

Ansteuerung von Formsignalen mit Ministeppern

Formsignale mit dem Mini-Stepper bewegen

Quelle: MLL Stammtisch Mai 2021

Tipps zu den Platinen, zum Einbau des Steppers und erste grundsätzliche Überlegungen zur Steuerung mit der MLL sind vom Anfang des Videos bis Minute 32 zu finden. Ab Minute 33 wird die Steuerung des zweibegriffigen Formsignals vorgeführt.

Nachträglich hat sich jedoch leider herausgestellt, dass die Einstellungen nicht oder nicht zuverlässig unkontrollierte Bewegungen des Signals beim Einschalten oder Restart verhindern. Gerade bei der Nutzung von Programmen wie ROCRAIL ist im automatischen oder halbautomatischen Betrieb ein definierter Anfangszustand sinnvoll.

Die Einstellungen wurden daher gegenüber der Stammtischversion geändert.

Benötigte Teile:

- Hauptplatine über USB- Kabel mit PC verbunden
- Stromversorgung für Steppermotor
- Formsignal mit eingebautem Mini-Steppermotor
- Bestückte Stepperplatine mit einer (550) oder drei Anschlussmöglichkeiten (551) für Steppermotoren

Ablauf: Den Pattern-Configurator aufrufen und, wenn bisher noch nicht durchgeführt, aus den Beispielen die Signale laden.

Das „Dep Signal4“ Beispiel aufrufen und über die Excel-Funktionen von diesem Beispiel eine Kopie anlegen. Im Bild ist die Kopie unter „Formsignal“ abgespeichert. Das Beispiel wird als Grundlage verwendet.

Nun in das Blatt „Formsignal“ wechseln und die folgenden Werte eintragen:

Für die Steuerung des Steppers brauchen wir zwei Steuerbefehle:

- Stepper ein-/ausschalten – über den roten Kanal mit dem Wert 127
- Drehrichtung des Steppers – über den gelben Kanal, Wert 255 eine Drehrichtung, Wert 0 entgegengesetzt

Anmerkung: Stimmen später im Prog-Gen die Tasten GRÜN oder ROT nicht mit der Einstellung des Signals überein, kann man hier einfach Grün in der zweiten Spalte auf 255 (oder x entspricht 255) und in der vierten Spalte auf 0 (oder . entspricht 0) setzen. Anschließend erneut zum Prog-Gen senden.

Der blaue Kanal kann zur Steuerung einer LED verwendet werden, z.B. der Signalbeleuchtung. Diese Funktion nutze ich im Prog-Gen, nicht hier im Pattern-Config.

Den Wert für die Ausgabekanäle auf „2“ stellen. In der unteren Tabelle könne die Werte für die anderen Ausgabekanäle gelöscht werden.

Für die anderen Einstellungen sind folgende Werte einzutragen bzw. zu ändern:

- Bits pro Wert: „8“
- Mode: „PM_SEQUENZ_NO_RESTART“ (Flanken getriggerte einmalige Sequenz. Kein Neustart während der Laufzeit)
- Analoges Überblenden: „0“ (schaltet das Überblenden ab)
- Goto Aktivierung: „N_OneTimeBut1“

Bild Prog-Gen einfügen

Bild Goto Aktivierung einfügen

Über die Taste „Programm Generator“ die Daten nun zum Prog-Gen schicken.

Im Prog-Gen die Funktion an der gewünschten Stelle einfügen. Hier wurde ein neues Blatt zu Testzwecke angelegt.

Zunächst nur die Zeilen 3, 16 (Heartbeat) und 11 bis 13 aktivieren und zum Arduino schicken. Diese Zeilen dienen zur erstmaligen Einstellung des Signals. Zeile 11 ist der rote Kanal, der den Stepper über „ ROT/GRÜN) ein- bzw. ausschaltet. Über Zeile 12 kann die Drehrichtung bestimmt werden. Eingeschaltet (GRÜN) wird der Wert 255 gesendet, ausgeschaltet (ROT) der Wert 0 und damit die entgegengesetzte Drehrichtung.

Zeile 13 dient zum Schalter der Signalbeleuchtung. Diese Funktion hat keinen Einfluss auf den Stepper und muss nicht genutzt werden.

Nach der Einstellung des Signals können die Zeilen 11 bis 13 deaktiviert und die Zeilen 6, 8, 9 aktiviert werden.

ACHTUNG: Mit der Zeile 6 wird der letzte Zustand gespeichert „#define ENABLE_STORE_STATUS()“. Damit wird erreicht, dass beim nächsten Einschalten der Anlage, bei einem Reset des Nano oder nach Unterbrechung der Stromversorgung der letzte Zustand wieder eingenommen wird.

(Die letzten Zustände bei Signalen oder anderen per DCC, Selectrix oder CAN gesteuerten Funktionen werden gespeichert und beim nächsten Start wieder aktiviert. Wenn der Modus nicht aktiviert ist, dann sind die entsprechenden Funktionen abgeschaltet bzw. beginnen mit dem in der Spalte „Start Wert“ definierten Zustand.)

Diese Funktion funktioniert hier **NICHT!** Wenn sie im Prog-Gen bereits vorhanden ist für andere Funktionen genutzt wird, muss sie durch eine „0“ in der Zeile „Startwert“ für die Signalfunktion abgeschaltet werden.

Zeile 8 beinhaltet die im Pattern-Configurator erzeugte und importierte Funktion.

Nach der Eingabe der Adresse muss der Typ als „Grün“ oder „Rot“ definiert werden

Mit Zeile 9 kann bei Bedarf eine LED, z.B. die Signalbeleuchtung, geschaltet werden.

<https://wiki.mobaledlib.de/playground/playground>

ALTER TEXT Quelle: **MLL Stammtisch Mai 2021**

Tipps zu den Platinen, zum Einbau des Steppers und erste grundsätzliche Überlegungen zur Steuerung mit der MLL sind vom Anfang des Videos bis Minute 32 zu finden. Ab Minute 33 wird die

Steuerung des zweibegriffigen Formsignals vorgeführt, die im Folgenden erläutert wird.

Benötigte Teile:

- Hauptplatine über USB- Kabel mit PC verbunden
- Stromversorgung für Steppermotor
- Formsignal mit eingebautem Mini-Steppermotor
- Bestückte Stepperplatine mit einer (550) oder drei Anschlussmöglichkeiten (551) für Stepper-Motoren

Ablauf:

Den Pattern-Configurator aufrufen und, wenn bisher noch nicht durchgeführt, aus den Beispielen die Signale laden. Das „Dep Signal4“ Beispiel aufrufen und über die Schaltfläche **Neues Blatt** eine Kopie anlegen. Die Einstellungen übernehmen und einen neuen Namen vergeben, hier „Formsignal (2)“. Das Beispiel wird als Grundlage verwendet.

Nun in das neue angelegte Blatt wechseln.

Neues Blatt
by Harald

Erste RGB LED: 1
 Startkanal der RGB LED: 0
 Schalter Nummer: SI_1
 Anzahl der Ausgabe Kanäle: 2
 Bits pro Wert: 8 => 256 Helligkeitsstufen (0..255)
 Wert Min: 0
 Wert Max: 255
 Wert ausgeschaltet: 0
 Mode: PM_SEQUENZ_NO_RESTART
 Analoges Überblenden: 0
 Goto Mode: 1
 Goto Aktivierung: Binary
 Grafische Anzeige: 1
 Spezial Mode:

Aktualisieren
Stop

Ergebnis: **PatternT1(1,28,SI_LocalVar,2,0,255,0,PM_SEQUENZ_NO_RESTART,4**

Makro Name: **_Dep_Signal4**

Makro: **#define _Dep_Signal4(LED) PatternT1(LED,28,SI_LocalV**
#define _Dep_Signal4_StCh(LED,StCh) PatternT1(LED,StCh+28,SI_Loc

Wenn gleiche Zeiten verwendet werden, dann sollten nur die ersten Zeite

Dauer	4 sec					
-------	-------	--	--	--	--	--

Flash Bedarf: 21 Bytes

Goto Tabelle

RGB LED

LED Nr	Spalte Nr ->	1	2	3	4	5	6
1	Rot 1	127		127			
2	Grün	x			.		

Formsignal (2) |
 Formsignal |
 Welding |
 Lok-Schuppen |
 Dep Signal4 RGB |
 Dep

Für die Steuerung des Steppers brauchen wir zwei Steuerbefehle:

- Stepper ein-/ausschalten - über den roten Kanal (LED Nr 1) mit dem Wert 127
- Drehrichtung des Steppers - über den grünen Kanal (LED Nr 2), Wert 255 eine Drehrichtung, Wert 0 entgegengesetzt

Der blaue Kanal, das wäre dann die LED Nr 3 wenn die Anzahl der Ausgabekanäle auf „3“ erhöht wird, kann zur Steuerung einer LED verwendet werden, z.B. der Signalbeleuchtung. Diese Funktion nutze ich im Prog-Gen, nicht hier im Pattern-Config.

Daher den Wert für die **Ausgabekanäle auf „2“** einstellen. In der unteren Tabelle könne die Werte für die anderen Ausgabekanäle gelöscht werden.

Für die anderen Einstellungen sind folgende Werte einzutragen bzw. zu ändern:

- Anzahl der Ausgabe Kanäle: 2
- Bits pro Wert: 8
- Mode: „PM_SEQUENZ_NO_RESTART“ (Flanken getriggert ein malige Sequenz. Kein Neustart während der Laufzeit)
- Analoges Überblenden: „0“ (schaltet das Überblenden ab)
- Goto Aktivierung: „Binary“ (Bild einfügen, mit dieser Einstellung kann dann im Prog-Gen der Ein/Aus-Schalter gewählt werden, um ein mehrfaches Bewegen der Signalfügel in die selbe Richtung auszuschließen.)

Auswahl der Goto Aktivierung ✕

Goto Aktivierung auswählen:

Wenn der Goto Mode verwendet wird, dann kann das Muster an verschiedenen Stellen gestartet werden. Diese Spalten sind in der Tabelle mit nummerierten Pfeilen markiert.
Mit diesem Dialog wird definiert wie die Startspalte im Betrieb ausgewählt wird.

Name	Beschreibung
N_Buttons	Bis zu 64 Taster steuern die Startspalte. Start bei Goto 0.
N_Buttons1	Bis zu 64 Taster steuern die Startspalte. Start bei Goto 1.
N_OneTimeBut	Bis zu 64 Taster steuern die Startspalte. Start bei Goto 0. Startposition wird nur ein mal aktiviert.
N_OneTimeBut1	Bis zu 64 Taster steuern die Startspalte. Start bei Goto 1. Startposition wird nur ein mal aktiviert.
Binary	Die Startposition wird durch die binäre Kombination von bis zu 6 Eingänge bestimmt.
Binary1	Die Startposition wird durch die binäre Kombination von bis zu 6 Eingänge bestimmt. Start bei Goto 1
Counter	Der Startpunkt wird über einen Zähler bestimmt welcher z.B. von einem Taster erhöht wird.
RandButton	Beim Betätigen eines Tasters wird eine zufällige Startposition ausgewählt
RandomTime	Nach einer zufälligen Zeit wird eine neue zufällige Startposition ausgewählt
RandomCount	Nach einer zufälligen Zeit wird die nächste Startposition ausgewählt
RandomPingPong	Nach einer zufälligen Zeit wird die nächste Startposition ausgewählt. Die Zählrichtung wird am ende umgedreht.
Nothing	Keine Goto Aktivierung verwenden. Sie ist in der vorangegangenen Zeile generiert.

Bei manche Zentralen können meherer Einzelne Zustände zu einem gemeinsahmen Zustand zusammengefasst werden. Damit reduziert sich die Anzahl der benötigten Adressen. Die einzelnen Eingänge werden Binär zu einer Zahl zusammengefasst.

Löschen
Abbruch
Auswahl

Über die Taste „**Programm Generator**“ die Daten zum Prog-Gen schicken. Im Prog-Gen die Funktion an der gewünschten Stelle einfügen.

Aktiv	Filter	Adresse oder Name	Typ	Start wert	Beschreibung	Verteiler Nummer	Stecker Nummer	Name	Beleuchtung, Sound, oder andere Effekte	Start LedNr	LEDs	InCh	Loc	LED/ Sound/ Vout
								Heartbeat LED_einstellbar	RGB_Heartbeat2(#LED, 5, 100)	0	1	0	0	0
									#define ENABLE_STORE_STATUS()					0
		1	AnAus		_Dep_Signall4 (pc)			Muster Pattern_Configurator	// Activation: Binary Bin_InCh_to_TmpVar(#InCh, 1) Pattern1(#LED,28,S1_LocalVar,2,0,255,0,PM_SEQUENZ_NO_RESTAR T.d_sec:127,0,0,0,127,255,0,0 ,0,63,128,63)	1	C1-2	1	0	0
		2	AnAus					LED_einstellbar	Const(#LED, C3, #InCh, 0, 40)	1	C3-3	1	0	0
		3	AnAus					LED_einstellbar	Const(#LED, C1, #InCh, 0, 127)			C1-1	1	0
		4	AnAus					LED_einstellbar	Const(#LED, C2, #InCh, 0, 255)			C2-2	1	0
		5	AnAus					LED_einstellbar	Const(#LED, C3, #InCh, 0, 50)			C3-3	1	0

Für Testzwecke habe ich ein neues Blatt angelegt.

- Zunächst nur die Zeilen **4** (Heartbeat) und **11 bis 13** aktivieren und zum Arduino schicken.

Diese Zeilen sind zur erstmaligen Einstellung des Signals vorgesehen.

- Zeile **11** ist der rote Kanal, der den Stepper über ROT/GRÜN ein- bzw. ausschaltet.
- Über Zeile **12** kann die Drehrichtung bestimmt werden. Eingeschaltet (GRÜN) wird der Wert 255 gesendet, ausgeschaltet (ROT) der Wert 0 und damit die entgegengesetzte Drehrichtung.
- Zeile **13** dient zum Schalten der Signalbeleuchtung. Diese Funktion hat keinen Einfluss auf den Stepper und muss nicht genutzt werden.

Nach der Grundeinstellung des Signals können die Zeilen 11 bis 13 deaktiviert, die Zeilen 6, 8, 9 aktiviert und zum Arduino geschickt werden.

- Mit der Zeile **6** wird der letzte Zustand gespeichert „#define ENABLE_STORE_STATUS()“. Damit wird erreicht, dass beim nächsten Einschalten der Anlage, bei einem Reset des Nano oder nach Unterbrechung der Stromversorgung der letzte definierte Zustand wieder eingenommen wird. *(Die letzten Zustände bei Signalen oder anderen per DCC, Selectrix oder CAN gesteuerten Funktionen werden gespeichert und beim nächsten Start wieder aktiviert. Wenn der Modus nicht aktiviert ist, dann sind die entsprechenden Funktionen abgeschaltet bzw. beginnen mit dem in in der Spalte „Start Wert“ definierten Zustand.)*
- Zeile 8 beinhaltet die vom Pattern-Config erzeugte Funktion. Nach der Eingabe der Adresse muss der Typ als „**Ein/Aus**“ definiert werden, nicht als Taster!, um eine mehrfache Bewegung der Flügel des Signals in eine Richtung auszuschließen.
- Mit Zeile **9** kann bei Bedarf eine LED, z.B. die Signalbeleuchtung, geschaltet werden.

From:

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

Permanent link:

<https://wiki.mobaledlib.de/anleitungen/stepper/signale?rev=1640103022>

Last update: **2021/12/21 17:10**

