwendet werden.




In diesem Fenster kann die Periode des Blinkers eingestellt werden. Zusätzlich kann man wählen, welche Led Kanäle verwendet werden sollen.

## Blinker (minimum)

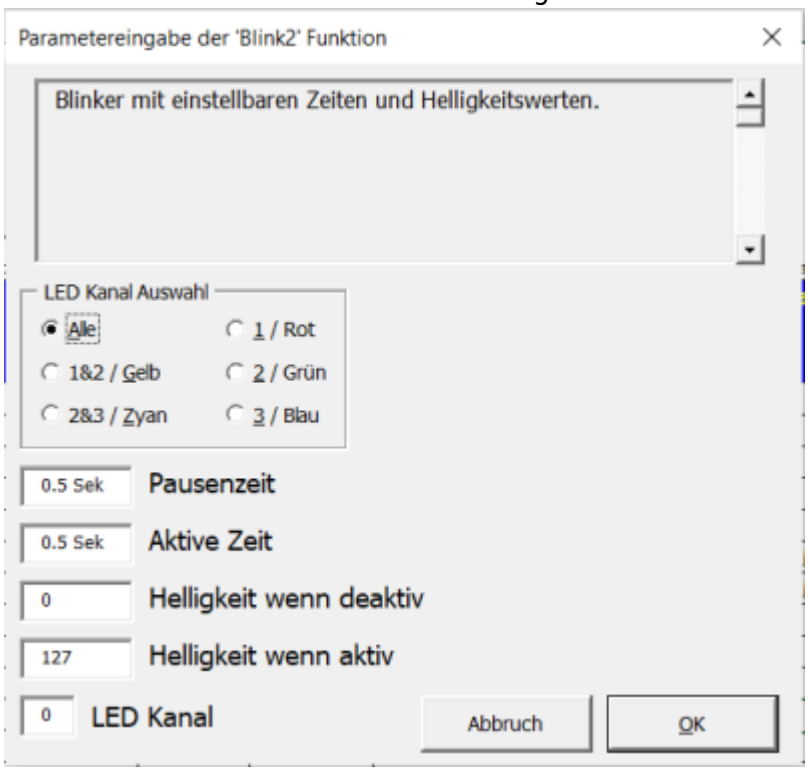
### BlinkerHD

Die Funktion BlinkerHD funktioniert wie die Funktion Blinker, allerdings gehen die LEDs nie vollständig aus.

## Blinker (Frequenz und Helligkeit)

### Blink2(LED, Cx, InCh, Pause, Act, Val0, Val1)

Bei der Funktion Blink2 können die Pausenzeit, die aktive Zeit, sowie die Helligkeitswerte bei angeschalteter und abgeschalteter LED eingestellt werden.
In diesem Fenster können die Werte eingestellt werden:




**Pausenzeit:** Hier wird die Zeit eingestellt, die die LED nicht leuchten soll.
**aktive Zeit:** Hier wird die Zeit eingestellt, die die LED leuchten soll.
**Helligkeit wenn deaktiv:** Hier wird die Helligkeit eingestellt, die die LED im ausgeschalteten Zustand haben soll.
**Helligkeit wenn aktiv:** Hier wird die Helligkeit eingestellt, die die LED im eingeschalteten Zustand haben soll.

**Anwendung KFZ-Blinker/Warnblinker:**

g
net sich einwandfrei für das Erstellen typischer KFZ-Blinker. Beim Einschalten blinkt dieser in Deutschland nach § 54 StVZO bzw. ECE-R 48 auf der jeweiligen Seite phasengleich mit einer Frequenz von 1,5 Hz ± 0,5 Hz (90 Lichterscheinungen pro Minute ± 30). Nach Betätigen des Fahrtrichtungsschalters muss das erste Aufleuchten des Blinklichtes nach spätestens einer Sekunde und das Verlöschen nach spätestens eineinhalb Sekunden erfolgen.

Der Blinkgeber muss nach ECE-R 6 so takten, dass die Hellzeit der Blinkleuchten gemessen bei 95 % der maximalen Lichtstärke mehr als 300 Millisekunden beträgt.

Um es für die MobaLedLib passend zu machen, bedeutet das:
60 Sek. / 60 Lichterscheinungen = 1 Sekunde (Abweichung Langsam)
60 Sek. / 90 Lichterscheinungen ≈ 700 ms (Intervall ideal)
60 Sek. / 120 Lichterscheinungen = 500 ms (Abweichung Schnell)

Davon muss das Licht mindestens 300 ms bei 95 % der maximalen Lichtstärke sein. Bei 500 mSek. je Lichterscheinung muss die Pause also kürzer sein als die aktive Phase.

## Blinker komplett einstellbar

### Blink3

Die Funktion „Blinker komplett einstellbar“ funktioniert wie die Funktion Frequenz und Helligkeit, allerdings kann zusätzlich die Helligkeit eingestellt werden, mit der die LED leuchten soll, wenn die Funktion abgeschaltet ist.

# Schweißlicht

## Schweißlicht dauerhaft

** WeldingCont**

Mit der Funktion „Schweißlicht dauerhaft“ kann ein Schweißlicht simuliert werden. Dieses Licht flackert solange der Eingang aktiv ist hell Weiß. Nach dem Schweißvorgang glüht die „Schweißstelle“ kurz rot nach.
Den besten Effekt erzielt man hier selbstverständlich mit einer RGB-LED.

## Schweißlicht einmalig

** Welding**

Mit der Funktion „Schweißlicht einmalig“ kann ein Schweißlicht simuliert werden. Dieses Licht flackert eine gewisse Zeit hell Weiß und verlischt dann für eine Weile. Nach dem Schweißvorgang glüht die „Schweißstelle“ kurz rot nach. Diese Funktion sollte von einer übergeordneten Funktion gesteuert werden („Der Arbeiter will ja auch mal eine Pause“).
Den besten Effekt erzielt man hier selbstverständlich mit einer RGB-LED.

## Schweißlicht zufällig

### RandWelding

Mit der Funktion „Schweißlicht zufällig“ wird das Schweißlicht zufällig gesteuert. Die Zeiten „MinTime“ und „MaxTime“ bestimmen den Zufälligen Startzeitpunkt. Über „MinOn“ und „MaxOn“ wird angegeben wie lange eine die Arbeit an einem Werkstück dauert.

# Signale

## Einfahrsignal

### EntrySignal3

Einfahrsignal mit 3 einzelnen LEDs welche über ein WS2811 Modul angesteuert werden (HP0, HP1, HP2). Es wird über drei Taster gesteuert.

## Einfahrsignal RGB

### EntrySignal3_RGB

Einfahrsignal mit 3 RGB LEDs zu Testzwecken (HP0, HP1, HP2). Es wird über drei Taster gesteuert.

### Ausfahrsignal

#### DepSignal4

Ausfahrsignal mit 6 einzelnen LEDs welche über zwei WS2811 Module angesteuert werden (HP0, HP1, HP2, HP0+SH1). Es wird über vier Taster gesteuert.

### Ausfahrsignal RGB

#### DepSignal4_RGB

Ausfahrsignal mit 6 RGB LEDs zu Testzwecken (HP0, HP1, HP2, HP0+SH1). Es wird über vier Taster gesteuert.

## KS-Signalsystem

### KS-Vorsignal Zs3V

#### KS_Vorsignal_Zs3V

### KS-Vorsignal Zs3V RGB

#### KS_Vorsignal_Zs3V_RGB

### KS-Hauptsignal Zs3 Zs1

#### KS_Hauptsignal_Zs3_Zs1

### KS-Hauptsignal Zs3 Zs1 RGB

#### KS_Hauptsignal_Zs3_Zs1_RGB

### KS-Hauptsignal Zs3 Zs6 Zs1

#### KS_Hauptsignal_Zs3_Zs6_Zs1

### KS-Hauptsignal Zs3 Zs6 Zs1 RGB

#### KS_Hauptsignal_Zs3_Zs6_Zs1_RGB

# Signale WS2812 by Matthias

## Hauptsignal links RGB

### HS_5l_RGB

## Hauptsignal rechts RGB

### HS_5r_RGB

## Haupt- und Vorsignal RGB

### HS_5_Plus_RGB

## Gleissperrsignal RGB

### Gleissperrsignal_RGB

# Signale (Trix)

## Einfahrsignal (Trix)

** EntrySignal3Bin**

Einfahrsignal mit 3 einzelnen LEDs welche über ein WS2811 Modul angesteuert werden. (HP0, HP1, HP2). Es wird über zwei binäre Eingänge gesteuert. Damit eignet es sich besonders für Selectrix.

## Einfahrsignal RGB (Trix)

** EntrySignal3Bin_RGB**

Einfahrsignal mit 3 RGB LEDs zu Testzwecken (HP0, HP1, HP2). Es wird über zwei binäre Eingänge gesteuert. Damit eignet es sich besonders für Selectrix.

## Ausfahrsignal (Trix)

### DepSignal4Bin

Ausfahrsignal mit 6 einzelnen LEDs welche über zwei WS2811 Module angesteuert werden (HP0, HP1, HP2, HP0+SH1). Es wird über zwei binäre Eingänge gesteuert. Damit eignet es sich besonders für

Selectrix.

### Ausfahrsignal RGB (Trix)

#### DepSignal4Bin_RGB

Ausfahrsignal mit 6 RGB LEDs zu Testzwecken (HP0, HP1, HP2, HP0+SH1). Es wird über zwei binäre Eingänge gesteuert. Damit eignet es sich besonders für Selectrix.

## Farbeinstellungen

### Farbe und Helligkeit einstellen

#### Set_ColTab

Mit dem Befehl Set_ColTab können die Farben der Leds individuell angepasst werden. Weitere Infos dazu findet man hier: Farbtabelle

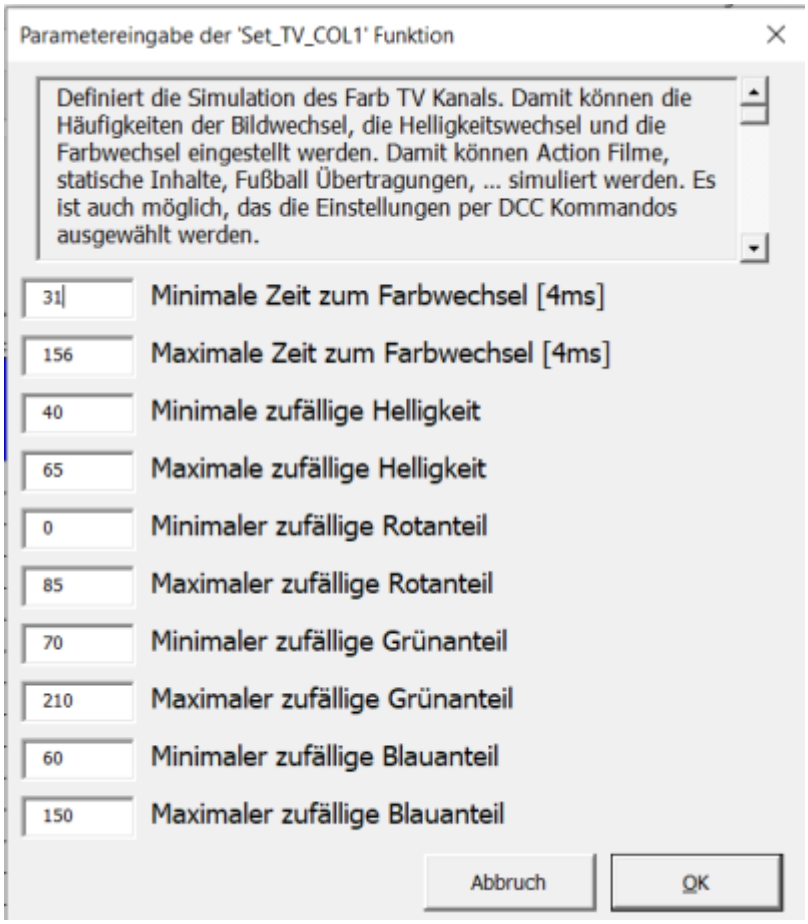
### Farb-TV Kanal 1 einstellen

#### Set_TV_COL1(InCh, Update_t_Min, Update_t_Max, Min_Brightness, Max_Brightness, R_Min, R_Max, G_Min, G_Max, B_Min, B_Max)

Der Effekt Set_TV_Col1 definiert die Parameter des TV Farbkanals 1. Damit können die Häufigkeiten der Bildwechsel, die Helligkeitswechsel und die Farbwechsel eingestellt werden. So können Action Filme, statische Inhalte, Fußball Übertragungen, … simuliert werden. Es ist auch möglich, die Einstellungen per DCC Kommando auszuwählen.

Die Funktion Set_TV_COL1 muss im Programm Generator unmittelbar vor die zu verändernde Funktion „Belebtes Haus“ gesetzt werden, um die Fernseher aller folgenden belebten Häuser zu verändern. Im Anschluss sollte direkt nach der Zeile „Belebtes Haus“ ein weiteres Set_TV_COL1 mit unveränderten Werten gesetzt werden, damit nachfolgende Häuser nicht denselben Fernsehkanal zeigen.

In diesem Fenster können die Einstellungen ausgewählt werden:



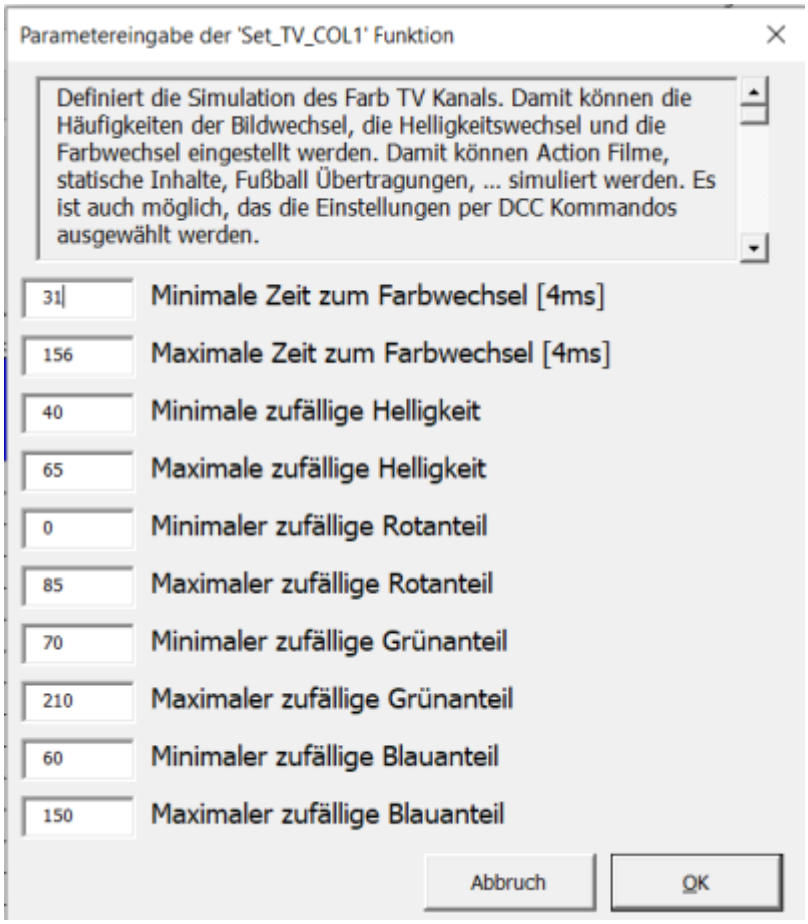

Für ein Fußballspiel sollte bspw. ein hoher Grünanteil ausgewählt werden, für einen Action Film schnelle Bildwechsel usw.

## Farb-TV Kanal 2 einstellen

**Set_TV_COL2(InCh, Update_t_Min, Update_t_Max, Min_Brightness, Max_Brightness, R_Min, R_Max, G_Min, G_Max, B_Min, B_Max)**
Der Effekt Set_TV_Col1 definiert die Parameter des TV Farbkanals 2. Damit können die Häufigkeiten der Bildwechsel, die Helligkeitswechsel und die Farbwechsel eingestellt werden. So können Action Filme, statische Inhalte, Fußball Übertragungen, … simuliert werden. Es ist auch möglich, die Einstellungen per DCC Kommando auszuwählen.

Die Funktion Set_TV_COL2 muss im Programm Generator unmittelbar vor die zu verändernde Funktion „Belebtes Haus“ gesetzt werden, um die Fernseher aller folgenden belebten Häuser zu verändern. Im Anschluss sollte direkt nach der Zeile „Belebtes Haus“ ein weiteres Set_TV_COL2 mit unveränderten Werten gesetzt werden, damit nachfolgende Häuser nicht denselben Fernsehkanal zeigen.

In diesem Fenster können die Einstellungen ausgewählt werden:




Für ein Fußballspiel sollte bspw. ein hoher Grünanteil ausgewählt werden, für einen Action Film schnelle Bildwechsel usw.

## S/W-TV Kanal 1 einstellen

### Set_TV_BW1(InCh, Update_t_Min, Update_t_Max, Min_Brightness, Max_Brightness, BW_R, BW_G, BW_B)

Der Befehl Set_TV_BW1 definiert die Parameter des Schwarz-weiß TV Kanals 1. Damit können die Häufigkeiten der Bildwechsel, die Helligkeitswechsel und die Anteile der verwendeten Farben ausgewählt werden. Damit kann das typisch bläuliche Licht eines alten Fernsehers erzeugt werden. Es ist auch möglich, die Einstellungen per DCC zu wählen. So kann zwischen Schwarz- weiß und Farbfernseher gewechselt werden.

Die Funktion Set_TV_BW1 muss im Programm Generator unmittelbar vor die zu verändernde Funktion „Belebtes Haus“ gesetzt werden, um die Fernseher aller folgenden belebten Häuser zu verändern. Im Anschluss sollte direkt nach der Zeile „Belebtes Haus“ ein weiteres Set_TV_BW1 mit unveränderten Werten gesetzt werden, damit nachfolgende Häuser nicht denselben Fernsehkanal zeigen.

## S/W-TV Kanal 2 einstellen

### Set_TV_BW2(InCh, Update_t_Min, Update_t_Max, Min_Brightness, Max_Brightness, BW_R, BW_G, BW_B)

Der Befehl Set_TV_BW1 definiert die Parameter des Schwarz-weiß TV Kanals 2. Damit können die

Häufigkeiten der Bildwechsel, die Helligkeitswechsel und die Anteile der verwendeten Farben ausgewählt werden. Damit kann das typisch bläuliche Licht eines alten Fernsehers erzeugt werden. Es ist auch möglich, die Einstellungen per DCC zu wählen. So kann zwischen Schwarz- weiß und Farbfernseher gewechselt werden.

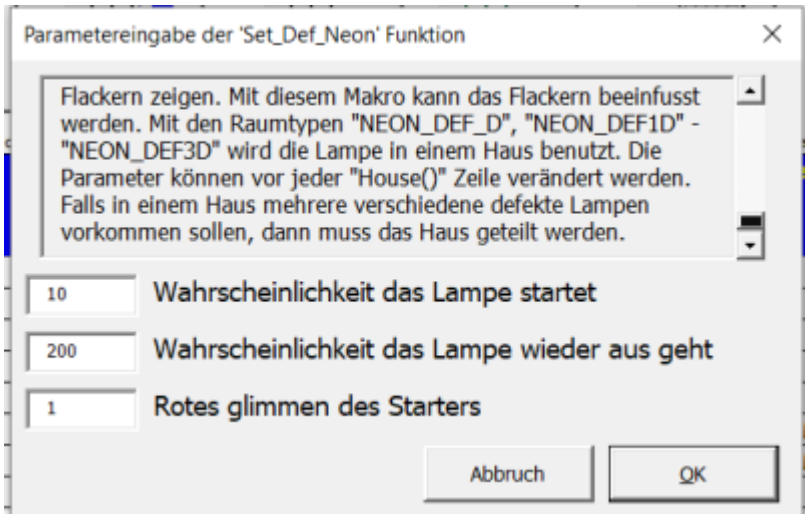
Die Funktion Set_TV_BW2 muss im Programm Generator unmittelbar vor die zu verändernde Funktion „Belebtes Haus“ gesetzt werden, um die Fernseher aller folgenden belebten Häuser zu verändern. Im Anschluss sollte direkt nach der Zeile „Belebtes Haus“ ein weiteres Set_TV_BW2 mit unveränderten Werten gesetzt werden, damit nachfolgende Häuser nicht denselben Fernsehkanal zeigen.

## Neonröhre Defekt einstellen

### Set_Def_Neon

Mit dem Befehl Set_Def_Neon können die Parameter zur Simulation einer defekten Neonröhre definiert werden. Eine defekte Neonröhre kann ganz unterschiedliches Flackern erzeugen. Mit diesem Makro kann das Flackern beeinflusst werden. Es können die Wahrscheinlichkeit, dass die Lampe startet, die Wahrscheinlichkeit, dass die Lampe wieder ausgeht und das rote Glimmen des Starters beeinflusst werden. In diesem Fenster können die Einstellungen ausgewählt werden.

Die Funktion Set_Def_Neon muss im Programm Generator unmittelbar vor die zu verändernde Funktion „Belebtes Haus“ oder „Straßenbeleuchtung“ gesetzt werden, um die defekten Neonröhren aller folgenden belebten Häuser/Straßenbeleuchtungen zu verändern. Im Anschluss sollte direkt nach der Zeile „Belebtes Haus“ oder „Straßenbeleuchtung“ ein weiteres Set_Def_Neon mit unveränderten Werten gesetzt werden, damit nachfolgende Lampen nicht dasselbe Muster zeigen.



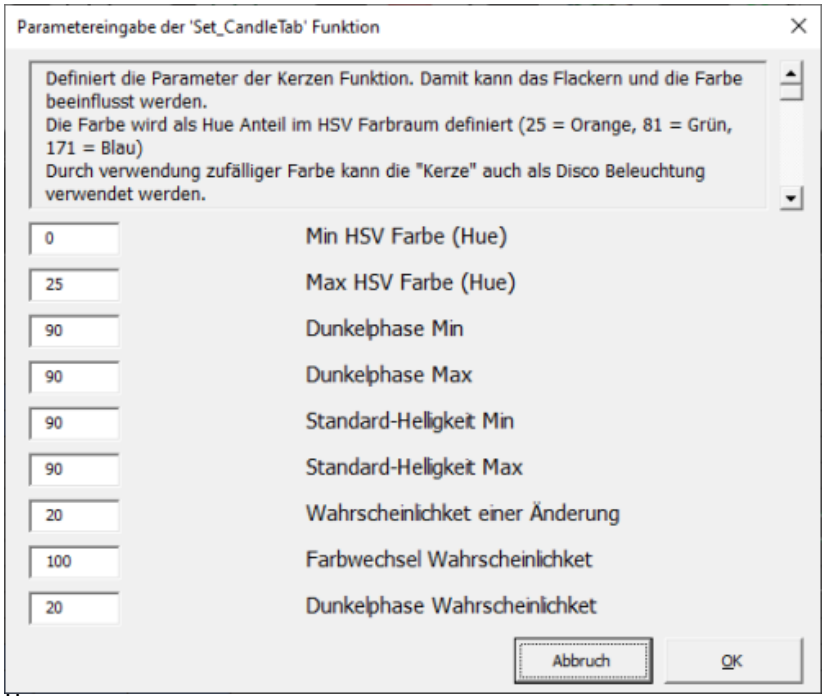

## Kerzen einstellen

### Set_CandleTab

Definiert die Parameter der Kerzen Funktion. Damit kann das Flackern und die Farbe beeinflusst werden. Die Farbe wird als Hue Anteil im HSV Farbraum definiert (25 = Orange, 81 = Grün, 171 = Blau).
Durch Verwendung zufälliger Farbe kann die „Kerze“ auch als Disco Beleuchtung verwendet werden.

Das Set_CandleTab muss im Programm Generator unmittelbar vor die zu verändernde Funktion „Belebtes Haus“ oder „Straßenbeleuchtung“ gesetzt werden. In einer dieser beiden Funktionen wählt man eine Kerze, die mit einer RGB-LED simuliert wird. Im Anschluss sollte direkt nach der Zeile „Belebtes Haus“ oder „Straßenbeleuchtung“ ein weiteres Set_CandleTab mit unveränderten Werten gesetzt werden, damit nachfolgende Kerzen nicht aussehen wie eine Disco oder wie im nachfolgenden Beispiel wie eine Glut.

| Aktiv | Filter | Adresse oder Name | Typ | Start-wert | Beschreibung | Verteiler-Nummer | Stecker-Nummer | Icon | Name | Beleuchtung, Sound, oder andere Effekte | Start LedNr | LEDs | InCnt | Loc InCh | LED/ Sound Kanal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ✓ | | | | | | | | | Heartbeat LED | RGB_Heartbeat(#LED) | 0 | 1 | 0 | 0 | 0 |
| | | | | | | | | | | | | | | | |
| ✓ | | | | | Farbeinstellungen für Glut | | | | Kerzen einstellen | Set_CandleTab(0, 25, 90, 90, 90, 90, 20, 100, 20) | | | 0 | 0 | |
| ✓ | | 1 | AnAus 0 | | Glut im Kamin | | | | Straßenbeleuchtung | GasLights(#LED, #InCh, CANDLE) | 1 | 1 | 1 | 0 | 0 |
| ✓ | | | | | Standardeinstellungen für Kerze | | | | Kerzen einstellen | Set_CandleTab(25, 25, 80, 100, 120, 145, 64, 50, 5) | | | 0 | 0 | |




mit entsprechender Anpassung auch gut als Glut in einem Grill/Kamin, da es im Grill selten lichterloh brennt. Damit es einer Glut nahekommt, muss man zunächst mit dem Set_CandleTab die „Kerze“ einstellen.

Das Geheimnis für eine ruhige Glut liegt zum einen in den Min/Max-Helligkeiten. Setzt man diese alle auf den gleichen Wert (90), kommt es der Glut deutlich näher, weil der Wechsel von hell zu dunkel entfällt. Zum anderen müssen die Wahrscheinlichkeiten für eine Helligkeitsänderung und die Dunkelphase runter, damit es ruhiger wird (weniger Wechseltätigkeit). Ein kleiner Wert (20) muss aber bleiben, damit ein klein wenig Bewegung ins Spiel kommt. Lediglich die Wahrscheinlichkeit eines Farbwechsels (100) muss drin bleiben. Die Farben werden dann im Bereich zwischen rot (0) und orange (25) gehalten.

# Dynamik

## Servo

Nach erfolgreichem **Zusammenbau** der Servo-Platine und **Programmierung des ATTiny85** stehen im Programm-Generator vier Funktionen für die Servo zur Verfügung.

### Servo mit 2 Positonen

#### Servo2

Das Servo bewegt sich nach dem Einschalten der Versorgungsspannung nicht. Erst wenn eine der Tasten betätigt wird, fährt es langsam zu der entsprechenden Position. Geschwindigkeit, Endlagen und Drehrichtung werden mit dem Servo_Pos Programm eingestellt.

Dieses Makro steuert zwei Positionen des Servos an. Die Positionen können über Taster angefahren werden. Die richtigen Positionen sollten vorher über das **Farbtest Programm** herausgefunden werden.

### Servo mit 3 Positonen

#### Servo3

Das Servo bewegt sich nach dem Einschalten der Versorgungsspannung nicht. Erst wenn eine der Tasten betätigt wird, fährt es langsam zu der entsprechenden Position. Geschwindigkeit, Endlagen und Drehrichtung werden mit dem Servo_Pos Programm eingestellt.

Dieses Makro steuert drei Positionen des Servos an. Die Positionen können über Taster angefahren werden. Die richtigen Positionen sollten vorher über das **Farbtest Programm** herausgefunden werden.

### Servo mit 4 Positonen

#### Servo4

Das Servo bewegt sich nach dem Einschalten der Versorgungsspannung nicht. Erst wenn eine der Tasten betätigt wird, fährt es langsam zu der entsprechenden Position. Geschwindigkeit, Endlagen und Drehrichtung werden mit dem Servo_Pos Programm eingestellt.

Dieses Makro steuert vier Positionen des Servos an. Die Positionen können über Taster angefahren werden. Die richtigen Positionen sollten vorher über das **Farbtest Programm** herausgefunden werden.

### Servo mit 5 Positonen

#### Servo5

Das Servo bewegt sich nach dem Einschalten der Versorgungsspannung nicht. Erst wenn eine der Tasten betätigt wird, fährt es langsam zu der entsprechenden Position. Geschwindigkeit, Endlagen und Drehrichtung werden mit dem Servo_Pos Programm eingestellt.

Dieses Makro steuert fünf Positionen des Servos an. Die Positionen können über Taster angefahren werden. Die richtigen Positionen sollten vorher über das **Farbtest Programm** herausgefunden werden.

## Herzstückpolarisierung bistabil v1.1

### Relaiskontakt A/B bis E/F

#### Herz_BiRelais

### Relaiskontakt A/B bis E/F, invers

#### Herz_BiRelais_I

## Herzstückpolarisierung bistabil v1.0

### Relaiskontakte A und B

#### Herz_BiRelais_V1_AB

### Relaiskontakte A und B, invers

#### Herz_BiRelais_I_V1_AB

### Relaiskontakte C und D

#### Herz_BiRelais_V1_CD

### Relaiskontakte C und D, invers

#### Herz_BiRelais_I_V1_CD

### Relaiskontakte E und F

#### Herz_BiRelais_V1_EF

### Relaiskontakte E und F, invers

#### Herz_BiRelais_I_V1_EF

## Herzstückpolarisierung monostabil

### Relaiskontakt A/B bis E/F

#### Herz_MoRelais

### Relaiskontakt A/B bis E/F, invers

#### Herz_MoRelais_I

### Relaiskontakt A bis F

#### Herz_2MoRelais

### Relaiskontakt A bis F, invers

#### Herz_2MoRelais_I

# Schalten

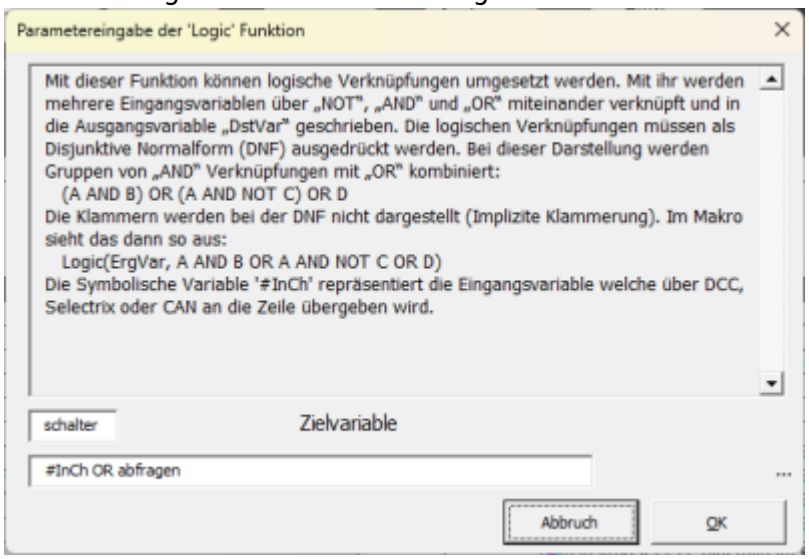
## Abhängigkeiten

### Logische Verknüpfung

#### Logic

Mit der Logic Funktion können mehrere Eingangsvariablen zu einer Ausgangsvariable verknüpft werden. So können logische Verknüpfungen umgesetzt werden. Mit der Logic Funktion werden mehrere Eingangsvariablen über „NOT“, „AND“ und „OR“ verknüpft und in die Ausgangsvariable geschrieben. Die logischen Verknüpfungen müssen als Disjunktive Normalform geschrieben werden.

Bei dieser Darstellung werden Gruppen von „AND“ Verknüpfungen mit „OR“ kombiniert. Das kann z.B. so aussehen:
A AND B OR A AND NOT C OR D
Als Beispiel dafür soll eine LED entweder zusammen mit einer anderen LED über eine vergebene Variable (z. B. über den Befehl LED-to-Var) oder über eine eigene DCC Adresse geschaltet werden. Dazu benötigt man zunächst die Logic Funktion. Wenn man diese aufruft, erscheint dieses Fenster:




Im Feld Zielvariable trägt man einen beliebigen Variablennamen ein. In diesem Beispiel wird der Name „schalter“ verwendet. Dieser wird mit der darunter eigegebenen logischen Verknüpfung aktiviert.
#InCh fragt die DCC-Adresse, den Schalter oder die Variable ab, die unter „Adresse oder Name“ im Programm Generator für die „Logische Verknüpfung“ angegeben wurde, in unserem Beispiel die DCC-Adresse 12.
Die Variable „abfragen“ ist mit einer LED-to-Var-Funktion verknüpft und wird aktiv, wenn die dort definierte Bedingung eintritt, in unserem Beispiel die rote LED.

Mit der Variable „schalter“ kann nun eine beliebige Funktion geschaltet werden. In unserem Beispiel wird ein Blinker immer zusammen mit der roten LED aktiviert oder separat über die DCC-Adresse 12.

| Aktiv | Filter | Adresse oder Name | Typ | Start wert | Beschreibung | Verteiler-Nummer | Stecker-Nummer | Icon | Name | Beleuchtung, Sound, oder andere Effekte | Start LedNr | LEDs |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ✓ | | 3 | AnAus 0 | | DCC#3 schaltet rote LED | | | | LED einstellbar | Const(#LED, C1, #InCh, 0, 127) | 0 | C1-1 |
| ✓ | | | | | Variable "abfragen" wird aktiv, wenn rote LED (DCC#3) an ist | | | | LED-Werte als Variable | LED_to_Var(abfragen, 0, >, 0) | | |
| ✓ | | 12 | AnAus 0 | | DCC#12 ODER Variable "abfragen" aktivieren "schalter" | | | | Logische Verknüpfung | Logic(schalter, #InCh OR abfragen) | | |
| ✓ | | schalter | | | Blinker wird entweder mit DCC#3 oder mit DCC#12 aktiv | | | | Blinker | Blinker(#LED, C_ALL, #InCh, 1 Sek) | 1 | 1 |

## LED-Werte kopieren

### CopyLED

Mit dem „CopyLED()“ Befehl wird die Helligkeit der drei Farben einer Quell-LED („SrcLED“) in eine andere LED („LED“) kopiert. Das ist zum Beispiel bei einer Ampel an einer Kreuzung sinnvoll. Hier sollen die gegenüberliegenden Ampeln das gleiche Bild zeigen. Ein anderes Beispiel ist ein Farbwechsel, der von vielen Flutlichtstrahlern erzeugt wird. Hier kann mit dem „CopyLED()“ Befehl auch Speicherplatz gespart werden.

Wenn zwei RGB LEDs das gleiche zeigen sollen, dann kann man das auch durch die elektrische Verkabelung erreichen. Hier ist allerdings darauf zu achten, dass nur von der Master-LED ein Datensignal zur nächsten LED führt.

## LED-Werte als Variable

### LED_to_Var

Mit dieser Funktion kann die Helligkeit anderer LEDs abgefragt und somit als Schalter genutzt werden. Das macht es beispielsweise möglich, mithilfe des Pattern Configurators einen Ablauf vorzugeben. Besonders interessant ist in dem Zusammenhang die Nutzung eines virtuellen LED Kanals, um nicht unnötig blinde WS2811 als Referenz zu nutzen.

**Zielvariable:**
Im Gegensatz zu den Zielvariablen bei der Logic-Verknüpfung o. Ä., muss hier keine Zahl hinter der Variable eingegeben werden. Hier ist man also völlig frei und kann als Variable den spezifischen Namen der gewünschten Funktion eingeben, z. B. „Heissluftballon“. Wichtig ist - wie bei allen Variablen, dass der verwendete Name einzigartig ist.

**LED Nummer Offset:**
Das aus dem englischen übersetzte Wort Offset bedeutet hier „Abstand“. Als Eselsbrücke kann man sich hier merken, dass der Abstand zu LED 0 der vorangegangenen Funktion abgefragt wird. Wenn beispielsweise ein belebtes Haus mit vier RGB-LEDs programmiert wurde, bei dem der blaue Kanal der letzten LED abgefragt werden soll, so beträgt der Abstand (Offset) 11. Der rote Kanal der ersten LED ist in dem Fall die LED 0. Die erste RGB-LED belegt die Kanäle 0-2, die zweite 3-5, die dritte 6-8 und die vierte 9-11.

**Vergleichstyp ( = != < > & !& ):**
Hier stehen gleich mehrere Operatoren zur Verfügung, mit denen man so gut wie jeden Zustand abfragen kann.

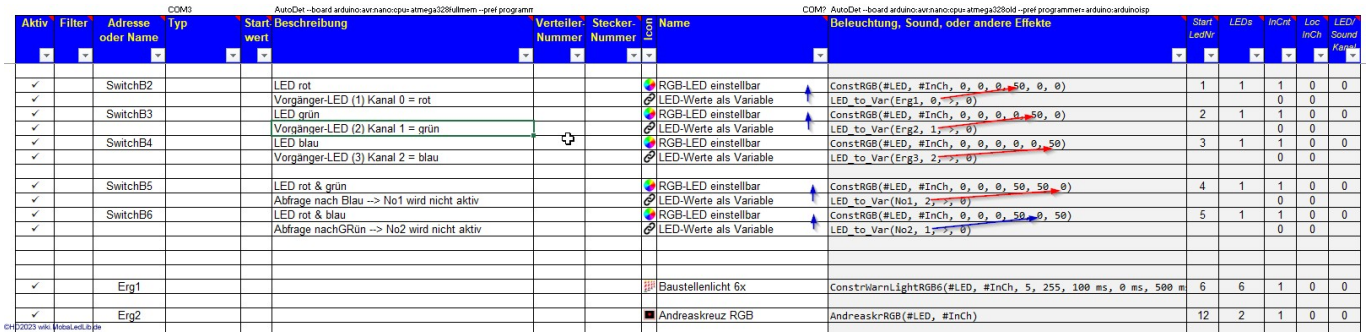
**Vergleichswert:**
Der Vergleichswert ist einfach gesagt der Helligkeitwert (0 bis 255) der LED, die durch die Funktion abgefragt werden soll.

### Zielvariable

Hier wird ein Variablen-Name angegeben, der im weiteren zur Steuerung z.B. eines Schweißlichtes verwendet wird.

### LED-Nummer Offset

Dies ist die entschiedenste und problematischste Stelle der Funktion. Die Funktion bezieht sich auf die LED aus der Zeile davor. Ist das eine RGB-LED, so ist hier der Wert 0 dem roten Kanal entsprechend. Der Wert 1 ist der Grün-Kanal und 2 der blaue Kanal. Wie gesagt von der LED davor.



| Aktiv | Filter | Adresse oder Name | Typ | Start-wert | Beschreibung | Verteiler-Nummer | Stecker-Nummer | Icon | Name | Beleuchtung, Sound, oder andere Effekte | Start LedNr | LEDs | InCnt | Loc InCh | LED/ Sound Kanal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ✓ | | SwitchB2 | | | LED rot | | | | RGB-LED einstellbar | ConstRGB(#LED, #InCh, 0, 0, 0, 50, 0, 0) | 1 | 1 | 1 | 0 | 0 |
| ✓ | | | | | Vorgänger-LED (1) Kanal 0 = rot | | | | LED-Werte als Variable | LED_to_Var(Erg1, 0, >, 0) | | | 0 | 0 | |
| ✓ | | SwitchB3 | | | LED grün | | | | RGB-LED einstellbar | ConstRGB(#LED, #InCh, 0, 0, 0, 50, 0) | 2 | 1 | 1 | 0 | 0 |
| ✓ | | | | | Vorgänger-LED (2) Kanal 1 = grün | | | | LED-Werte als Variable | LED_to_Var(Erg2, 1, >, 0) | | | 0 | 0 | |
| ✓ | | SwitchB4 | | | LED blau | | | | RGB-LED einstellbar | ConstRGB(#LED, #InCh, 0, 0, 0, 0, 0, 50) | 3 | 1 | 1 | 0 | 0 |
| ✓ | | | | | Vorgänger-LED (3) Kanal 2 = blau | | | | LED-Werte als Variable | LED_to_Var(Erg3, 2, >, 0) | | | 0 | 0 | |
| ✓ | | SwitchB5 | | | LED rot & grün | | | | RGB-LED einstellbar | ConstRGB(#LED, #InCh, 0, 0, 0, 50, 50, 0) | 4 | 1 | 1 | 0 | 0 |
| ✓ | | | | | Abfrage nach Blau --> No1 wird nicht aktiv | | | | LED-Werte als Variable | LED_to_Var(No1, 2, >, 0) | | | 0 | 0 | |
| ✓ | | SwitchB6 | | | LED rot & blau | | | | RGB-LED einstellbar | ConstRGB(#LED, #InCh, 0, 0, 0, 50, 0, 50) | 5 | 1 | 1 | 0 | 0 |
| ✓ | | | | | Abfrage nachGRün --> No2 wird nicht aktiv | | | | LED-Werte als Variable | LED_to_Var(No2, 1, >, 0) | | | 0 | 0 | |
| ✓ | | Erg1 | | | | | | | Baustellenlicht 6x | ConstrWarnLightRGB6(#LED, #InCh, 5, 255, 100 ms, 0 ms, 500 m | 6 | 6 | 1 | 0 | 0 |
| ✓ | | Erg2 | | | | | | | Andreaskreuz RGB | AndreaskrRGB(#LED, #InCh) | 12 | 2 | 1 | 0 | 0 |

Es ist aber auch möglich, LED-Zustände von LED’s zu testen, die erst nach der LED_to_Var Zeile kommen. In diesem Fall geht der Offset bei 3 los

| LED | Kanal | Offset |
|---|---|---|
| 2 | Rot | 3 |
| 2 | Grün | 4 |
| 2 | Blau | 5 |
| 3 | Rot | 6 |

usw. bis zu einem Offset von 31, also bis zu 9 RGB-LED’s weit.

| Aktiv | Filter | Adresse oder Name | Typ | Start-wert | Beschreibung | Verteiler-Nummer | Stecker-Nummer | Icon | Name | Beleuchtung, Sound, oder andere Effekte | Start LedNr | LEDs | InCnt | Loc InCh | LED/ Sound Kanal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ✓ | | SwitchB2 | | | RGB-LED (1) Offset: 0,1,2 | | | | RGB-LED einstellbar | ConstRGB(#LED, #InCh, 0, 0, 0, 127, 127, 127) | 1 | 1 | 1 | 0 | 0 |
| ✓ | | | | | | | | | LED-Werte als Variable | LED_to_Var(Erg1, 8, >, 0) | | | 0 | 0 | |
| ✓ | | SwitchB3 | | | RGB-LED (2) Offset: 3,4,5 | | | | RGB-LED einstellbar | ConstRGB(#LED, #InCh, 0, 0, 0, 127, 127, 127) | 2 | 1 | 1 | 0 | 0 |
| ✓ | | SwitchB4 | | | RGB-LED (3) Offset: 6,7,8 | | | | RGB-LED einstellbar | ConstRGB(#LED, #InCh, 0, 0, 0, 127, 127, 127) | 3 | 1 | 1 | 0 | 0 |
| ✓ | | Erg1 | | | Aktiv nur, wenn bei RGB-LED 3 der Blau-Kanal ein ist | | | | Andreaskreuz | Andreaskreuz(#LED, C23, #InCh) | 4 | C2-3 | 1 | 0 | 0 |

### Vergleichstyp

Es gibt 6 Vergleichswerte

| Operator | Bedeutung | Beispiel für „wahr“ |
|---|---|---|
| = | Beide Werte müssen gleich sein | LED-Kanal 10, Vergleichswert 10 |
| != | Beide Werte müssen ungleich sein | LED-Kanal 10, Vergleichswert 20 |
| < | LED muss kleiner sein als Vergleich | LED-Kanal 10, Vergleichswert 11 |
| > | LED muss größer sein als Vergleich | LED-Kanal 11, Vergleichswert 10 |
| & | Binärer Vergleich | Siehe unten |

| Operator | Bedeutung | Beispiel für „wahr“ |
|---|---|---|
| !& | Binärer Vergleich negiert | Siehe unten |

Der bitweise AND-Operator (einzelnes kaufmännisches Und &), bearbeitet die Bitpositionen der umgebenden Ausdrücke unabhängig voneinander gemäß dieser Regel: Wenn beide Eingangsbits 1 sind, ist das resultierende Ergebnis 1, andernfalls ist das Ergebnis 0.
Eine andere Möglichkeit, dies auszudrücken, ist:

| | |
|---|---|
| 0 0 1 1 | LED-Kanal (3) |
| 0 1 0 1 | Vergleich (5) |
| - - - - - - | |
| 0 0 0 1 | (operand1 & operand2) – Ergebnis ist wahr |

**Beispielcode**

| | | |
|---|---|---|
| Led-Kanal | 92 | Binär: 01011100 |
| Led-Kanal | 101 | Binär: 01100101 |
| Led_to_Var; | Ergebnis: | 01000100, oder 68 dezimal und damit wahr. |

Um die Binär-Vergleiche zu nutzen, sind ausführliche Tests notwendig. Sie eignen sich nicht für eine schnelle Lösung. Hierfür sind die 4 anderen Vergleichstypen gut geeignet.

### Vergleichswert

Hier wird der Wert eingetragen, der mit dem Wert des LED-Kanals verglichen wird.

## Variable für 256 Zustände erstellen

### New_Local_Var

Mit dem Makro „New_Local_Var()“ wird eine Variable vom Typ „ControlVar_t“ angelegt. Sie kann Werte zwischen 0 und 255 annehmen und besitzt zusätzliche Flags mit denen Änderungen erkannt werden können.

## Verknüpfung zur MLL-Bibliothek

** Use_GlobalVar**

Mit der Funktion „Use_GlobalVar()“ können die eigenen Programmteile mit den bibliotheksinternen Funktionen Daten austauschen.

## Temporäre 8bit Variable erstellen

### InCh_to_TmpVar

Mit diesem Befehl wird eine temporäre 8 Bit Variable mit den Werten aus mehreren logischen Variablen gefüllt. Die Variable wird auf 0 gesetzt, wenn die erste logische Variable aktiv ist. Sie wird auf 1 gesetzt, wenn die zweite logische Variable aktiviert wurde…|Das wird z.B. zum steuern eines Signals über mehrere DCC Adressen benötigt.|Im Gegensatz zur „New_Local_Var()“ Funktion und zur „Use_GlobalVar()“ Funktion wird hier kein zusätzlicher Speicher benötigt. Das ist möglich, weil derselbe Speicher mehrfach benutzt werden kann. In den beiden anderen Fällen muss gespeichert werden ob sich der Eingang verändert denn nur dann soll eine Aktion ausgelöst. Bei dieser Funktion wird die Änderung der Eingangsvariablen benutzt.

## Temporäre 8bit Variable erstellen, binär

### Bin_InCh_to_TmpVar

Dieser Befehl kombiniert mehrere logische Variablen binär zu einer 8 Bit Variable. Wenn zwei Eingangsvariablen benutzt werden, dann ergeben sich 4 mögliche Kombinationen:

| Var 1 | Var 0 | Ausgang |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 2 |
| 1 | 1 | 3 |

Bei InCh_Cnt = 3 ergeben sich entsprechend 8 verschiedene Kombinationen. Diese Funktion wird z.B. zur Ansteuerung von Signalen bei Selectrix verwendet.

## Eingangsvariablen definieren

### Define Input

Damit wird ein Variable angelegt welche als Parameter in anderen Makros benutzt werden kann.
Aufbau der Variable: INCH_<Typ>_<Adr>_<Tast> <Typ>: „DCC“, „SX“, „CAN“ / <Adr> Adresse / <Typ>: „RED“, „GREEN“, „ONOFF“
Bei Selectix <Adr>: <Channel>_<BitNr> / <Typ>: „ONOFF“, „TAST“

#### Beispiel des Variablen-Namen (hier DCC)

- INCH_DCC_12_ONOFF
- INCH_DCC_13_RED
- INCH_DCC_13_GREEN

#### Beispiel einer Programmierung

Hier soll ein Andreaskreuz aktiviert werden durch 2 DCC-Adressen: Egal welche DCC-Adresse auf ON ist, soll das Andreaskreuz blinken.

| Aktiv | Filter | Adresse oder Name | Typ | Start wert | Beschreibung | Verteiler-Nummer | Stecker-Nummer | Icon | Name | Beleuchtung, Sound, oder andere Effekte | Start LedNr | LEDs | InCnt | Loc InCh | LED/ Sound Kanal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ✓ | | 11 | AnAus 0 | | | | | | Eingangsvariablen definieren | // Define Input() | | | 1 | 0 | |
| ✓ | | 12 | AnAus 0 | | | | | | Eingangsvariablen definieren | // Define Input() | | | 1 | 0 | |
| ✓ | | | | | | | | | Logische Verknüpfung | Logic(Schranke1, INCH_DCC_12_ONOFF OR INCH_DCC_13_ONOFF) | | | 1 | 0 | |
| ✓ | | Schranke1 | | | | | | | Andreaskreuz RGB | AndreaskrRGB(#LED, #InCh) | 1 | 2 | 1 | 0 | 0 |

Hier werden für die DCC-Adresse 11 und 12 jeweils eine Inputvariable (ICH_DCC_11_ONOFF und INCH_DCC_12_ONOFF) definiert. Diese werden über eine OR-Verknüpfung verknüpft.

# Automatisierung

## Zeitplan

### Schedule

Dieser Plan gibt nur die groben Rahmenbedingungen vor. Wann die Ausgänge tatsächlich geschaltet werden, bestimmt das Programm zufällig damit ein realer Eindruck entsteht. Geschaltet werden die Ausgangsvariablen „DstVar1“ bis „DstVarN“. Sie werden zufällig zwischen dem Zeitpunkt „Start“ und „End“ eingeschaltet, wenn es „Abend“ ist und genauso Zufällig wieder am „Morgen“ ausgeschaltet. Ob es „Abend“ oder „Morgen“ ist, bestimmt die globale Variable „DayState“. Sie ist „Abends“ auf „SunSet“ gesetzt und „Morgens“ auf „SunRise“. Die zweite Variable „Darkness“ bestimmt über eine Zahl zwischen 0 und 255 wie „Dunkel“ es ist. Damit repräsentiert sie die Zeit.

## Zählwerk

### **Counter** (Experteneinstellung)

Mit der Counter Funktion können verschiedene Funktionen implementiert werden. Das sind nicht nur Zähler sondern auch Funktionen wie Flip-Flops, Mono Flops, … Der Befehl wird über einen Mode Parameter gesteuert.
Details dazu findet man in „MobaLedLib Ein kurzer Ueberblick.docx“

## Timer mit Abbruch

### **Button** (Experteneinstellung)

Dieses Makro speichert ein Ereignis(z.B. Tastendruck) für eine bestimmte Zeit mit Abbruchmöglichkeit. Damit kann z.B. der Rauchgenerator im brennenden Haus aktiviert werden. Der Ausgang kann vor Ablauf der Zeit wieder deaktiviert werden, wenn der Taster ein zweites Mal gedrückt wird.

## Timer ohne Abbruch

### **ButtonNOff**

Dieses Makro speichert ein Ereignis (z.B. Tastendruck) für eine bestimmte Zeit. Damit kann Beispielsweise der Rauchgenerator in einem „Brennenden“ Haus aktiviert werden. Der Ausgang kann

NICHT vor dem Ablauf der Zeit deaktiviert werden.

## Treppenhausschalter

### **ButtonFunc** (Experteneinstellung)

Die Ausgangsvariable „DstVar” wird 1, wenn der Eingang „InCh“ aktiv ist. Der Ausgang bleibt nachdem der Eingang deaktiviert wurde für „Duration” Millisekunden aktiv. Das Makro entspricht einem statischen, retriggerbaren Mono Flop. Die Zeit kann auch durch anhängen von „Sec“ oder „Min“ angegeben werden. Die maximale Zeit ist 17 Minuten.

## Zufallsschaltung

Die Zufallsschaltungen haben einen Parameter namens „MODE“. Hier kann folgendes angegeben werden:

```
  RM_NORMAL    Standardwert.
  RF_SLOW      Die Zeitbasis wird durch 16 geteilt. Dieses Flag wird
automatisch gesetzt, wenn die Zeit > 65535 ms ist.
  RF_SEQ       Schalten Sie die Ausgänge der Funktion RandMux() sequenziell
und nicht zufällig um.
  RF_STAY_ON       Flag für die Funktion Ranom(). Der Ausgang bleibt aktiv,
bis der Eingang abgeschaltet wird. MinOn und MaxOn legen die Einschaltdauer
fest.
  RF_NOT_SAME      Verhindern Sie, dass dieselbe Ausgabe zweimal
hintereinander verwendet wird.
```

Die Parameter können auch mittels „ | “ kombiniert werden. Z.B.: „RF_SEQ | RF_SLOW“

### **RandMux**

Die Ausgänge werden über die Zahlen „DstVar1“ bis „DstVarN“ definiert. „DstVar1“ wird aktiviert, wenn „InCh“ nicht aktiviert ist. Bei aktivem „InCh“ werden zufällig „DstVar2“ bis „DestVarN“ aktiviert. Über „MinTime“ wird bestimmt, wie lange ein Ausgang minimal aktiv ist. „MaxTime“ beschreibt analog die maximale Zeit, die der Ausgang aktiv bleiben soll. Das Programm bestimmt zwischen diesen beiden Eckpunkten einen zufälligen Zeitpunkt zu dem ein zufälliger anderer Kanal aktiviert wird. Mit dieser Funktion kann zum Beispiel zwischen verschieden Lichteffekten für eine Disco umgeschaltet werden. Details dazu findet an in „MobaLedLib Ein kurzer Ueberblick.docx“

## Zufallsschaltung 1 Ausgang

### **Random**

Die Funktion „Random()“ aktiviert einen Ausgang nach einer zufälligen Zeit. Die Einschaltdauer kann ebenfalls zufällig sein. Damit kann man einen Effekt zufällig steuern. Eine Anwendung dafür ist das Blitzlicht eines Fotografens, das ab und zu blitzen soll. Mit den Parametern „MinTime“ und „MaxTime“ wird bestimmt in welchem zeitlichen Bereich die Funktion aktiv werden soll. Über „MinOn“ und

„MaxOn“ wird die Dauer vorgegeben.

### Zufallsschaltung sequenziell

**RandCntMux** (Experteneinstellung)

Diese Funktion ist ähnlich wie die „RandMux()“ Funktion. Hier werden aber die Ausgänge nacheinander aktiviert wenn der nächste zufällige Zeitpunkt erreicht ist.

### Multiplexer

**Multiplexer** (Experteneinstellung)

Mit dem Multiplexer können Gruppen von Mustern (Pattern) erstellt werden, welche Nacheinander oder nach dem Zufallsprinzip aktiviert werden. Die Anzahl der verwendeten LEDs kann angepasst werden. Hier wird die Verwendung des Multiplexers erklärt:
Verwendung des Multiplexers

## Charlieplexing

### Charlieplexing Taster

** Charlie_Buttons**

Mit dem Charlieplexing Modul können bis zu 12 LEDs über 4 Kabel angesteuert werden. Das wird z.B. bei Signalen oder Ampeln verwendet damit die Kabel in den Mast passen. Die Ansteuerung per Taster eignet sich z.B. für die MS2. Es können bis zu 11 Taster benutzt werden.

### Charlieplexing binär

** Charlie_Binary**

Mit dem Charlieplexing Modul können bis zu 12 LEDs über 4 Kabel angesteuert werden. Das wird z.B. bei Signalen oder Ampeln verwendet damit die Kabel in den Mast passen. Die binäre Ansteuerung wird bei vielen Zentralen benutzt. Dabei werden weniger Kanäle belegt als bei der Ansteuerung per Taster.

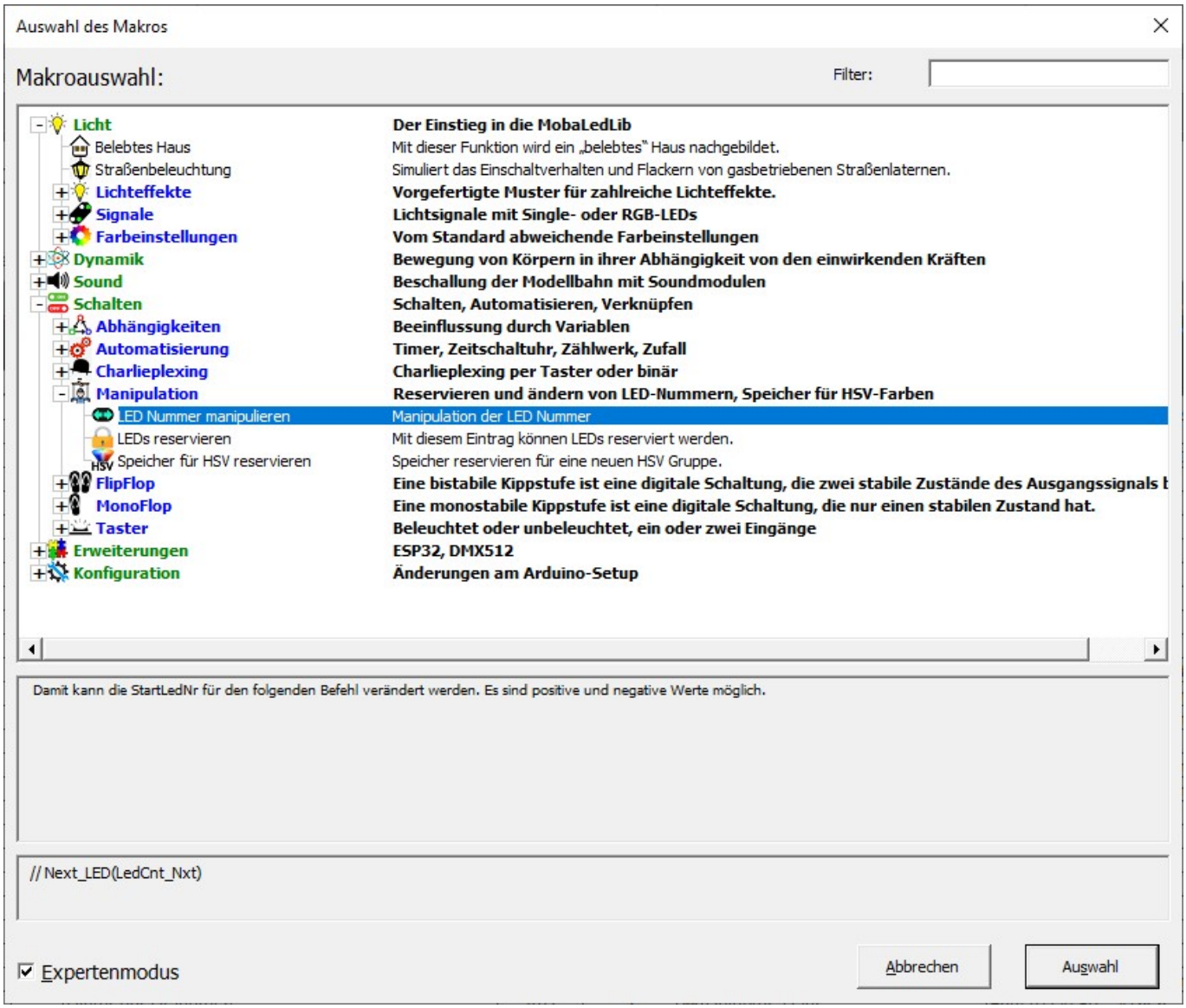
## Manipulation

### LED Nummer manipulieren

**Next_LED** (Experteneinstellung)

Diese Funktion eignet sich beispielsweise, um nicht genutzte Ausgänge eines WS2811-Chips zu nutzen. Hat man beispielsweise in einem belebten Haus ein Ladengeschäft mit zwei Einzel-LEDs beleuchtet und will dieses Ladengeschäft als eigenes „belebtes Haus“ steuern, dass unabhängig vom Wohnhaus schaltet, kann der dritte Ausgang des WS2811 nicht für die nächste LED im Wohnbereich genutzt werden. Der Programm Generator fängt beim nächsten „belebten Haus“ wieder beim ersten Kanal des nächsten WS2811 an.
Gibt man nun zunächst den Befehl „Next LED = -1“ ein, wird eine LED-Nummer (bzw. ein WS281x) zurück gezählt und man kann den Ausgang „Blau“ als erstes Glied in der nächsten Kette nutzen nutzen.

Die Funktion erreicht man bei aktiviertem Expertenmodus unter Schalten > Manipulation > LED Nummer manipulieren




Im Menü kann nun festgelegt werden, ob man zurück- (-N) oder vorspringen (+N) möchte. Im Beispiel des Wechselblinkers muss man eine LED-Nummer zurück, also „-1“.

Im Programm-Generator sieht die Programmierung dann aus wie folgt:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Ladengeschäft | V01 | 2 | Belebtes Haus | HouseT(#LED, #InCh, 2, 2, 0, 0, NEON_LIGHT1M, NEON_LIGHT2M) | 9 | 1 | 1 |
| | V01 | 2 | LED Nummer manipulieren | // Next_LED(-1) | 10 | -1 | 0 |
| Wohnhaus | V01 | 2 | Belebtes Haus | HouseT(#LED, #InCh, 2, 4, 15, 60, NEON_LIGHT3M, SINGLE_LED1, | 9 | 3 | 1 |

Achtung: Will man vom ersten WS281x in der Kette den Rot-Kanal (1) und vom zweiten WS281x in der Kette den Blau-Kanal (3), muss „0“ addiert werden.

| | | | | |
|---|---|---|---|---|
| LED einstellbar | Const(#LED, C1, #InCh, 0, 255) | 32 | C1-1 | 1 |
| LED Nummer manipulieren | // Next_LED(0) | 33 | 0 | 0 |
| LED einstellbar | Const(#LED, C3, #InCh, 0, 255) | 33 | C3-3 | 1 |

Da mit diesem Befehl maximal 20 RGB-LEDs vor- bzw. zurückgesprungen werden kann, eignet sich diese Funktion weniger zum Reservieren von LEDs. Dafür gibt es die Funktion **LEDs reservieren**.

## LEDs reservieren

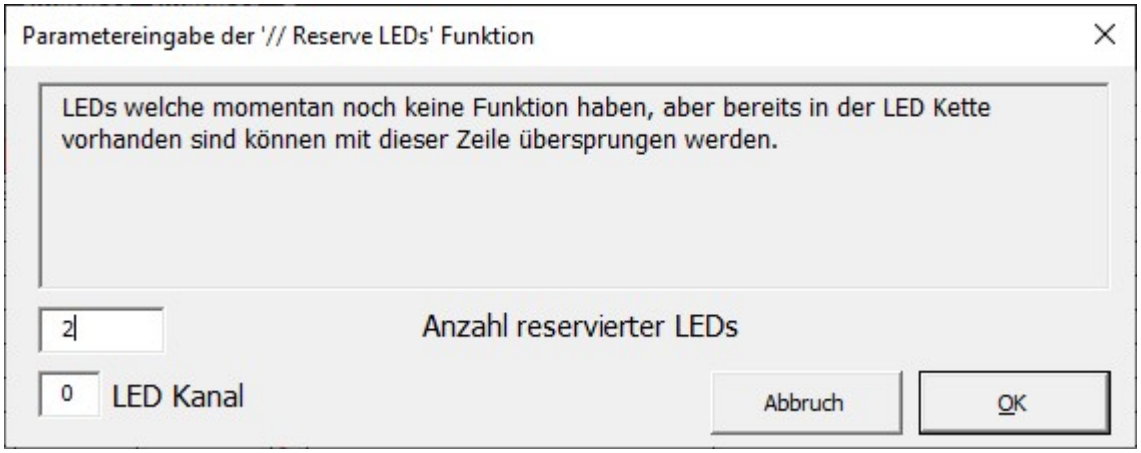
### Reserve LEDs

Der Reserve LED Befehl kann vielfältig eingesetzt werden.
Im **Anlagenbetrieb** eignet er sich beispielsweise zum Überspringen ungenutzter WS2811 auf der **24LED Connector** Platine. Hat man diese nicht zu 100% ausgenutzt, müssen nicht genutzte WS2811 Chips solange reserviert werden, damit der nächste WS281X nach der 24LED Connector Platine korrekt angesprochen wird. Hat man am Connector beispielsweise nur 15 Straßenlaternen angeschlossen, muss man drei LEDs bzw. WS2811 reservieren.
Es können aber auch WS2811 mit angeschlossenen LEDs oder WS2812B LEDs übersprungen werden.

Im **Testbetrieb** eignet sich der Befehl beispielsweise in Zusammenhang mit der 64 LED-Matrix. Testet man dort z. B. ein Muster mit zehn LEDs und möchte gleichzeitig die nachfolgenden Funktionen oder die Funktion der Heartbeat N auf der Hauptplatine erhalten, muss man zunächst die 54 ungenutzten LEDs der Matrix reservieren um sie zu überspringen.

**Anleitung:** Mit aktivierten Experteneinstellungen findet man den Eintrag unter Schalten > Manipulation > LEDs reservieren.
Nach dem Doppelklick trägt man im Menü die Anzahl der zu überspringenden WS2812-RGB-LEDs bzw. WS2811-Chips ein, **nicht** die Anzahl zu überspringender Kanäle (RGB)!

Hat man beispielsweise 16 LEDs am 24er Connector angeschlossen, sind damit sechs WS2811 in Betrieb (5 WS2811 à 3 LEDs + 1 WS2811 à 1 LED = 6 WS2811). Die Kanäle G und B des sechsten müssen nicht mehr reserviert werden. Um die letzten beiden Chips des Connectors zu überspringen, reserviert man in dem Fall zwei LEDs.




### Speicher für HSV reservieren

#### New_HSV_Group

Effekte welche HSV Farben und nicht RGB Farben benutzen benötigen Speicherplatz für die HSV Werte. Da u.U. mehrere Zeilen benötigt werden zur Steuerung einer LED muss dieser Befehl vor jeder Gruppe zusammengehöriger HSV Zeilen stehen.

## Flip-Flop

Ein Flipflop (auch Flip-Flop), oft auch bistabile Kippstufe oder bistabiles Kippglied genannt, ist eine elektronische Schaltung, die zwei stabile Zustände des Ausgangssignals besitzt. Dabei hängt der aktuelle Zustand nicht nur von den gegenwärtig vorhandenen Eingangssignalen ab, sondern außerdem vom Zustand, der vor dem betrachteten Zeitpunkt bestanden hat. Eine Abhängigkeit von der Zeit besteht nicht, sondern nur von Ereignissen. Durch die Bistabilität kann die Kippstufe eine Datenmenge von einem Bit über eine unbegrenzte Zeit speichern. Dazu muss, anders als bei nichtflüchtigen Datenspeichern, jedoch die Spannungsversorgung dauernd gewährleistet sein.

### RS FlipFlop

#### RS_Flip-Flop (Experteneinstellung)

Ein Flip-Flop kann zwei Zustande annehmen (0 oder 1). Bei einem RS Flip-Flop werden die Zustände über zwei Eingänge bestimmt. Eine positive Flanke an „S_InCh“ setzt das Flip-Flop (Ausgang = 1), eine Flanke an “InCh“ löscht das Flip-Flop (Ausgang = 0).

### RS Flip-Flop, Auszeit

**RS_FlipFlopTimeout** (Experteneinstellung)

Ein Flip-Flop kann zwei Zustande annehmen (0 oder 1). Bei einem RS Flip-Flop werden die Zustände über zwei Eingänge bestimmt. Eine positive Flanke an „S_InCh“ setzt das Flip-Flop (Ausgang = 1), eine Flanke an “InCh“ löscht das Flip-Flop (Ausgang = 0). Hier existiert zusätzlich ein Parameter „Timeout“ der bestimmt, wann das Flip-Flop automatisch gelöscht wird.

## Toggle Flip-Flop

**T_FlipFlopReset** (Experteneinstellung)

Der Ausgang eines „Toggle Flip-Flops” wird bei jeder positiven Flanke an Eingang umgeschaltet. Diese Funktion hat zusätzlich einen „Reset“ Eingang mit dem das Flip-Flop auf Null gesetzt werden kann. Wenn dieser Eingang nicht benötigt wird, dann kann er mit „SI_0“ belegt werden.

## Toggle Flip-Flop, Auszeit

**T_FlipFlopResetTimeout** (Experteneinstellung)

Der Ausgang eines „Toggle Flip-Flops” wird bei jeder positiven Flanke an Eingang umgeschaltet. Diese Funktion hat zusätzlich einen „Reset“ Eingang mit dem das Flip-Flop auf Null gesetzt werden kann. Wenn dieser Eingang nicht benötigt wird, dann kann er mit „SI_0“ belegt werden. Der Parameter „Timeout“ definiert die Zeit nach der das Flipflop automatisch zurück gesetzt wird.

## RS Flip-Flop invers

** RS_FlipFlopInv** (Experteneinstellung)

Ein Flip-Flop kann zwei Zustande annehmen (0 oder 1). Bei einem RS Flip-Flop werden die Zustände über zwei Eingänge bestimmt. Eine positive Flanke an „S_InCh“ setzt das Flip-Flop (Ausgang = 1), eine Flanke an “InCh“ löscht das Flip-Flop (Ausgang = 0). Bei diesem Flip-Flop ist der Ausgang zu Programmbeginn aktiv.

## RS Flip-Flop invers, Auszeit

** RS_FlipFlopInvTimeout** (Experteneinstellung)

Ein Flip-Flop kann zwei Zustande annehmen (0 oder 1). Bei einem RS Flip-Flop werden die Zustände über zwei Eingänge bestimmt. Eine positive Flanke an „S_InCh“ setzt das Flip-Flop (Ausgang = 1), eine Flanke an “InCh“ löscht das Flip-Flop (Ausgang = 0). Bei diesem Flip-Flop ist der Ausgang zu Programm Begin aktiv. Außerdem kann es per Timeout zurück gesetzt werden. Das dieses Flip-Flop invers funktioniert wird dar Ausgang per Tastendruck abgeschaltet und geht nach der angegebenen Zeit wieder an.

## Toggle Flip-Flop invers

**T_FlipFlopInvReset** (Experteneinstellung)

Der Ausgang eines „Toggle Flip-Flops” wird bei jeder positiven Flanke an Eingang umgeschaltet. Diese Funktion hat zusätzlich einen „Reset“ Eingang mit dem das Flip-Flop auf Null gesetzt werden kann. Wenn dieser Eingang nicht benötigt wird, dann kann er mit „SI_0“ belegt werden. Bei diesem Flip-Flop ist der Ausgang zu Programmbeginn aktiv.

## Toggle Flip-Flop invers, Auszeit

**T_FlipFlopInvResetTimeout** (Experteneinstellung)

Der Ausgang eines „Toggle Flip-Flops” wird bei jeder positiven Flanke an Eingang umgeschaltet. Diese Funktion hat zusätzlich einen „Reset“ Eingang mit dem das Flip-Flop auf Null gesetzt werden kann. Wenn dieser Eingang nicht benötigt wird, dann kann er mit „SI_0“ belegt werden. Der Parameter „Timeout“ definiert die Zeit nach der das Flipflop automatisch zurück gesetzt wird. Bei diesem Flip-Flop ist der Ausgang zu Programmbeginn aktiv.

## RS FlipFlop bipolar

**RS_Flip-Flop2** (Experteneinstellung)

Ein Flip-Flop kann zwei Zustande annehmen (0 oder 1). Bei einem RS Flip-Flop werden die Zustände über zwei Eingänge bestimmt. Eine positive Flanke an „S_InCh“ setzt das Flip-Flop (Ausgang = 1), eine Flanke an “InCh“ löscht das Flip-Flop (Ausgang = 0). Dieses FlipFlop hat zwei Ausgangsvariablen welche abwechselnd aktiv sind.

## RS Flip-Flop bipolar, Auszeit

**RS_FlipFlop2Timeout** (Experteneinstellung)

Ein Flip-Flop kann zwei Zustande annehmen (0 oder 1). Bei einem RS Flip-Flop werden die Zustände über zwei Eingänge bestimmt. Eine positive Flanke an „S_InCh“ setzt das Flip-Flop (Ausgang = 1), eine Flanke an “InCh“ löscht das Flip-Flop (Ausgang = 0). Hier existiert zusätzlich ein Parameter „Timeout“ der bestimmt, wann das Flip-Flop automatisch gelöscht wird. Dieses FlipFlop hat zwei Ausgangsvariablen welche abwechselnd aktiv sind.

## Toggle Flip-Flop bipolar

**T_FlipFlop2Reset** (Experteneinstellung)

Der Ausgang eines „Toggle Flip-Flops” wird bei jeder positiven Flanke an Eingang umgeschaltet. Diese Funktion hat zusätzlich einen „Reset“ Eingang mit dem das Flip-Flop auf Null gesetzt werden kann. Wenn dieser Eingang nicht benötigt wird, dann kann er mit „SI_0“ belegt werden. Dieses FlipFlop hat

zwei Ausgangsvariablen welche abwechselnd aktiv sind.

### Toggle Flip-Flop bipolar, Auszeit

**T_FlipFlop2ResetTimeout** (Experteneinstellung)

Der Ausgang eines „Toggle Flip-Flops” wird bei jeder positiven Flanke an Eingang umgeschaltet. Diese Funktion hat zusätzlich einen „Reset“ Eingang mit dem das Flip-Flop auf Null gesetzt werden kann. Wenn dieser Eingang nicht benötigt wird, dann kann er mit „SI_0“ belegt werden. Der Parameter „Timeout“ definiert die Zeit nach der das Flipflop automatisch zurück gesetzt wird. Dieses FlipFlop hat zwei Ausgangsvariablen welche abwechselnd aktiv sind.

## Mono-Flop

Eine monostabile Kippstufe, auch monostabiler Multivibrator, Monoflop oder Univibrator, ist eine digitale Schaltung, die nur einen stabilen Zustand hat. Von einem von außen eintreffenden Trigger-Signal angestoßen, ändert die Schaltung für eine bestimmte Zeit ihren Schaltzustand. Anschließend kehrt die Kippstufe wieder in die Ruhelage zurück. Man unterscheidet zwischen nachtriggerbaren (auch: retriggerbar) und nicht nachtriggerbaren Monoflops. Nachtriggerbar bedeutet, dass ein während des Zeitablaufes eintreffendes Triggersignal die interne Zeit jeweils erneut startet und der aktive Schaltzustand dementsprechend zeitlich verlängert wird. Bei einem nicht nachtriggerbaren Monoflop hat ein Triggersignal während der aktiven Phase keine Wirkung. /

### Mono-Flop

**MonoFlop** (Experteneinstellung)

Ein Mono Flop ist eine Funktion welche den Ausgang für eine bestimmte Zeit aktiviert, wenn am Eingang ein Wechsel von Null nach Eins (Positive Flanke) erkannt wurde. Die Zeitdauer wird mit jeder weiteren Flanke verlängert, aber nicht, wenn der Eingang dauerhaft aktiv ist.

Das Mono-Flop eignet sich beispielsweise zur Ansteuerung von Entkupplern. Dabei spielt es keine Rolle, ob diese mit Magnetspulen oder mit Servo betrieben werden. Mit einer freien DCC-Adresse wird das Mono-Flop aktiviert, welches wiederum den Ausgang für das Servo oder ein Relais für die eingestellte Zeit aktiviert. Der Entkuppler geht nach der voreingestellten Zeit wieder nach unten.
Ein weiteres Beispiel könnte ein Rauchgenerator sein. Um zu vermeiden, dass dieser „trocken läuft“, lässt man ihn von einem übergeordneten Mono-Flop nach fünf Minuten deaktivieren.

### Mono-Flop, Reset

**MonoFlopReset** (Experteneinstellung)

Ein Mono Flop ist eine Funktion welche den Ausgang für eine bestimmte Zeit aktiviert, wenn am Eingang ein Wechsel von Null nach Eins (Positive Flanke) erkannt wurde. Die Zeitdauer wird mit jeder weiteren Flanke verlängert, aber nicht, wenn der Eingang dauerhaft aktiv ist. Der Mono Flop wird

zurückgesetzt wenn der Reset Eingang 1 ist.

## Mono-Flop, Reset nach Zeit

**MonoFlopLongReset** (Experteneinstellung)

Die Dauer kann ebenso wie bei der vorangegangenen Funktion verlängert werden.

## Mono-Flop invers

**MonoFlopInv** (Experteneinstellung)

Dieser Mono Flop besitzt einen inversen Ausgang. Das bedeutet, der Ausgang ist zu Beginn aktiv (1) und wird deaktiviert mit einer positiven Flanke an „InCh“. Die Zeit kann wie beim normale MF mit einer steigenden Flanke verlängert werden.

## Mono-Flop invers, Reset nach Zeit

**MonoFlopInvLongReset** (Experteneinstellung)

Hier werden die Eigenschaften „Invers“ und „Reset“ kombiniert, wenn der Eingang länger als 1.5 Sekunden aktiv ist.

## Mono-Flop bipolar

**MonoFlop2** (Experteneinstellung)

Ein Mono Flop ist eine Funktion welche den Ausgang für eine bestimmte Zeit aktiviert, wenn am Eingang ein Wechsel von Null nach Eins (Positive Flanke) erkannt wurde. Die Zeitdauer wird mit jeder weiteren Flanke verlängert, aber nicht, wenn der Eingang dauerhaft aktiv ist. Dieses Monoflop hat zwei Ausgangsvariablen welche abwechselnd aktiv sind.

## Mono-Flop bipolar, Reset

**MonoFlop2LongReset** (Experteneinstellung)

Ein Mono Flop ist eine Funktion welche den Ausgang für eine bestimmte Zeit aktiviert, wenn am Eingang ein Wechsel von Null nach Eins (Positive Flanke) erkannt wurde. Die Zeitdauer wird mit jeder weiteren Flanke verlängert, aber nicht, wenn der Eingang dauerhaft aktiv ist. Der Mono Flop wird zurückgesetzt wenn der Reset Eingang 1 ist. Dieses Monoflop hat zwei Ausgangsvariablen welche abwechselnd aktiv sind.

# Taster

Der Bereich für die Taster wurde ausgelagert, da hier eine umfangreiche Erklärung mit Beispielen zu jedem einzelen Tastermakro erstellt wurde.
Die Seite ist ab sofort hier zu finden: Effekte-MLL Tasterfunktionen

## Status-LED

# Konfiguration

### Letzte Zustände speichern

#### Enable_Store_Status

Durch die Funktion Enable_Store_Status wird der letzte Status von Variablen gespeichert. Dadurch zeigen beim erneuten Einschalten der Anlage z.B. Signale immer noch das gleiche Signalbild. Die Funktion bezieht sich immer auf **alle** ~~nachfolgenden~~ Zeilen. Es reicht also, wenn die Funktion einmal am Anfang der Konfiguration eingefügt wird. Es werden aber nur bestimmte Variablen / Funktionen gespeichert:

1. DCC An/Aus (nicht DCC rot oder DCC grün!)
2. Flip-Flops aller Art
3. Signale
4. Charlieplexing
5. Relaiskontakte für Herzpolarisierung
6. und ggf. noch andere...

Nicht gespeichert werden Servostellungen oder Servopositionen.

Um das Speichern eines einzelnen Elements zu verhindern (ohne die ganze Funktion auf zugeben) wird für diese Funktion in der Spalte „Startwert“ eine „0“ eingetragen.

| Aktiv | Filter | Adresse oder Name | Typ | Start-wert | Beschreibung |
|---|---|---|---|---|---|
| ✓ | | 10 | AnAus 0 | | Hier wird der Zustand gespeichert |
| ✓ | | 11 | AnAus 0 | 0 | Hier nicht |

### LEDs der Hauptplatine steuern

** Mainboard_Led**

Mit diesem Befehl können die drei Taster LEDs auf der Hauptplatine und die Grüne HB LED angesteuert werden (1 = linke LED, 2 = mittlere LED, 3 = rechte LED, 4 = HB LED zwischen den Nanos.) Es können auch die Arduino Pin Nummern D2, D3..D13, A0..A5) angegeben werden. Das ist

vor allem für Testzwecke sinnvoll.

# Pinnummern

## Pins Schalter Gruppe A definieren

Definiert die verwendeten Eingangs-Pins für die analogen Schalter (Gruppe A)
Die Liste enthält einen oder mehrere analoge Pins das Arduinos welche per Komma getrennte sind. Pro Eingangs Pin können 10 analoge Schalter eingelesen werden (Standard: A6)

## Pins Schalter Gruppe B definieren

Definiert die verwendeten Eingangs-Pins für die Gruppe B (Border)
Die Gruppe B ist für Taster (oder Schalter) am Anlagenrand (Border) gedacht. Hier wird normalerweise nur ein Eingangs Pin benutzt. (Standard: A2). Mit einer PushButon_4017 Platine können 10 Taste eingelesen werden. Durch Kaskadierung mehrerer Platinen kann die Anzahl der Eingänge erhöht werden.

## Pins Schalter Gruppe C definieren

Definiert die verwendeten Eingangs-Pins für die Gruppe C (Console)
Mit der Gruppe C können sehr viele Schalter eingelesen werden. 80 Schalter können mit einer PushButton_4017 Platine erfasst werden. Durch Kaskadierung mehrerer Platinen kann die Anzahl der Eingänge erhöht werden. (Standard: 2 10 11 12 A5)

## Pins Schalter Gruppe D definieren

Definiert die verwendeten Eingangs-Pins für die Gruppe D (Direct) - direkt auf der Hauptplatine
Auf der Hauptplatine befinden sich 3 Taster. Diese können direkt verwendet werden. Die verwendeten Anschlüsse und einige Weiter sind auch auf den Stecker „KEY_80“ gelegt damit externe Schalter angeschlossen werden können. Mit diesem Befehl können die Anzahl der Schalter erweitert werden. (Standard: 7 8 9)

## Pins LED Bus definieren

Definiert die Ausgangs-Pins zur Ansteuerung der LEDs

Mit der MobaLedLib können mehrere LED Kanäle angesteuert werden. Der erste Kanal wird für die normalen LEDs benutzt. Kanal 2 ist für die Taster gedacht.
**Arduino:** Es können bis zu 4 Kanäle benutzt werden. (Standard: 6 A4)
**ESP32:** Es können bis zu 8 Kanäle benutzt werden. (Standard: 27 32 16 14 18 19 23 0 17)

**DMX:** Bei Verwendung von DMX erhöht sich die Anzahl der Kanäle um 1.

**Virtuelle Pins:** Bei Verwendung des virtueller Pins 'V' erhöht sich die Anzahl der Kanäle um 1. Virtuelle Kanäle eignen sich, um versteckte Operationen auszuführen, die man u. A. mit anderen Funktionen abfragen kann.\\  Eine Anleitung zu diesem Thema gibt es hier: Virtuelle LED-Kanäle

# Tag/Nacht

## Fotowiderstand aktivieren

### Read_LDR

Mit dieser Funktion wird das Auslesen des LDR aktiviert. Zu Testzwecken können die gemessenen Helligkeitswerte im seriellen Monitor angezeigt werden. Weitere Infos zur Tag und Nachtsteuerung sind hier zu finden: Tag und Nachtsteuerung

## Helligkeitswerte anzeigen

### Read_LDR_Debug

Mit diesem Makro können die gemessenen Helligkeitswerte des LDR im seriellen Monitor angezeigt werden. Das ist vor allem für Testzwecke sinnvoll. Später sollte diese Funktion jedoch wieder deaktiviert werden, da sonst unnötig viel Speicherplatz und Rechenzeit benötigt wird.

## Tag/Nacht-Modus aktivieren

### DayAndNightTimer

Mit dieser Funktion wird ein Zeitgeber generiert mit dem die Schedule Funktion gesteuert wird. Der Zeitgeber kann automatisch ablaufen oder über ein DCC Signal, einen Schalter oder eine Variable gesteuert werden (1 = Nacht, 0 = Tag). Es kann angegeben werden, wie lange ein Tag/Nachtzyklus dauern soll. Weitere Infos sind hier zu finden: Tag und Nachtsteuerung

## Uhrzeit beibehalten bei Tag/Nacht

### KeepDarknessCtr

Wenn diese Funktion aktiv ist, wird die Uhrzeit beim Umschalten zwischen Tag und Nacht nicht neu initialisiert. Normalerweise wird die Uhrzeit initialisiert wenn von Tag auf Nacht umgeschaltet wird. 0→1: Es wird dunkler ⇒ Zeit wird auf 12:Uhr gesetzt. 1→0: Es wird heller ⇒ Zeit wird auf 0:00 gesetzt. Wenn der Schalter aktiviert ist, dann wird sofort von dunkler werden auf heller werden umgeschaltet. Das bedeutet aber auch, das die Zeit gespiegelt wird. Aus 18:00 wird 6:00 Uhr…

## Tageszeiten anzeigen

### DayAndNightTimer_Debug

Mit dieser Funktion kann die aktuelle Uhrzeit im seriellen Monitor ausgegeben werden. Das ist vor allem für Testzwecke sinnvoll.

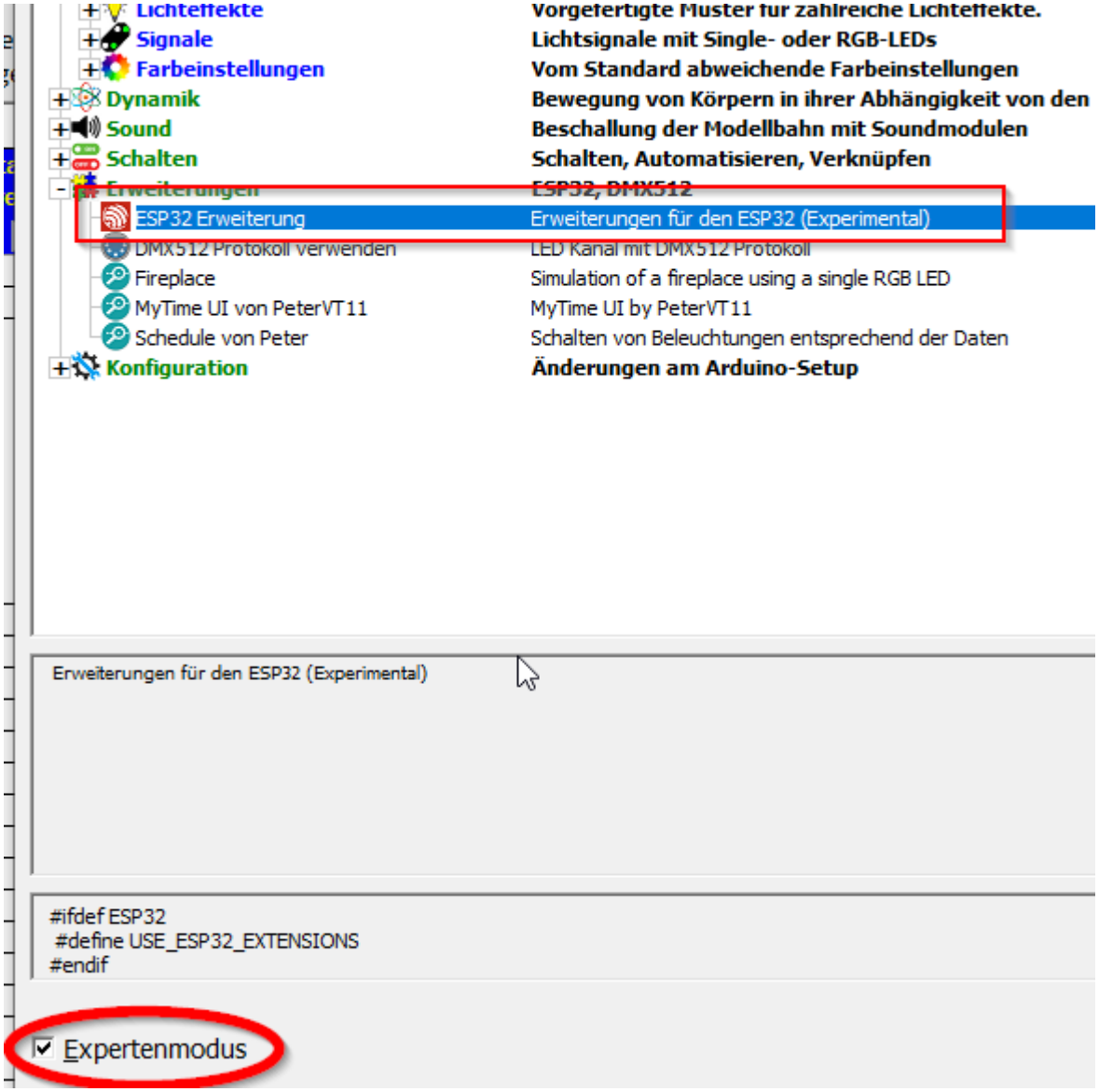
# Erweiterungen

## ESP32 Erweiterung

### EspExtensions

Dieses Makro aktiviert die Erweiterungen für den Esp32. Diese Erweiterungen befinden sich aber noch in der Entwicklungsphase. Um den Esp32 mit der Lichtmaschine (100 oder 101) zu verwenden, muss diese Adapterplatine auf die Hauptplatine aufgesteckt werden: Adapterplatine für Esp32
Mit der Lichtmaschine Pro ist der ESP32 ja schon vorhanden. Mit dem Esp32 können deutlich mehr Leds und Effekte als mit dem normalen Arduino Nano angesteuert werden.

Um die erweiterten Möglichkeiten zu nutzen muss im Programm-Generator ein Eintrag gemacht werden:




Damit wird im Programm-Manager folgendes eingetragen:

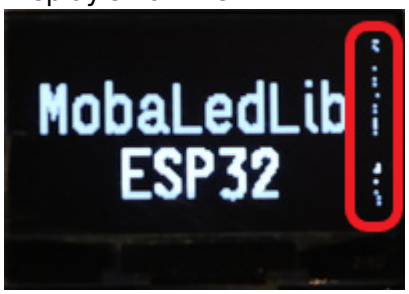
```
#ifdef ESP32
  #define USE_ESP32_EXTENSIONS
#endif
```

Um das Display der Lichtmaschine Pro zu aktivieren ist ein weiterer Eintrag notwendig:

```
#ifdef ESP32
  #define USE_ESP32_EXTENSIONS
  #define USE_UI
#endif
```

**Ab der Software 3.3.2G bzw. 3.4.0 gibt es noch weitere Möglichkeiten.**

Je nach verwendeten Display (0,96„ oder 1.3“ Display) kann es zu Pixelfehlern am rechten Rand des Display's kommen.




Da gibt es einen Parameter, der die Displaygröße einstellt:

```
OLED_TYP 1    ist das 0,96" Display
OLED_TYP 2    ist das 1,3" Display und ist der Standard (auch wenn nicht
eingetragen)
```

Eingetragen wird

```
#ifdef ESP32
  #define USE_ESP32_EXTENSIONS
  #define USE_UI
  #define OLED_TYP 2
#endif
```

Des weiteren kann auch die **MobaLedLib-Uhrzeit** angezeigt werden. Voraussetzung ist allerdings die Zeile „Tag/Nacht-Modus aktivieren“. Mit dem Eintrag

```
UI_MLLTIME 1
```

Das kann dann so im Excelblatt stehen:

```
 #ifdef ESP32
  #define USE_ESP32_EXTENSIONS
  #define USE_UI
  #define UI_MLLTime 1
#endif
```

Das ergibt dann die Anzeige:


Für Nutzer des **Fotowiderstandes (LDR)** gibt es die Möglichkeit, die Daten anzeigen zu lassen. Hier

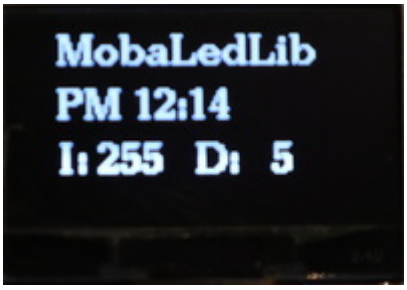
muss allerdings die Zeile „Fotowiderstand aktivieren“ vorhanden sein.

```
UI_MLLTime 2
```

oder

```
UI_MLLTime 3
```

Bei „MLL_Time 2“ gibt es diese Anzeige:


Bei „MLL_Time 3“ gibt es diese Anzeige: 


Die angezeigten Werte sind wie die LDR-Werte, die über die serielle Schnittstelle ausgegeben werden. „I“ ist der Istwert des Sensors (regiert also schnell), „D“ ist der „Dumped“ Wert, also ein langsam nachfolgender Wert, den die MLL ausrechnet.

**Wie kann ich das ins Excel-Blatt eintragen?**
Ich ziehe mir das Eingabefeld in Excel (oben über unseren Buttons) groß (also nach unten) und dann kann man die Werte problemlos eingeben. Oder in einem externen Texteditor eintragen, alles kopieren und in das Excel-Feld einfügen.

## DMX512 Protokoll verwenden

### UseDMX512

Auf diesem LED Kanal wird das DMX512 Protokoll ausgegeben. Was mit DMX möglich ist, kann man in der Aufzeichnung des Stammtischs Januar 2021 sehen: Aufzeichnung Januar 2021

Die Auswahl eines DMX Geräts erfolgt neben den optischen Werten auch immer über dessen Datenblatt. Ohne Datenblatt, welches die DMX Kanäle beschreibt, ist das Gerät nutzlos.
Wichtig ist auch, dass die RGB Kanäle möglichst auf drei aufeinanderfolgende DMX Kanäle liegen.
Bitte achtet unbedingt auf den Abstrahlwinkel, meistens werden in der Bühnentechnik „Spots“ mit geringem Abstrahlwinkel angeboten. Diese sind für große Räume gedacht. In kleineren Räumen braucht ihr

unbedingt „Flood“ Varianten mit großem Abstrahlwinkel. Ein gutes Beispiel für einen **schlecht** geeigneten Scheinwerfer ist dieses Modell: 27x3W Led Flat Par Light RGB.
Der Scheinwerfer hat vor jeder RGB LED ein Linsen, die das Licht bündelt und einen geringen Abstrahlwinkler erzeugt. Diese Linsen können zwar entfernt werden, dann wird das Abstrahlverhalten besser, das ist aber etwas Bastelarbeit.

**Einfachstes Beispiel:** Ein 12 Kanal DMX Modul, mit dem man eine Raumbeleuchtung mit einfarbigen Led-Stripes betreiben kann.

Diese Modul hat 12 DMX Kanäle. Die DMX Basisadresse wird über DIP-Schalter eingestellt.
Unter der Annahme dass DMX Adresse 1 eingestellt ist und an die LED Känal jeweils Led-Stripes mit rot/grün/blau angeschlossen sind heißt das
DMX Adresse 1-3 rot/grün/blau
DMX Adresse 4-6 rot/grün/blau
DMX Adresse 7-9 rot/grün/blau
DMX Adresse 10-12 rot/grün/blau

Im ProgGenerator entspricht das 4 RGB Leds.
Die DMX Adresse errechnet sich aus der Excel Spalte „Start-LedNr “, es gilt die Formel

DmxKanal = Start-LedNr * 3
+ 1 für rot
+ 2 für grün
+ 3 für blau

| 2 | Aktiv | Filter | Adresse oder | Typ | | Start- | Beschreibung | Icon | Name | Beleuchtung, Sound, oder andere Effekte | Start LedNr | LEDs | InCnt | Loc InCh | LED/ Sound Kanal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | ✓ | | | | | | | | DMX512 Protokoll verwenden | // Use_DMX512() | 0 | 0 | 0 | 0 | 2 |
| 4 | ✓ | | 1 | AnAus | 0 | | | | RGB-LED einstellbar | ConstRGB(#LED, #InCh, 0, 0, 0, 230, 220, 24) | 0 | 1 | 1 | 0 | 2 |
| 5 | ✓ | | 2 | AnAus | 0 | | | | RGB-LED einstellbar | ConstRGB(#LED, #InCh, 0, 0, 0, 79, 43, 211) | 1 | 1 | 1 | 0 | 2 |
| 6 | ✓ | | 3 | AnAus | 0 | | | | RGB-LED einstellbar | ConstRGB(#LED, #InCh, 0, 0, 0, 246, 8, 99) | 2 | 1 | 1 | 0 | 2 |
| 7 | ✓ | | 4 | AnAus | 0 | | | | RGB-LED einstellbar | ConstRGB(#LED, #InCh, 0, 0, 0, 56, 198, 130) | 3 | 1 | 1 | 0 | 2 |

Mit Zeile 3 wird der ProgGenerator angewiesen, auf Led Kanal 2 das DMX Protokoll auszugeben.

Dieses Beispiel war ja ganz einfach. Etwas aufwändiger wird es mit z.B. diesem Modell: E-Lektron PAR-18 RGB DMX Scheinwerfer
Gemäß Handbuch sieht die DMX Kanalbelegung so aus:

## Steuerung über DMX

DMX-Modus aktivieren mit den Drucktasten unter dem Display oder mit der Fernbedienung.

| DMX Kanal | DMX Wert | |
|---|---|---|
| CH1 | 0-255 | RGB Dimmer 0-100% |
| CH2 | 0-255 | Dimmer rot 0-100% |
| CH3 | 0-255 | Dimmer grün 0-100% |
| CH4 | 0-255 | Dimmer blau 0-100% |
| CH5 | 0-7 | Stroboskop aus |
| | 8-255 | Stroboskop Geschwindigkeit |
| CH6 | 0-10 | Bedienung über CH1-CH5 |
| | 11-60 | Farbe wählen mit CH7 |
| | 61-110 | Farbwechsel überblenden - Geschwindigkeit mit CH7 |
| | 111-160 | Farbwechsel abwechselnd - Geschwindigkeit mit CH7 |
| | 161-210 | Farbwechsel - Geschwindigkeit mit CH7 |
| | 211-255 | Farbwechsel Musiktakt gesteuert |
| CH7 | 0-255 | Farb-/Geschwindigkeitswahl für CH6 |

Zum Aktivieren der Kanäle CH1-CH5 soll CH6 auf Minimum gestellt werden.

Ganz blöd ist in dem Fall die Kanalanzahl 7, denn dem ProgrammGenerator sind vielfache von drei deutlich lieber.
Wichtig ist hier, dass die RGB Kanäle auf drei hinereinanderfolgenden DMX Adresse liegen.

Wie würde ich es lösen: Der Scheinwerfer bekommt DMX Adresse 3, d.h.
CH1 = DMX Adresse 3: Gesamthelligkeit
CH2 = DMX Adresse 4: Helligkeit rot
CH3 = DMX Adresse 5: Helligkeit grün
CH4 = DMX Adresse 6: Helligkeit blau
CH5 = DMX Adresse 7: Stroboskop
CH6 = DMX Adresse 8: Betriebsmodus\ CH7 = DMX Adresse 9: Geschwindigkeit

Wir brauchen CH6 Betriebsmodus auf einen Wert von 0-10 = „Bedienung über CH1-CH5“. So ist es möglich, Gesamthelligkeit (CH1) und Helligkeit der einzelen Farben (CH2-CH4) zu setzten. CH5 muss 0 sein, weil wir kein Stroboskop brauchen, und CH7 ist egal.

In diesem Fall müssen wir die DMX Kanäle CH1, CH5 und CH6 einzeln ansprechen.
Die DMX Adresse errechnet sich aus den Excel Spalten „Start-LedNr und „Leds“
Die Formel ist Start-LedNr * 3 + X, wobei X von CX aus Spalte LEDS kommt, bzw. X=1 wenn kein Cx in LEDS Spalte.

| 2 | Aktiv | Filter | Adresse oder | Typ | Start- | Beschreibung | Icon | Name | Beleuchtung, Sound, oder andere Effekte | Start LedNr | LEDs | InCnt | Loc InCh | LED/ Sound Kanal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | ✓ | | | | | | | DMX512 Protokoll verwenden | // Use_DMX512() | 0 | 0 | 0 | 0 | 2 |
| 4 | ✓ | | | | | | | LED einstellbar | Const(#LED, C3, #InCh, 255, 255) | 0 | C3-3 | 1 | 0 | 2 |
| 5 | ✓ | | 1 | AnAus 0 | | | | CH2-4: RGB-LED einstellbar | ConstRGB(#LED, #InCh, 0, 0, 0, 35, 219, 92) | 1 | 1 | 1 | 0 | 2 |
| 6 | ✓ | | | | | | | CH5: Stroboskop immer aus | Const(#LED, C1, #InCh, 0, 0) | 2 | C1-1 | 1 | 0 | 2 |
| 7 | ✓ | | | | | | | CH6: Betriebsmodus 0 | Const(#LED, C2, #InCh, 0, 0) | 2 | C2-2 | 1 | 0 | 2 |
| 8 | ✓ | | | | | | | CH7: Geschwindigkeit egal | Const(#LED, C3, #InCh, 0, 127) | 2 | C3-3 | 1 | 0 | 2 |

| | | |
|---|---|---|
| Zeile 4 | Start-Led 0 (auf Led Kanal 2) und C3 → 0*3+3 = DMX Adresse 3 | Das Const-Makro hat sowohl für Helligkeit aus wie ein den Wert 255, somit immer 255. |
| Zeile 5 | Start-Led 1, kein Cx → 1*3+1 = DMX Adresse 4 | Weil RGB LEDs immer drei Kanäle haben, gilt dies somit auch DMX Adresse 5 und 6. |
| Zeile 6 | Start-Led 2 und C1 → 2*3+1 = DMX Adresse 7 | Das Const-Makro hat sowohl für Helligkeit aus wie ein den Wert 0, somit immer 0. |
| Zeile 7 | Start-Led 2 und C2 → 2*3+2 = DMX Adresse 8 | Das Const-Makro hat sowohl für Helligkeit aus wie ein den Wert 0, somit immer 0. |
| Zeile 8 | Start-Led 2 und C3 → 2*3+3 = DMX Adresse 9 | Das Const-Makro hat sowohl für Helligkeit aus wie ein den Wert 0, somit immer 0. |

Damit lässt sich dieses Modell steuern. Das Beispiel kann noch vereinfacht werden, indem die Zeilen 6-8 zusammengefasst werden.

| 2 | Aktiv | Filter | Adresse oder | Typ | Start | Beschreibung | Icon | Name | Beleuchtung, Sound, oder andere Effekte | Start LedNr | LEDs | InCnt | Loc InCh | LED/ Sound Kanal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | ✓ | | | | | | | DMX512 Protokoll verwenden | // Use_DMX512() | 0 | 0 | 0 | 0 | 2 |
| 4 | ✓ | | | | | | | CH1: Gesamthelligkeit | Const(#LED, C3, #InCh, 255, 255) | 0 | C3-3 | 1 | 0 | 2 |
| 5 | ✓ | | 1 | AnAus 0 | | | | CH2-4: RGB-LED einstellbar | ConstRGB(#LED, #InCh, 0, 0, 0, 35, 219, 92) | 1 | 1 | 1 | 0 | 2 |
| 6 | ✓ | | | | | | | CH5-7: alles 0 | ConstRGB(#LED, #InCh, 0, 0, 0, 0, 0, 0) | 2 | 1 | 1 | 0 | 2 |

| | | |
|---|---|---|
| Zeile 6 | Start-Led 2 und C1 → 2*3+1 = DMX Adresse 7 | Das ConstRGB Makro verwendet drei Kanäle, welche sowohl für Helligkeit aus wie ein für alle Werte Wert 0 haben, somit immer alles 0. |

Am schönsten wäre natürlich ein RGB Scheinwerfer ohne jegliche Schnicht-Schnack Funktionen wie Strobe, Fade, Musiksteuerung..., welcher nur drei DMX Kanäle für rot/grün/blau hat. So ein Modell habe ich bisher noch nicht gefunden.

**Beispiel für Fortgeschrittene:**
Dieser DMX Bar kann laut Handbuch mit 6/9/16/30 oder 58 DMX Adressen gesteuert werden, je mehr Kanäle desto gezielter können einzelne LEDs des Bars angesprochen werden.
Sehen wir uns die Variante 30 Adressen genauer an:
Hier können 4 LED Segment einzeln gesteuert werden, pro Segment werden 7 DMX Kanäle verwendet, vier für die RGBW Helligkeit, zusätzlich noch Kanäle für Master-Helligkeit und Strobe-Effekt. Dann kommt noch ein Channel für Sondereffekte, der aber in der Konfiguration nicht berücksichtig wurde.

Das sieht im Excel dann so aus. Man sieht auch gut die Verschiebung wegen der 7 Channels, Segment 1 startet mit C1-1, Segment 2 mit C2-2, Segment 3 mit C3-3. Die Master Helligkeit wird hier mit DCC Adresse 1 für alle Segment geschaltet, die RGBW pro Segment jeweils einzeln.

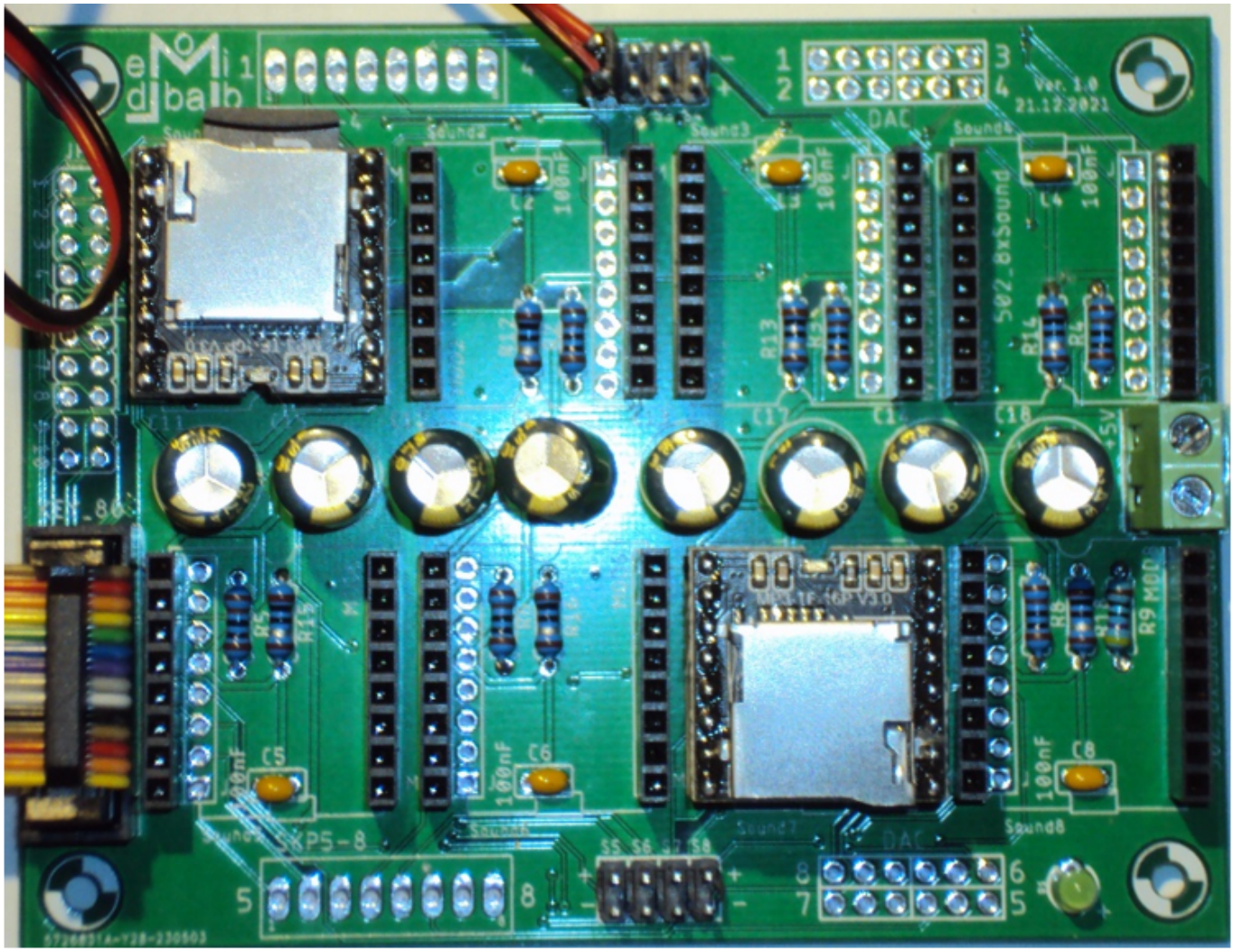
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | AnAus | 0 | | C1 Master | | Const(#LED, C1, #InCh, 0, 255) | 0 | C1-1 |
| 55 | AnAus | 0 | | C1 Strobe | | Const(#LED, C2, #InCh, 0, 10) | 0 | C2-2 |
| 11 | AnAus | 1 | | C1 R | | Const(#LED, C3, #InCh, 0, 255) | 0 | C3-3 |
| 12 | AnAus | 0 | | C1 G | | Const(#LED, C1, #InCh, 0, 255) | 1 | C1-1 |
| 13 | AnAus | 0 | | C1 B | | Const(#LED, C2, #InCh, 0, 255) | 1 | C2-2 |
| 14 | AnAus | 0 | | C1 W | | Const(#LED, C3, #InCh, 0, 255) | 1 | C3-3 |
| | | | | | | | | |
| 1 | AnAus | 0 | | C2 Master | | Const(#LED, C2, #InCh, 0, 255) | 2 | C2-2 |
| 15 | AnAus | 0 | | C2 R | | Const(#LED, C1, #InCh, 0, 255) | 3 | C1-1 |
| 16 | AnAus | 0 | | C2 G | | Const(#LED, C2, #InCh, 0, 255) | 3 | C2-2 |
| 17 | AnAus | 0 | | C2 B | | Const(#LED, C3, #InCh, 0, 255) | 3 | C3-3 |
| 18 | AnAus | 0 | | C2 W | | Const(#LED, C1, #InCh, 0, 255) | 4 | C1-1 |
| | | | | | | | | |
| 1 | AnAus | 0 | | C3 Master | | Const(#LED, C3, #InCh, 0, 255) | 4 | C3-3 |
| 19 | AnAus | 0 | | C3 R | | Const(#LED, C2, #InCh, 0, 255) | 5 | C2-2 |
| 20 | AnAus | 0 | | C3 G | | Const(#LED, C3, #InCh, 0, 255) | 5 | C3-3 |
| 21 | AnAus | 0 | | C3 B | | Const(#LED, C1, #InCh, 0, 255) | 6 | C1-1 |
| 22 | AnAus | 0 | | C3 W | | Const(#LED, C2, #InCh, 0, 255) | 6 | C2-2 |

# Effekte - Sound

# Sound Funktionen für das MP3-TF-16P Modul

## Hauptplatine

Soundsteuerung mit der 8xMP3-Multi-Soundplatine.




Hierzu muss ein Pin der Hautplatine zur seriellen Ansteuerung eines Sound Moduls sowie den Typ des angeschlossenen Soundmoduls definiert werden. Danach können Sound-Tracks dieses Moduls abgespielt werden. Die Pin Nummer kann als Zahl, oder als symbolische Konstante angegeben werden. Beispiel: KEY80_P1 Der Typ bestimmt das Protokoll, mit dem das Soundmodul gesteuert wird. Manche MP3-TF Module haben einen Fehler und müssen mit der Variante NO_CRC (keine Prüfsumme) angesprochen werden. Module mit folgenden Sound Chip Nummern wurde bisher als fehlerhaft identifiziert: MH2024K-16SS Beispiel einer Programmierung im Prog_Generator:

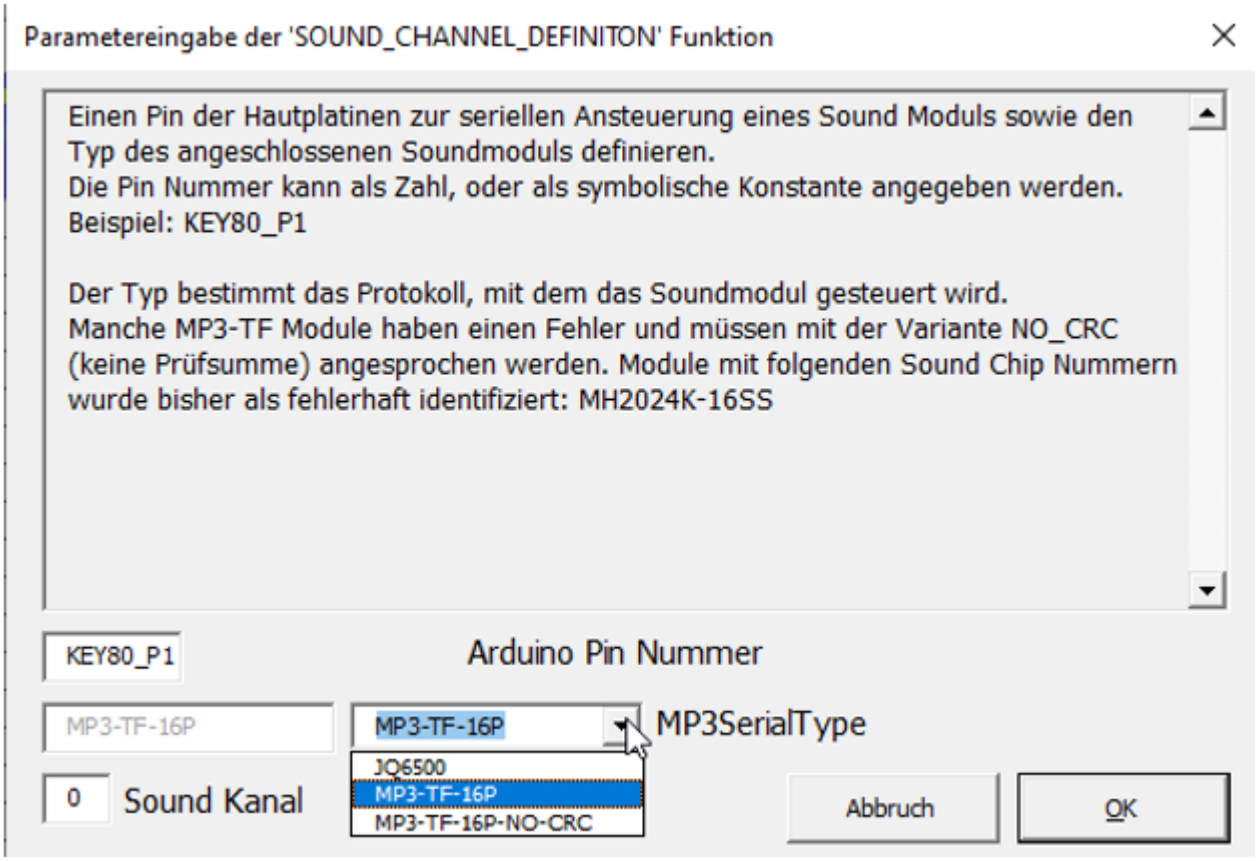
| Aktiv | Filter | Adresse oder Name | Typ | Start wert | Beschreibung | Verteiler-Nummer | Stecker-Nummer | Icon | Name | Beleuchtung, Sound, oder andere Effekte | Start LedNr | LEDs | InCnt | Loc InCh | LED/ Sound Kanal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ✓ | | | | | | | | | Soundmodul wählen | SOUND_CHANNEL_DEFINITON(KEY80_P1, MP3-TF-16P) | S0 | S | 0 | 0 | 0 |
| ✓ | | 11 | AnAus 0 | | Kirchenglocken | | | | Titel # abspielen | SOUND_CHANNEL_PLAY_TRACK(#LED, #InCh, 1) | S0 | S | 1 | 0 | 0 |
| ✓ | | 12 | AnAus 0 | | Andreas-Kreuz | | | | Titel # abspielen | SOUND_CHANNEL_PLAY_TRACK(#LED, #InCh, 2) | S0 | S | 1 | 0 | 0 |

Bezeichnung Beschreibung

| Soundmodul wählen | Pin des Soundmoduls definieren |
|---|---|
| Titel # abspielen | Track # vom angegebenen Modul abspielen (Rootverzeichnis) |
| Zufälligen Titel abspielen | Zufälligen Titel vom angegebenen Modul abspielen (Rootverzeichnis) |
| Wiedergabe | Setzt die Sound Wiedergabe fort |
| Pause | Hält die Sound Wiedergabe an |
| Endloswiedergabe | Stellt den Wiederholmodus ein |
| Lauter | Lautstärke erhöhen |
| Leiser | Lautstärke reduzieren |
| Lautstärke definieren | Lautstärke setzen auf 0 ..100% |

# Soundmodul wählen

Einen Pin der Hautplatinen zur seriellen Ansteuerung eines Sound Moduls sowie den Typ des angeschlossenen Soundmoduls definieren. Die Pin Nummer kann als Zahl, oder als symbolische Konstante angegeben werden.




Beispiel: KEY80_P1

| Soundmodul | Jumperpin | Pin „KEY_80“ | Kennung Programm-Generator | Arduino-Pin |
|---|---|---|---|---|
| #1 | A1 → B1 | 1 | KEY80_P1 | D2 |
| #2 | A2 → B2 | 2 | KEY80_P2 | D7 |
| #3 | A3 → B3 | 3 | KEY80_P3 | D8 |
| #4 | A4 → B4 | 4 | KEY80_P4 | D9 |
| #5 | A5 → B5 | 5 | KEY80_P5 | D10 |
| #6 | A6 → B6 | 6 | KEY80_P6 | D11 |
| #7 | A7 → B7 | 7 | KEY80_P7 | D12 |
| #8 | A8 → B8 | 12 | KEY80_P12 | A5 |
| n.c. | A9 | 8 | KEY80_P8 | A1 |
| n.c. | B9 | 9 | KEY80_P9 | A2 |
| n.c. | A10 | 10 | KEY80_P10 | A3 |
| n.c. | B10 | 11 | KEY80_P11 | A4 |

Der Typ bestimmt das Protokoll (MP3-TF-16P, MP3-TF16P-NO-CRC oder JQ6500), mit dem das Soundmodul gesteuert wird. Manche MP3-TF Module haben einen Fehler und müssen mit der Variante NO_CRC (keine Prüfsumme) angesprochen werden. Module mit folgenden Sound Chip Nummern wurde bisher als fehlerhaft identifiziert: MH2024K-16SS

## Titel # abspielen

Track # auf einem angegeben Soundkanal starten (Rootverzeichnis). Sound Kanal 0 – 7 (entspricht Soundmodul 1 – 8) Tracknummer 1 – 256

## Zufälligen Titel abspielen

Im Makro wird ein Bereich von Track x bis Track y angegeben. Aus diesem Bereich wird ein zufälliger Track abgespielt.

## Wiedergabe

Setzt die Sound Wiedergabe fort, nach dem sie mit dem Pause-befehl unterbrochen wurde.

## Pause

Hält die Sound Wiedergabe an.

## Endloswiedergabe

Stellt den Wiederholmodus ein LOOP_ALL: spielt alle Titel wiederholt ab LOOP_FOLDER: spielt alle Titel eines Verzeichnisses wiederholt ab LOOP_ONE: spielt den aktuellen Titel wiederholt ab LOOP_OFF: keine Wiederholung

## Lauter

Lautstärke erhöhen.

## Leiser

Lautstärke reduzieren.

## Lautstärke definieren

Der angegebene Prozentwert wird je nach Soundmodul in einen passenden Lautstärkenwert umgewandelt, welches das Soundmodul unterstützt.

## Sound_Prev

Die Sound_Prev Funktion spielt die vorherige Sounddatei ab.

## Sound_Next

Mit der Sound_Next Funktion wird die nächste Sound Datei abgespielt.

## Sound_PausePlay

Mit dem Befehl Sound_PausePlay wird die Sound- Wiedergabe angehalten oder wieder fortgesetzt.

## Sound_Loop

## Sound_USDSPI

## Sound_PlayMode

## Sound_DecVol

## Sound_IncVol

## Sound_Seq1

Mit diesem Makro wird die Sound Datei 1 auf einem MP3-TF-16P Soundmodul wiedergegeben.

## Sound_Seq2

Mit diesem Makro wird die Sound Datei 2 auf einem MP3-TF-16P Soundmodul wiedergegeben.

## Sound_Seq3

Mit diesem Makro wird die Sound Datei 3 auf einem MP3-TF-16P Soundmodul wiedergegeben.

## Sound_Seq4

Mit diesem Makro wird die Sound Datei 4 auf einem MP3-TF-16P Soundmodul wiedergegeben.

## Sound_Seq5

Mit diesem Makro wird die Sound Datei 5 auf einem MP3-TF-16P Soundmodul wiedergegeben.

## Sound_Seq6

Mit diesem Makro wird die Sound Datei 6 auf einem MP3-TF-16P Soundmodul wiedergegeben.

## Sound_Seq7

Mit diesem Makro wird die Sound Datei 7 auf einem MP3-TF-16P Soundmodul wiedergegeben.

## Sound_Seq8

Mit diesem Makro wird die Sound Datei 8 auf einem MP3-TF-16P Soundmodul wiedergegeben.

## Sound_Seq9

Mit diesem Makro wird die Sound Datei 9 auf einem MP3-TF-16P Soundmodul wiedergegeben.

## Sound_Seq10

Mit diesem Makro wird die Sound Datei 10 auf einem MP3-TF-16P Soundmodul wiedergegeben.

## Sound_Seq11

Mit diesem Makro wird die Sound Datei 11 auf einem MP3-TF-16P Soundmodul wiedergegeben.

## Sound_Seq12

Mit diesem Makro wird die Sound Datei 12 auf einem MP3-TF-16P Soundmodul wiedergegeben.

## Sound_Seq13

Mit diesem Makro wird die Sound Datei 13 auf einem MP3-TF-16P Soundmodul wiedergegeben.

## Sound_Seq14

Mit diesem Makro wird die Sound Datei 1 auf einem MP3-TF-16P Soundmodul wiedergegeben.

### Sound_PlayRandom

Mit diesem Makro wird eine zufällige Sound Datei abgespielt.

### Sound_Next_of_N_Reset

### Sound_Next_of_N

## Sound Funktionen für das JQ6500 Sound Modul

### Sound_JQ6500_Prev

Mit diesem Makro wird die vorherige Sounddatei abgespielt.

### Sound_JQ6500_Next

Mit diesem Makro wird die nächste Sound Datei abgespielt.

### Sound_JQ6500_DecVol

### Sound_JQ6500_IncVol

### Sound_JQ6500_Seq1

Mit diesem Makro wird die Sound Datei 1 auf einem JQ6500 Soundmodul abgespielt.

### Sound_JQ6500_Seq2

Mit diesem Makro wird die Sound Datei 2 auf einem JQ6500 Soundmodul abgespielt.

### Sound_JQ6500_Seq3

Mit diesem Makro wird die Sound Datei 3 auf einem JQ6500 Soundmodul abgespielt.

### Sound_JQ6500_Seq4

Mit diesem Makro wird die Sound Datei 4 auf einem JQ6500 Soundmodul abgespielt.

### Sound_JQ6500_Seq5

Mit diesem Makro wird die Sound Datei 5 auf einem JQ6500 Soundmodul abgespielt.

### Sound_JQ6500_PlayRandom

### Sound_JQ6500_Next_of_N_Reset

### Sound_JQ6500_Next_of_N

### Sound_JQ6500_BG_Prev

### Sound_JQ6500_BG_Next

### Sound_JQ6500_BG_DecVol

### Sound_JQ6500_BG_IncVol

### Sound_JQ6500_BG_Seq1

### Sound_JQ6500_BG_Seq2

### Sound_JQ6500_BG_Seq3

### Sound_JQ6500_BG_Seq4

### Sound_JQ6500_BG_Seq5

### Sound_JQ6500_BG_PlayRandom

### Sound_JQ6500_BG_Next_of_N_Reset

### Sound_JQ6500_BG_Next_of_N

## Sound_ADKey

## Sound_JQ6500_ADKey

From:
https://wiki.mobaledlib.de/ - **MobaLedLib Wiki**

Permanent link:
**https://wiki.mobaledlib.de/anleitungen/effekte_mll?rev=1791894509**

Last update: **2026/10/13 12:28**