Application Note: Digital Pick-to-Light System

Inhaltsverzeichnis

1 DPD – Digitales Pick-to-light Device

1.1 Einleitung

Dieser Implementierungsleitfaden beschreibt die Integration der digitalen Pick-to-light-Module in eine SPS-Umgebung. Weitere technische Daten und rechtliche Hinweise gibt das Einzeldatenblatt des jeweiligen Moduls.

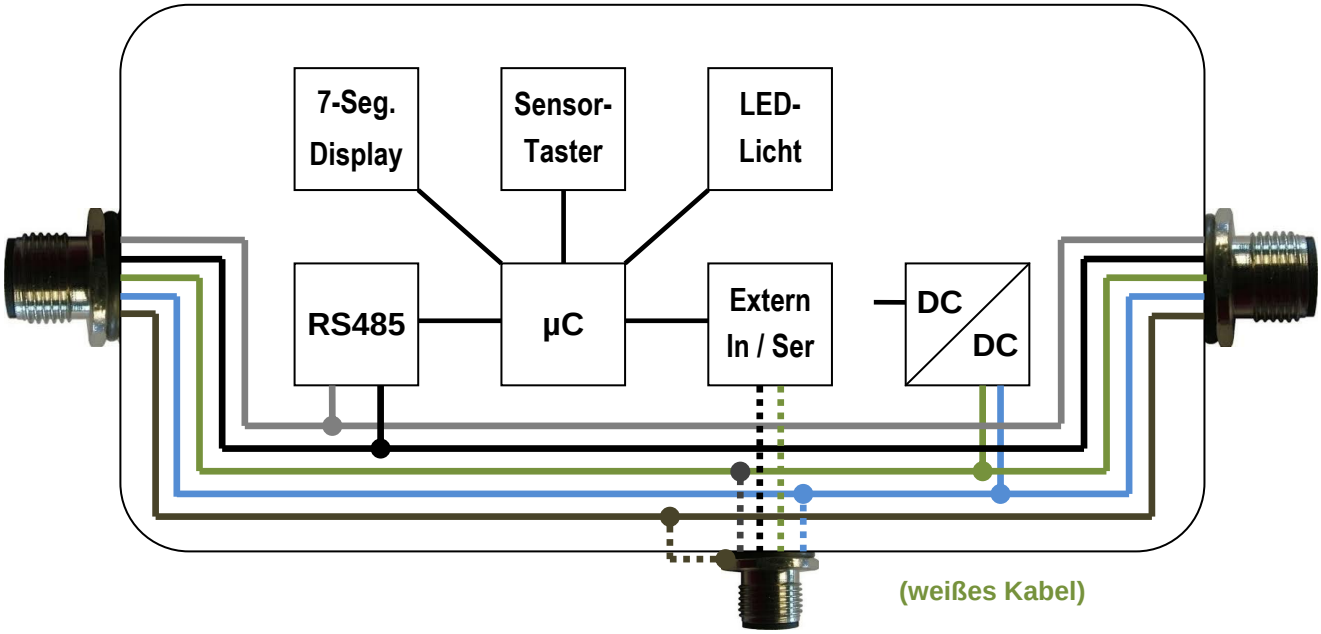
DPD Varianten			
Bestellnummer	**Anzeige**	**Befestigung**	**Ein- und Ausgänge**
7508001876	4-Zeichen LED	Rundrohr 28mm	-
7508001928	4-Zeichen LED	C-Profil 36mm	-
7508002015	4-Zeichen LED	Rundrohr 28mm	Digitaleingang via M8 Kontrolles eines LED-Streifens via M8
7508002014	4-Zeichen LED	C-Profil 36mm	Digitaleingang via M8 Kontrolles eines LED-Streifens via M8
7508001907	4-Zeichen LED	-	Digitaleingang via M8 Kontrolles eines LED-Streifens via M8

Technische Daten (Ausschnitt)		
Bussystem		
Typ	Modbus RTU	
Baudrate	57600 Bd	
Datenformat	8N2	
Übertragung	RS-485 Halbduplex	
Versorgungsspannung		
Spannung	24V DC … 36V DC	
Maximalstrom	50mA	
Anschlüsse		
Versorgungsspannung und Datenbus	M12 Stecker & Buchse A-codiert	
Verbindungskabel	STP 4-polig	Weidmüller 196471xxxx oder 1060132xxxx (xxxx = Länge in cm)
Max. Kabellänge	1200m (Daten)	Der Spannungsabfall bei langen Kabeln muss berücksichtigt werden!

1.2 Funktionale Übersicht

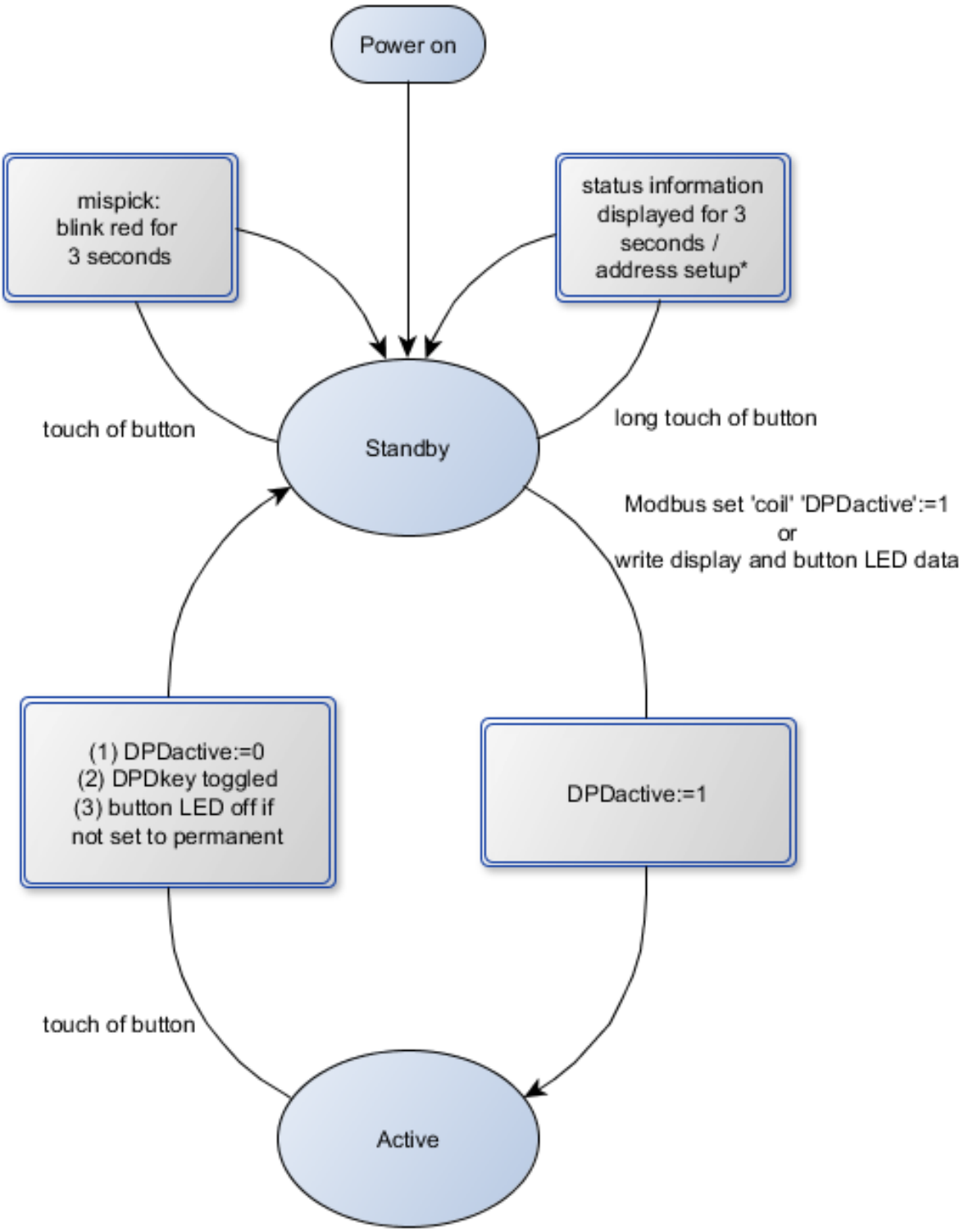
- Anzeige von 4 über Modbus-RTU empfangenen ASCII-Zeichen soweit mit 7 Segmenten darstellbar
- Sensortaste leuchtet mehrfarbig entsprechend der über Modbus-RTU empfangenen Farbinformation
- Statusbyte ist auslesbar über Modbus-RTU
- Licht der Sensortaste schaltet ab bei Berührung
- Blinken für 3 Sekunden, wenn die Sensortaste im inaktiven Zustand berührt wird
- Ansteuerung eines LED-Streifens mit WS281x-Treibern über den M8-Anschluss[1]

1.3 Blockdiagramm des DPDs

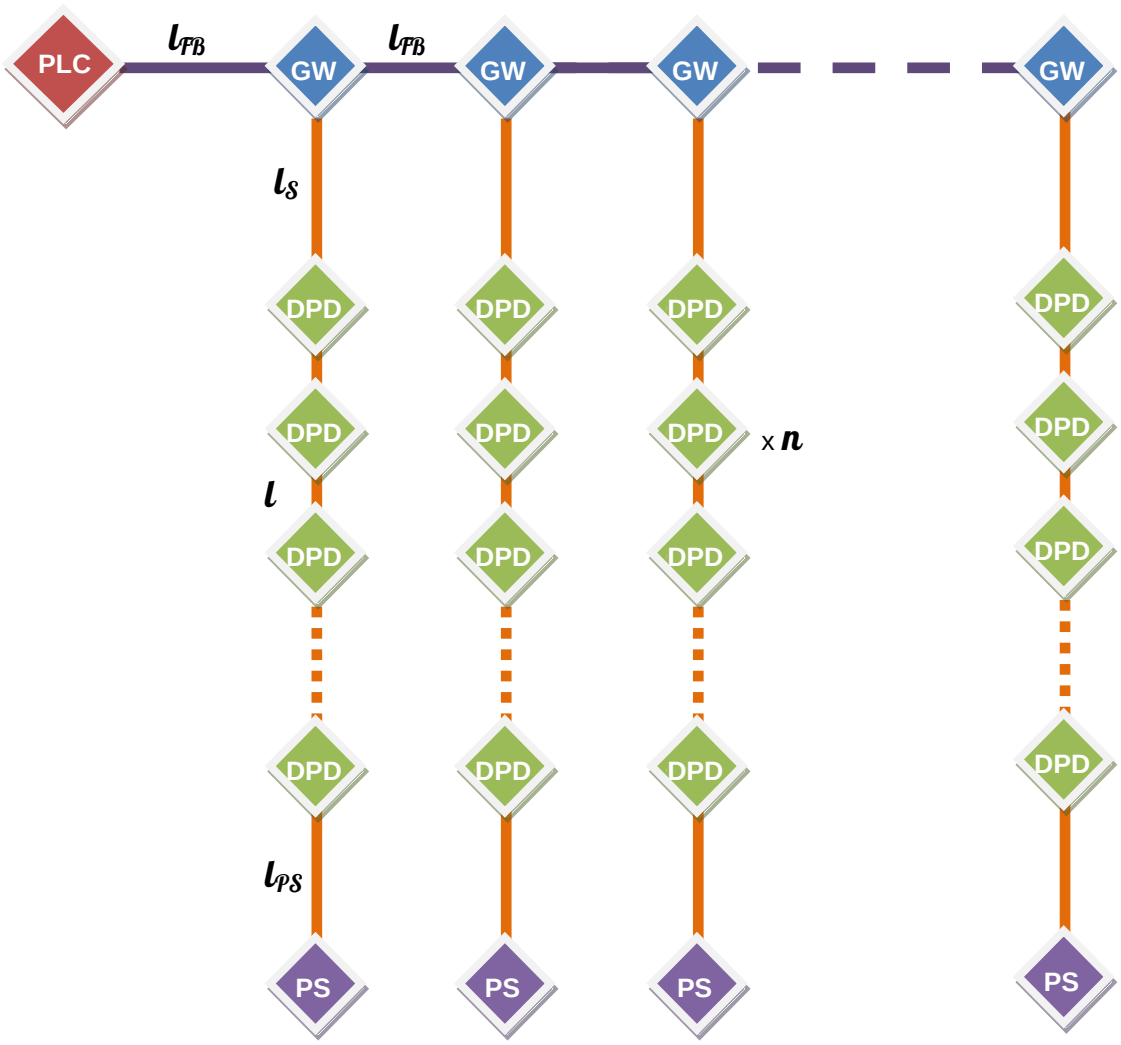


[1] Konfigurierbar für Firmwareversionen 1.2.7 und höher beim „DPD advanced“

1.4 Zustands-Übergangs-Diagramm eines DPDs



1.5 Systemtopologie



Legende:

PLC	SPS
GW	Gateway zum übergeordneten Systembus zur DPD-Line
PS	Versorgungsspannung
DPD	Digitales Pick-to-light Device
n	Anzahl der DPDs pro Linie
l_{FB}	Kabellänge des Feldbussegmentes
l_S	Kabellänge zwischen Gateway und ersten DPD
l	Kabellänge zwischen zwei DPDs
l_{PS}	Kabellänge zwischen PS und dem ersten DPD

Dieser Verdrahtungsplan zeigt eine Beispielanordnung. In realen Anwendungen sind meistens mehrere Gateways in einem Schaltkasten sowie die Spannungsversorgung angeordnet. Zum Stromverbrauch müssen auch extern verbundene Lichtschranken und LED-Streifen hinzugerechnet werden.

Sind Spannungsversorgung und Gateway in einem Gehäuse, so ist, l_S gleich l_{PS}.

Wird die Spannungsversorgung in der Mitte der Linie angeschlossen, so ist n die Anzahl der DPDs geteilt durch 2.

1.6 Berechnung der maximalen Kabellänge

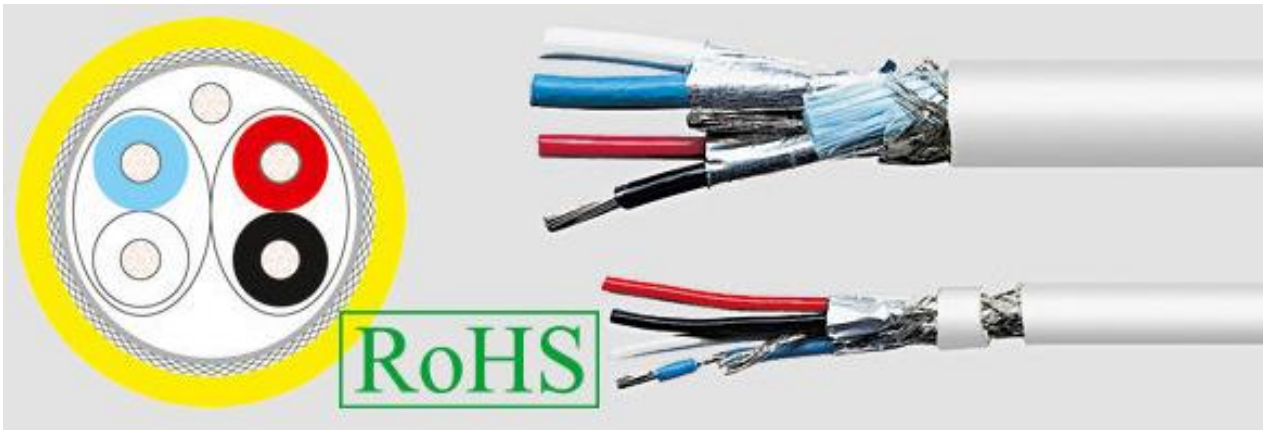
l_{PB}	Gegeben durch die Feldbus-Technik, z. B. 100m für Ethernet-basierte Feldbusse
$l_S + n \times l <$ 1200m	Linientopologie
$l_S + n \times l <$ 100m	Beliebige Topologie
$n \times l_{PS} + ½\, n\,(n - 1) \times l < K$	K = 2000m für Versorgungsspannung = 31.5V K = 1000m für Versorgungsspannung Spannung = 28.5V K = 10000m für Versorgungsspannung Spannung = 48V[2]

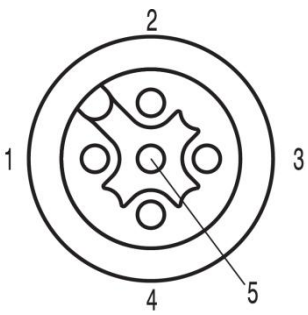
1.7 Busverbindung

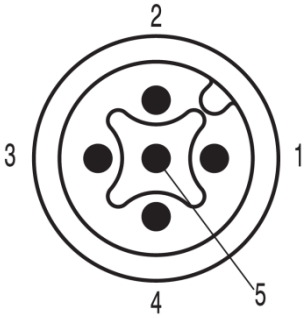
Jedes DPD hat einen M12-Stecker und eine M12-Buchse. Bis zu 31 DPDs können in einer Linie hintereinandergeschaltet werden. Geeignete Kabel sind z. B. Weidmüller 196471xxxx oder 1060132xxxx (xxxx = Länge in cm). Modbus-RTU-Daten und Versorgungsspannung werden über ein Kabel übertragen. Die beiden Leitungen für Modbus-RTU müssen paarweise verseilt sein. Geeignete Kabel werden z. B. für CAN, DeviceNet oder CC-Link verwendet.

[2] Hardwareversionen x2.0.0 oder höher

Verkabelung des DPD-Buses			
M12 Pin	**Farbe innerhalb Kabel (meistens)**	**Farbe innerhalb DPD**	**Signal**
1 = Schirm	(Schirm)		Schirm = FE = + Versorgungsspannung LED-Streifen (Option)
2			+Versorgungsspannung
3			0V
4			RS-485 (A / Y / TxRx+)
5			RS-485 (B / Z / TxRx-)

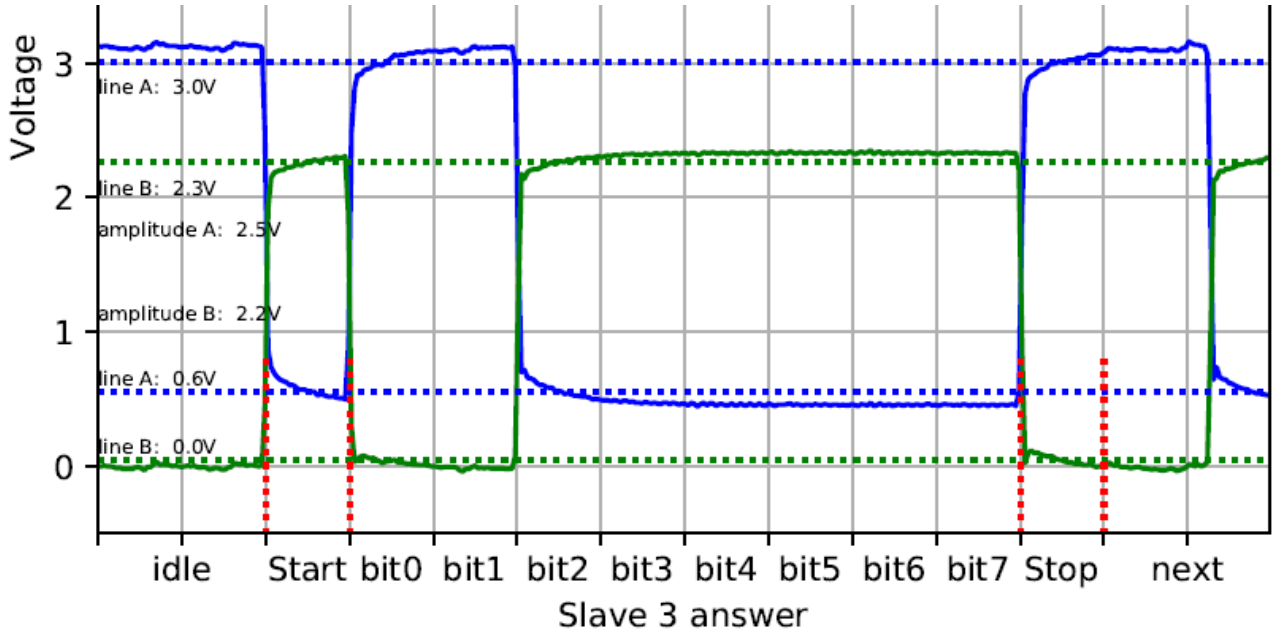






M8-Anschluss		
M8 Pin	**Farbe innerhalb DPD**	**Signal**
1		+Versorgungsspannung des Sensors
2		WS281x serieller Ausgang
3		0V
4		Eingang
Schirm	-	+Versorgungsspannung für LED-Streifen

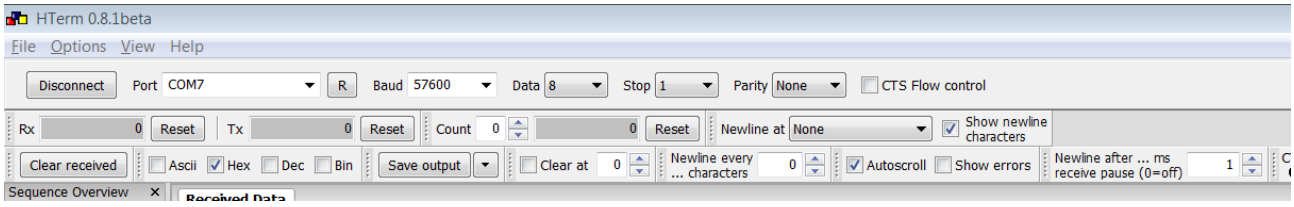
Bespiel-Sample der RS485-Signale A und B des ersten Bytes der Antwort von DPD 3:



1.8 Testsetup

Ein einfacher Testaufbau kann mit Hilfe eines PCs und eines sehr preiswerten USB-RS-485-Adapters zusammen mit einem 24V-Netzteils durchgeführt werden..

Für einfache Tests kann sehr gut ein Terminalprogramm wie  “HTerm” verwendet werden. Konfiguration: COM-Port (COM7 in diesem Beispiel), 57600Bd, 8N1, no handshake, Newline after 1ms.



Beispielcode: Initialisierung Modbus-RTU mit Python und der Lib “minimalmodbus” (Linux / Windows):

```
device_uart_size = 64
ComPort = sys.argv[1]
minimalmodbus.BAUDRATE = 57600
minimalmodbus.PARITY = 'N'
minimalmodbus.BYTESIZE = 8
minimalmodbus.STOPBITS = 1
minimalmodbus.TIMEOUT = 0.01
minimalmodbus.CLOSE_PORT_AFTER_EACH_CALL = True

try:
    dpd_bus = minimalmodbus.Instrument("COM7", 1)
except:
    print("Wrong COM port: COM7")
    sys.exit(-1)
```

1.9 Integrierte Servicefunktionen des DPDs

- Anzeige der Firmware-Version (2 Zeichen)[3] und der Modbus-RTU-Adresse für 3.5 Sekunden nach dem Einschalten der Spannungsversorgung
- Anzeige der Firmware-Version (2 Zeichen)[4] und der Modbus-RTU-Adresse für 3 Sekunden, wenn die Sensortaste mehr als 4 Sekunden lang berührt wird
- Rücksetzen der Modbus-RTU-Adresse auf 31 (Auslieferungszustand), wenn die Sensortaste mehr als 4 Sekunden lang innerhalb der ersten 15 Sekunden nach Einschalten der Spannungsversorgung berührt wird

1.10 Modbus-RTU-Funktionen

Modbus unterstützt zwei unterschiedliche Datenformate: Bit-granulare Befehle, die “coils” für Ausgänge und für Eingänge “discrete inputs” genannt werden und 16-bit-Wörter, die “holding registers” für Ausgänge und “input registers” für Eingänge genannt werden. Achtung: Die Adresszählung startet bei 1 und die erste Ziffer spezifiziert den Datentyp![5]

[3] Firmware-Version 1.2.6 und höher
[4] Firmware-Version 1.2.6 und höher
[5] Vgl. http://www.chipkin.com/files/liz/MODBUS_2010Nov12.pdf für Details

Tabellarische Übersicht der Modbus-Register		
„Coils" = bit-granulare Ausgänge mit Schreib- und Lesezugriff über die Modbus-Funktion 01/05		
0 00001	DPDactive	(normalerweise unbenutzt) muss gesetzt werden, um das DPD zu aktivieren = Touch-Sensor freizuschalten; erfolgt durch Schreibbefehl von Anzeige- oder LED-Daten automatisch
0 00002	*reserviert*	reserviert für zukünftige Erweiterungen (Output0)
0 00003	*reserviert*	reserviert für zukünftige Erweiterungen (Output1)
0 00004	*reserviert*	reserviert für zukünftige Erweiterungen (Output level bit 0)
0 00005	*reserviert*	reserviert für zukünftige Erweiterungen (Output level bit 1)
0 00006	*reserviert*	reserviert für zukünftige Erweiterungen
0 00007	*reserviert*	reserviert für zukünftige Erweiterungen
0 00008	*reserviert*	reserviert für zukünftige Erweiterungen
"Inputs" = bit-granulare Eingänge mit Lesezugriff über die Modbus-Funktion 02		
1 00001	*reserviert*	reserviert für zukünftige Erweiterungen
1 00002	*reserviert*	reserviert für zukünftige Erweiterungen
1 00003	DPDButton	gesetzt während der Sensortaster berührt wird oder Input0 aktiv ist
1 00004	DPDBToggle	wechselt wenn die Sensortaster berührt wird oder Input0 aktiviert wird, wenn das DPD vorher aktiviert wurde
1 00005	*reserviert*	reserviert für zukünftige Erweiterungen (I0Toggle XOR DPDBToggle)
1 00006	*reserviert*	reserviert für zukünftige Erweiterungen (I1Toggle)
1 00007	*Reserviert*	reserviert für zukünftige Erweiterungen (Input0 active XOR Button touched)
1 00008	*Reserviert*	reserviert für zukünftige Erweiterungen (Input1)
"Input registers" = wortweise Eingänge mit Lesezugriff über die Modbus-Funktion 04		
3 00001	*res.*/HwVer0	high-Byte: reserviert, low-Byte: Hardware-Versions-ID
3 00002	*res.*/HwVer1	high-Byte: reserviert, low-Byte: Hardware-Versions-ID
3 00003	*res.*/HwVer2	high-Byte: reserviert, low-Byte: Hardware-Versions-ID
3 00004	*res.*/FwVer0	high-Byte: reserviert, low-Byte: Firmware-Versions-ID
3 00005	*res.*/FwVer1	high-Byte: reserviert, low-Byte: Firmware-Versions-ID
3 00006	*res.*/FwVer2	high-Byte: reserviert, low-Byte: Firmware-Versions-ID
3 00007..21	*res./Serial*	high-Byte: reserviert, low-Byte: Seriennummer (15 Bytes)
3 00022	*Reserviert*	reserviert für zukünftige Erweiterungen
"Holding registers" = wortweise Ausgänge mit Schreib- und Lesezugriff über die Modbus-Funktionen 06/16 und 03/23		
4 00001	*DPDASCII4 /* DPDASCII0	high-Byte: fünftes Zeichen des LED-Displays (ASCII) low-Byte: erstes Zeichen des LED-Displays (ASCII)
4 00002	*DPDASCII5 /* DPDASCII1	high-Byte: sechstes Zeichen des LED-Displays (ASCII) low-Byte: zweites Zeichen des LED-Displays (ASCII)
4 00003	*DPDASCII6 /* DPDASCII2	high-Byte: siebtes Zeichen des LED-Displays (ASCII) low-Byte: drittes Zeichen des LED-Displays (ASCII)

Tabellarische Übersicht der Modbus-Register		
„Coils" = bit-granulare Ausgänge mit Schreib- und Lesezugriff über die Modbus-Funktion 01/05		
4 00004	*DPDASCII7 /* DPDASCII3	high-Byte: achtes Zeichen des LED-Displays (ASCII) low-Byte: viertes Zeichen des LED-Displays (ASCII)
4 00005	*res.*/DPDFarbe	high-Byte: reserviert, low-Byte: gültige ASCII-Werte: "R", "V", "B", "O" für rot, grün, blau und orange
4 00006	DPDColAttr	gültige ASCII-Werte: "FI", "CL", "CR" für Dauerlicht / schnelles und langsames Blinken; erstes ASCII-Zeichen im high-Byte
4 00007	*reserviert*	reserviert für zukünftige Erweiterungen[6]
4 00008	DPDstripe0	Controlwort0 LED-Streifen[7]
4 00009	DPDstripe1	Controlwort1 LED-Streifen[7]
4 00010..48	*reserved*	reserved for future use for various non-permanent data
4 00049..64	*reserved*	reserved for future use for various permanent data

Anmerkung zu Big Endian:

Bei Modbus-RTU wird das high-Byte wird zuerst übertragen gefolgt vom low-Byte.

[6] Schreibzugriff mit Modbus-Befehl 06 setzt die Modbus-RTU-Adresse
[7] Konfigurierbar für Firmwareversionen 1.2.7 und höher beim „DPD advanced"

1.11 Modbus-RTU Beispiel-Datenübertragung

1.11.1 Lesen des Status von DPD 18 = 0x12 (alle acht “Input” Bits 1 00001…1 00008)

Der Modbus-RTU Master sendet:

Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7
DPD address	Modbus Cmd	Modbus Addr	Modbus Addr	Modbus Cnt	Modbus Cnt	CRChigh	CRClow
0x12	0x02	0x00	0x00	0x00	0x08	0x7b	0x6f
18	2	0	0	0	8	123	111

DPD 18 antwortet z. B.:

Data0	Data1	Data2	Data3	Data4	Data5
DPD address	Modbus Cmd	Modbus Cnt	DPD status	Modbus CRC	Modbus CRC
0x12	0x02	0x01	0x01	0x64	0xcc
18	2	1	1	100	204 = -52

Die angeforderten Statusdaten befinden sich im vierten Datenbyte der Antwort. Bitte die CRC auswerten, um Übertragungsfehler aufzudecken! Wenn kein DPD 18 verbunden ist, wird keine Antwort empfangen. Ein geeignetes Timeout ist z. B. 5ms.

Das Master-Kommando ist immer gleich, so dass die 31 Pakete mit CRC vorberechnet werden können:

1	2	3	4	5	6	7	8
001	002	000	000	000	008	121	204
002	002	000	000	000	008	121	255
003	002	000	000	000	008	120	046
004	002	000	000	000	008	121	153
005	002	000	000	000	008	120	072
006	002	000	000	000	008	120	123
007	002	000	000	000	008	121	170
008	002	000	000	000	008	121	085
009	002	000	000	000	008	120	132
010	002	000	000	000	008	120	183
011	002	000	000	000	008	121	102
012	002	000	000	000	008	120	209
013	002	000	000	000	008	121	000
014	002	000	000	000	008	121	051
015	002	000	000	000	008	120	226
016	002	000	000	000	008	122	141
017	002	000	000	000	008	123	092
018	002	000	000	000	008	123	111
019	002	000	000	000	008	122	190
020	002	000	000	000	008	123	009
021	002	000	000	000	008	122	216
022	002	000	000	000	008	122	235
023	002	000	000	000	008	123	058
024	002	000	000	000	008	123	197
025	002	000	000	000	008	122	020
026	002	000	000	000	008	122	039
027	002	000	000	000	008	123	246
028	002	000	000	000	008	122	065
029	002	000	000	000	008	123	144
030	002	000	000	000	008	123	163
031	002	000	000	000	008	122	114

1	2	3	4	5	6	7	8
01	02	00	00	00	08	79	CC
02	02	00	00	00	08	79	FF
03	02	00	00	00	08	78	2E
04	02	00	00	00	08	79	99
05	02	00	00	00	08	78	48
06	02	00	00	00	08	78	7B
07	02	00	00	00	08	79	AA
08	02	00	00	00	08	79	55
09	02	00	00	00	08	78	84
0A	02	00	00	00	08	78	B7
0B	02	00	00	00	08	79	66
0C	02	00	00	00	08	78	D1
0D	02	00	00	00	08	79	00
0E	02	00	00	00	08	79	33
0F	02	00	00	00	08	78	E2
10	02	00	00	00	08	7A	8D
11	02	00	00	00	08	7B	5C
12	02	00	00	00	08	7B	6F
13	02	00	00	00	08	7A	BE
14	02	00	00	00	08	7B	09
15	02	00	00	00	08	7A	D8
16	02	00	00	00	08	7A	EB
17	02	00	00	00	08	7B	3A
18	02	00	00	00	08	7B	C5
19	02	00	00	00	08	7A	14
1A	02	00	00	00	08	7A	27
1B	02	00	00	00	08	7B	F6
1C	02	00	00	00	08	7A	41
1D	02	00	00	00	08	7B	90
1E	02	00	00	00	08	7B	A3
1F	02	00	00	00	08	7A	72

1.11.2 Schreiben des Displays und der LED für DPD 18 = 0x12 (4 00001…4 00006)

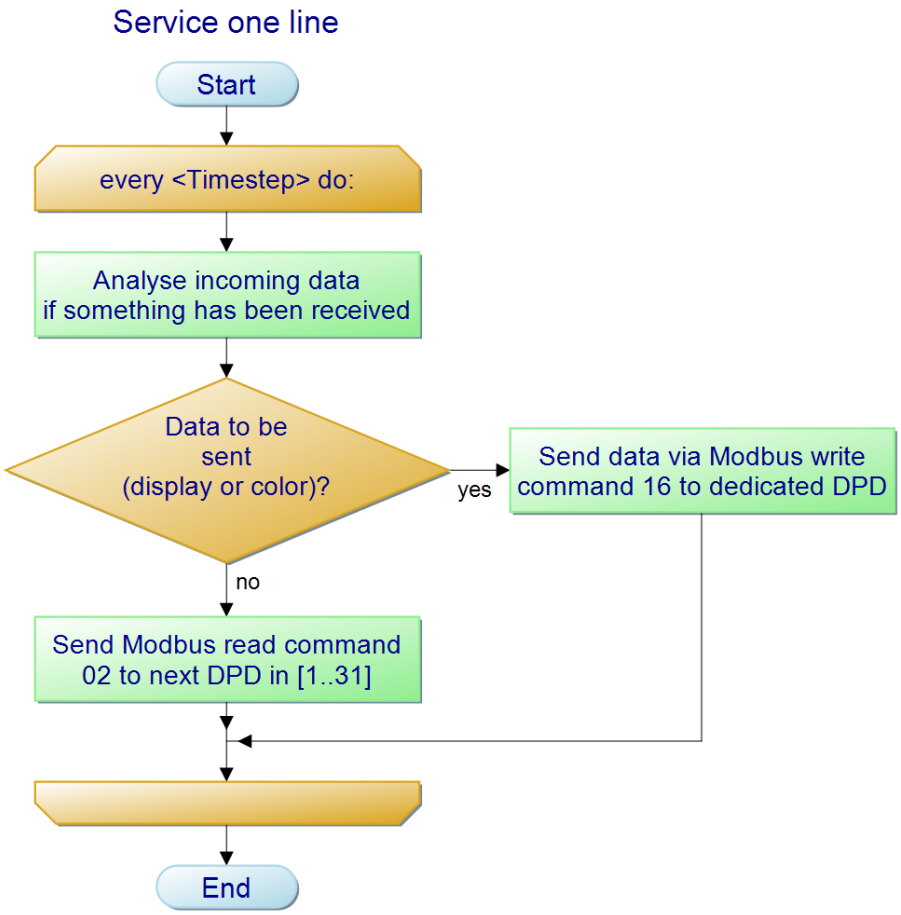
Der Modbus RTU-Master sendet beispielsweise:

Data0	Data1	Data2	Data3	Data4	Data5	Data6
DPD address 0x12 18	Modbus Cmd 0x10 16	Modbus Addr 0x00 0	Modbus Addr 0x00 0	Modbus Cnt 0x00 0	Modbus Cnt 0x06 6	Modbus Cnt2 0x0c 12
Data7	**Data8**	**Data9**	**Data10**	**Data11**	**Data12**	**Data13**
0x00 0	Display0 0x31 = '1' 49	0x00 0	Display1 0x2d = '-' 45	0x00 0	Display2 0x31 = '1' 49	0x00 0
Data14	**Data15**	**Data16**	**Data17**	**Data18**	**Data19**	**Data20**
Display3 0x38 = '8' 56	0x00 0	DPDColor 0x42 = 'B' 66	DPDColAttr1 0x46 = 'F' 70	DPDColAttr0 0x49 = 'I' 73	Modbus CRC 0x62 98	Modbus CRC 0x79 121

DPD 18 antwortet darauf immer:

Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7
DPD address 0x12 18	Modbus Cmd 0x10 16	Modbus Addr 0x00 0	Modbus Addr 0x00 0	Modbus Cnt 0x00 0	Modbus Cnt 0x06 6	Modbus CRC 0x42 66	Modbus CRC 0xa8 168 = -88

Mit diesen beiden Kommandos kann ein DPD-System vollständig betrieben werden. Das Scannen aller maximal 31 DPDs erfolgt in weniger als 0,5 Sekunden. Das Programm für eine SPS mag folgende Struktur haben:



Beispiel in Python mit der Lib “minimalmodbus”: Zeigt auf den DPD deren Addresse an und leuchtet grün. Danach wird fortlaufend das Toggle-Bit gelesen und bei Tastendruck “TEST”/blaue LED ausgegeben:

```
# write address to each DPD at startup
for dpd_bus.address in range(1, 32):
    try:
        DPDcounter[dpd_bus.address] = dpd_bus.read_bit(4, 2)
        print ("DPD", dpd_bus.address, "detected.", DPDcounter[dpd_bus.address])
        status = dpd_bus.write_registers(0, [ord("A"), ord("d"), 48 + dpd_bus.address // 10, 48 + dpd_bus.address % 10,
                                             ord("V"), 256 * ord("F") + ord("I")])
    except:
        pass

###################################
# main loop:
#   read
#   service DPD inputs as demo
#   write output data
###################################

while True:
    for dpd_bus.address in range(1, 32):
        try:
            newDPDcounter = dpd_bus.read_bit(4, 2)
            if DPDcounter[dpd_bus.address] != newDPDcounter:
                DPDcounter[dpd_bus.address] = newDPDcounter
                print("Key detected: Device", dpd_bus.address, ", new counter value:", newDPDcounter)
                status = dpd_bus.write_registers(0, [ord("T"), ord("e"), ord("s"),ord("t"), ord("B"), 0x100 * ord("F") + ord("I")])
        except:
            pass
```

1.12 Modbus-RTU-Adresseinstellung über Modbus

1.12.1 Setzen einer neuen Modbus-Adresse für DPD 31 = 0x1f (wird dauerhaft gespeichert)

Ein fabrikneues DPD ist immer auf die Modbus-Adresse 31 programmiert. Der Modbus-Master sendet z. B. folgende Bytefolge, um das DPD auf Adresse 18 zu setzen:

Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7
DPD address	Modbus Cmd	Modbus Addr	Modbus Addr	Modbus Data	Modbus Data	Modbus CRC	Modbus CRC
0x1f	0x06	0x00	0x06	0x00	0x12	0xea	0x78
31	6	0	6	0	18	234 = -22	120

Das DPD prüft die Adresse. Nur Adressen im Bereich von 1..30 werden akzeptiert. Adressen im Bereich von 129..158 (also 1..30 + 128) werden genau dann akzeptiert, wenn die Taste des DPDs gleichzeitig berührt wird. DPD 31 antwortet mit der gleichen Antwort. Anschließend antwortet das DPD nur noch auf Adresse 18.

Da auch hier das Kommando immer gleich ist, lassen sich die Kommandos ebenso vorberechnen für die Zieladressen 1..30 bzw. 1..30 + 128:

1	2	3	4	5	6	7	8
1F	06	00	06	00	01	AB	B5
1F	06	00	06	00	02	EB	B4
1F	06	00	06	00	03	2A	74
1F	06	00	06	00	04	6B	B6
1F	06	00	06	00	05	AA	76
1F	06	00	06	00	06	EA	77
1F	06	00	06	00	07	2B	B7
1F	06	00	06	00	08	6B	B3
1F	06	00	06	00	09	AA	73
1F	06	00	06	00	0A	EA	72
1F	06	00	06	00	0B	2B	B2
1F	06	00	06	00	0C	6A	70
1F	06	00	06	00	0D	AB	B0
1F	06	00	06	00	0E	EB	B1
1F	06	00	06	00	0F	2A	71
1F	06	00	06	00	10	6B	B9
1F	06	00	06	00	11	AA	79
1F	06	00	06	00	12	EA	78
1F	06	00	06	00	13	2B	B8
1F	06	00	06	00	14	6A	7A
1F	06	00	06	00	15	AB	BA
1F	06	00	06	00	16	EB	BB
1F	06	00	06	00	17	2A	7B
1F	06	00	06	00	18	6A	7F
1F	06	00	06	00	19	AB	BF
1F	06	00	06	00	1A	EB	BE
1F	06	00	06	00	1B	2A	7E
1F	06	00	06	00	1C	6B	BC
1F	06	00	06	00	1D	AA	7C
1F	06	00	06	00	1E	EA	7D

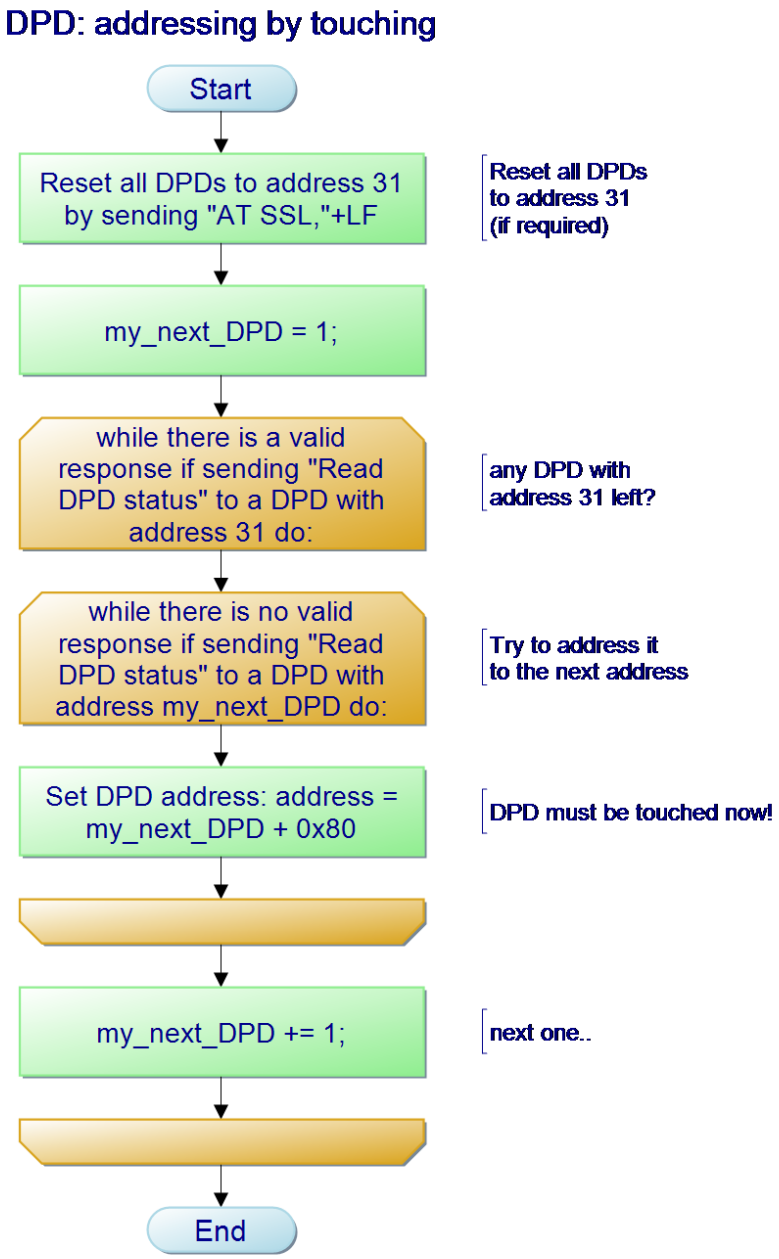
1	2	3	4	5	6	7	8
031	006	000	006	000	001	171	181
031	006	000	006	000	002	235	180
031	006	000	006	000	003	042	116
031	006	000	006	000	004	107	182
031	006	000	006	000	005	170	118
031	006	000	006	000	006	234	119
031	006	000	006	000	007	043	183
031	006	000	006	000	008	107	179
031	006	000	006	000	009	170	115
031	006	000	006	000	010	234	114
031	006	000	006	000	011	043	178
031	006	000	006	000	012	106	112
031	006	000	006	000	013	171	176
031	006	000	006	000	014	235	177
031	006	000	006	000	015	042	113
031	006	000	006	000	016	107	185
031	006	000	006	000	017	170	121
031	006	000	006	000	018	234	120
031	006	000	006	000	019	043	184
031	006	000	006	000	020	106	122
031	006	000	006	000	021	171	186
031	006	000	006	000	022	235	187
031	006	000	006	000	023	042	123
031	006	000	006	000	024	106	127
031	006	000	006	000	025	171	191
031	006	000	006	000	026	235	190
031	006	000	006	000	027	042	126
031	006	000	006	000	028	107	188
031	006	000	006	000	029	170	124
031	006	000	006	000	030	234	125

1	2	3	4	5	6	7	8
1F	06	00	06	00	81	AA	15
1F	06	00	06	00	82	EA	14
1F	06	00	06	00	83	2B	D4
1F	06	00	06	00	84	6A	16
1F	06	00	06	00	85	AB	D6
1F	06	00	06	00	86	EB	D7
1F	06	00	06	00	87	2A	17
1F	06	00	06	00	88	6A	13
1F	06	00	06	00	89	AB	D3
1F	06	00	06	00	8A	EB	D2
1F	06	00	06	00	8B	2A	12
1F	06	00	06	00	8C	6B	D0
1F	06	00	06	00	8D	AA	10
1F	06	00	06	00	8E	EA	11
1F	06	00	06	00	8F	2B	D1
1F	06	00	06	00	90	6A	19
1F	06	00	06	00	91	AB	D9
1F	06	00	06	00	92	EB	D8
1F	06	00	06	00	93	2A	18
1F	06	00	06	00	94	6B	DA
1F	06	00	06	00	95	AA	1A
1F	06	00	06	00	96	EA	1B
1F	06	00	06	00	97	2B	DB
1F	06	00	06	00	98	6B	DF
1F	06	00	06	00	99	AA	1F
1F	06	00	06	00	9A	EA	1E
1F	06	00	06	00	9B	2B	DE
1F	06	00	06	00	9C	6A	1C
1F	06	00	06	00	9D	AB	DC
1F	06	00	06	00	9E	EB	DD

1	2	3	4	5	6	7	8
031	006	000	006	000	129	170	021
031	006	000	006	000	130	234	020
031	006	000	006	000	131	043	212
031	006	000	006	000	132	106	022
031	006	000	006	000	133	171	214
031	006	000	006	000	134	235	215
031	006	000	006	000	135	042	023
031	006	000	006	000	136	106	019
031	006	000	006	000	137	171	211
031	006	000	006	000	138	235	210
031	006	000	006	000	139	042	018
031	006	000	006	000	140	107	208
031	006	000	006	000	141	170	016
031	006	000	006	000	142	234	017
031	006	000	006	000	143	043	209
031	006	000	006	000	144	106	025
031	006	000	006	000	145	171	217
031	006	000	006	000	146	235	216
031	006	000	006	000	147	042	024
031	006	000	006	000	148	107	218
031	006	000	006	000	149	170	026
031	006	000	006	000	150	234	027
031	006	000	006	000	151	043	219
031	006	000	006	000	152	107	223
031	006	000	006	000	153	170	031
031	006	000	006	000	154	234	030
031	006	000	006	000	155	043	222
031	006	000	006	000	156	106	028
031	006	000	006	000	157	171	220
031	006	000	006	000	158	235	221

1.12.2 One-touch-Adressierung

Dadurch, dass ein DPD nur auf Adress-Settings > 128 reagiert, wenn die Taste gleichzeitig berührt wird, lässt sich eine einfache Adressierung einer fabrikneuen Linie wie folgt implementieren. Wenn eine bereits adressierte Linie umadressiert werden soll, muss vorher das Kommando „AT SSL,31“ gesendet werden. Es ist hilfreich, diesen Algorithmus ebenfalls in der SPS zu hinterlegen.



Das folgende Python-Script zeigt dies exemplarisch (ohne vorheriges Rücksetzen mit “AT SSL,31”):

```
# start writing addresses with address # ..
my_next_address = 1
exitcnt = 1                                 # counts reads to address 31 with errors
writeit = True                              # used to print user request just once

while my_next_address < 31:
    time.sleep(.1)                          # "pseudo-isochronous 100ms"
    #dpd_bus.debug = True
    dpd_bus.address = my_next_address       # look if DPD with current address is available
    try:
        status = dpd_bus.read_bit(2, 2)
        print ("DPD with address", my_next_address, "found.")
        my_next_address += 1
        exitcnt = 2
        writeit = True
    except:
        my_next_address += 0
    time.sleep(.1)                          # "pseudo-isochronous 100ms"
    dpd_bus.address = 31  # brand new module: address := 31
    #print (my_next_address)
    try:
        status = dpd_bus.read_bit(2, 2)
        if exitcnt>0:
            exitcnt -= 1
    except:
        exitcnt += 1
        if exitcnt>10:
            print ("no further module with address=31 detected.")
            sys.exit(-1)
    if writeit & (exitcnt==0):
        print ("Please touch next DPD still having address 31 to set it to address", my_next_address)
        writeit = False
    time.sleep(.1)  # "pseudo-isochronous 100ms"
    try:
        status = dpd_bus.write_register(6, my_next_address+0x80, 0, 6, False)
        time.sleep(.1)                          # "pseudo-isochronous 100ms"
    except:
        my_next_address += 0
```

1.13 Setzen einer neuen Modbus-Adresse für DPDs mit Adresse 31

Durch Berühren der Sensorfläche für mehr als 5 Sekunden gelangt man in den eingebauten Adress-Setup-Modus eines DPDs, wenn dieses bisher auf die Adresse 31 programmiert war.

Der Ablauf ist wie folgt:

1. Sensortaster für mehr als 5 Sekunden berühren ➔ Modbus-Adresse wird angezeigt und Taster blinkt schnell rot


2. Sensortaster berühren zum Hochzählen der Adresse bis die gewünschte Adresse angezeigt wird

3. Sensortaster für mehr als 5 Sekunden berühren ➔ Neue Modbus-Adresse wird gespeichert, Taste leuchtet dauerhaft grün


Wenn der Sensortaster mehr als 3 Sekunden nicht berührt wird, beendet das DPD den Adresseingabe-modus.

Das Zurücksetzen der Adresse auf die Adresse 31 kann durch Betätigen der Sensortaste für 5 Sekunden innerhalb der ersten 15 Sekunden nach Kaltstart oder durch das unten gezeigte AT-Kommando durchgeführt werden.

1.14 Zurücksetzen aller DPDs auf Adresse 31 (kein Modbus-Kommando – Vorsicht!)

Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Data8	Data9
"A"	"T"	Space	"S"	"S"	"L"	","	"3"	"1"	LF
0x41	0x54	0x20	0x53	0x53	0x4c	0x2c	0x33	0x31	0x0a
65	84	32	83	83	76	44	52	49	10

Achtung: Alle DPDs werden auf Modbus-Adresse 31 zurückgesetzt!

Dieses Kommando kann z. B. mit einem Terminalprogramm wie “HTerm” gesendet werden.

Das Kommando muss als ein String ohne zeitliche Lücken gesendet werden und mit „LF“ abgeschlossen werden. HTerm muss z. B. auf “send on enter: LF” eingestellt sein.

1.15 Steuerung der LED-Streifens mit integrierten WS281x-Treiberbausteinen

LED-Streifen, bei denen die einzelnen LED-Gruppen mit einem auf dem LED-Streifen bestückten WS281x-Treiber gesteuert werden, können über die M8-Buchse eines DPDs gesteuert werden (nicht in allen Versionen vorhanden!).

Ein weiteres Netzteil zur Versorgung der LED-Streifen wird benötigt. Dieses Netzteil muss auf die Versorgungsspannung der LED-Streifen abgestimmt sein (12V oder 5V)!

Die High-Byte der ASCII-Register des LED-Displays werden für die Kontrolle des LED-Streifen verwendet:

DPD_Stripe0							
WS2811mode	number of the first (of two) pixel to light: pixel 0						
bit 15 (1 = high speed)	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8

DPD_Stripe0							
NoOfPixels0		Pixel0Red1	Pixel0Red0	Pixel0Green1	Pixel0Green0	Pixel0Blue1	Pixel0Blue0
bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

DPD_Stripe1							
KeepOnOnTouch	number of the second (of two) pixel to light: pixel 1						
bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8

DPD_Stripe1							
NoOfPixels1		Pixel1Red1	Pixel1Red0	Pixel1Green1	Pixel1Green0	Pixel1Blue1	Pixel1Blue0
bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

Prinzip: Zwei Gruppen von maximal 4 hintereinander liegenden LED-Pixeln können gleichzeitig leuchten. Das erste Pixel wird mit dem Wert DPD_StripeAddr*x* festgelegt (erstes leuchtende Pixel). NoOfPixel*x* gibt die Anzahl der hintereinander leuchtenden Pixel an (-1). Jede Pixelgruppe wird über RGB-Werte gesteuert. Für jede Farbe gibt es 4 Helligkeitsstufen:

Pixel*x*Col1	Pixel*x*Col0	Light Intensity
0	0	0%
0	1	25%
1	0	50%
1	1	100%

Beispiel:

0x83 ➔ LED-Streifen im High-Speed Mode des WS281x verbunden. Das 4. Pixel ist das erste leuchtende der ersten Gruppe.

0x09 ➔ Das 10. Pixel ist das erste leuchtende der zweiten Gruppe. Der LED-Streifen wird nicht ausgeschaltet, wenn die Sensortaste berührt wird.

0x70 ➔ Die erste Gruppe besteht aus zwei leuchtenden Pixeln und leuchtet 100% rot.

0x02 ➔ Die zweite Gruppe besteht nur aus einem einzelnen Pixel, das 50% blau leuchtet.

Anmerkung:

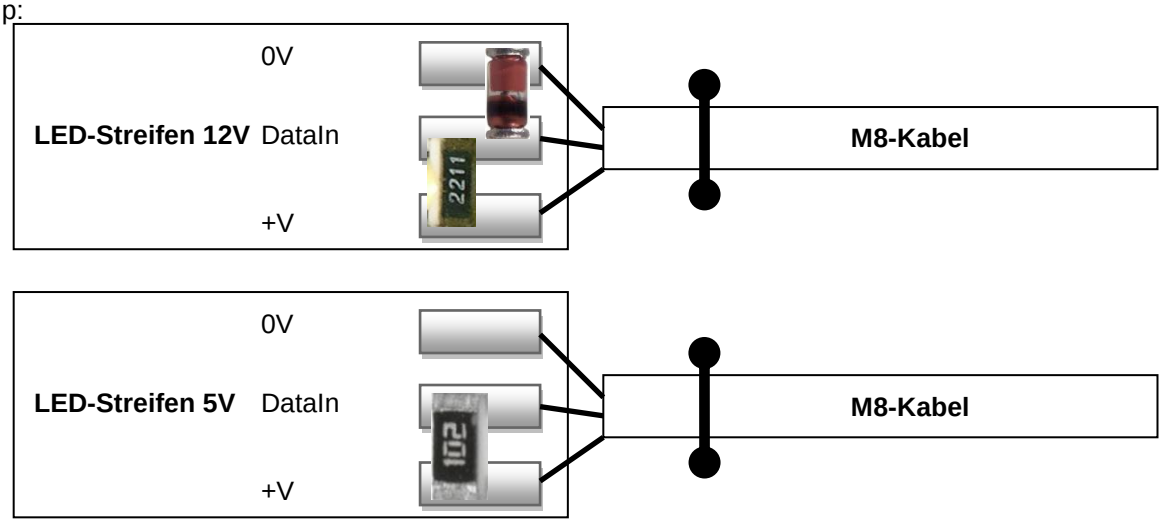
Es gibt LED-Streifen, bei der die Farben vertauscht sind.

Der LED-Streifen wird automatisch bei Berührung der Sensortaste oder Aktivierung des externen Eingangs abgeschaltet, wenn das Bit KeepOnTouch nicht gesetzt ist.

Hardwareverbindung:
LED-Streifen werden über ein geschirmtes M8-Kabel an das DPD angeschlossen. Geeignet sind z. B. die Kabel Weidmüller 190659xxxx (xxxx = Länge in cm). Achtung: Ein- oder Ausstecken des M8-Kabels nur bei stromlosem DPD!

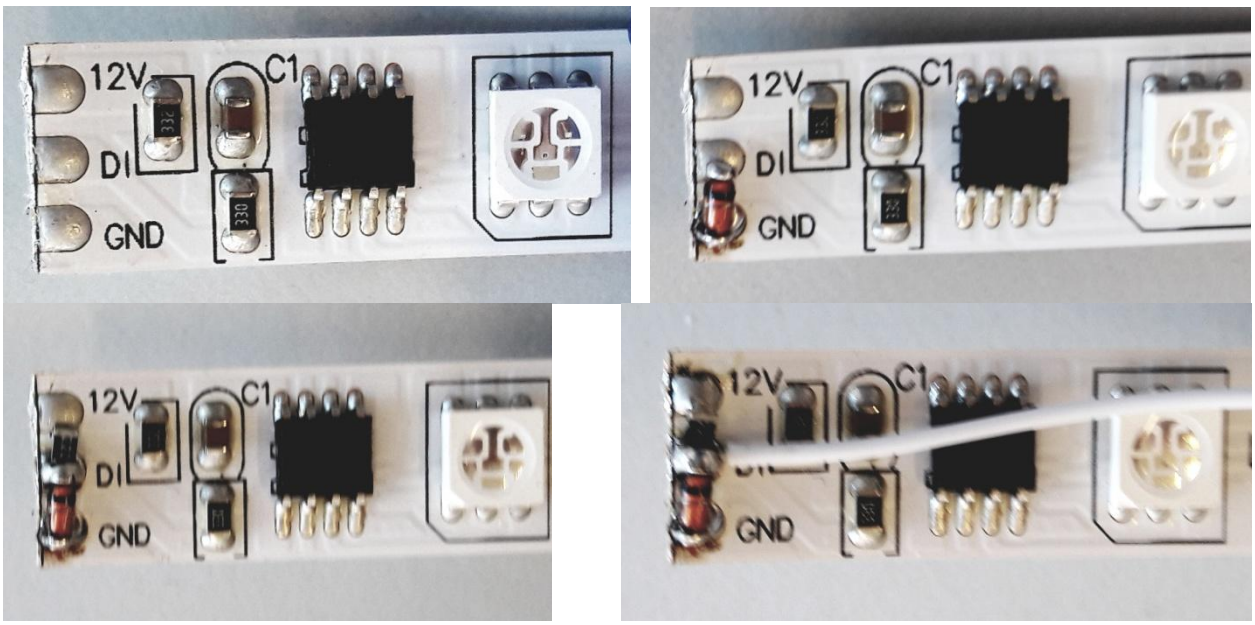
Bei einem LED-Streifen mit 5V-Versorgung muss ein 1kOhm pull-up-Widerstand in die serielle Datenleitung am Eingang des Streifens eingelötet werden. Bei Streifen mit 12V-Versorgungsspannung ist eine Kombination aus 2.2kOhm pull-up-Widerstand und 5,1V-Zener-Diode zu 0V notwendig. Im Gegensatz zum Foto unten verwenden Sie vorteilhafterweise SMD-Bauteile. Achten Sie auf eine geeignete Zugentlastung für das M8-Kabel und sichern Sie den Anschluss zusätzlich mit Schrumpfschlauch o. ä..

Prinzip:

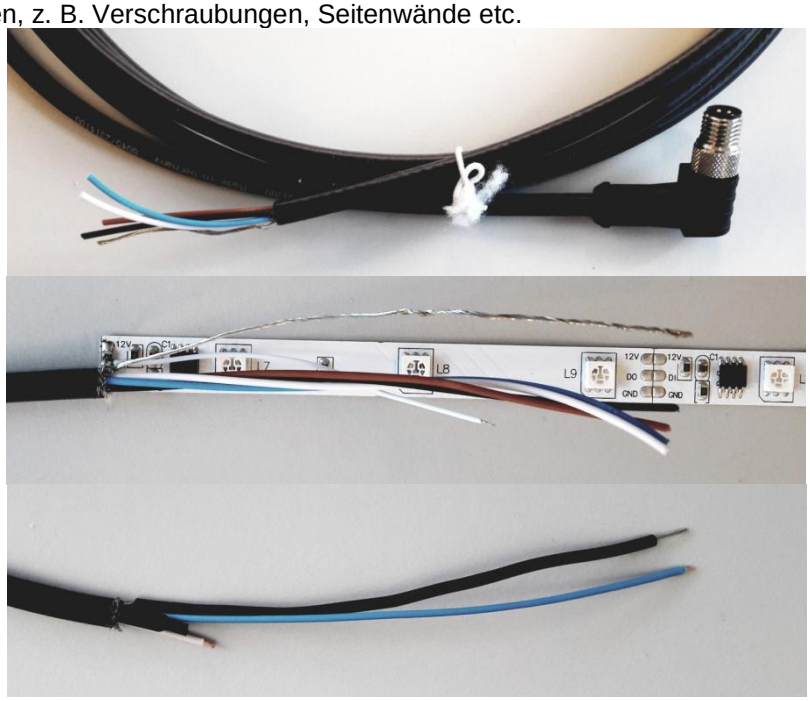


Konfektionierungsbeispiel eines LED-Streifens für 12V:

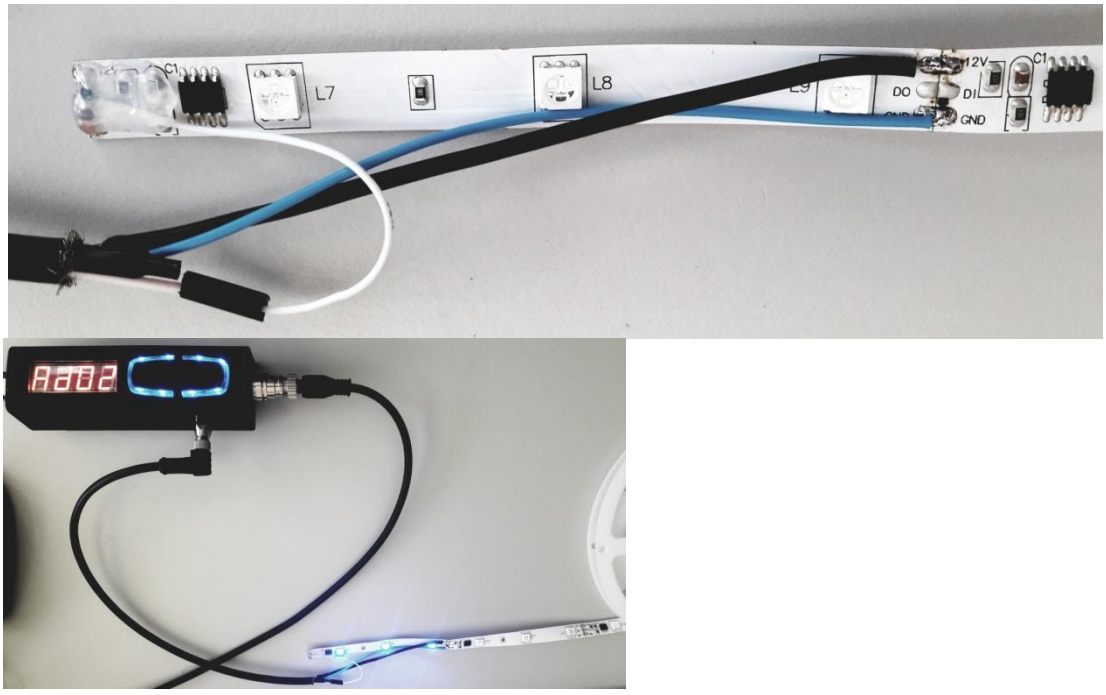
1. LED-Streifen auf die gewünschte Länge abschneiden, GND-Pad verzinnen, Diode bestücken und beidseitig anlöten (dieser Schritt entfällt bei einem mit 5V zu versorgenden LED-Streifen; Ausrichtung = schwarzer Ring beachten!), Widerstand bestücken und beidseitig anlöten. Hochflexibles Kabel an den Anschuss „DI“ anlöten. Bereich mit Heißkleber isolieren und mechanisch stabilisieren.



2. M8-Kabel auf die gewünschte Länge zuschneiden, wie gezeigt abmanteln, unbenutzte Drähte kürzen und ebenso wie den Beidraht mit Schrumpfschlauch isolieren. Nun durch alle Durchführungen durchstecken, z. B. Verschraubungen, Seitenwände etc.



3. Die drei Drähte anlöten, die Verbindung der beiden weißen Drähte mit Schrumpfschlauch isolieren und einen Funktionstest durchführen. Wichtig: Zuerst das M8-Kabel an das stromlose DPD anschließen und dann das M12-Kabel an das DPD anschließen.



4. LED-Streifen einbauen. Dabei für eine Zugentlastung sorgen.

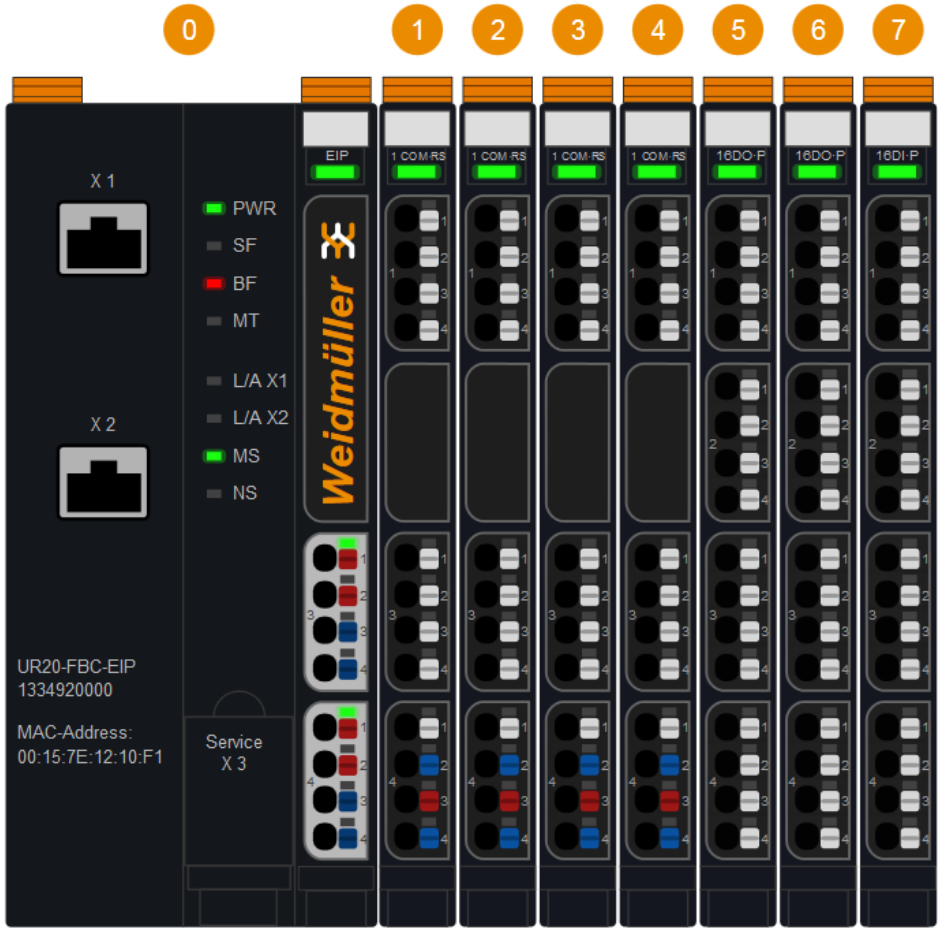
2 Weidmüller u-remote als Modbus-RTU-Master

2.1 Einführung

Das Weidmüller u-remote-System verbindet sich mit nahezu allen relevanten Feldbussen und ist deshalb auch die optimale Lösung, wenn es darum geht, die DPDs mit einer SPS zu verbinden. Bis zu 64 DPD-Linien können dabei in einer Station angesteuert werden.

2.2 Beispiel-Aufbau einer u-remote-Station

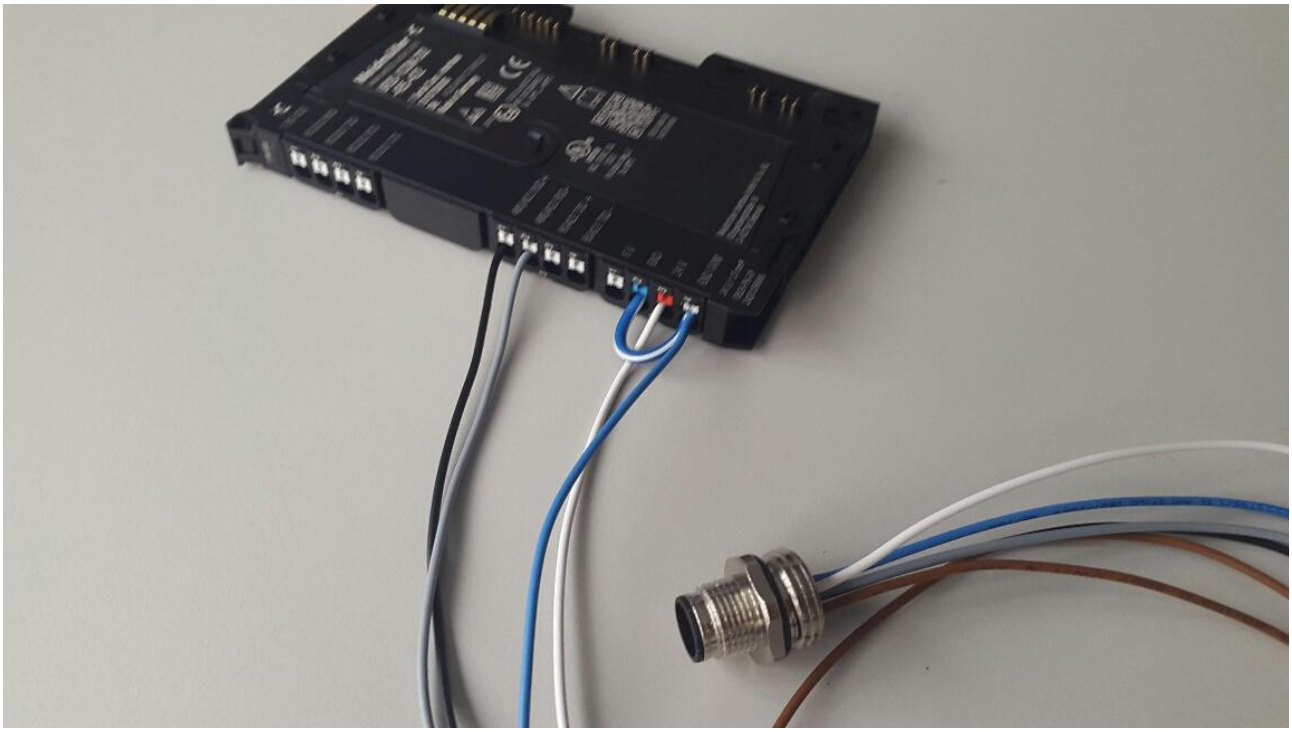
u-remote			
Bestellnr.	**Modul**	**Funktion**	**Bemerkungen**
1334920000	UR20-FBC-EIP	Buskoppler Ethernet-IP	Muss zur SPS passen
1315750000	UR20-1COM-485	1. Modbus RTU master	
1315750000	UR20-1COM-485	2. Modbus RTU master	
1315750000	UR20-1COM-485	3. Modbus RTU master	
1315750000	UR20-1COM-485	4.Modbus RTU master	
1315250000	UR20-16DO-P	16 digitale Ausgänge 0,5A	Für allgemeine Aufgaben
1315250000	UR20-16DO-P	16 digitale Ausgänge 0,5A	Für allgemeine Aufgaben
1315200000	UR20-16DI-P	16 digitale Eingänge Typ 3	Für allgemeine Aufgaben



2.3 Verbinden eines UR20-1COM-485 Moduls mit einer DPD-Linie

Zur Verbindung des Moduls mit der DPD-Line via Modbus-RTU kann z. B. der Weidmüller-Steckverbinder SAIE-M12S-5-0.5U-M16(1861230000 verwendet werden.

Bus Wiring			
M12 Pin	**UR20-1COM-485**	**Farbe 1861230000**	**Signal**
1 = Schirm	(connect to c-bar)		Schirm
2	FWT 4.3	|_____|	+28.8V (Strom < 4A!)
3	FWT 4.1 & 4.4		0V
4	FWT 3.1		RS-485 (A / Y / TxRx+)
5	FWT 3.2		RS-485 (B / Z / TxRx-)



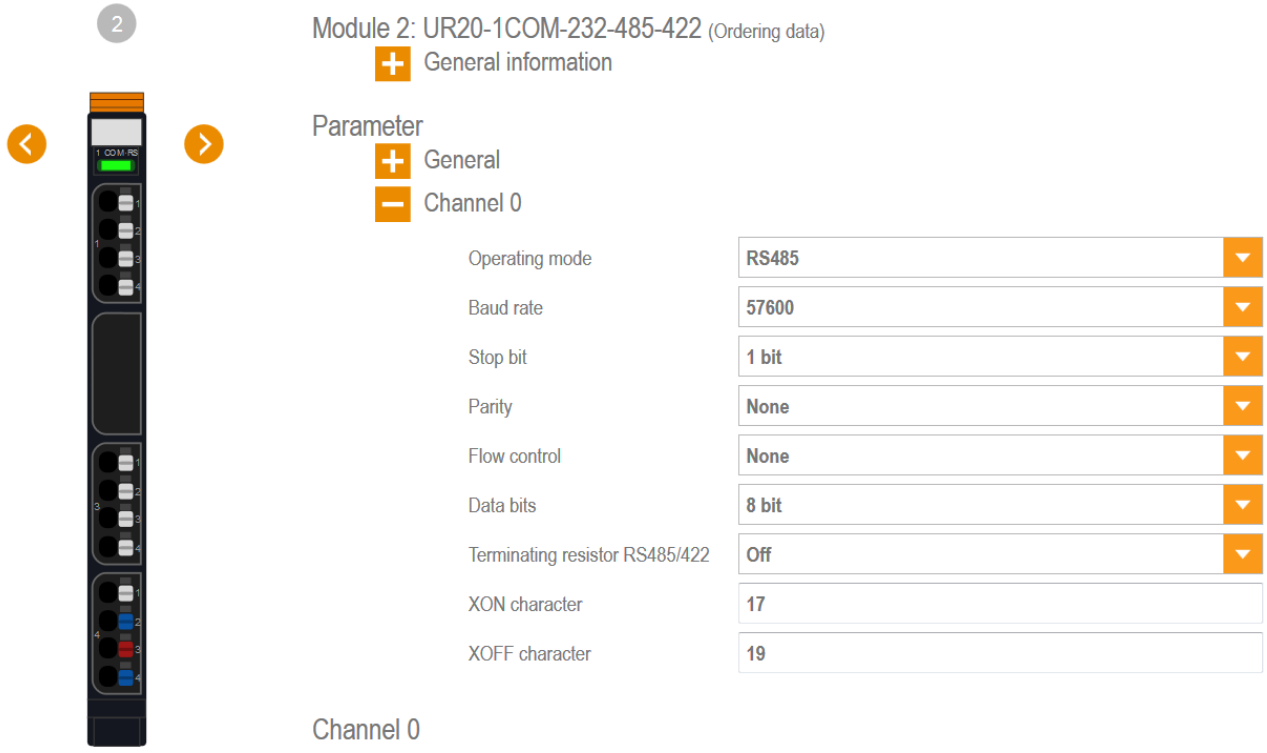
2.4 Modulkonfiguration UR20-1COM-232-485-422

Die Parameter können auf zwei verschiedene Arten eingestellt werden:

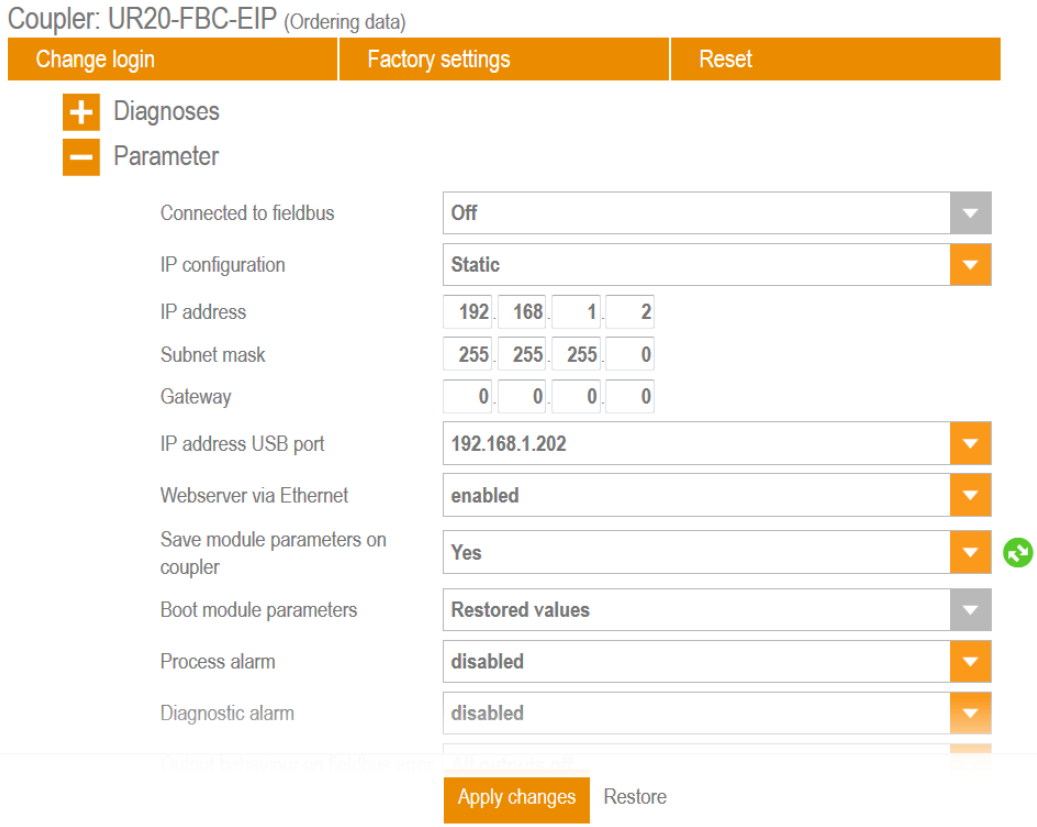
1. In der SPS
2. Durch Konfiguration und dauerhafte Speicherung im Feldbuskoppler via Web-Interface

Die erste Option steht nicht in allen Entwicklungsumgebungen zur Verfügung. Unterstützt wird diese Funktion beispielsweise von Siemens-SPSn. Die zweite Option ist unabhängig von der verwendeten SPS. Dazu wird eine Browser-Verbindung über Ethernet oder USB (virtual LAN) hergestellt. Der genaue Ablauf ist im Handbuch des Buskopplers beschrieben.

Stellen Sie dabei für jedes Modul den passenden Modus und die passende Baud-Rate ein:



Danach den FBC auswählen und “Save Modul parameters on coupler” auf “Yes” setzen. Zum Schluss “Apply changes”anklicken:



Bemerkung:

Die IP-Adresse ist beispielhaft und muss entsprechend den Anforderungen der SPS eingestellt werden.

2.5 u-remote-Datenmodell

Jedes UR20-1COM-485-Modul hat 16 Bytes Prozessdaten für Eingangs- und für Ausgangsdaten. 14 Bytes sind dabei als Datenspeicher für die RS-485-Kommunikation vorgesehen, ein Byte für die Anzahl der Datenbytes und jeweils eines für Kontroll- und Steuerungsaufgaben..

Achten Sie auf die Datenintegrität, soweit dies nicht bereits von der SPS sichergestellt wird. Im Zweifel sollte das Byte QB0 als letztes gesetzt werden.

Die beiden folgenden Tabellen zeigen die Ein- und Ausgangsdaten eines UR20-1COM-Moduls. Details entnehmen Sie bitte dem u-remote-Handbuch.

Process input data UR20-1COM-232-485-422

Byte	Format	Name	Bit	Description	Remarks
IB0	Word	Status and diagnosis	IX0.0	Data in the receive buffer	RX = 0: Receive buffer is empty RX = 1: A telegramm or telegramm segment in the receive buffer is ready for transmission.
			IX0.1	Receive buffer nearly full	Only 10 characters are left in the receive buffer. XOFF will be set if parametrerised.
			IX0.2	Not used *	
			IX0.3	RX_CNT	The RX_CNT value is assigned to each data segment of the process input data while transmission. The sequence or the RX_CNT values is: Binary: 00, 01, 10, 11, 00, ... Decimal: 0, 1, 2, 3, 0, ... A faulty data seqence indicates missing data segments.
			IX0.4	RX_CNT	
			IX0.5	TX_CNT_ACK	The TX_CNT_ACK value is a copy of the TX_CNT value, which has been transferred together with the last data segment of the process output data. TX_CNT_ACK acknowledges that the data has been taken over successfully.
			IX0.6	TX_CNT_ACK	
			IX0.7	STAT	STAT = 1: Communication with the device is without fault. STAT = 0: Faulty communication with the device.
IB1		Length of the data segment / of the subsequent diagnosis data		RX	Length of the data/diagnosis data in this frame
IB 2 ... IB 7 or IB 2 ... IB 15		Received data		User data of the transferred telegramm segment	

Anmerkung:

* IX0.2 wird als CRC_OK-Flag verwendet (gesetzt, wenn die CRC des empfangenen Paketes richtig ist); überprüfen Sie auf der Weidmüller-Internetseite, dass das Modul den aktuellen Firmwarestand besitzt.

Process output data UR20-1COM-232-485-422

Byte	Format	Name	Bit	Description	Remarks
QB0	Word	Status and diagnosis	QX0.0	RXBUF FLUSH	Bit 0: RXBUF FLUSH The receive buffer can be cleared using this bit. STATRES = 1: A requirement with RXBUF FLUSH = 1 will be ignored. STATRES = 0: The receive buffer will be cleared with RX-BUF FLUSH = 1.
			QX0.1	TXBUF FLUSH	Bit 1: TXBUF FLUSH The transmit buffer can be cleared using this bit. STATRES = 1: A requirement with TXBUF FLUSH = 1 will be ignored. STATRES = 0: The transmit buffer will be cleared with TXBUF FLUSH = 1.
			QX0.2	TX_HWBUFFER	Bit 2: DisableSend_TX_HWBUFFER This bit controls the hardware transmit buffer: DisableSend_TX_HWBUFFER = 0: The hardware transmit buffer is released. A character (Byte) will be sent as soon as it reaches the buffer. DisableSend_TX_HWBUFFER = 1: The hardware transmit buffer is locked. Characters (Bytes) will only be sent, when Disable-Send_TX_HWBUFFER is set to 0 again.
			QX0.3	TX_CNT	The TX_CNT value is assigned to each data segment of the process output data. The sequence or the TX_CNT values is: Binary: 00->01->10->11->00... Decimal: 0->1->2->3->0... A faulty data sequence indicates missing data segments.
			QX0.4	TX_CNT	
			QX0.5	RX_CNT_ACK	RX_CNT_ACK must include a copy of the RX_CNT value. The RX_CNT value has been transferred together with the last data segment of the process input data.
			QX0.6	RX_CNT_ACK	RX_CNT_ACK must be set in analogy with RX_CNT (in the status byte). It indicates that the data segment has been transferred successfully by using RX_CNT and enables to receive new data.
			QX0.7	Communication status	The input data status bit STAT will be reset using this bit. When changing from 1 to 0 (falling edge) STAT will be reset from 0 to 1. STAT = 0: All changes in the data fields TX_BYTE_CNT, TX_CNT and RX_CNT_ACK will be ignored. The receive or transmit buffer can be cleared using RXBUF FLUSH or TXBUF FLUSH respectively. STAT = 1 or changing from 0 to 1: The buffers cannot be cleared.
QB1		Length of the data segment *			
QB 2 ... QB 7 or QB 2 ... QB 15		Transmission data		User data of the transferred telegramm segment	

Anmerkung:

* QB1.7 wird als ADD_CRC-Flag verwendet (die UR20-1COM fügt dann automatisch 2 Bytes Modbus-CRC zu jedem Telegramm hinzu)[8]

[8] Firmware-Version 1.0.12 oder höher

2.6 Modbus-RTU Beispiel-Telegrammrahmen, gesendet über u-remote

Dieses Beispiel verwendet dieselben Dateninhalte wie das in Kapitel 1.11.

2.6.1 Leses des DPD-Statusbytes von DPD 18 = 0x12 (alle 8 “Input” Bits 1 00001…1 00008)

Ausgangsprozessdaten:

Control data									Modbus RTU data to send					
QB0 - Control byte								QB1	QB2	QB3	QB4	QB5	QB6	QB7
bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Length/CRC	Data0	Data1	Data2	Data3	Data4	Data5
1	RX_CNT_ACK	TX_CNT		0	0	0		0x86 134 = -122	DPD address 0x12 18	Modbus Cmd 0x02 2	Modbus Addr 0x00 0	Modbus Addr 0x00 0	Modbus Cnt 0x00 0	Modbus Cnt 0x08 8

Anmerkung:

“TX_CNT” muss hochgezählt werden, wenn neue Daten für das Modul UR20-1COM-485 verfügbar sind. QB1.7 ist gesetzt, sodass die CRC automatisch angefügt wird.

Spätestens 5ms nachdem das UR20-1COM-Modul das Datenpaket gesendet hat, sind die Prozesseingangsdaten verfügbar, wenn ein DPD mit der Nummer 18 verbunden war:

Status data									Modbus RTU data received (sent by DPD)					
IB0 - Status byte								IB1	IB2	IB3	IB4	IB5	IB6	IB7
bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Length	Data0	Data1	Data2	Data3	Data4	Data5
1	TX_CNT_ACK	RX_CNT		CRC OK	0	1		0x06 6	DPD address 0x12 18	Modbus Cmd 0x02 2	Modbus Cnt 0x01 1	DPD status 0x01 1	Modbus CRC 0x64 100	Modbus CRC 0xcc 204 = -52

Die angeforderte Information befindet sich im vierten empfangenen Byte. Überprüfen Sie zusätzlich das CRC_OK-Flag auf mögliche Übertragungsfehler. Im Falle eines Fehlers verwerfen Sie das empfangene Paket und fordern die Information erneut an. Die beiden empfangenen CRC-Bytes in den Registern IB6 und IB7 müssen nicht überprüft werden.
Wenn kein DPD mit der Nummer 18 verbunden ist, wird keine Antwort empfangen.

Anmerkung:
Die Prozesseingangsdaten bleiben solange unverändert, wie das Auslesen nicht durch kopieren der RX_CNT-Bits des Registers IB0 auf die Bits RX_CNT_ACK im Command Data Byte QB0 kopiert wurden!

2.6.2 Schreiben des Displays und Setzen der LEDs von DPD 18 = 0x12 (4 00001…4 00006)

Prozessausgangsdaten:

Control data									Modbus RTU data to send												
QB0 - Control byte								QB1	QB2	QB3	QB4	QB5	QB6	QB7	QB8	QB9	QB10	QB11	QB12	QB13
bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Length/CRC	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Data8	Data9	Data10	Data11
1	RX_CNT_ACK		TX_CNT		1	0	0	0x87 135 = -121	DPD address 0x12 18	Modbus Cmd 0x10 16	Modbus Addr 0x00 0	Modbus Addr 0x00 0	Modbus Cnt 0x00 0	Modbus Cnt 0x06 6	Modbus Cnt2 0x0c 12					
1	RX_CNT_ACK		TX_CNT+1		0	0	0	0x8c 140 = -116	0x00 0	Display0 0x31 49	0x00 0	Display1 0x2d 45	0x00 0	Display2 0x31 49	0x00 0	Display3 0x38 56	0x00 0	color 0x42 66	Cattribute1 0x46 70	Cattribute0 0x49 73

Anmerkung:

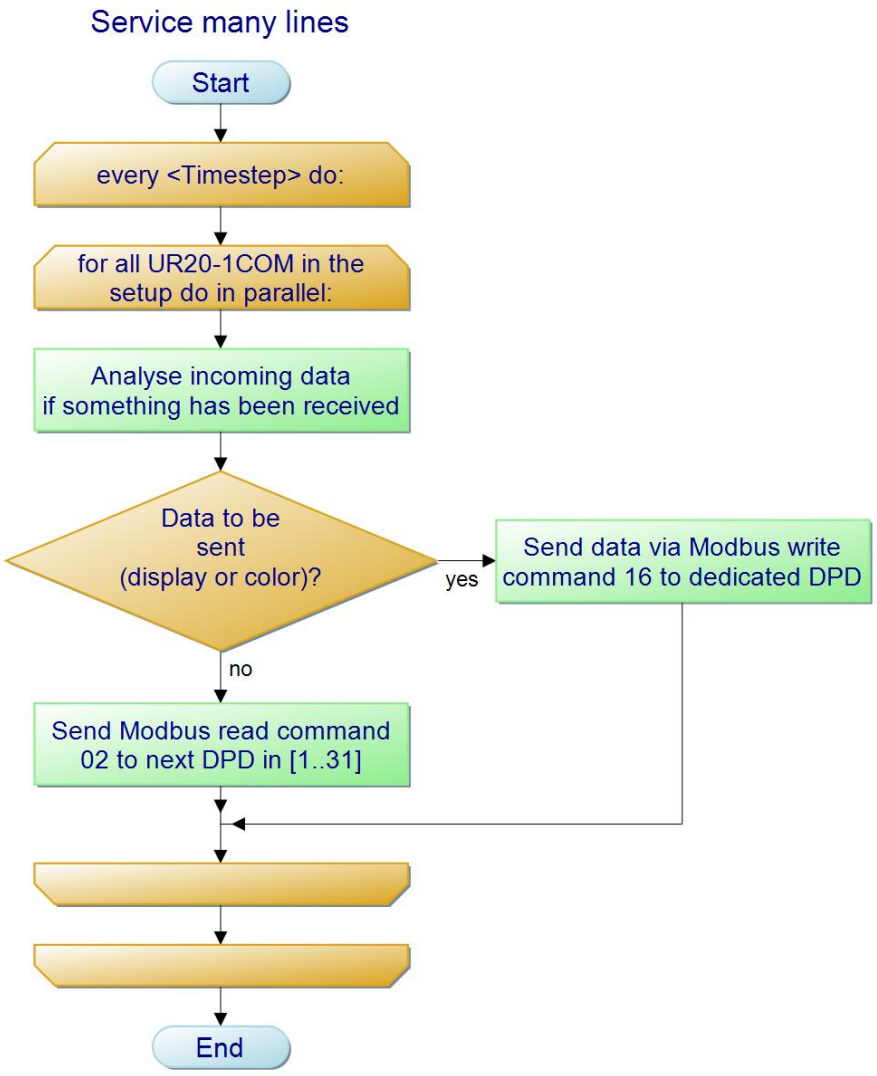
Es werden zwei Buszugriffszyklen für das Übertragen per “Modbus RTU multiple register write“ benötigt. Werten Sie die zurückgelesenen Bits RX_CNT_ACK aus, wenn Sie den Gesamtzyklus auf eine Minimum beschleunigen möchten oder warten Sie mindestens 1ms zwischen zwei Schreibzugriffen auf das u-remote-System!

Spätestens 6ms nachdem das Modbus-Kommando gesendet wurde, stellt das Modul UR20-1COM-485 die Antwort in den Prozesseingangsdaten zur Verfügung, wenn ein DPD mit der Nummer 18 angeschlossen war:

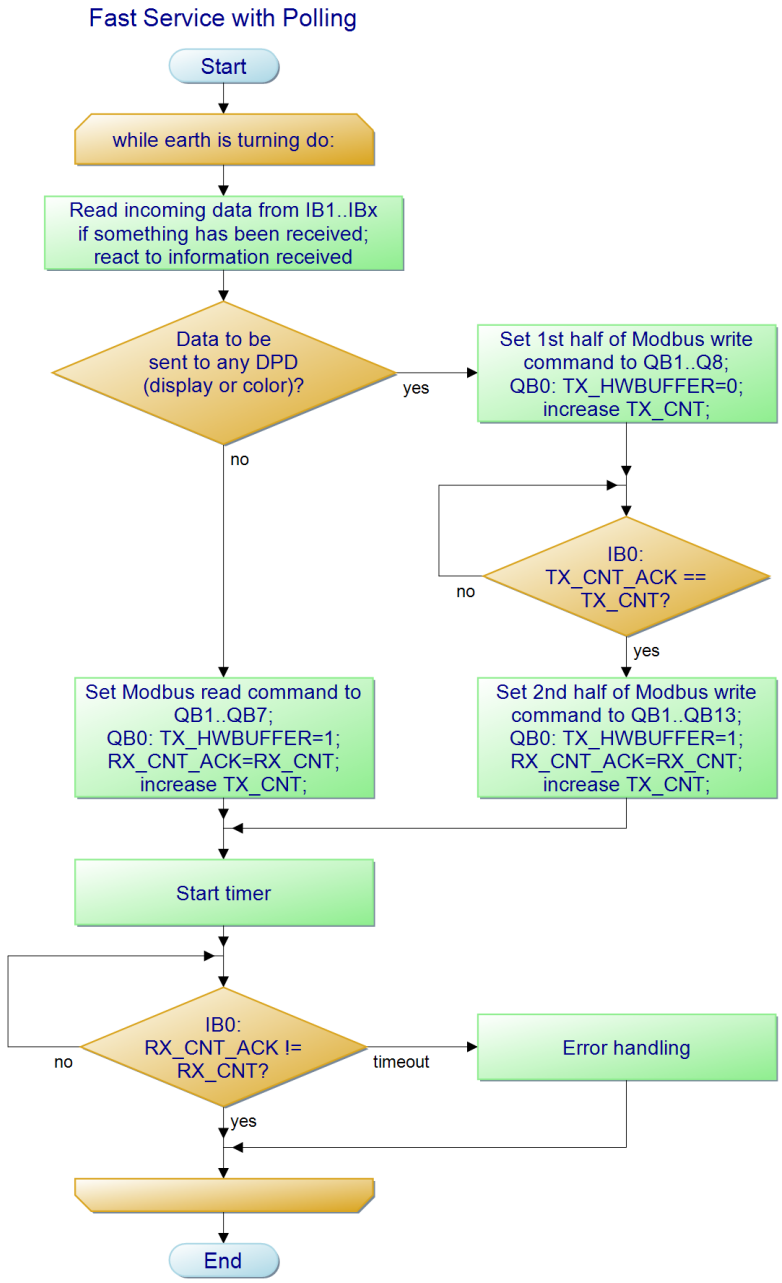
Status data									Modbus RTU data received (sent by DPD)							
IB0 - Status byte								IB1	IB2	IB3	IB4	IB5	IB6	IB7	IB8	IB9
bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Length	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7
1	TX_CNT_ACK		RX_CNT		CRC OK	0	1	0x08 8	DPD address 0x12 18	Modbus Cmd 0x10 16	Modbus Addr 0x00 0	Modbus Addr 0x00 0	Modbus Cnt 0x00 0	Modbus Cnt 0x06 6	Modbus CRC 0x42 66	Modbus CRC 0xa8 168 = -88

Mit nur diesen beiden Kommandos können Sie ein DPD-System vollständig betreiben. Da moderne SPS-Feldbusse vielfach schneller sind als Modbus-RTU, können Sie alle UR20-1COM-485 Module parallel mit daten versorgen, so dass bei 31 DPDs die Gesamtzeit für die Abfrage aller installierten DPDs weniger als 0,5 Sekunden beträgt.

Ein Beispielalgorithmus für eine SPS ist nebenstehend abgebildet:



Ein zeitminimierter Gesamtalgorithmus kann wie folgt implementiert werden:



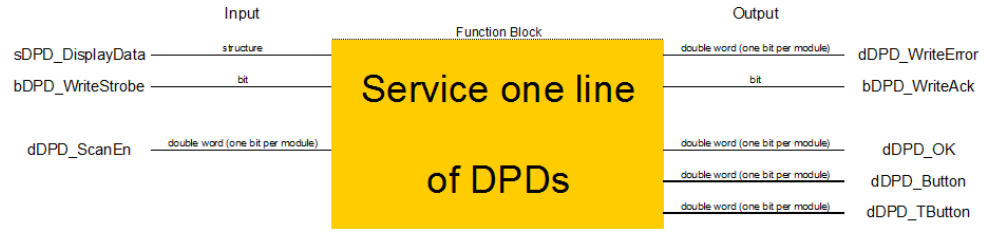
2.6.3 Einstellen der Modbus RTU-Adresse des DPDs

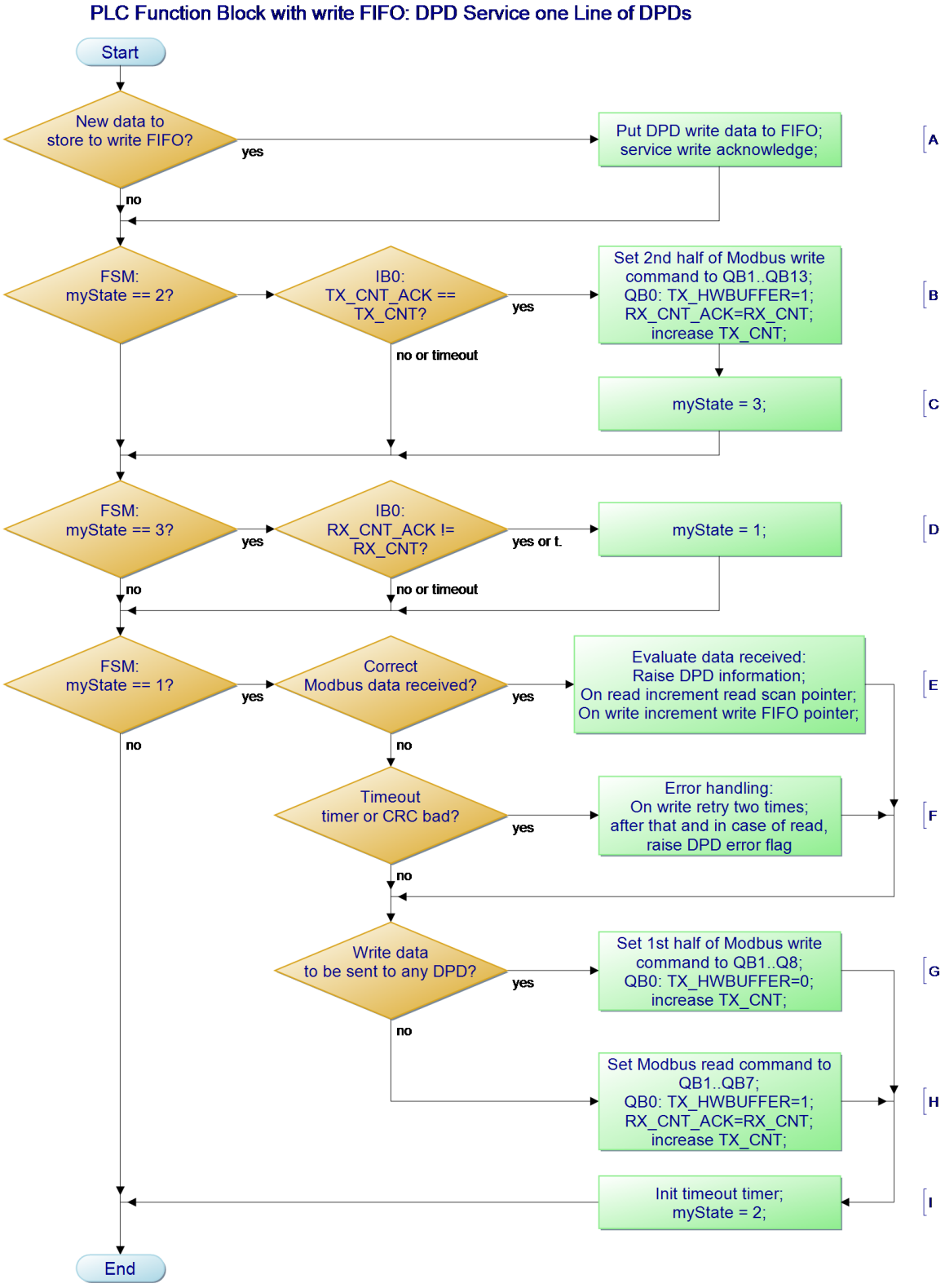
Weitere Modbus-RTU-Kommandos können in Anlehnung an das oben vorgestellte Schema übertragen werden.

Alle DPDs einer Linie können mit folgendem nicht-Modbus-Kommando auf die Default-Adresse 0x1f = 31 zurückgesetzt werden. Es wird keine CRC benötigt:

Control data									Serial data to send (no Modbus)									
QB0 - Control byte								QB1	QB2	QB3	QB4	QB5	QB6	QB7	QB8	QB9	QB10	QB11
bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Length/CRC	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Data8	Data9
1	RX_CNT_ACK		TX_CNT		0	0	0	0x0a 10	"A" 0x41 65	"T" 0x54 84	Space 0x20 32	"S" 0x53 83	"S" 0x53 83	"L" 0x4c 76	"," 0x2c 44	"3" 0x33 51	"1" 0x31 49	LF 0x0a 10

2.7 Funktionsbaustein für SPSn (Vorschlag)





3 Änderungshistorie der Firmware

Änderungshistorie der Firmware	
Version	**Beschreibung**
1.0.0	Initiale Version
1.1.0	Implementierung des “toggle bits” bei Tastendruck zusätzlich zum key flag Unterstützt Modbus-Funktion 04 HW- und FW-Versions-IDs integriert
1.2.0	Bugfix für Ziffernverschiebung im Display
1.2.4	Adressen eines DPDs im Auslieferungszustand (Adresse 31) können im Zusammenspiel mit dem Tastendruck gesetzt werden
1.2.6	Tasten-Flag wird beim _Loslassen_ der Taste gesetzt. Display zeigt beim Start zusätzlich die hinteren beiden Stellen der Firmware-ID an Wird der externe Eingang aktiv, wird Bit7 des Status-Registers zusätzlich negiert Liefert Seriennummer beim Modbus read_multiple_register-Kommando zurück Modus zum direkten Einstellen der Modbus-Adresse eingebaut Selbsttest beim Einschalten: Alle Farben und das LED-Display werden getestet
1.2.7	Unterstützung von LED-Streifen am M8-Stecker Unterstützung von Modbus-Adressen 1-60 Kürzere Anzeige, wenn DPD auf Adresse 31 zurückgesetzt wird Reagiert zusätzlich auf Modbus-Adresse 0 (broadcast)

Weidmüller reserves the right to make changes without further notice to any products herein. Weidmüller makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Weidmüller assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation special, consequential or incidental damages.Buyer is responsible for its products and applications using Weidmüller products, including compliance with all laws, regulations and safety requirements or standards, regardless of any support or applications information provided by Weidmüller. “Typical” parameters which may be provided in Weidmüller data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals” must be validated for Jedes customer application by customer’s technical experts. Weidmüller does not convey any license under its patent rights nor the rights of others. Weidmüller products are not designed, intended, or authorized for use as a critical component in life support systems or any FDA Class 3 medical devices or medical devices with a same or similar classification in a foreign jurisdiction or any devices intended for implantation in the human body. Should Buyer purchase or use Weidmüller products for any such unintended or unauthorized application, Buyer shall indemnify and hold Weidmüller and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Weidmüller was negligent regarding the design or manufacture of the part. Weidmüller is an Equal Opportunity/Affirmative Action Employer. This literature is subject to all appliKabel copyright laws and is not for resale in any manner.

Weidmüller Interface GmbH & Co. KG
Klingenbergstraße 16
32758 Detmold
Germany

Phone +49 (0) 5231 14-0
Fax +49 (0) 5231 14-2083
E-Mail info@weidmueller.com
Internet www.weidmueller.com