

# DCC-EX v 5.0.9

03d - Erweiterungen - NANO



## Tests wurden unter Windows 11 durchgeführt

Die folgenden Einstellungen und Vorgehensweise kann natürlich jeder selbst bestimmen.  
Es sind nur meine Erfahrungen.

Wichtig.....	2
Arduino NANO vorbereiten .....	3
Vorbereitung: Arduino MEGA .....	3
Vorbereitung: Arduino NANO.....	5
Vorbereitung: Arduino NANO mit altem Bootloader.....	8
Gesamt Ergebnis mit einem Arduino NANO .....	9
Händische Einrichtung .....	10
Einstellungen für den Arduino NANO .....	10
Einrichtung in der Datei „myAutomation.h“ / „mySetup.h“ .....	15
Gerätetreiber .....	15
Befehle für Eingänge .....	16
Befehle für Ausgänge .....	18

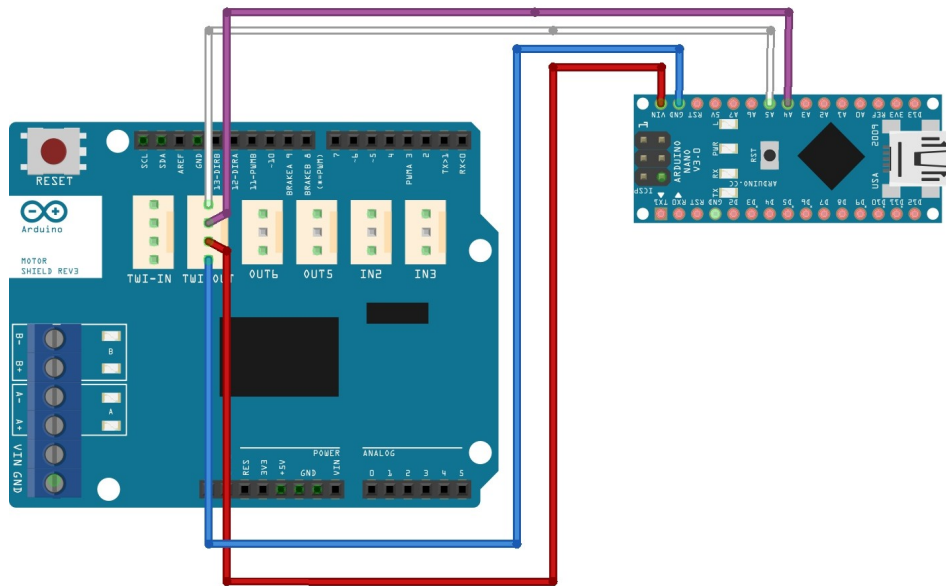
## Wichtig

- **Ich übernehme keine Garantie/ Haftung auf Richtigkeit, Vollständigkeit usw. Es beruht alles auf eigener Erfahrung.**
- **Bei den nachfolgenden Schaltungs-/ Anschlussbeispielen ist unbedingt selbst auf die Pinbelegung zu achten, es gibt Bauelemente mit gleichen Daten aber anderer Pinbelegung.**
- **Auch auf die Spannungsversorgung muss unbedingt selbst geachtet werden, manche Komponenten benötigen 3,3 Volt, andere aber 5 Volt usw.**
- **Bei der Versorgungsspannung für den Arduino über USB unbedingt darauf achten, daß es nicht zu einer Überlastung des speisenden Raspberry Pi kommt.**

**Ich habe daher, nachdem ich die Hardwareinstallation (siehe DCC-EX Beschreibung) vorgenommen habe, zusätzlich zum USB-Anschluss des Arduinos noch ein Netzteil für den Arduino und ein weiteres natürlich für die Gleisspannung vorgesehen.**



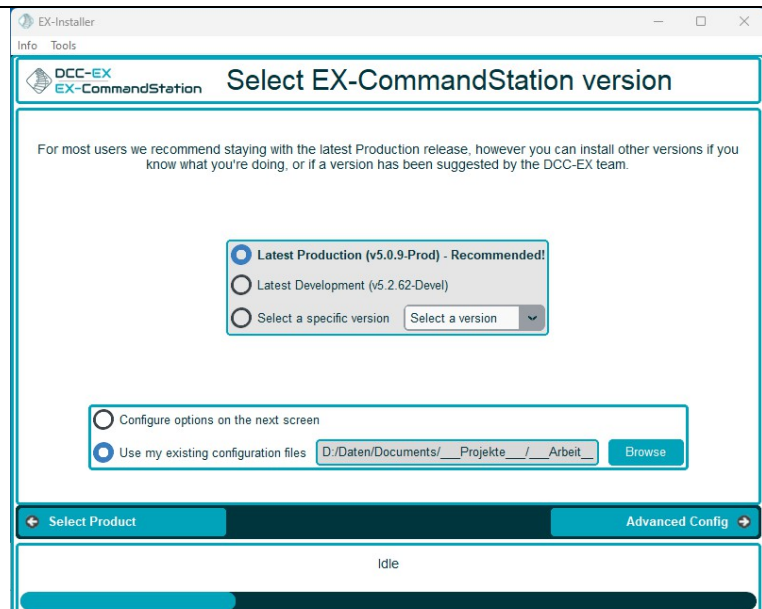
## Arduino NANO vorbereiten



fritzing

**Eigene Anschlussbelegung beachten!**

## Vorbereitung: Arduino MEGA

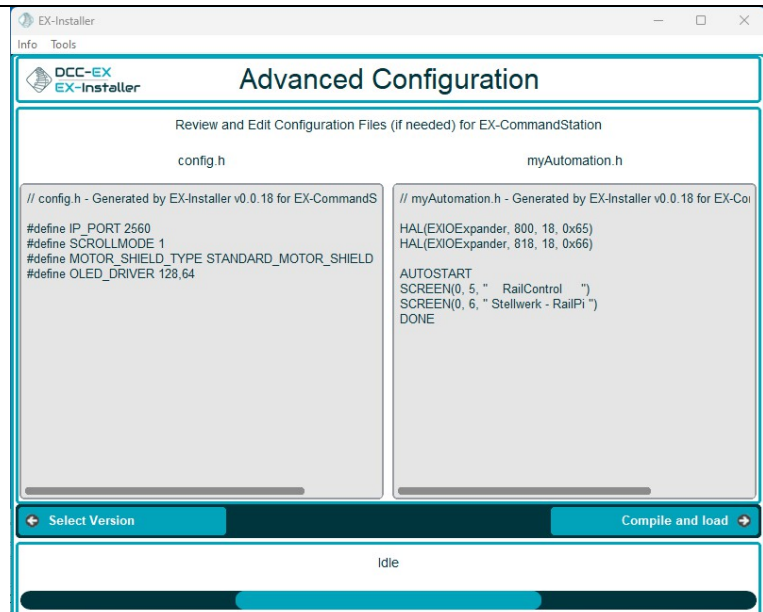
**Installation mit EX-Installer****Die ersten Fenster sind bereits unter „Standard Installation“ beschrieben****Laden der eventuell vorhande-  
nen Back-up Dateien**

**Bei mir**

**Danach folgende Einstellungen vornehmen:**

- **HAL(EXIOExpander, 800, 18, 0x65)**
- **HAL(EXIOExpander, 818, 18, 0x66)**

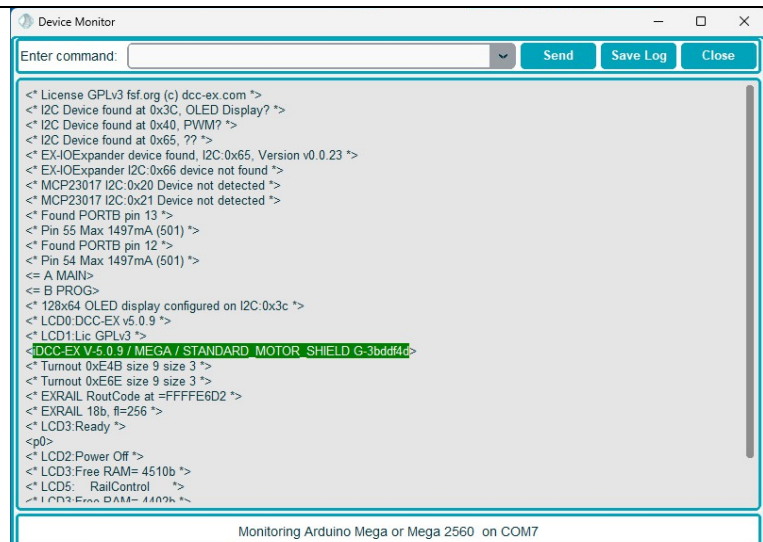
**Hier für 2 Arduino's NANO**



**Die weiteren Fenster bzw. die Installationsschritte sind wie unter Kapitel 1 „02 - Standardinstallation.pdf“ beschrieben durchzuführen**

**Ergebnis in einem Terminal:**

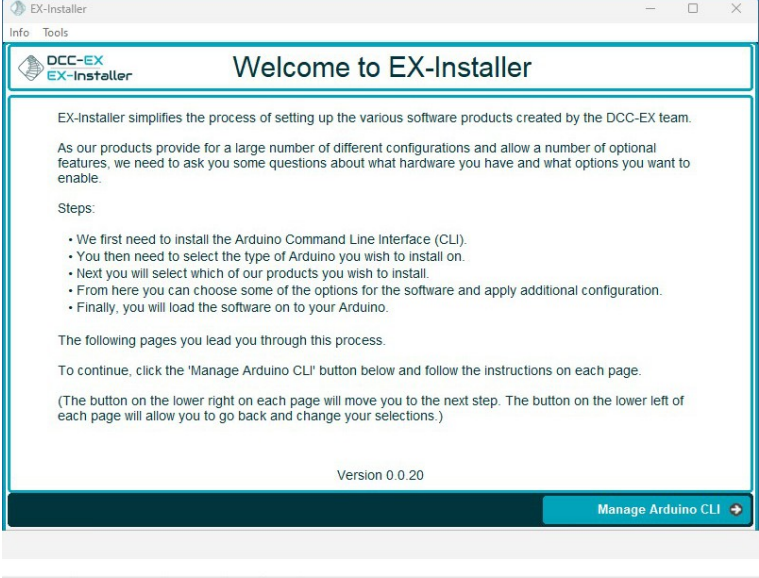
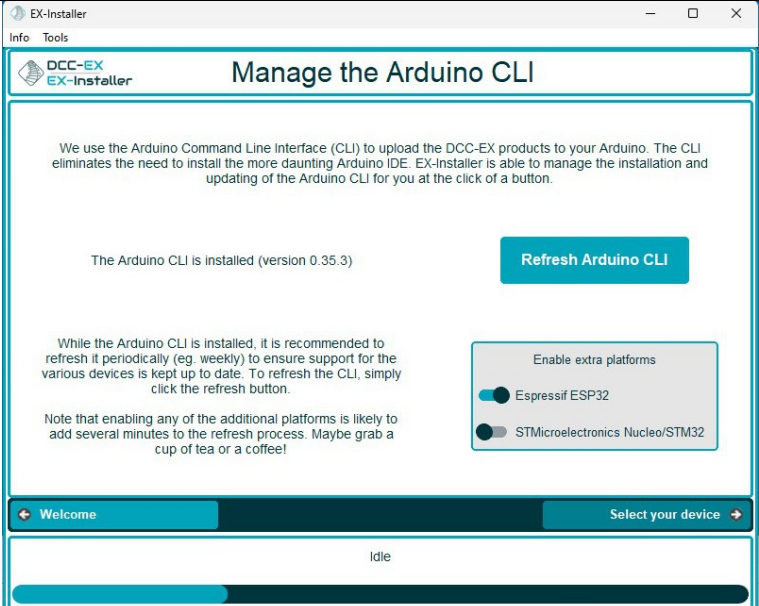
**Man sieht daß das Device 0x65 am I2C Bus gefunden wurde.**



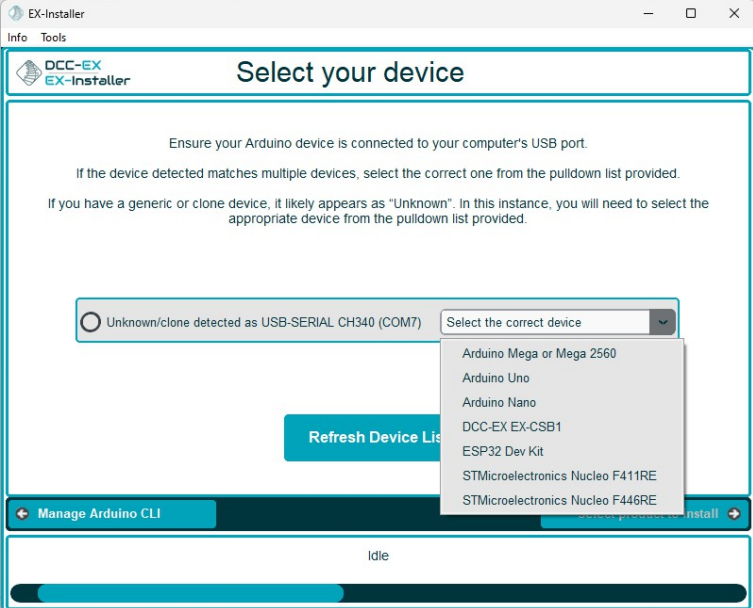
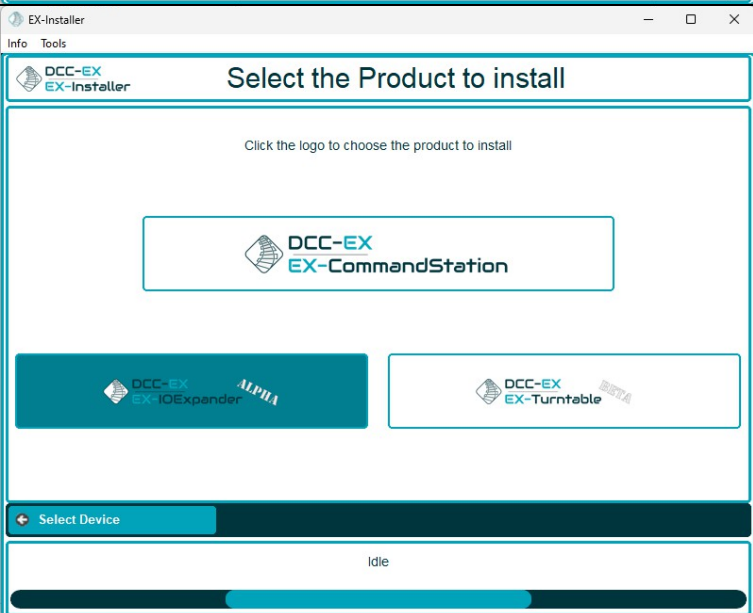
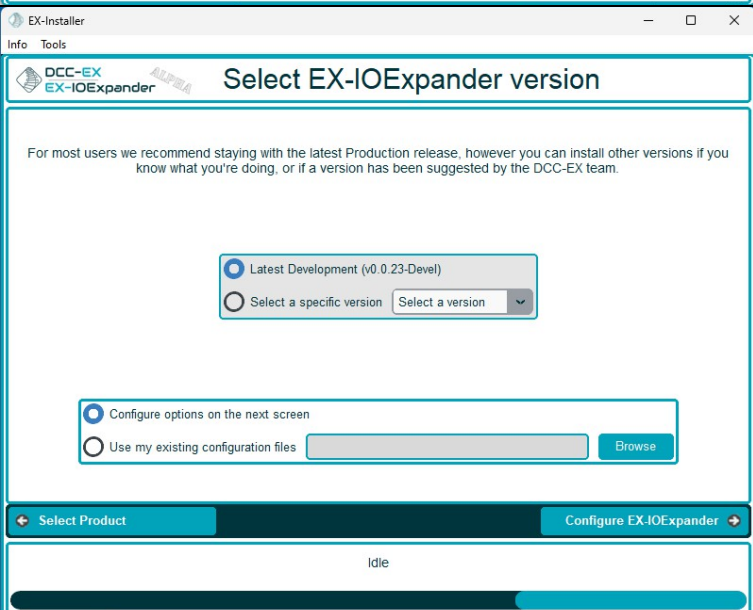
## Vorbereitung: Arduino NANO

**Achtung:** Arduino NANO's mit einem alten Bootloader konnte ich auf diesem Weg nicht installieren. Mit dem Arduino Programm oder VSCode mit PlatformIO klappt es aber (s.a. Seite 8).

### Installation mit EX-Installer

<b>EX-Installer Version 0.0.20</b>	
<b>aktuelle Arduino CLI - Datei laden</b>	



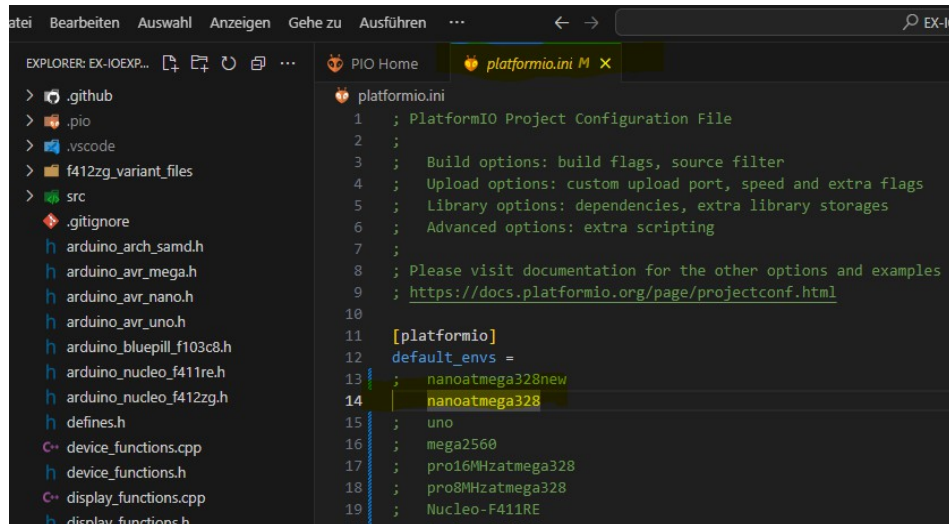
<p><b>Schnittstelle und Hardware auswählen</b></p>	
<p><b>„DCC-EX IOExpander“ auswählen</b></p>	
<p><b>hier kann man bei einer nochmaligen Installation die gesicherten Daten wieder einlesen</b></p>	

<p><b>hier werden, wenn schon vorhanden, die I2C Pullup's abgeschaltet und die I2CAdresse festgelegt.</b></p>	
	
<p><b>Ergebnis</b></p> <p><b>I2C Adresse 0x65, ohne Pullup Widerstände</b></p>	

## Vorbereitung: Arduino NANO mit altem Bootloader

### Installation unter VSCode mit PlatformIO

Meine Einstellungen habe ich so vor genommen:



```

1 ; PlatformIO Project Configuration File
2 ;
3 ; Build options: build flags, source filter
4 ; Upload options: custom upload port, speed and extra flags
5 ; Library options: dependencies, extra library storages
6 ; Advanced options: extra scripting
7 ;
8 ; Please visit documentation for the other options and examples
9 ; https://docs.platformio.org/page/projectconf.html
10
11 [platformio]
12 default_envs =
13 ; nanoatmega328new
14 nanoatmega328
15 ; uno
16 ; mega2560
17 ; pro16MHzatmega328
18 ; pro8MHzatmega328
19 ; Nucleo-F411RE

```

### Arduino NANO (alter Bootloader)



```

37 // PULLUP - equivalent of <T P>
38 //
39 // #define TEST_MODE ANALOGUE_TEST
40 // #define TEST_MODE INPUT_TEST
41 // #define TEST_MODE OUTPUT_TEST
42 // #define TEST_MODE PULLUP_TEST
43
44 ///////////////////////////////////////////////////////////////////
45 // Uncomment to disable internal I2C pullup re
46 // NOTE: This will not apply to all supported
47 #define DISABLE_I2C_PULLUPS
48
49 #endif

```

ohne pullup Widerstände

Bitte unbedingt beachten!

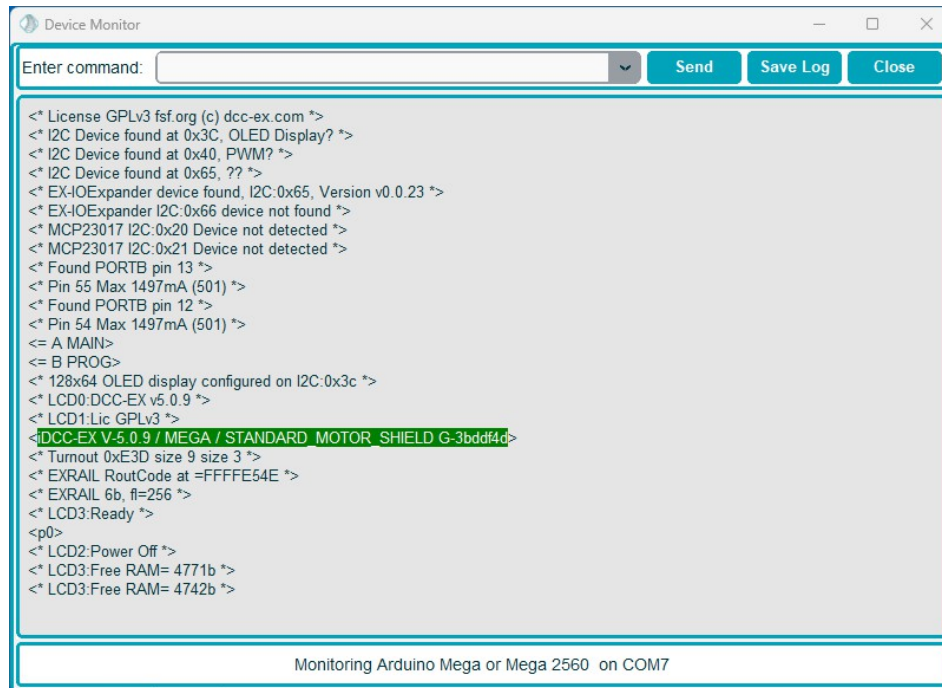
Die Beschreibung findet sich unter:

„Configure I2C address via serial“

Link: [Übersicht und Konfiguration — DCC-EX Modellbahndokumentation](#)



## Gesamt Ergebnis mit einem Arduino NANO



```
< * License GPLv3 fsf.org (c) dcc-ex.com * >
< * I2C Device found at 0x3C, OLED Display? * >
< * I2C Device found at 0x40, PWM? * >
< * I2C Device found at 0x65, ?? * >
< * EX4IOExpander device found, I2C:0x65, Version v0.0.23 * >
< * EX4IOExpander I2C:0x66 device not found * >
< * MCP23017 I2C:0x20 Device not detected * >
< * MCP23017 I2C:0x21 Device not detected * >
< * Found PORTB pin 13 * >
< * Pin 55 Max 1497mA (501) * >
< * Found PORTB pin 12 * >
< * Pin 54 Max 1497mA (501) * >
< = A MAIN >
< = B PROG >
< * 128x64 OLED display configured on I2C:0x3c * >
< * LCD0:DCC-EX v5.0.9 * >
< * LCD1:Lic GPLv3 * >
< DCC-EX V-5.0.9 / MEGA / STANDARD MOTOR SHIELD G-3bddf4d >
< * Turnout 0xE3D size 9 size 3 * >
< * EXRAIL RoutCode at =FFFFE54E * >
< * EXRAIL 6b, fl=256 * >
< * LCD3:Ready * >
< p0 >
< * LCD2:Power Off * >
< * LCD3:Free RAM= 4771b * >
< * LCD3:Free RAM= 4742b * >
```

Monitoring Arduino Mega or Mega 2560 on COM7

### Arduino NANO I2C 0x65

## Händische Einrichtung

Die manuelle Installation habe ich getestet, um die Befehle von DCC-EX besser zu verstehen. Es ist viel Arbeit und man darf keinen Schritt vergessen, sonst war alles umsonst.

Es funktioniert alles einwandfrei.

Der bessere Weg ist jedoch, die Einstellungen in der Datei  
„myAutomation.h“  
und  
„mySetup.h“  
vorzunehmen.

Hier werden dann gemäß des Inhaltes der Konfigurationsdateien bei jedem Neustart alle Einstellungen automatisch vorgenommen und man erspart es sich - auch bei einem Wechsel des Arduinos - wieder alles händisch neu einrichten zu müssen. Einfach nur das Programm in den neuen Arduino Mega einspielen - fertig.

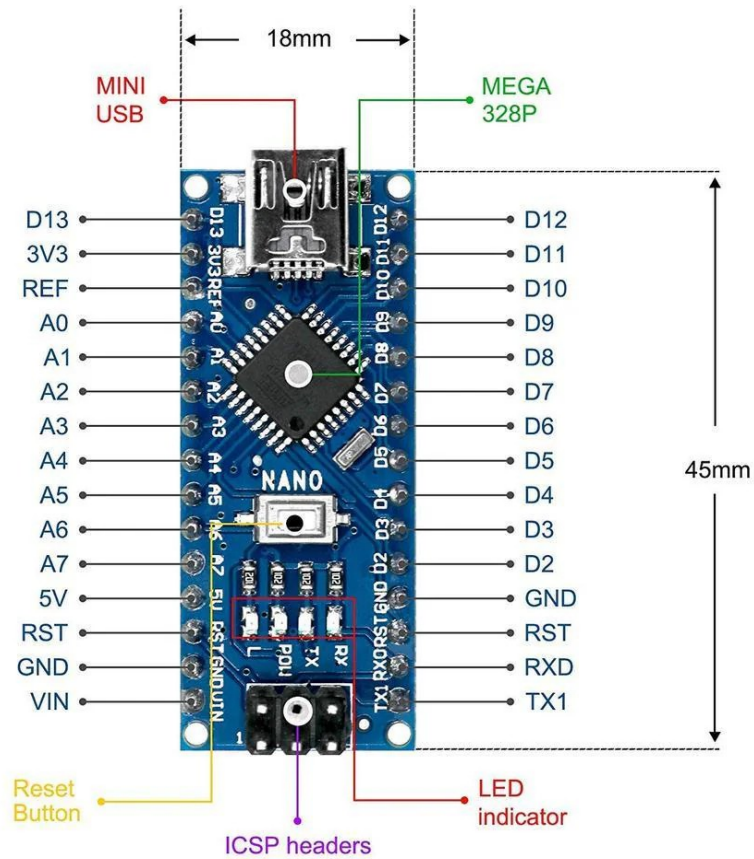
## Einstellungen für den Arduino NANO

1. Arduino Nano Adr. 0x65		
Ppin Nano	Vpin Zentrale	id
D2	800	800
D3	801	801
D4	802	802
D5	803	803
D6	804	804
D7	805	805
D8	806	806
D9	807	807
D10	808	808
D11	809	809
D12	810	810
D13	811	811
A0	812	812
A1	813	813
A2	814	814
A3	815	815
A6	816	
A7	817	

2. Arduino Nano Adr. 0x66		
Ppin Nano	Vpin Zentrale	id
D2	818	818
D3	819	819
D4	820	820
D5	821	821
D6	822	822
D7	823	823
D8	824	824
D9	825	825
D10	826	826
D11	827	827
D12	828	828
D13	829	829
A0	830	830
A1	831	831
A2	832	832
A3	833	833
A6	834	
A7	835	

HAL(EXIOExpander, Vpin, $\Sigma$ Pins, Adr)	
HAL(EXIOExpander, 800, 18, 0x65)	HAL(EXIOExpander, 818, 18, 0x66)

<S id vpin pullup>	
<S 800 800 1>	<S 818 818 1>

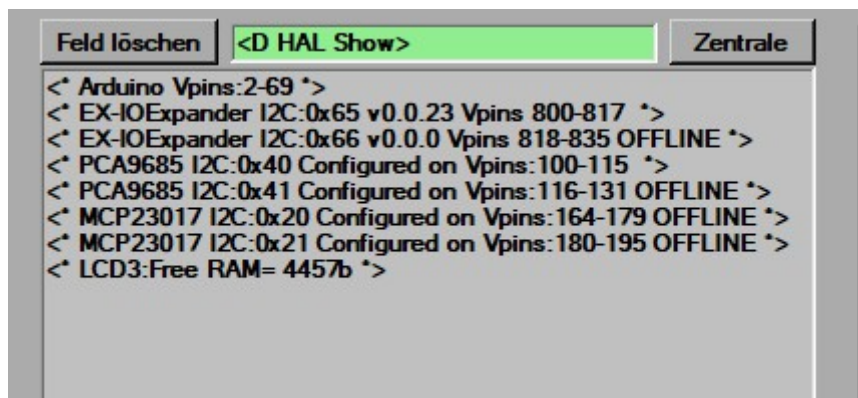


Vpin	Pin	Digital	Analog	PWM
800	D2	Ja	Nein	Nein
801	D3	Ja	Nein	Ja
802	D4	Ja	Nein	Nein
803	D5	Ja	Nein	Ja
804	D6	Ja	Nein	Ja
805	D7	Ja	Nein	Nein
806	D8	Ja	Nein	Nein
807	D9	Ja	Nein	Ja
808	D10	Ja	Nein	Ja
809	D11	Ja	Nein	Ja
810	D12	Ja	Nein	Nein
811	D13	Ja	Nein	Nein
812	A0	Ja	Ja	Nein
813	A1	Ja	Ja	Nein
814	A2	Ja	Ja	Nein
815	A3	Ja	Ja	Nein
816	A6	Nein	Ja	Nein
817	A7	Nein	Ja	Nein

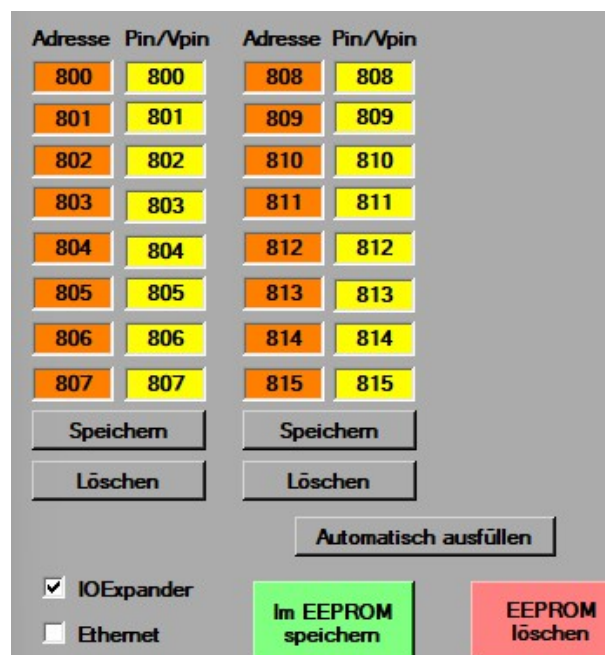
### 1. Arduino Nano ab Vpin 800

Vpin 816 und 817 sind nur Analoge - Pin's  
somit können wir nur die Vpin's von 800 bis 815 benutzen

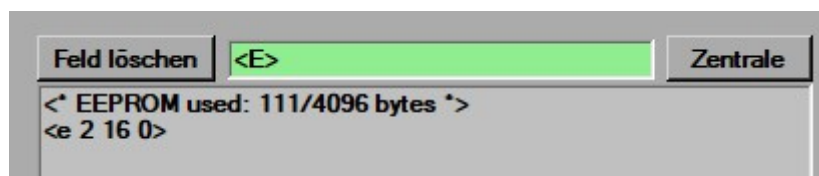
Folgende Einstellungen habe ich mit dem Programm „PinSetting“ (siehe Video von Wilfried - [link](#) unter „00 - Allgemeines und Wichtiges.pdf“) vorgenommen.

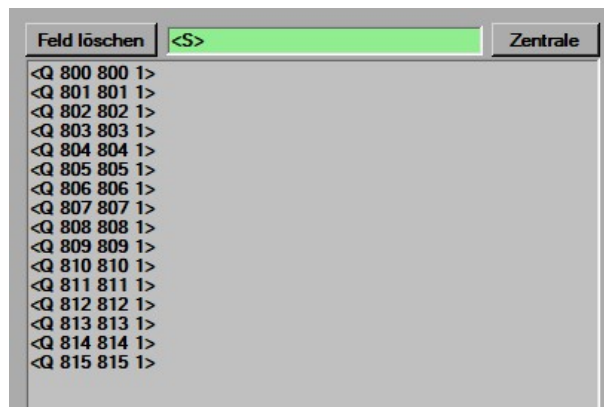


Anzeige der bei mir angeschlossenen Geräte.



Bei mir gilt das für den ersten Arduino NANO, speichern nicht vergessen, sowie ins EEPROM speichern!





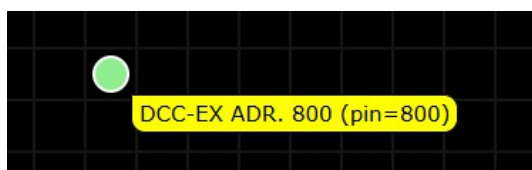
**Belegung der DCC - Adressen mit den Vpin's sowie Pullup's**  
Die DCC - Adressen müssen nicht die gleichen sein wie die Vpin's  
(das macht es aber übersichtlicher)

### DCC-EX ADR. 800

Basisdaten	Position
Name:	DCC-EX ADR. 800
Anschluss:	800 - +
Invertiert:	<input type="checkbox"/>
Typ:	standard
Fahrstrasse ausführen:	-

X ✓

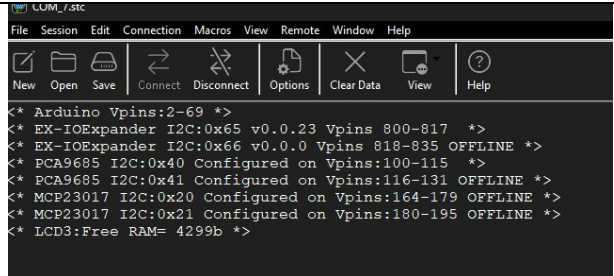
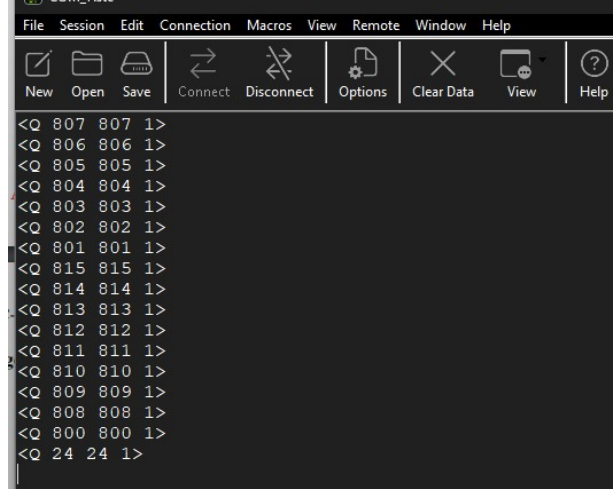
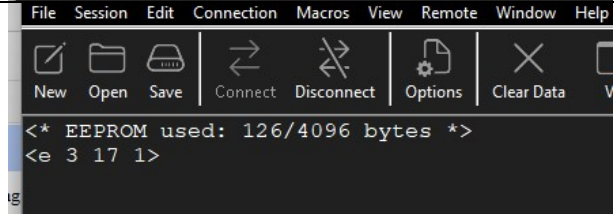
#### Einstellung der Rückmelder in RailControl



#### Ergebnis der Rückmeldung



Die einzelnen Befehle kann man natürlich auch mit einem Terminal Programm eingeben, hier ein paar Beispiele mit [CoolTerm](#). Immer die Befehle in spitzen Klammern eingeben.

<p><b>&lt;D HAL Show&gt;</b></p> <p><b>Zeigt die Konfiguration der konfigurierten Servoplatine und der GPIO-Extenderplatine sowie die verwendeten Pins</b></p>	 <pre>&lt; Arduino Vpins:2-69 *&gt; &lt; EX-IOExpander I2C:0x65 v0.0.23 Vpins 800-817 *&gt; &lt; EX-IOExpander I2C:0x66 v0.0.0 Vpins 818-835 OFFLINE *&gt; &lt; PCA9685 I2C:0x40 Configured on Vpins:100-115 *&gt; &lt; PCA9685 I2C:0x41 Configured on Vpins:116-131 OFFLINE *&gt; &lt; MCP23017 I2C:0x20 Configured on Vpins:164-179 OFFLINE *&gt; &lt; MCP23017 I2C:0x21 Configured on Vpins:180-195 OFFLINE *&gt; &lt; LCD3:Free RAM= 4299b *&gt;</pre>
<p><b>&lt;S&gt;</b></p> <p><b><a href="#">Liste aller definierten Sensoren</a></b></p>	 <pre>&lt;Q 807 807 1&gt; &lt;Q 806 806 1&gt; &lt;Q 805 805 1&gt; &lt;Q 804 804 1&gt; &lt;Q 803 803 1&gt; &lt;Q 802 802 1&gt; &lt;Q 801 801 1&gt; &lt;Q 815 815 1&gt; &lt;Q 814 814 1&gt; &lt;Q 813 813 1&gt; &lt;Q 812 812 1&gt; &lt;Q 811 811 1&gt; &lt;Q 810 810 1&gt; &lt;Q 809 809 1&gt; &lt;Q 808 808 1&gt; &lt;Q 800 800 1&gt; &lt;Q 24 24 1&gt;</pre>
<p><b>&lt;E&gt;</b></p> <p><b><a href="#">Definitionen im EEPROM speichern</a></b></p>	 <pre>&lt; * EEPROM used: 126/4096 bytes *&gt; &lt;e 3 17 1&gt;</pre>

## Einrichtung in der Datei „myAutomation.h“ / „mySetup.h“

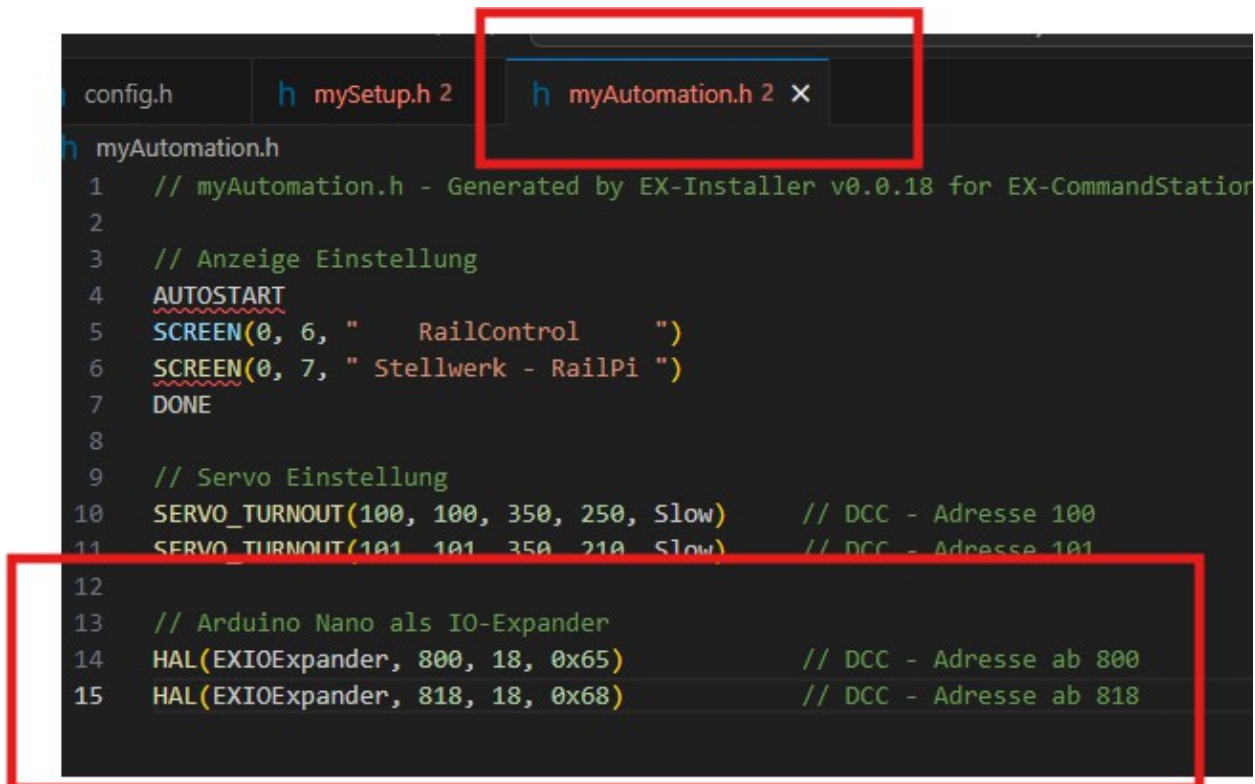
### Gerätetreiber

Um die Unterstützung für EX-IOExpander zu aktivieren, muss das Gerät entweder in "myHal.cpp" oder "myAutomation.h" in der EX-CommandStation erstellt/definiert werden.

Der Gerätetreiber ist standardmäßig enthalten, sodass lediglich die Geräte definiert werden müssen.

Ich habe die Gerätedefinition direkt in "myAutomation.h" mit dem Befehl HAL erstellt.

- HAL(EXIOExpander, vpin, npins, address)
  - vpin = Unbenutzte Adresse
  - npins = Gesamtzahl der VPs, die dem Gerät zugewiesen werden sollen
  - address = Ein verfügbares i2C-Adresse (Standard-0x65)



```
config.h  mySetup.h 2  myAutomation.h 2 x
h myAutomation.h
1 // myAutomation.h - Generated by EX-Installer v0.0.18 for EX-CommandStation
2
3 // Anzeige Einstellung
4 AUTOSTART
5 SCREEN(0, 6, " RailControl ")
6 SCREEN(0, 7, " Stellwerk - RailPi ")
7 DONE
8
9 // Servo Einstellung
10 SERVO_TURNOUT(100, 100, 350, 250, Slow) // DCC - Adresse 100
11 SERVO_TURNOUT(101, 101, 350, 210, Slow) // DCC - Adresse 101
12
13 // Arduino Nano als IO-Expander
14 HAL(EXIOExpander, 800, 18, 0x65) // DCC - Adresse ab 800
15 HAL(EXIOExpander, 818, 18, 0x68) // DCC - Adresse ab 818
```

Auszug aus meiner Datei: myAutomation.h

## Befehle für Eingänge

Die **EX-CommandStation** unterstützt Sensoreingänge, die an jedem Arduino PIN angeschlossen werden können, der von der DCC-EX-CommandStation nicht selbst verwendet wird.

Zusätzlich können natürlich PIN's an externen I/O-Expandern und anderen Geräten genutzt werden..

Der Sensor gilt als **INAKTIV**, wenn er ein Potential von +5 V hat, und als **AKTIV**, wenn der Pin auf 0 V heruntergezogen wird (PullUp =1 s.u.)

Um die richtigen Spannungspegel zu gewährleisten, MUSS die Sensorschaltung wieder an die gleiche Masse gebunden werden, die auch vom Arduino verwendet wird.

Der Sensorcode verwendet eine Entprelllogik, um Kontaktprellen zu eliminieren, das durch mechanische Schalter bei Schaltvorgängen erzeugt wird. Dadurch wird vermieden, dass für jeden Sensor eine eigene Glättungsschaltung erstellt werden muss.

Möglicherweise müssen die Parameter in der Datei „Sensor.cpp“ durch Versuche mit speziellen Sensoren geändert werden. Die Standardparameter schützen jedoch bis zu 20 Millisekunden vor Kontaktprellen.

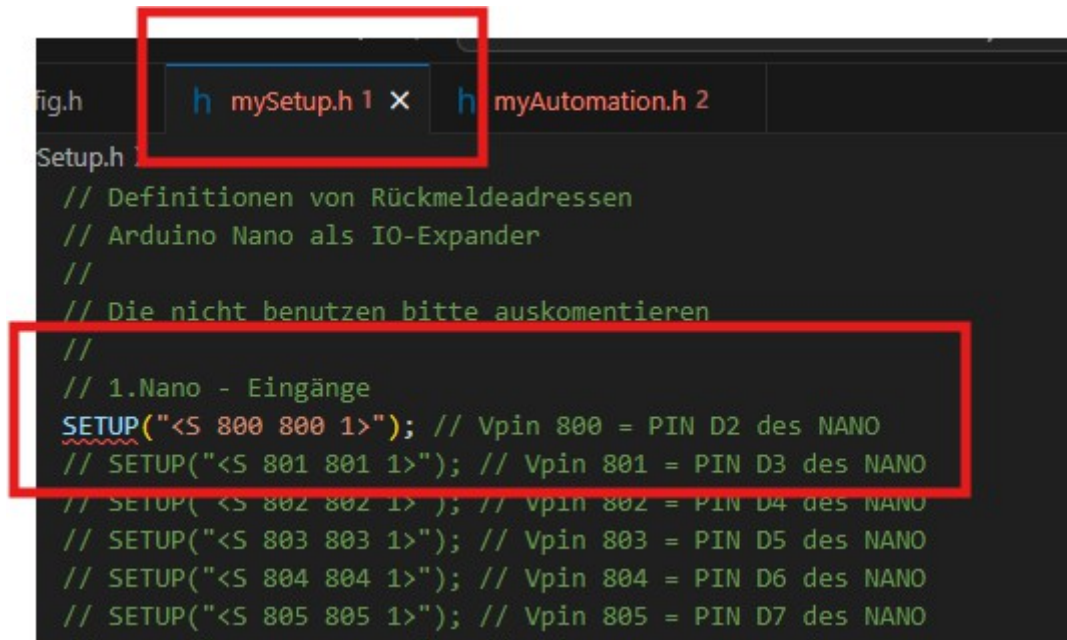
Damit die EX-CommandStation einen oder mehrere Arduino-PIN's auf Sensor-Trigger überwacht, ist die Sensordefinition durch den Befehl <S> in verschiedenen Variationen zu definieren/ bearbeiten/ löschen:

- **<S id vpin pullup>** : Erstellt eine neue Sensor-ID mit angegebener PIN und PULLUP, wenn die Sensor-ID bereits vorhanden ist, wird sie mit der angegebenen PIN und PULLUP aktualisiert
- **<S id>** : Löscht die Definition der Sensor-ID
- **<S>** : Listet alle definierten Sensoren auf
- **Ergebnis:** für jeden definierten Sensor oder wenn keine Sensoren definiert sind **<Q id vpin pullup>**
  - **id:** Die numerische ID (0 - 32767) des Sensors (DCC - Adresse)
  - **vpin :** Die PIN-Nummer des Eingangs, der vom Sensorobjekt gesteuert werden soll. Bei Arduino - Eingabepins entspricht dies der digitalen PIN-Nummer. Bei Servoeingängen und E/A-Expandern ist dies die für das HAL-Gerät definierte PIN-Nummer.
  - **pullup :**
    - 1 = interner Pull-up-Widerstand für PIN verwenden (ACTIVE = LOW),
    - 0 = interner Pull-up-Widerstand für PIN nicht verwenden (ACTIVE=HIGH).

Alle Sensoren, die wie oben definiert sind, werden innerhalb der Hauptschleife des Programmes zyklisch wiederholt und sukzessive überprüft.

Wenn ein Sensor-PIN seinen Zustand wechselt, wird eine der folgenden seriellen Nachrichten generiert:

- <Q id> - für den Übergang der Sensor-ID vom INACTIVE-Zustand in den ACTIVE-Zustand (d.h. der Sensor ist ausgelöst)
- <q id> - für den Übergang der Sensor-ID vom ACTIVE-Zustand in den INACTIVE-Zustand (d.h. der Sensor ist nicht mehr ausgelöst)
- <Q> - Listet den Status aller definierten Sensoren auf  
RÜCKGABE: <Q-ID> (aktiv) oder <q-ID> (nicht aktiv)



```
fig.h  mySetup.h 1 x  myAutomation.h 2
Setup.h
// Definitionen von Rückmeldeadressen
// Arduino Nano als IO-Expander
//
// Die nicht benutzen bitte auskommentieren
//
// 1.Nano - Eingänge
SETUP("<S 800 800 1>"); // Vpin 800 = PIN D2 des NANO
// SETUP("<S 801 801 1>"); // Vpin 801 = PIN D3 des NANO
// SETUP("<S 802 802 1>"); // Vpin 802 = PIN D4 des NANO
// SETUP("<S 803 803 1>"); // Vpin 803 = PIN D5 des NANO
// SETUP("<S 804 804 1>"); // Vpin 804 = PIN D6 des NANO
// SETUP("<S 805 805 1>"); // Vpin 805 = PIN D7 des NANO
```

Auszug aus meiner Datei: mySetup.h

## Befehle für Ausgänge

Die EX-CommandStation unterstützt die optionale **OUTPUT-Steuerung** von ungenutzten Arduino PIN's für benutzerdefinierte Zwecke. PIN's können aktiviert oder deaktiviert werden.

Standardmäßig werden die AKTIVEN PIN's auf HIGH und INACTIVE AUF LOW gesetzt. Dieses Standardverhalten kann jedoch für jeden PIN invertiert werden, in diesem Fall ACTIVE = LOW und INACTIVE = HIGH.

Definitionen und Status (ACTIVE/ INACTIVE) für Ausgabe-PIN's werden im EEPROM beibehalten und beim Einschalten wiederhergestellt.

Standardmäßig wird jeder definierte PIN entsprechend seinem wiederhergestellten Zustand auf aktiv oder inaktiv gesetzt. Das Standardverhalten kann jedoch so geändert werden, dass jeder PIN gezwungen werden kann, beim Einschalten entweder aktiv oder inaktiv zu sein, unabhängig von seinem vorherigen Zustand vor dem Ausschalten.

Damit die EX-CommandStation einen oder mehrere Arduino -PIN's als benutzerdefinierte Ausgänge verwendet, ist per Befehl <Z> der jeweilige Output-Pin zu definieren

- <Z id vpin iflag> : Erstellt eine neue Ausgabe-ID  
mit den angegebenen Werten für vpin und iflag  
Ergebnis: O bzw X - wenn erfolgreich bzw. nicht erfolgreich  
(z. B. nicht genügend Speicher).

Wenn die Ausgabe-ID bereits vorhanden ist, wird sie mit den angegebenen Werten aktualisiert.

### Hinweis:

Der Ausgangszustand wird sofort auf ACTIVE/ INACTIVE gesetzt und der Pin wird entsprechend dem angegebenen Wert auf HIGH/ LOW gesetzt.

- <Z id>: Löscht die Definition der Ausgabe-ID  
Ergebnis: O bzw. X - wenn erfolgreich bzw. nicht erfolgreich  
(z. B. ID existiert nicht)
- <Z> : Listet alle definierten Ausgangspins auf  
Ergebnis: für jeden definierten Ausgangspin oder wenn keine Ausgangspins definiert sind: <Y id vpin iflag state>
- id : Die numerische ID (0 - 32767) der Ausgabe (DCC - Adresse)
- vpin : Die PIN-Nummer des Ausgangs, der vom Ausgabeobjekt gesteuert werden soll. Bei Arduino - Ausgangspins ist das die PIN-Nummer.

Bei Servoausgängen und E/A-Expandern ist dies die für das HAL-Gerät definierte PIN - Nummer (falls vorhanden), z.B.:

100 - 115 für Servos am ersten PCA9685 Servocontroller-Modul ,

116 - 131 für Servos am zweiten PCA9685 Modul,

164 - 179 für PIN's am ersten MCP23017 GPIO-Erweiterungsmodul und

180 - 195 für PIN's am zweiten MCP23017 Modul.



- **state** : Der Status der Ausgabe (0 = INAKTIV / 1 = AKTIV)
- **iflag** : Definiert das Betriebsverhalten der Ausgabe basierend auf den Bits 0, 1 und 2 wie folgt:
  - **iflag, bit 0**: 0 = vorwärts Operation (ACTIVE = HIGH / INACTIVE=LOW)  
1 = invertiert Operation (ACTIVE = LOW / INACTIVE = HIGH)
  - **iflag, bit 1**: 0 = Zustand des PIN's, der beim Einschalten wiederhergestellt wird, auf ACTIVE oder INACTIVE abhängig vom Zustand vor dem Ausschalten.  
1 = Status des Pins, der beim Einschalten oder bei der ersten Erstellung festgelegt wurde, entweder auf ACTIVE oder INACTIVE je nach IFLAG, bit 2
  - **iflag, bit 2**: 0 = Status des PIN's, der beim Einschalten oder bei der ersten Erstellung auf INACTIVE gesetzt ist  
1 = Status des PIN's, der beim Einschalten oder bei der ersten Erstellung auf ACTIVE gesetzt ist

Für die Änderung von definierten Ausgängen gilt:

- **<Z id state>** : Setzt die Ausgabe entweder auf den ACTIVE- oder INACTIVE-Zustand.

Ergebnis: X bei Fehler oder wenn die Ausgabe-ID vorhanden ist

**<Y id state>**

**id** : Die numerische ID (0 - 32767) des zu steuernden Ausganges.

**state** : Der Status der Ausgabe (0 = INAKTIV / 1 = AKTIV)

Die Stati der Ausgabepins eines Arduinos werden im EEPROM ausfallsicher gespeichert.

Eine Liste der aktuellen Zustände jedes Ausgabepins wird von der **EX-CommandStation** generiert, wenn der Statusbefehl aufgerufen wird. Dies bietet eine effiziente Möglichkeit, den Zustand aller Ausgänge zu initialisieren, die von einer separaten Schnittstelle oder einem GUI-Programm überwacht oder gesteuert werden.

```
45
46
47 // 1.Nano - Ausgänge
48 // SETUP("<Z 800 800 0>"); // Vpin 800 = PIN D2 des NANO
49 SETUP("<Z 801 801 0>"); // Vpin 801 = PIN D3 des NANO
50 // SETUP("<Z 802 802 0>"); // Vpin 802 = PIN D4 des NANO
51 // SETUP("<Z 803 803 0>"); // Vpin 803 = PIN D5 des NANO
52 // SETUP("<Z 804 804 0>"); // Vpin 804 = PIN D6 des NANO
```

Auszug aus meiner Datei: mySetup.h