

Sound Servoplatine

Über die Servo-Platine 510 können drei Soundmodule angesteuert werden.

Es werden dabei JQ6500 Module oder MP3-TF16-p/DFPlayer Mini Module verwendet. Dabei spielt es keine Rolle ob nur JQ6500 oder nur MP3-TF16-p/DFPlayer Mini oder diese beliebig gemischt, eingesetzt werden.

Die drei Module können auch parallel je einen Sound abspielen.

Das JQ6500 hat dabei den Vorteil das es bereits 2 MByte Speicher enthält, was für etliche Soundprojekte ausreichend sein dürfte.

Das MP3-TF-16p nutzt hingegen eine MicroSD-Karte mit bis zu 32 GByte und vereinfacht durch die mögliche Ordnerstruktur die Verwaltung der Sounddaten.

Vorbereitung Hardware

ATTiny 85

Als Servoplatine eignet sich jede 510DE-Platine, bestückt als Servoplatine.

Die Lötjumper für den Servobetrieb SERVO, SERVO1, SERVO2, SERVO3 müssen geschlossen werden.

Zunächst muss der ATTiny für die Servoplatine programmiert werden. Die Anleitung dazu [Programmierung von ATTinys für Servo, Charlieplexing oder Sound](#) ist im WIKI zu finden. Im Pattern-Configurator unter spezielle Module **Servo-MP3** auswählen.

55)

Pattern Configurator

Beispiele | **Spezielle Module** | Extras

Diese Seite enthält Funktionen mit denen spezielle, auf dem ATTiny basierte Module programmiert und getestet werden können. [by Hardi](#)

Programmieradapter

Prog. ISP Ein ATTiny hat keinen USB Anschluss. Darum benötigt man zur Programmierung einen Programmieradapter (In Circuit Programmer). Das kann ein Arduino mit besonderen Programm sein. Mit dem Knopf Links wird das Programm zu Arduino übertragen.

Dieses Programm kann auf dem Tiny_UniProg und auf einen "Nackten" Arduino in einem Steckbrett eingesetzt werden.
Der "HV Reset" ist allerdings nur mit der "Tiny_UniProg" Platine möglich. Er wird für das Programmieren des Servo Programms benötigt.

Charlieplexing | **Servo** | **Servo-MP3**

Prog.Servo/MP3 Man kann ein Servo-Modul auf zur Ansteuerung von Soundmodulen nutzen (JQ6500/MP3-TF-16p) nutzen.

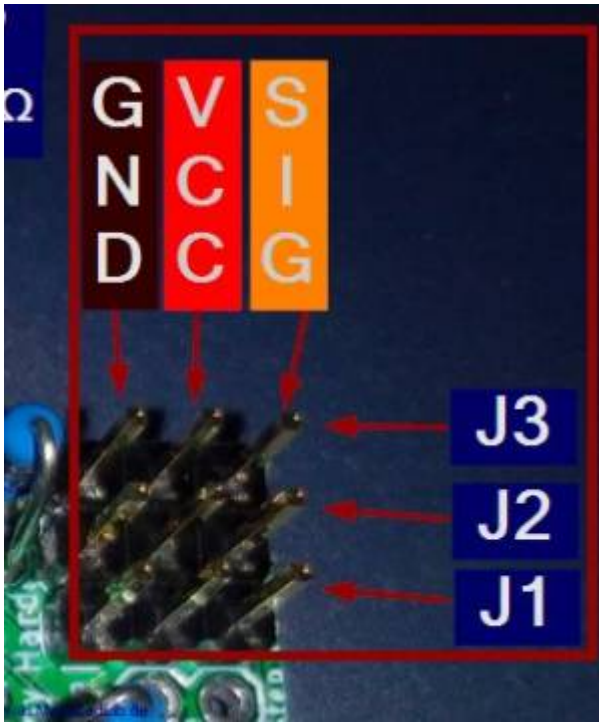
Dabei werden die Soundmodule mit je einem 1k Widerstand an die Signal-Pins der Servoausgänge angeschlossen. Zusätzlich ist die Masse zu verbinden. Die Stromversorgung kann man getrennt lösen - meist funktioniert es aber auch die 5V von den Servoanschlüssen zu nutzen.

Zur Programmierung des ATTiny85 wird ein Programmieradapter benötigt (siehe oben) in der der ATTiny eingesteckt wird.

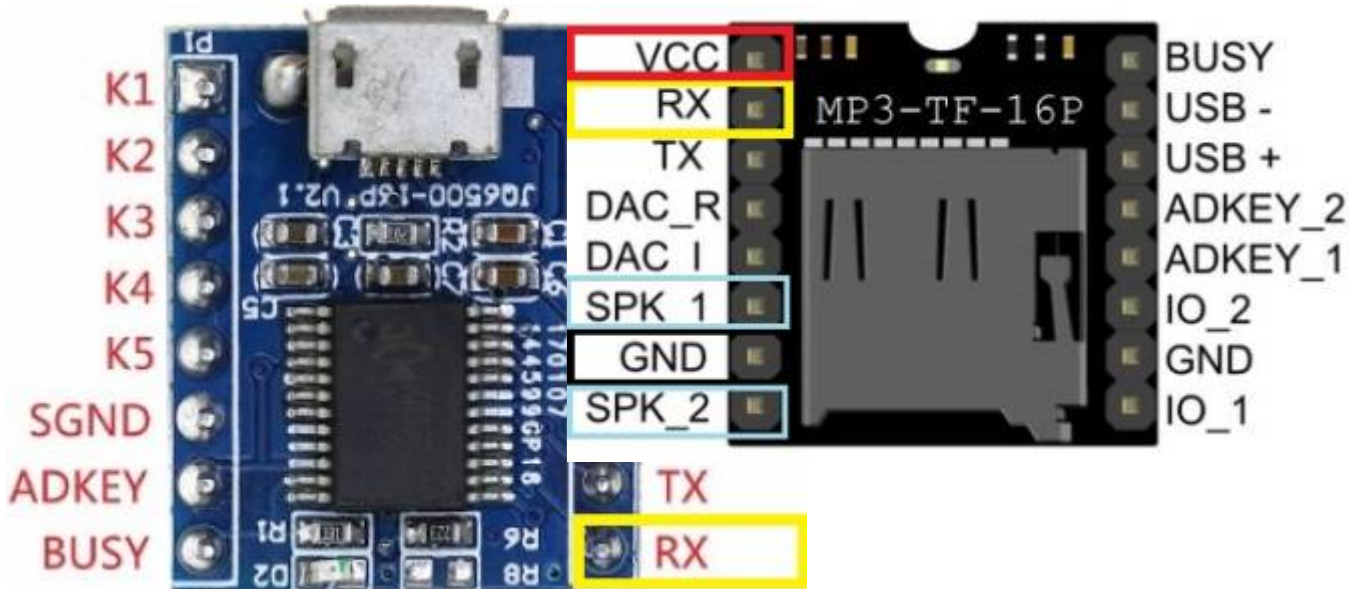
Achtung: Die Software für das Servo Modul ist noch in der Entwicklung.

Dialog schließen

Das folgende Bild zeigt die Pins am Ausgang der Servo-Platine. Der SIG-J1 - Pin ist mit dem Eingang des ersten Soundmodules zu verbinden, entsprechend J2 mit Modul 2 und J3 mit Modul3.



JQ6500 und MP3-TF16-p/DFPlayer Mini



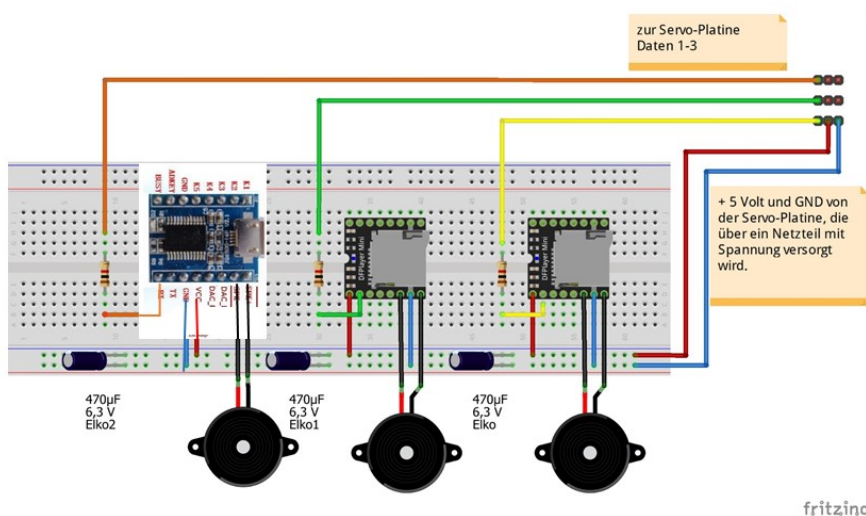
Verdrahtung

Die drei Ausgänge (SIG) der Servo-Platine werden nun über je einen 1 kOhm Widerstand mit den Eingängen (RX) der Sound-Module verbunden.

Die Versorgungsspannung sollte über eine Verteilerplatine mit angeschlossener stabiler Spannungsversorgung erfolgen, da die Leistung an der Hauptplatine zur Versorgung der Sound-Module nicht ausreicht.

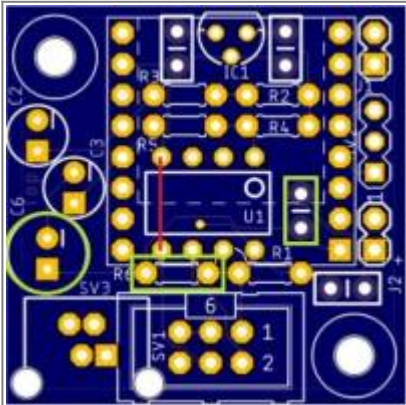
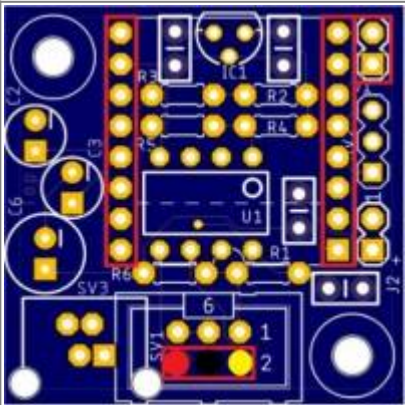
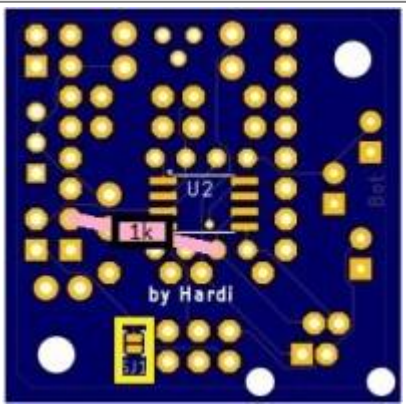
Die 470µF Elektrolytkondensatoren dienen als Puffer für die recht hohen Einschaltströme der Soundmodule.

Eine separate 5V Spannungsversorgung ist ebenfalls möglich.



Als weitere Möglichkeit bietet sich der Umbau einer unbestückten 501de_Soundplatine (MP3-TF-16p)

an:

		
Drahtbrücke (rot) an der Oberseite	Buchsenleisten: 2x8 Soundmodul	Widerstand 1kOhm an der Unterseite
R6 Widerstand 10Ohm	1x3 rot(VCC), schwarz(GND), gelb(SIG)	Jumper SJ1 schließen
Keramikkondensator 100nf	ACHTUNG-ServoAnschluss rot/schwarz vertauscht	
C6 Elko 470µF	1x2 Lautsprecher	

SD Karte

Die Verzeichnisstruktur der Sounddateien auf der SD-Karte ist relativ starr vorgeschrieben. Es können Sounddateien (wahlweise im mp3- oder wav-Format) in folgende Verzeichnisse der SD-Karte abgelegt werden:

- Wurzelverzeichnis (Dateinamen 4-stellig 0001.mp3 etc.)
- Verzeichnis ADVERT (Dateinamen 4-stellig)
- Verzeichnis mp3 (Dateinamen 4-stellig)
- Verzeichnisse 01 bis 32 - (Dateinamen 3-Stellig 001.mp3 etc.)

Es werden nur die ersten Zeichen der Dateinamen ausgewertet. Der tatsächliche Dateiname kann länger sein. Damit sind Dateinamen der Art

0023Yesterday.mp3

möglich. Dieser Dateiname wird vom Soundmodul als „0023.mp3“ behandelt. Das erleichtert den Umgang mit den Sounddateien deutlich und sollte ausgiebig genutzt werden.

Programm-Generator

Im Prog-Gen gibt es für den Servo-Sound folgende Befehle:

ATTiny85		Soundmodule über ATTiny85	
	Befehl an Soundmodul		Befehl an Soundmodul über Servoplatine
	Titel # abspielen		Track auf Soundmodul über Servoplatine abspielen
	Einstellungen		MP3-TF-16p einstellen
	Titel aus Ordner abspielen		MP3-TF-16p, Track aus Ordner abspielen
	Pin MP3-Modul definieren		Anschluss für MP3-Modul auswählen
	Soundmodul definieren		Typ für angeschlossenes MP3-Modul einstellen
	Titel # aus Hauptverzeichnis abspielen		Track # vom angegebenen Modul abspielen (Rootverzeichnis)
	Titel # aus mp3 abspielen		Track # aus mp3 auf Modul abspielen

Nach einer **Neuprogrammierung** des ATTiny sind die Ausgänge wie folgt vorbelegt:

SERV01	JQ6500	PIN 5 des ATTiny PB0(MOSI)
SERV02	MP3- TF- 16p	PIN 6 des ATTiny PB1(MISO)
SERV03	JQ6500	PIN 7 des ATTiny PB2(SCK/ADC1)

Möchte man andere Modultypen anschließen, so muss man **einmalig** die verwendeten Module mit dem Befehl <Soundmodul definieren> `MP3_SET_TYPE` einstellen.

Der ATTiny merkt sich diese Einstellung, daher kann man das einmal nach der Installation mit ein paar Zeilen im Programmgenerator machen.

Eine Änderung ist nur dann notwendig, wenn die angeschlossenen Modul-Typen verändert werden. Die Einstellungen werden im Beispiel mit den Tastern SwitchD1-D3 auf der Hauptplatine programmiert:

Aktiv	Filter	Adresse oder Name	Typ	Startwert	Beschreibung	Verteiler Nummer	Stecker Nummer	Name	Beleuchtung, Sound, oder andere Effekte	Start Lednr	LEDS	inCrt
<input checked="" type="checkbox"/>					LED auf dem Mainboard			Heartbeat LED	RGB_Heartbeat(#LED)	0	1	0
<input checked="" type="checkbox"/>		SwitchD1	Soundmodul Type Servo 1 definieren					Soundmodul definieren	MP3_SET_TYPE(#LED, #InCh, 1, MP3_TF_16p)	1	C1-1	1
<input checked="" type="checkbox"/>		SwitchD2	Soundmodul Type Servo 2 definieren					Soundmodul definieren	MP3_SET_TYPE(#LED, #InCh, 2, MP3_JQ6500)	1	C1-1	1
<input checked="" type="checkbox"/>		SwitchD3	Soundmodul Type Servo 3 definieren					Soundmodul definieren	MP3_SET_TYPE(#LED, #InCh, 3, MP3_TF_16p)	1	C1-1	1

neu - SERV01 J1 = MP3-TF-16p Modul
 neu - SERV02 J2 = JQ6500 Modul
 neu - SERV03 J3 = MP3-TF-16p Modul

Beispiel der Soundsteuerung mit den Befehlen:

99												
100	<input checked="" type="checkbox"/>		80	Rot				Soundmodul definieren	MP3_SET_TYPE(#LED, #InCh, 1, MP3_JQ6500)	1	C1-1	1 0 0
101	<input checked="" type="checkbox"/>		81	Rot				Soundmodul definieren	MP3_SET_TYPE(#LED, #InCh, 2, MP3_TF_16p)	1	C1-1	1 0 0
102	<input checked="" type="checkbox"/>		82	Rot				Soundmodul definieren	MP3_SET_TYPE(#LED, #InCh, 3, MP3_TF_16p)	1	C1-1	1 0 0
103												
104	<input checked="" type="checkbox"/>		90	Rot				Pin MP3-Modul definieren	MP3_SELECT_MODULE(#LED, #InCh, 1)	1	C1-1	1 0 0
105	<input checked="" type="checkbox"/>		91	Rot				Pin MP3-Modul definieren	MP3_SELECT_MODULE(#LED, #InCh, 2)	1	C1-1	1 0 0
106	<input checked="" type="checkbox"/>		92	Rot				Pin MP3-Modul definieren	MP3_SELECT_MODULE(#LED, #InCh, 3)	1	C1-1	1 0 0
107												
108	<input checked="" type="checkbox"/>		93	Rot				Befehl an Soundmodul	MP3_CHD(#LED, #InCh, MP3_DECREASE_VOLUME)	1	C1-1	1 0 0
109	<input checked="" type="checkbox"/>		94	Rot				Befehl an Soundmodul	MP3_CHD(#LED, #InCh, MP3_INCREASE_VOLUME)	1	C1-1	1 0 0
110	<input checked="" type="checkbox"/>		95	Rot				Titel # abspielen	MP3_TRACK(#LED, #InCh, MP3_PLAY_TRACK, 1)	1	C1-1	1 0 0
111	<input checked="" type="checkbox"/>		96	Rot				Titel # abspielen	MP3_TRACK(#LED, #InCh, MP3_PLAY_TRACK, 2)	1	C1-1	1 0 0
112												
113												
114												
115												

In den Zeilen 100-102 wird der an die Sound-Platine angeschlossene Modultyp festgelegt. Diese Befehle müssen nur einmalig gesendet werden. Der ATtiny speichert die Einstellung. Eine Änderung ist nur dann notwendig, wenn Änderungen bei den angeschlossenen Modul-Typen vorgenommen werden. Im Beispiel

- Ch1 → JQ6500
- Ch2 → MP3-TF16-p
- Ch3 → MP3-TF16-p

Zeile 104 - 106: legt den Ausgang fest auf den der nächste Befehl gehen soll. Hier:

- Zeile 104 - Ausgang 1, JQ6500.
- Zeile 105 - Ausgang 2, MP3-TF16-p
- Zeile 106 - Ausgang 3, MP3-TF16-p

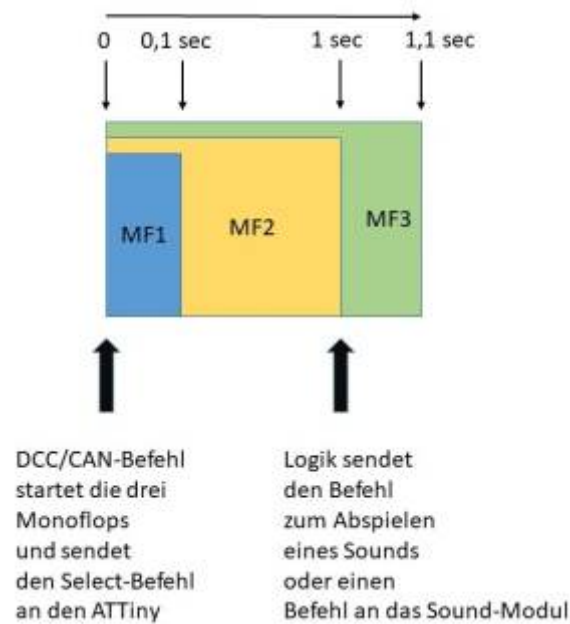
Beispiel:

- DCC Befehl 90 und anschließend 95 spielt Track 1 vom JQ6500 ab.
- DCC Befehl 91 und anschließend 95 spielt Track 1 vom ersten MP3-TF16-p ab.
- DCC Befehl 92 und anschließend 95 spielt Track 1 vom zweiten MP3-TF16-p ab.
- DCC Befehl 91 und anschließend 93 verringert die Laustärke beim zweiten MP3-TF16-p.

Anmerkungen:

- In der Macroauswahl wird der Hinweis „Sendet einen Befehl an ein MP3-TF-16p-Soundmodul welches über SERVO3 an einer Servoplatine angeschlossen ist“ gegeben. Das trifft so nicht zu, da die Befehle an alle drei Ausgänge gesendet werden können und auch für beide Modultypen gelten. Einschränkungen gibt es natürlich beim JQ 6500, da nur fünf Tracks gespeichert werden können.
- Alle Befehle an diese ATtiny Soundplatine müssen über eine LED Adresse laufen, im Beispiel LED 1. Sollte es (ungewollt) eine Verschiebung geben mit dem Befehl „next LED -1“ wieder auf die LED Adresse der Sound-Platine zurück gehen. [Hilfe](#)
- Über den [Kleinen Verteiler](#) mit der Copy-Funktion, Einstellung über den Jumper, kann man parallel Test-LEDs anschließen und optisch die Funktion überprüfen.

Steuerung über DCC/CAN-Befehle



Vor jedem Sound-Befehl kann das Modul ausgewählt werden auf dem sich die Sound-Datei befindet. So können Sound-Dateien in unterschiedlicher Reihenfolge von den drei Modulen abgespielt werden. Die Logik stellt sicher, dass zunächst über den ATTiny das Modul ausgewählt wird, der Befehl umgesetzt werden kann und dann, mit zeitlichem Verzug, der Track ausgewählt oder eine andere Funktion des Moduls aufgerufen wird.

Anmerkung:

Mit den Befehlen <Titel # aus Hauptverzeichnis abspielen> `MP3_PLAY_TRACK_ON` und <Titel # aus mp3 abspielen> `MP3_PLAY_MP3_ON` wird das Modul mit den eingetragenen Parametern bereits direkt angesprochen und der gewählte Sound abgespielt. Leider funktioniert das mit dem Modul JQ6500 nicht zuverlässig. Hier sollte trotzdem eine Vorauswahl mittels Monoflops (wie im Beispiel) getroffen werden.

Beispiele:

Im folgenden Beispiel wird bei Aufruf des:

- DCC-Befehl „3“ der erste Sound des ersten Moduls (JP6500) abgerufen
- DCC-Befehl „4“ der erste Sound des zweiten Moduls (DFPlayer Mini) abgerufen
- DCC-Befehl „5“ der erste Sound des dritten Moduls (DFPlayer Mini) abgerufen

Dialog Z. Arduino schicken Zeile einfügen Lösche Zeilen Verschiebe Zeilen Kopiere Zeilen Aus- oder Einblenden Alle Einblenden Lösche Tabelle Optionen Help Ver. 3.1.0 by Hardi

Aktiv	Filter	Adresse oder Name	Typ	Startwert	Beschreibung	Verteiler-Nummer	Stecker-Nummer	Icon	Name	Beleuchtung, Sound, oder andere Effekte	Start LedNr	LEDs	InCh	Loc InCh	LED/ Sound Kanal
					Modul 1 JQ6500										
✓		3	Rot		startet MonoFlo (MF) 1			⚡	! Mono-Flop	MonoFlo(MF11, #InCh, 0.1 Sek)			1	0	
✓		MF11	Rot		sendet Select Befehl für Modul 1			⚡	Pin MP3-Modul definieren	MP3_SELECT_MODULE(#LED, #InCh, 1)	1	^ C1-1	1	0	0
✓		3	Rot		startet MonoFlo (MF) 3			⚡	! Mono-Flop	MonoFlo(MF13, #InCh, 1.1 Sek)			1	0	
✓		MF12	Rot		logische Verknüpfung der drei MFs			⚡	! Logische Verknüpfung	Logic(MF12u13, NOT #InCh AND MF13)			1	0	
✓		MF12u13	Rot		spielt Track 1 vom JQ6500 ab (Welding)			🎵	! Titel # abspielen	MP3_TRACK(#LED, #InCh, MP3_PLAY_TRACK, 1)	1	^ C1-1	1	0	0
					Modul 2 DFPlayer Mini										
✓		4	Rot		startet MF1			⚡	! Mono-Flop	MonoFlo(MF21, #InCh, 0.1 Sek)			1	0	
✓		MF21	Rot		sendet Select Befehl für Modul 2			⚡	Pin MP3-Modul definieren	MP3_SELECT_MODULE(#LED, #InCh, 2)	1	^ C1-1	1	0	0
✓		4	Rot		startet MF2			⚡	! Mono-Flop	MonoFlo(MF22, #InCh, 1 Sek)			1	0	
✓		5	Rot		startet MF3			⚡	! Mono-Flop	MonoFlo(MF23, #InCh, 1.1 Sek)			1	0	
✓		MF22	Rot		logische Verknüpfung der drei MFs			⚡	! Logische Verknüpfung	Logic(MF22u23, NOT #InCh AND MF23)			1	0	
✓		MF22u23	Rot		spielt Track 1 aus Root von DFPlayer 1 ab			🎵	! Titel # abspielen	MP3_TRACK(#LED, #InCh, MP3_PLAY_TRACK, 1)	1	^ C1-1	1	0	0
					Modul3 DFPlayer Mini										
✓		5	Rot		startet MF1			⚡	! Mono-Flop	MonoFlo(MF31, #InCh, 0.1 Sek)			1	0	
✓		MF31	Rot		sendet Select Befehl für Modul 3			⚡	Pin MP3-Modul definieren	MP3_SELECT_MODULE(#LED, #InCh, 3)	1	^ C1-1	1	0	0
✓		5	Rot		startet MF2			⚡	! Mono-Flop	MonoFlo(MF32, #InCh, 1 Sek)			1	0	
✓		5	Rot		startet MF3			⚡	! Mono-Flop	MonoFlo(MF33, #InCh, 1.1 Sek)			1	0	
✓		MF32	Rot		logische Verknüpfung der drei MFs			⚡	! Logische Verknüpfung	Logic(MF32u33, NOT #InCh AND MF33)			1	0	
✓		MF32u33	Rot		spielt Track 1 aus Root von DFPlayer 2 ab			🎵	! Titel # abspielen	MP3_TRACK(#LED, #InCh, MP3_PLAY_TRACK, 1)	1	^ C1-1	1	0	0
					Welding Modul 1 JQ6500										
✓		MF12u13	Rot		Imported_Pattern (pc)			⚡	Muster Pattern_Configurator	// Activation: N_Buttons1InCh to TempVar1(#InCh, 1)PatternT26	2	1	1	0	0

Über parallel angeschlossene Test-LEDs, Stichwort Mini-Verteiler, kann der Ablauf optisch sehr gut verfolgt werden. Andere/kürzere Zeitintervalle für die MonoFlops sind möglich und ggf. durch Tests zu ermitteln.

In Zeile 133 wird über die Variable MF12u13 zeitgleich mit dem dazu gehörigen Geräusch vom Sound-Modul 2 ein Schweißlicht ausgelöst. Geräuschlänge und Länge des Lichts können leicht durch Anpassung des **Schweisslicht** über den Pattern-Configurator angepasst werden.

Damit man den Schweißler nicht immer persönlich wecken muss, hier eine Lösung mit der Zufallsschaltung (Random-Funktion).

					Modul 1 JQ6500										
✓		3	AnAus					🎲	Zufallsschaltung 1 Ausgang	Random(WL1, #InCh, RN_NORMAL, 3 sec, 15 sec, 1 sec, 1 sec)			1	0	
✓		WL1	Rot		startet MonoFlo (MF) 1			⚡	! Mono-Flop	MonoFlo(MF11, #InCh, 0.1 Sek)			1	0	
✓		MF11	Rot		sendet Select Befehl für Modul 1			⚡	Pin MP3-Modul definieren	MP3_SELECT_MODULE(#LED, #InCh, 1)	1	^ C1-1	1	0	0
✓		WL1	Rot		startet MonoFlo (MF) 2			⚡	! Mono-Flop	MonoFlo(MF12, #InCh, 1 Sek)			1	0	
✓		WL1	Rot		startet MonoFlo (MF) 3			⚡	! Mono-Flop	MonoFlo(MF13, #InCh, 1.1 Sek)			1	0	
✓		MF12	Rot		logische Verknüpfung der drei MFs			⚡	! Logische Verknüpfung	Logic(MF12u13, NOT #InCh AND MF13)			1	0	
✓		MF12u13	Rot		spielt Track 1 vom JQ6500 ab (Welding)			🎵	! Titel # abspielen	MP3_TRACK(#LED, #InCh, MP3_PLAY_TRACK, 1)	1	^ C1-1	1	0	0
✓		MF12u13	Rot		Imported_Pattern (pc)			⚡	Muster Pattern_Configurator	// Activation: N_Buttons1InCh to TempVar1(#InCh, 1)PatternT26	2	1	1	0	0
✓								🔧	LED Nummer manipulieren	// Next_LED(-2)	3	-2	0	0	0
					Modul 2 DFPlayer Mini										
✓		4	Rot		startet MF1			⚡	! Mono-Flop	MonoFlo(MF21, #InCh, 0.1 Sek)			1	0	
✓		MF21	Rot		sendet Select Befehl für Modul 2			⚡	Pin MP3-Modul definieren	MP3_SELECT_MODULE(#LED, #InCh, 2)	1	^ C1-1	1	0	0
✓		4	Rot		startet MF2			⚡	! Mono-Flop	MonoFlo(MF22, #InCh, 1 Sek)			1	0	
✓		4	Rot		startet MF3			⚡	! Mono-Flop	MonoFlo(MF23, #InCh, 1.1 Sek)			1	0	
✓		MF22	Rot		logische Verknüpfung der drei MFs			⚡	! Logische Verknüpfung	Logic(MF22u23, NOT #InCh AND MF23)			1	0	
✓		MF22u23	Rot		spielt Track 1 aus Root von DFPlayer 1 ab			🎵	! Titel # abspielen	MP3_TRACK(#LED, #InCh, MP3_PLAY_TRACK, 1)	1	^ C1-1	1	0	0
					Modul3 DFPlayer Mini										
✓		5	Rot		startet MF1			⚡	! Mono-Flop	MonoFlo(MF31, #InCh, 0.1 Sek)			1	0	
✓		MF31	Rot		sendet Select Befehl für Modul 3			⚡	Pin MP3-Modul definieren	MP3_SELECT_MODULE(#LED, #InCh, 3)	1	^ C1-1	1	0	0
✓		5	Rot		startet MF2			⚡	! Mono-Flop	MonoFlo(MF32, #InCh, 1 Sek)			1	0	
✓		5	Rot		startet MF3			⚡	! Mono-Flop	MonoFlo(MF33, #InCh, 1.1 Sek)			1	0	
✓		MF32	Rot		logische Verknüpfung der drei MFs			⚡	! Logische Verknüpfung	Logic(MF32u33, NOT #InCh AND MF33)			1	0	
✓		MF32u33	Rot		spielt Track 1 aus Root von DFPlayer 2 ab			🎵	! Titel # abspielen	MP3_TRACK(#LED, #InCh, MP3_PLAY_TRACK, 1)	1	^ C1-1	1	0	0

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

Permanent link: https://wiki.mobaledlib.de/anleitungen/spezial/sound_servoplatine

Last update: **2024/09/29 19:37**

