Die microSPS und der CAN BUS

Die microSPS wurde mit einem CAN Bus erweitert. Nun ist es möglich ein BUS System aufzubauen, damit wird die Steuerung noch flexiebler. Die Anwendung kann nach Aufgaben oder Orten aufgeteilt werden.

Dazu einige Beispiele:

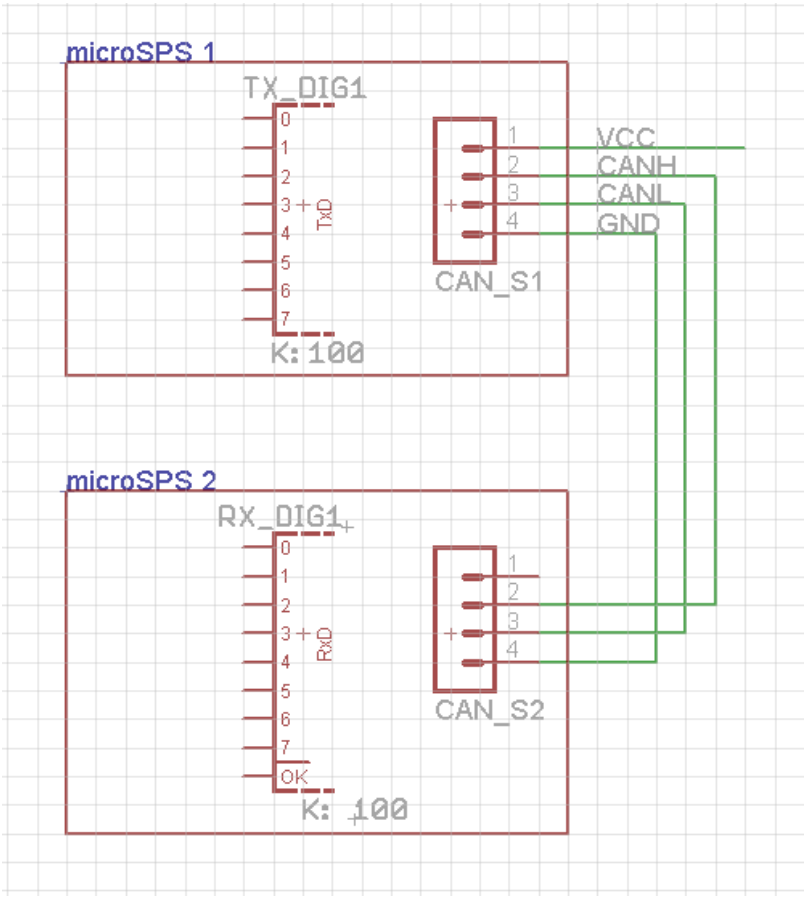
Falls man mehr Ein / Ausgänge benötigt, können somit in einer Anwendung mehrere Module eingesetzt und miteinander verknüpft werden.

Die Anzeige befindet sich an einem anderen Ort wie die Steuerung.

Über die CAN Schnittstellte könnte auch eine übergeordnete Steuerung / Computer auf die microSPS zugreifen.

Ein Webinterface könnte über den CAN BUS mit der microSPS Daten austauschen

Angeschlossen wird der CAN Bus über 3 oder 4 Leitungen. Beim 4 Leiter Anschluss wird die Versorgung mitgeführt. Damit würde das angeschlossene Modul über den CAN Bus mit 5V versorgt und benötigt so keine zusätzliche Spannungsversorgung. Hier eine Skizze, wie zwei Steuerungen miteinander verbunden werden.



In der microSPS ist ein Baustein als Sender und ein Baustein als Empfänger eingetragen. Beide Bausteine müssen die gleiche Adresse haben. Über die Adresse wird der Sender mit dem Empfänger verbunden.

Der CAN BUS muss an den Enden mit einem Abschlusswiderstand von 120 Ohm abgeschlossen werden. Dies erreicht man, indem die Brücke neben dem CAN Bus mit einem Jumper verbunden wird.

Aufbau der Telegramme

Für die Telegramme wird ein (Priorisierung und Adresse) 11 Bit Identifier verwendet. Das Telegramm hat folgende Struktur:

Statusfeld	Kontrollfeld		byte 1	byte 2	byte 3	byte 3 & 5	byte 6
Identifier 11 Bit	RTR	DLC	Massage Type	Adresse	Message Number	Daten	CRC

Identifier	11 Bit	0x200 ist als Filter für die SPS Datenübertragung freigegeben
RTR	1 Bit	wird mit 0 belegt
DCL	4 Bit	Datenlänge
byte 1 bis 8	8 Byte	Als Nutzdaten im Telegramm stehen bis zu 8 Datenbytes zur Verfügung. Für die microSPS werden aber nur 6 byte benötigt.

Für die CRC werden alle Bytes addiert und dann das Zweierkoplement gebildet.

```
uint8_t get_CRC(const can_t *msg)
{
  uint16_t crc16 = 0;
  uint8_t i, crc, length;

  length = msg->length & 0x0f;

  // Die Bytes addieren
  for (i=0;i<length;i++)
  {
    crc16 += msg->data[i];
  }

  crc = crc16;     // lo Byte isolieren
  crc ^= 0xFF;     // XOR
  crc++;           // Zweierkomplement bilden

  return crc;
```

Das erste Byte (byte 1) codiert den Typ und das Kommando der Nachricht: Bisher wurden zwei Message Typen definiert (00 und 01).

Message Type 00

Mit diesem Type werden die Daten aus der Digitalen und Analogen Bausteine übertragen. Zur Überwachung der Telegramme wird ein Telegramm Zähler mitgeführt. Der Sender muss nach jeder Nachricht den Telegramm Zähler um eins erhöhen. Der Empfänger kann somit prüfen ob ein Telegramm verloren gegangen ist.

	Bits							
	7	6	5	4	3	2	1	0
0	Message Type							
1	Bausteinadresse 0 bis 255							
2	Telegramm Zähler 0 bis 255							
3	hi byte							
4	lo byte							
5	CRC für die Quersummenprüfung							

Message Type 01

Dieses Telegramm entspricht dem Type 00 bis dass bei diesem Telegramm der Telegramm Zähler gesetzt wird.

	Bits							
	7	6	5	4	3	2	1	0
0	Message Type							
1	Bausteinadresse 0 bis 255							
2	Telegramm Zähler 0 bis 255							
3	hi byte							
4	lo byte							
5	CRC für die Quersummenprüfung							

Für die Kommunikation der microSPS sind diese zwei Telegramme ausreichend. Für weitere Anforderungen können hier zusätzliche Telegramme implementiert werden.