Applikationsbeschreibung

EDIZIOdue colore
UNI-Taster 392x – Steckverbinder

10.UNI3924-D.2104/210409

Feller
by Schneider Electric

1 Einleitung

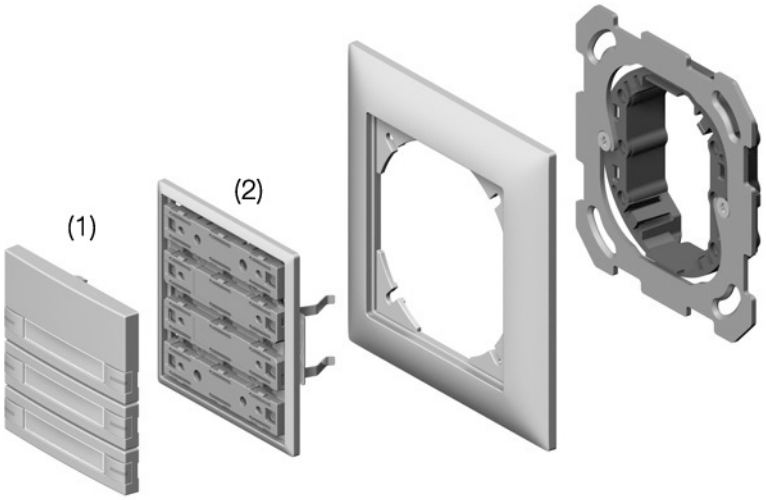
Dieses Dokument dient als Spezifikation des EDIZIOdue colore UNI-Tasters 392x. Es legt die Schnittstelle offen und erläutert alle Funktionen, welche das Gerät anbietet.

Dieses Dokument hat Gültigkeit für die Produkte mit den Frontplatten :

- 900-3924.FMI… für UNI-Taster 1-4fach, 1/1 und 1/2 Tasten, ohne LED's
- 900-3924.FMI.L… für UNI-Taster 1-4fach, 1/1 und 1/2 Tasten, mit LED's
- 900-3928.FMI… für UNI-Taster 4-8fach, 1/1, 1/2 und 1/4 Tasten, ohne LED's
- 900-3928.FMI.L… für UNI-Taster 4-8fach, 1/1, 1/2 und 1/4 Tasten, mit LED's

1.1 System-Aufbau

Der UNI-Taster (1) kann durch Drücken der Tasten Befehle absetzen und über die LEDs Statusinformationen anzeigen.
Über die physikalische Schnittstelle kann der Taster auf einen Busankoppler (2) aufgesteckt werden. Über diese Schnittstelle kommuniziert der Busankoppler mit dem Taster.
Der Busankoppler kann grundsätzlich auf ein beliebiges System aufsetzen und wird vom Kunde selbst gebaut oder eingekauft.
Die Art der Befehle/Funktionen des Tasters wird durch den Busankoppler bestimmt.



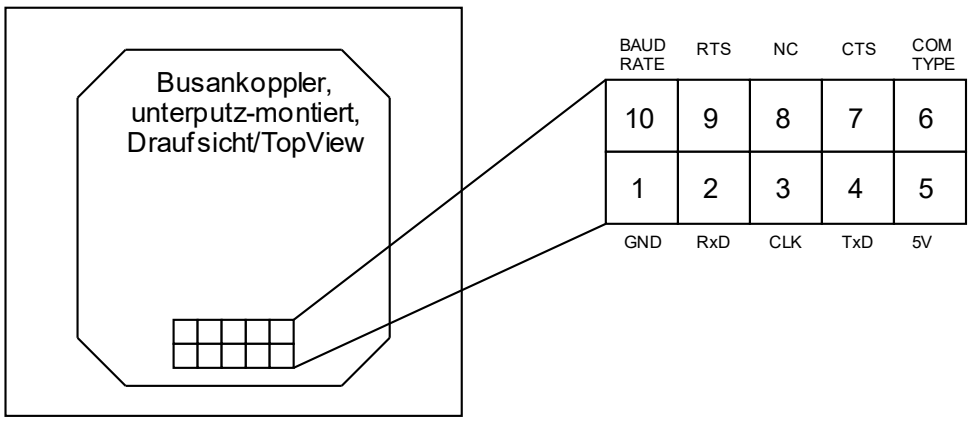
2 Physikalische Schnittstelle

Als physikalische Schnittstelle zwischen dem Busankoppler und dem Taster wird eine 2x5-polige Steckverbindung im Raster 2,54 mm verwendet:

- Busankoppler: Buchsenleiste
- Taster: Stiftleiste

Es wird empfohlen, dass die Buchsenleiste des Busankopplers über vergoldete Kontakte verfügt. Somit können Kontaktprobleme durch Langzeit-Korrosion vermieden werden.

Die Pinbelegung der physikalische Schnittstelle ist nachfolgend aus Sicht des Busankopplers (Datenflussrichtung) definiert:



Pin	Bezeichnung	Beschreibung
1	GND	Masse
2	RxD	Empfangsleitung Daten (Busankoppler empfängt / Taster sendet über diese Leitung)
3	CLK	Takt für Datenleitungen
4	TxD	Sendeleitung Daten (Busankoppler sendet / Taster empfängt über diese Leitung)
5	5V	Speisung 5 V DC
6	COM TYPE	Kommunikations-Modus (wird durch Spannungswert an diesem Pin definiert, siehe Modus)
7	CTS	Empfangsleitung Handshake, Flusskontrolle (Busankoppler empfängt / Taster sendet über diese Leitung)
8	NC	nicht angeschlossen (reserviert für zukünftige Anwendungen, Pin offen lassen)
9	RTS	Sendeleitung Handshake, Flusskontrolle (Busankoppler sendet / Taster empfängt über diese Leitung)
10	BAUD RATE	Baud Rate (wird durch Spannungswert an diesem Pin definiert, siehe Baud Rate)

2.1 Speisung

Der Taster muss vom Busankoppler gespeist werden. Die Qualitätsanforderungen an die Speisung sind wie folgt:

Bezeichnung	min.	typ.	max.
Spannung	4,75 V DC	5 V DC	5,25 V DC
Stromverbrauch @ 0 LED's	-	1 mA DC	-
Stromverbrauch @ 4 LEDs rot/grün, 50% Helligkeit	-	3,4 mA DC	-
Stromverbrauch @ 4 LEDs rot/grün, 100% Helligkeit	-	5,5 mA DC	-
Stromverbrauch @ 8 LEDs rot/grün, 50% Helligkeit	-	5,7 mA DC	-
Stromverbrauch @ 8 LEDs rot/grün, 100% Helligkeit	-	9,8 mA DC	-
Stromverbrauch @ 4 LEDs blau, 50% Helligkeit	-	5,8 mA DC	-
Stromverbrauch @ 4 LEDs blau, 100% Helligkeit	-	10,2 mA DC	-
Stromverbrauch @ 8 LEDs blau, 50% Helligkeit	-	10,5 mA DC	-
Stromverbrauch @ 8 LEDs blau, 100% Helligkeit	-	19,2 mA DC	-

Die Pegel der Kommunikationsleitungen sind ebenfalls 5 V DC.

3 Kommunikations-Modus

3.1 Voreinstellungen

3.1.1 Kommunikations-Modus

Der Kommunikations-Modus wird mittels einem Widerstand zwischen Pin 6 (COM TYPE) und Pin 5 (5V) der physikalischen Schnittstelle bestimmt.
Der Kommunikations-Modus wird bei jedem Programmstart (Hardware Power On Reset, Software Reset) ermittelt. Während dem Betrieb kann der Kommunikations-Modus auch per Software verändert werden.

Com Type Nr.	Beschreibung	Widerstandswert, Toleranz
99	nicht definiert, Default Kommunikations-Modus Nr. 11 wird verwendet	Kein Widerstand
12	reserviert	82 kΩ, 1%
11	**Protokoll, ohne Handshake, asynchron, Default**	**68 kΩ, 1%**
10	reserviert	56 kΩ, 1%
09	Protokoll, mit Handshake, asynchron	47 kΩ, 1%
08	reserviert	39 kΩ, 1%
07	Protokoll, mit Handshake, synchron	33 kΩ, 1%
06	Byteaustausch, synchron, LED-Farbe: rot	27 kΩ, 1%
05	Byteaustausch, synchron, LED-Farbe: grün	22 kΩ, 1%
04	Byteaustausch, synchron, LED-Farbe: blau	18 kΩ, 1%
03	Byteaustausch, asynchron, LED-Farbe: rot	15 kΩ, 1%
02	Byteaustausch, asynchron, LED-Farbe: grün	12 kΩ, 1%
01	Byteaustausch, asynchron, LED-Farbe: blau	10 kΩ, 1%

3.1.2 Baud Rate

Die Baud Rate wird mittels einem Widerstand zwischen Pin 10 (BAUD RATE) und Pin 5 (5V) der physikalischen Schnittstelle bestimmt.
Die Baud Rate wird bei jedem Programmstart (Hardware Power On Reset, Software Reset) ermittelt. Während dem Betrieb kann die Baud Rate auch per Software verändert werden. Die Baud Rate ist nur bei den asynchronen Kommunikations-Modi wirksam.

Baud Rate Nr.	Beschreibung	Widerstandswert, Toleranz
99	nicht definiert, Default Baud Rate Nr. 05 wird verwendet	Kein Widerstand
12	reserviert	82 kΩ, 1%
11	115200 Baud	68 kΩ, 1%
10	reserviert	56 kΩ, 1%
09	57600 Baud	47 kΩ, 1%
08	38400 Baud	39 kΩ, 1%
07	19200 Baud	33 kΩ, 1%
06	reserviert	27 kΩ, 1%
05	**9600 Baud, Default**	**22 kΩ, 1%**
04	4800 Baud	18 kΩ, 1%
03	2400 Baud	15 kΩ, 1%
02	1200 Baud	12 kΩ, 1%
01	reserviert	10 kΩ, 1%

3.2 Byteaustausch

Der Kommunikations-Modus Byteaustausch besteht aus einem einfachen Protokoll mit fest eingestellten Kommunikations-Parametern.
Durch Verwendung dieses Protokolls können nicht alle Funktionen des Tasters verwendet werden.

Der Busankoppler startet die Kommunikation, in dem er den Zustand der LEDs mittels 1 Byte sendet. Der Taster antwortet darauf, in dem er den Zustand der Tasten mittels 1 Byte sendet.

Das LED-Byte ist wie folgt codiert:

D7	D6	D5	D4	D3	D2	D1	D0
L8	L7	L6	L5	L4	L3	L2	L1

Bezeichnung	Beschreibung
L8..L1	Zustände der LEDs L8..L1. Die LED-Farbe wird mit dem Widerstandswert beim Kommunikations-Modus gewählt. 1 = LED eingeschaltet 0 = LED ausgeschaltet

Das Tasten-Byte ist wie folgt codiert:

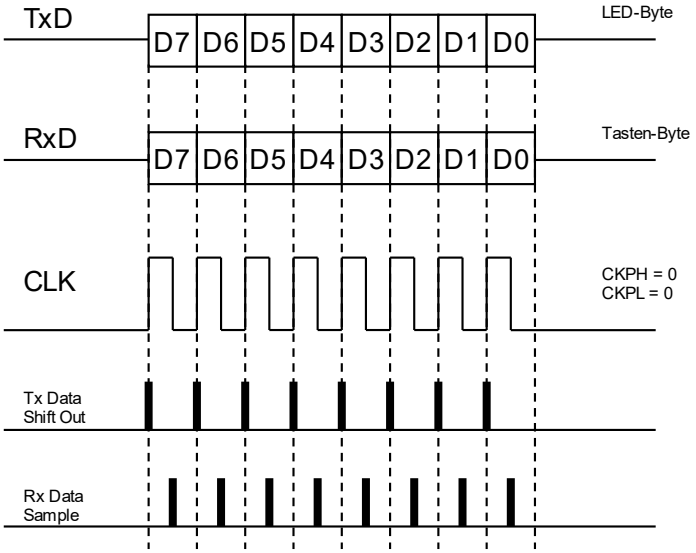
D7	D6	D5	D4	D3	D2	D1	D0
T8	T7	T6	T5	T4	T3	T2	T1

Bezeichnung	Beschreibung
T8..T1	Zustände der Tasten T8..T1. Die Tasten sind bereits entprellt. 1 = Taste gedrückt 0 = Taste losgelassen

3.2.1 Synchron

Im Synchron-Modus (SPI) wird der Busankoppler als Clock-Master vorausgesetzt. Das heisst, der Busankoppler generiert den Takt auf der CLK-Leitung und definiert somit auch implizit die Baud Rate. Die maximal zulässige Baud Rate ist beim Service SetSystemSettings.request ersichtlich.

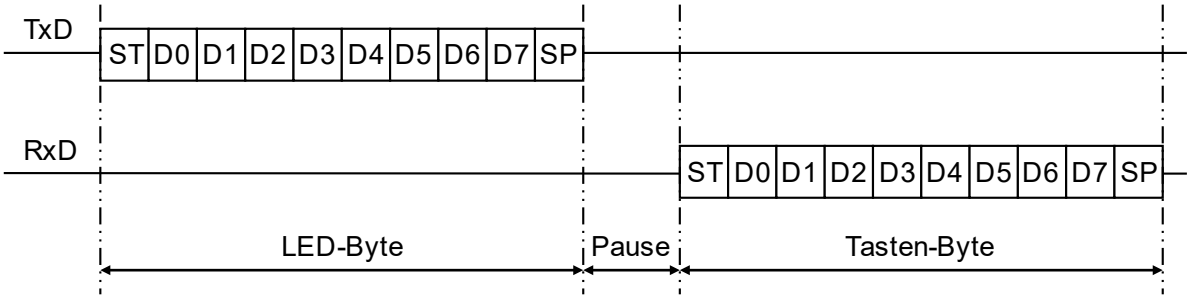
Jedes Byte wird nach folgendem Ablauf gesendet und empfangen:



Bezeichnung	Beschreibung
D7..D0	Daten-Bits
CKPH	Clock Phase normal (nicht verzögert)
CKPL	Clock Parität Idle LOW

3.2.2 Asynchron

Jedes Byte wird nach folgendem Ablauf gesendet und empfangen:



Bezeichnung	Beschreibung
ST	Start-Bit
D0..D7	Daten-Bits
SP	1 Stop-Bit
Parität	keine
Baud Rate	Gemäss Widerstand Pin 10/5, siehe physikalische Schnittstelle
Pause	100 * (1 / Baud Rate) [s]

3.3 Protokoll, mit Handshake

Der Kommunikations-Modus Protokoll mit Handshake besteht aus einem fortgeschrittenen Protokoll mit einstellbaren Kommunikations-Parametern.
Durch Verwendung dieses Protokolls können alle Funktionen des Tasters voll verwendet werden.

Die Verwendung eines Handshakes bringt neben dem technischen Mehraufwand folgende Vorteile mit sich:

- Flusskontrolle auf Byte-Ebene durch Hardware Handshake
- Zusätzliche Datenrichtungs-Flusskontrolle durch Software Handshake
- Einfacher zu implementieren auf Systemen mit niedriger Kommunikationsgeschwindigkeit, grosser Scheduler-Durchlaufzeit oder fehlender Interrupt-Fähigkeit.

Der Hardware Handshake findet bei jedem Byte statt.

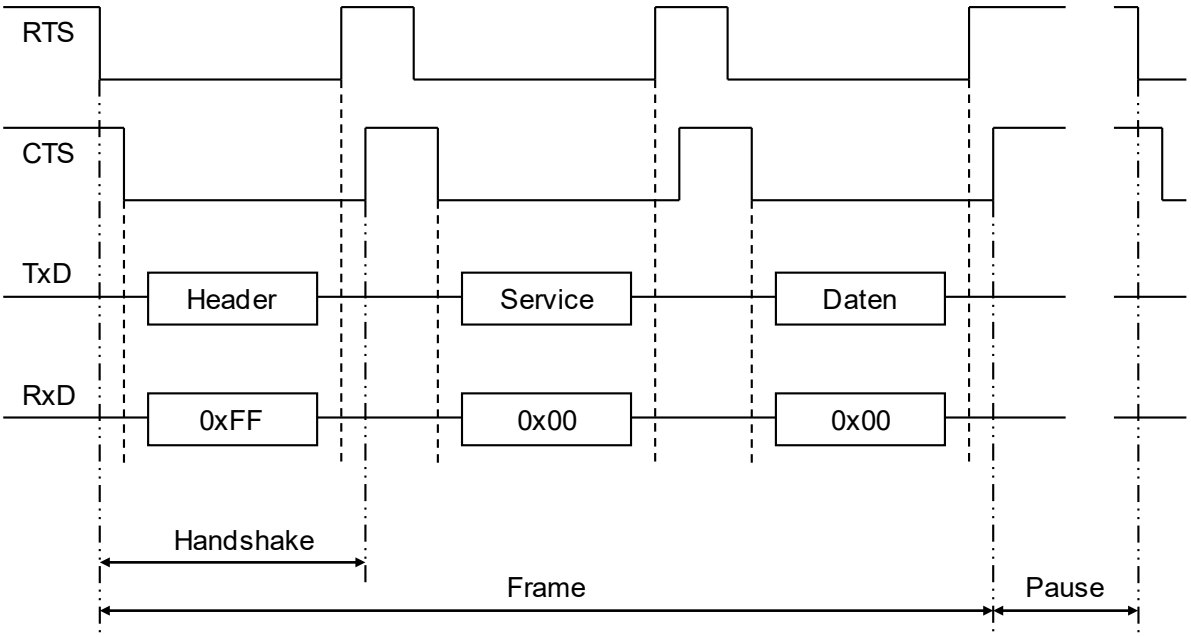
- Der Sender setzt seine RTS-Leitung auf LOW und wartet, bis der Empfänger die Bereitschaft signalisiert, indem er seine RTS-Leitung (CTS-Leitung des Senders) ebenfalls auf LOW setzt.
- Die Daten (1 Byte) werden übertragen.
- Der Sender setzt seine RTS-Leitung auf HIGH und wartet, bis der Empfänger das Kommunikationsende signalisiert, indem er seine RTS-Leitung (CTS-Leitung des Senders) ebenfalls auf HIGH setzt.

Der Sofware Handshake findet bei jedem Byte statt.

- Beim ersten Byte sendet der Sender den Frame Header (Wert ungleich 0xFF). Der Empfänger quittiert das erste Byte mit dem Wert 0xFF.
- Bei jedem weiteren Byte sendet der Sender die Daten. Der Empfänger quittiert jedes Byte mit dem Wert 0x00.

Zwischen jedem Frame besteht eine minimale Pause, bevor ein nächster Frame gsendet werden kann. Mehr Informationen dazu sind zu finden beim Service SetSystemSettings.request.

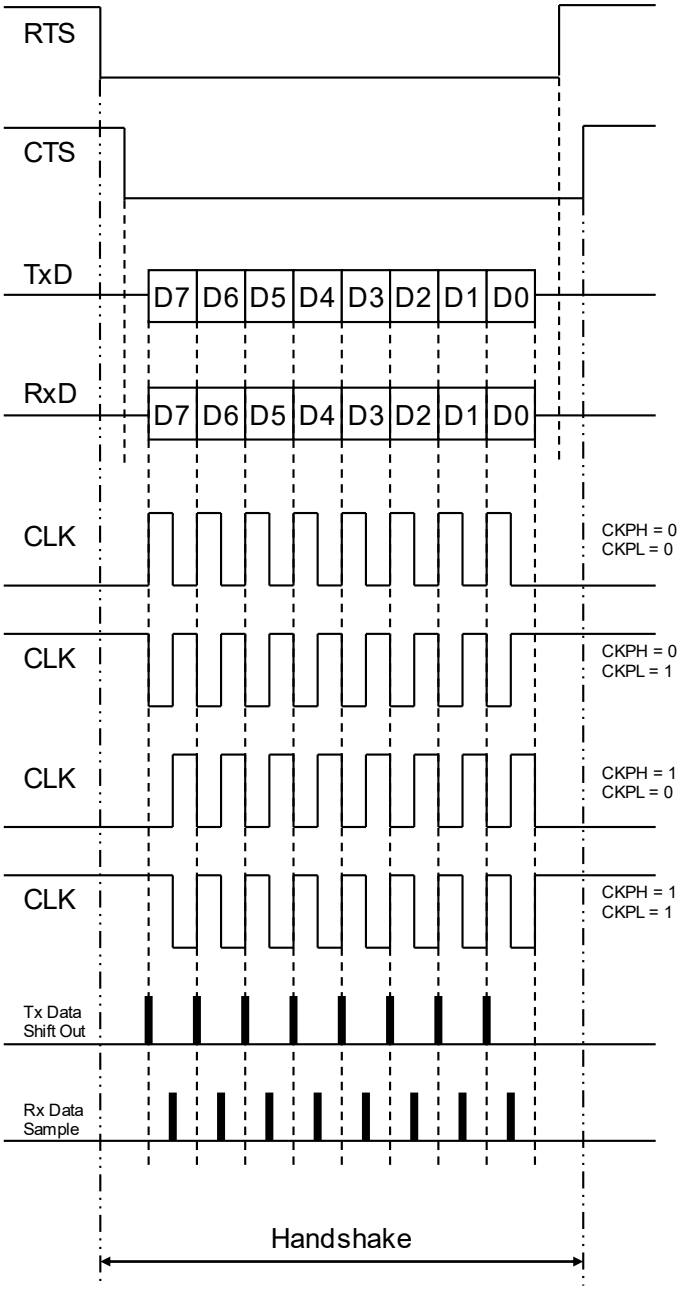
Nachfolgend ist das Prinzip des Handshakes anhand eines Frames mit 1 Byte Daten im zeitlichen Ablauf dargestellt:



3.3.1 Synchron

Im Synchron-Modus (SPI) wird der Busankoppler als Clock-Master vorausgesetzt. Das heisst, der Busankoppler generiert den Takt auf der CLK-Leitung und definiert somit auch implizit die Baud Rate. Die maximal zulässige Baud Rate ist beim Service SetSystemSettings.request ersichtlich.

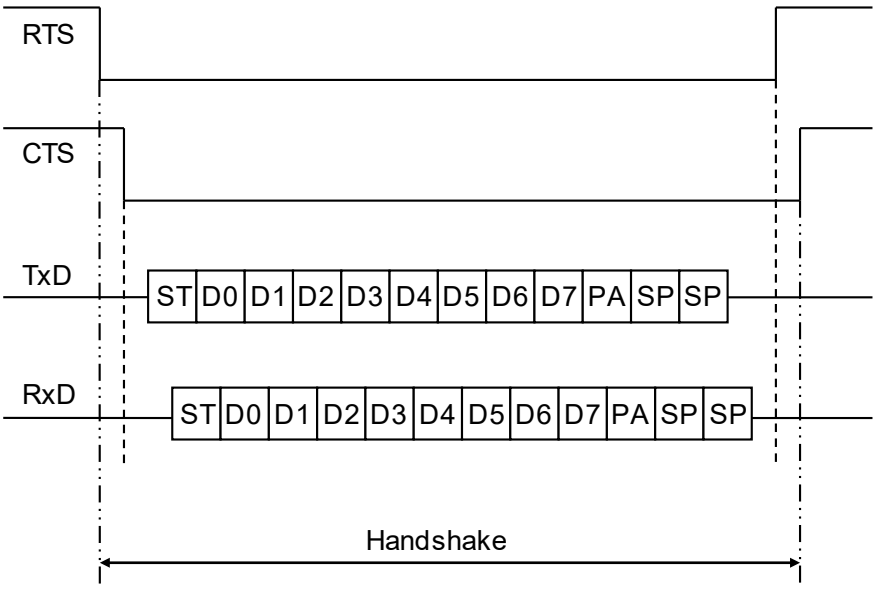
Jedes Byte wird nach folgendem Ablauf gesendet und empfangen:



Bezeichnung	Beschreibung
D7..D0	Daten-Bits
CKPH	Clock Phase konfigurierbar über Service SetSystemSettings.request
CKPL	Clock Parität konfigurierbar über Service SetSystemSettings.request

3.3.2 Asynchron

Jedes Byte wird nach folgendem Ablauf gesendet und empfangen:



Bezeichnung	Beschreibung
ST	Start-Bit
D0..D7	Daten-Bits
PA	Paritäts-Bit konfigurierbar über Service SetSystemSettings.request
SP	1..2 Stop-Bits konfigurierbar über Service SetSystemSettings.request

3.3.3 Fehlerbehandlung

Für die Übermittlung eines Bytes und den korrekten Abschluss des Handshakes besteht ein Timeout gemäss Service SetSystemSettings.request. Falls diese Zeit überschritten wird, besteht ein Kommunikationsfehler.

Nach einem Protokoll-/ oder Kommunikationsfehler wird eine bestehende Verbindung abgebrochen und ein allfällig unvollständiger Frame gelöscht. Der Taster setzt seine RTS-Leitung auf HIGH, bevor er eine allfällig nächste Verbindung startet. Setzt der Busankoppler seine RTS-Leitung nicht ebenfalls auf HIGH, wird ein neuer Kommunikationsversuch seitens Busankoppler angenommen und akzeptiert.

3.3.4 Protokoll-Master

Der Busankoppler ist als Protokoll-Master definiert.

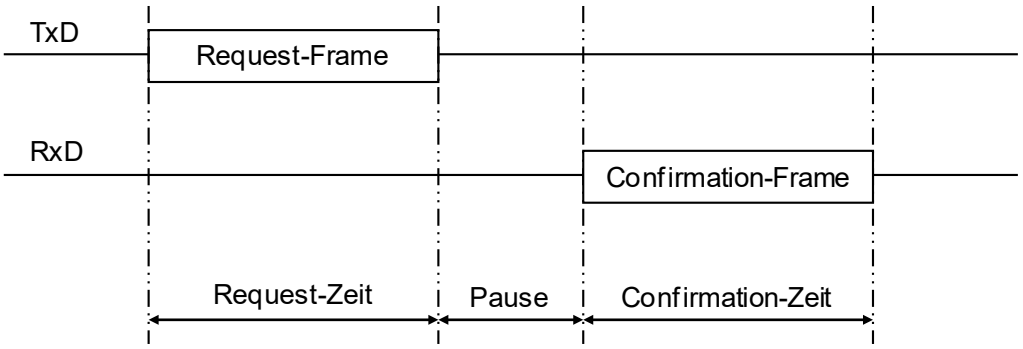
Es besteht die Möglichkeit, dass sowohl Busankoppler wie auch Taster gleichzeitig eine Kommunikation aufbauen möchten. Dies äussert sich dadurch, dass beide Teilnehmer im 1. Byte eines Frames einen gültigen Frame Header (Wert ungleich 0xFF) senden.
In diesem Fall kann der Busankoppler (als Protokoll-Master) wie gewünscht seinen Frame senden und der Taster bestätigt jedes Byte gemäss dem Software Handshake. Das heisst, der Taster verwirft seine Sendeanforderung.

3.4 Protokoll, ohne Handshake

Der Kommunikations-Modus Protokoll ohne Handshake besteht aus einem fortgeschrittenen Protokoll mit einstellbaren Kommunikations-Parametern.
Durch Verwendung dieses Protokolls können alle Funktionen des Tasters voll verwendet werden.

Die Verwendung des Protokolls ohne Handshake bringt neben dem Fehlen der Flusskontrolle folgende Vorteile mit sich:

- Möglichkeit zu einem vollständig Ereignis-orientierten Softwareaufbau.
- Einfacher zu implementieren auf Systemen mit hoher Kommunikationsgeschwindigkeit, kleiner Scheduler-Durchlaufzeit und vorhandener Interrupt-Fähigkeit.

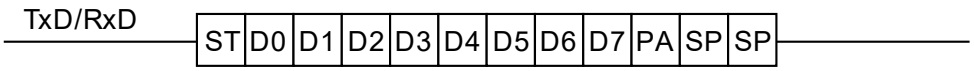


Zwischen jedem Frame besteht eine minimale Pause, bevor ein nächster Frame gsendet werden kann. Mehr Informationen dazu sind zu finden beim Service SetSystemSettings.request.

3.4.1 Asynchron

Beim Protokoll ohne Handshake wird nur die asynchrone Kommunikation unterstützt (keine SPI).

Jedes Byte wird nach folgendem Ablauf gesendet und empfangen:



Bezeichnung	Beschreibung
ST	Start-Bit
D0..D7	Daten-Bits
PA	Paritäts-Bit konfigurierbar über Service SetSystemSettings.request
SP	1..2 Stop-Bits konfigurierbar über Service SetSystemSettings.request

3.4.2 Fehlerbehandlung

Für die maximale Zeit zwischen 2 Bytes eines Frames besteht ein Timeout gemäss Service SetSystemSettings.request. Falls diese Zeit überschritten wird und der Frame noch nicht abgeschlossen ist, besteht ein Kommunikationsfehler.
Nach einem Protokoll-/ oder Kommunikationsfehler wird eine bestehende Verbindung abgebrochen und ein allfällig unvollständiger Frame gelöscht. Eine erneute Kommunikation ist frühestens nach Ablauf des Timeouts möglich.

4 Protokoll

4.1 Frame

Das Protokoll besteht grundsätzlich aus Paaren von Request-/ und Confirmation-Frames. Das heisst, auf einen Request-Frame folgt auch immer ein Confirmation-Frame, sofern die Kommunikation und die Verarbeitung erfolgreich war.
Ein Request-Frame wird immer vom Buskoppler gesendet. Ein Confirmation-Frame wird immer vom Taster gesendet.
Ferner bestehen auch automatische Indication-Frames, welche bei einem bestimmten Ereignis vom Taster gesendet werden. Die Indication-Frames werden nicht bestätigt.

Ein Frame der Länge n besteht aus Frame Header, Service und Daten:

Byte 0	Byte 1	Byte 2 .. (n-1)
Frame Header	Service	Daten

Bezeichnung	Beschreibung
Frame Header	Der Frame Header ist immer das erste Byte eines Services. Er dient zur Daten-/ und Übertragungssicherung des Frames.
Service	Der Service bestimmt die Art der nachfolgenden Daten und deren Verwendung.
Daten	Die Daten beinhalten die zum Service gehörenden Detailangaben.

4.1.1 Frame Header

Der Frame Header ist wie folgt codiert:

D7	D6	D5	D4	D3	D2	D1	D0
P	0	1	L	L	L	L	L

Bezeichnung	Beschreibung
P	Das Paritäts-Bit D7 wird so gewählt, dass über das ganze Byte gerade Parität entsteht.
01	Die Bits D6..D5 bilden einen konstanten Teil im Frame Header.
LLLLL	Die Bits D4..D0 definieren die Länge des Frames. Dabei wird die Anzahl Bytes ohne das Frame Header Byte selbst gezählt. Also die Anzahl Bytes, welche nach dem Frame Header folgen.

Spezielle Werte des Frame Headers:

0xA0: Der Taster führt einen Software Reset durch, wenn er 0xA0 als Frame Header empfängt.
Der Taster sendet 0xA0 als Frame Header, wenn er neu aufgestartet hat.

0xFF: Der Empfänger eines Frames schickt 0xFF als Frame Header, wenn Handshake aktiviert ist.

4.1.2 Service

Die Services werden nachfolgend detailiert beschrieben. Hier eine Übersicht:

Wert	Beschreibung
0x10	Service SetSystemSettings.request
0x11	Service SetSystemSettings.confirm
0x12	Service GetSystemSettings.request
0x13	Service GetSystemSettings.confirm
0x18	Service GetSystemState.request
0x19	Service GetSystemState.confirm
0x1A	Service SystemState.indication
0x1C	Service GetSystemInfo.request
0x1D	Service GetSystemInfo.confirm
0x30	Service SetLedState.request
0x31	Service SetLedState.confirm
0x32	Service GetLedState.request
0x33	Service GetLedState.confirm
0x38	Service SetLedBrightness.request
0x39	Service SetLedBrightness.confirm
0x3A	Service GetLedBrightness.request
0x3B	Service GetLedBrightness.confirm
0x40	Service GetButtonState.request
0x41	Service GetButtonState.confirm
0x42	Service ButtonState.indication

4.2 Service SetSystemSettings.request

Dieser Service fordert den Taster auf, die System Einstellungen zu ändern.
Nach einer Änderung der System Einstellungen muss innerhalb eines Timeouts von 10s eine erfolgreiche Kommunikation mit den neuen System Einstellungen stattfinden. Ist dies nicht der Fall, werden die alten System Einstellungen wiederhergestellt.
Wenn dieser Service nie benutzt wird oder Werte ausserhalb des definierten Bereichs übermittelt werden, wird automatisch der Default-Wert verwendet.

Der Frame ist wie folgt codiert:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
Frame Header	Service	Daten[0]	Daten[1]	Daten[2]
0xA6	0x10	Kommunikation	Baud Rate	Timeout
		Byte 5	Byte 6	
		Daten[3]	Daten[4]	
		Tasten Ind.	Sys. Status Ind.	

Über das Kommunikations-Byte werden die grundsätzlichen Einstellungen der Kommunikation vorgenommen. Es ist zu beachten, dass nicht alle möglichen Kombinationen unterstützt werden. Mehr dazu unter Kommunikations-Modus.
Das Kommunikations-Byte ist wie folgt codiert:

D7	D6	D5	D4	D3	D2	D1	D0
C1	C0	P1	P0	S	H	M1	M0

Bezeichnung	Beschreibung
C1	Clock Phase (CKPH) 1 = 1 Halbzyklus verzögert 0 = normal (nicht verzögert), Default (nur wirksam, wenn Kommunikations Art = synchron)
C0	Clock Polarität (CKPL) 1 = Idle HIGH 0 = Idle LOW, Default (nur wirksam, wenn Kommunikations Art = synchron)
P1	Parität 1 = aktiviert 0 = deaktiviert, Default (nur wirksam, wenn Kommunikations Art = asynchron)
P0	Parität Art 1 = gerade 0 = ungerade, Default (nur wirksam, wenn Kommunikations Art = asynchron)
S	Anzahl Stop-Bits 1 = 2 Stop-Bits 0 = 1 Stop-Bit, Default (nur wirksam, wenn Kommunikations Art = asynchron)
H	Handshake 1 = aktiviert 0 = deaktiviert, Default (nur wirksam, wenn Protokoll Art = Protokoll)
M1	Protokoll Art 1 = Byteaustausch 0 = Protokoll, Default
M0	Kommunikations Art 1 = synchron (SPI) 0 = asynchron (UART), Default

Die Baud Rate ist für die Übertragungsgeschwindigkeit nur massgebend, wenn die Kommunikations Art = asynchron gewählt wurde.
Bei Kommunikations Art = synchron wird die Baud Rate nur zur Berechnung des Timeouts verwendet. Die effektive Übertragungsgeschwindigkeit kann jedoch von der Baud Rate abweichen.
Das Baud Rate-Byte ist wie folgt codiert:

Wert	Beschreibung
0x01	1200 Baud
0x02	2400 Baud
0x04	4800 Baud
0x08	**9600 Baud, Default**
0x10	19200 Baud
0x20	38400 Baud
0x30	57600 Baud
0x60	115200 Baud (max.)

Das Timeout definiert die maximale Zeit zwischen 2 Bytes eines Frames. Haben 2 Bytes eines Frames einen grösseren Abstand, entseht ein Protokollfehler.
Bei Handshake = deaktiviert, wird anhand dieses Timeouts auch das Frameende detektiert. Somit ist das Timeout auch die minimale Pausenzeit zwischen einem Request-/ und Confirmationframe.
Das Timeout-Byte ist wie folgt codiert:

Wert	Beschreibung
0..1	**automatisches Timeout, Default** Timeout = 10 * 10 * (1 / Baud Rate) [s]
2..255	Faktor für Timeout Timeout = Faktor * 10 * (1 / Baud Rate) [s] Das max. Timeout ist begrenzt auf 500 ms.

Das Tasten Indikations-Byte ist wie folgt codiert:

Wert	Beschreibung
0x00	**Tasten Indikation deaktiviert, Default** Der Zustand der Tasten wird über den Service GetButtonState.request abgefragt.
0x01	Tasten Indikation aktiviert Der Zustand der Tasten wird über den Service ButtonState.indication automatisch bei jeder Änderung gesendet.

Das System Status Indikations-Byte ist wie folgt codiert:

Wert	Beschreibung
0x00	**System Status Indikation deaktiviert, Default** Der System Status wird über den Service GetSystemState.request abgefragt.
0x01	System Status Indikation aktiviert Der System Status wird über den Service SystemState.indication automatisch bei jeder Änderung gesendet.

4.3 Service SetSystemSettings.confirm

Über diesen Service wird bestätigt, dass die System Einstellungen geändert wurden.
Dies ist die Antwort auf den Service SetSystemSettings.request.
Dieser Service wird noch mit den alten System Einstellungen gesendet. Ab der nächsten Kommunikation werden die neuen System Einstellungen verwendet.

Der Frame ist wie folgt codiert:

Byte 0	Byte 1
Frame Header	Service
0x21	0x11

4.4 Service GetSystemSettings.request

Dieser Service fordert den Taster auf, die System Einstellungen zu senden.

Der Frame ist wie folgt codiert:

Byte 0	Byte 1
Frame Header	Service
0x21	0x12

4.5 Service GetSystemSettings.confirm

Über diesen Service sendet der Taster die System Einstellungen.
Dies ist die Antwort auf den Service GetSystemSettings.request.

Der Frame ist wie folgt codiert:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
Frame Header	Service	Daten[0]	Daten[1]	Daten[2]
0xA6	0x13	Kommunikation	Baud Rate	Timeout
		Byte 5	Byte 6	
		Daten[3]	Daten[4]	
		Tasten Ind.	Sys. Status Ind.	

Über das Kommunikations-Byte werden die grundsätzlichen Einstellungen der Kommunikation vorgenommen. Es ist zu beachten, dass nicht alle möglichen Kombinationen unterstützt werden. Mehr dazu unter Kommunikations-Modus.
Das Kommunikations-Byte ist wie folgt codiert:

D7	D6	D5	D4	D3	D2	D1	D0
C1	C0	P1	P0	S	H	M1	M0

Bezeichnung	Beschreibung
C1	Clock Phase (CKPH) 1 = 1 Halbzyklus verzögert 0 = normal (nicht verzögert), Default (nur wirksam, wenn Kommunikations Art = synchron)
C0	Clock Polarität (CKPL) 1 = Idle HIGH 0 = Idle LOW, Default (nur wirksam, wenn Kommunikations Art = synchron)
P1	Parität 1 = aktiviert 0 = deaktiviert, Default (nur wirksam, wenn Kommunikations Art = asynchron)
P0	Parität Art 1 = gerade 0 = ungerade, Default (nur wirksam, wenn Kommunikations Art = asynchron)
S	Anzahl Stop-Bits 1 = 2 Stop-Bits 0 = 1 Stop-Bit, Default (nur wirksam, wenn Kommunikations Art = asynchron)
H	Handshake 1 = aktiviert 0 = deaktiviert, Default (nur wirksam, wenn Protokoll Art = Protokoll)
M1	Protokoll Art 1 = Byteaustausch 0 = Protokoll, Default
M0	Kommunikations Art 1 = synchron (SPI) 0 = asynchron (UART), Default

Die Baud Rate ist für die Übertragungsgeschwindigkeit nur massgebend, wenn die Kommunikations Art = asynchron gewählt wurde.
Bei Kommunikations Art = synchron wird die Baud Rate nur zur Berechnung des Timeouts verwendet. Die effektive Übertragungsgeschwindigkeit kann jedoch von der Baud Rate abweichen.
Das Baud Rate-Byte ist wie folgt codiert:

Wert	Beschreibung
0x01	1200 Baud
0x02	2400 Baud
0x04	4800 Baud
0x08	**9600 Baud, Default**
0x10	19200 Baud
0x20	38400 Baud
0x30	57600 Baud
0x60	115200 Baud (max.)

Das Timeout definiert die maximale Zeit zwischen 2 Bytes eines Frames. Haben 2 Bytes eines Frames einen grösseren Abstand, entseht ein Protokollfehler.
Bei Handshake = deaktiviert, wird anhand dieses Timeouts auch das Frameende detektiert. Somit ist das Timeout auch die minimale Pausenzeit zwischen einem Request-/ und Confirmationframe.
Das Timeout-Byte ist wie folgt codiert:

Wert	Beschreibung
0..1	**automatisches Timeout, Default** Timeout = 10 * 10 * (1 / Baud Rate) [s]
2..255	Faktor für Timeout Timeout = Faktor * 10 * (1 / Baud Rate) [s] Das max. Timeout ist begrenzt auf 500 ms.

Das Tasten Indikations-Byte ist wie folgt codiert:

Wert	Beschreibung
0x00	**Tasten Indikation deaktiviert, Default** Der Zustand der Tasten wird über den Service GetButtonState.request abgefragt.
0x01	Tasten Indikation aktiviert Der Zustand der Tasten wird über den Service ButtonState.indication automatisch bei jeder Änderung gesendet.

Das System Status Indikations-Byte ist wie folgt codiert:

Wert	Beschreibung
0x00	**System Status Indikation deaktiviert, Default** Der System Status wird über den Service GetSystemState.request abgefragt.
0x01	System Status Indikation aktiviert Der System Status wird über den Service SystemState.indication automatisch bei jeder Änderung gesendet.

4.6 Service GetSystemState.request

Dieser Service fordert den Taster auf, den System Status zu senden.

Der Frame ist wie folgt codiert:

Byte 0	Byte 1
Frame Header	Service
0x21	0x18

4.7 Service GetSystemState.confirm

Über diesen Service sendet der Taster den System Status.
Dies ist die Antwort auf den Service GetSystemState.request.

Der Frame ist wie folgt codiert:

Byte 0	Byte 1	Byte 2
Frame Header	Service	Daten[0]
0x22	0x19	Status

Im Status-Byte ist der Fehler enthalten, welcher sich zuletzt ereignet hat. Wird Status = kein Fehler gemeldet, so bedeutet dies, dass bis zum Zeitpunkt der Anzeige kein einziger Fehler aufgetreten ist.
Das Status-Byte ist wie folgt codiert:

Wert	Beschreibung
0x00	kein Fehler
0x10	ungültiger Service
0x11	ungültiger Sub-Service
0x12	empfangene Framelänge stimmt nicht mit Länge im Frame Header überein
0x13	Framelänge stimmt nicht mit Service überein
0x14	ungültiger Frame Header
0x17	ungültiger Software Handshake
0x18	gleichzeitige Sendeanforderung, Taster hat Sendeanforderung verworfen
0x20	Fehler beim Ändern der System Einstellungen
0x21	Handshake wurde nicht aktiviert, wird aber benötigt bei SPI
0x22	ungültige Baud Rate
0x23	Timeout wurde begrenzt auf das Maximum von 500ms
0x30, 0x32	Empfangsbuffer Overflow
0x31, 0x33	Sendebuffer Overflow
0x34	Fehler während Empfangen eines Bytes
0x35	Fehler während Senden eines Bytes
0x36	Timeout überschritten während Empfangen eines Bytes
0x37	Timeout überschritten während Senden eines Bytes
0x40	USART Frame Fehler
0x41	USART Parität Fehler
0x42	USART Overflow
0x43	USART Break

4.8 Service SystemState.indication

Über diesen Service sendet der Taster den System Status automatisch bei Auftreten eines Fehlers.
Dieser Service kann aktiviert werden über den Service SetSystemSettings.request.

Der Frame ist wie folgt codiert:

Byte 0	Byte 1	Byte 2
Frame Header	Service	Daten[0]
0x22	0x1A	Status

Im Status-Byte ist der Fehler enthalten, welcher sich zuletzt ereignet hat. Wird Status = kein Fehler gemeldet, so bedeutet dies, dass bis zum Zeitpunkt der Anzeige kein einziger Fehler aufgetreten ist.
Das Status-Byte ist wie folgt codiert:

Wert	Beschreibung
0x00	kein Fehler
0x10	ungültiger Service
0x11	ungültiger Sub-Service
0x12	empfangene Framelänge stimmt nicht mit Länge im Frame Header überein
0x13	Framelänge stimmt nicht mit Service überein
0x14	ungültiger Frame Header
0x17	ungültiger Software Handshake
0x18	gleichzeitige Sendeanforderung, Taster hat Sendeanforderung verworfen
0x20	Fehler beim Ändern der System Einstellungen
0x21	Handshake wurde nicht aktiviert, wird aber benötigt bei SPI
0x22	ungültige Baud Rate
0x23	Timeout wurde begrenzt auf das Maximum von 500ms
0x30, 0x32	Empfangsbuffer Overflow
0x31, 0x33	Sendebuffer Overflow
0x34	Fehler während Empfangen eines Bytes
0x35	Fehler während Senden eines Bytes
0x36	Timeout überschritten während Empfangen eines Bytes
0x37	Timeout überschritten während Senden eines Bytes
0x40	USART Frame Fehler
0x41	USART Parität Fehler
0x42	USART Overflow
0x43	USART Break

4.9 Service GetSystemInfo.request

Dieser Service fordert den Taster auf, die System Information zu senden.

Der Frame ist wie folgt codiert:

Byte 0	Byte 1
Frame Header	Service
0x21	0x1C

4.10 Service GetSystemInfo.confirm

Über diesen Service sendet der Taster die System Information.
Dies ist die Antwort auf den Service GetSystemInfo.request.

Der Frame ist wie folgt codiert:

Byte 0	Byte 1	Byte 2	Byte 3
Frame Header	Service	Daten[0]	Daten[1]
0xA3	0x1D	SW-Info	HW-Info

Unter SW-Info ist die SW-Version des Mikrokontrollers auf dem Taster ersichtlich.
Das SW-Info-Byte ist wie folgt codiert:

Wert	Beschreibung
z.B. 0x12	High Nibble: Haupt-Version Low Nibble: Unter-Version Das gezeigte Beispiel bedeutet: Version 1.2

Unter HW-Info ist die HW-Bestückungsvariante des Tasters ersichtlich.
Das HW-Info-Byte ist wie folgt codiert:

Wert	Beschreibung
0x00	4 Kurzhubtaster, 0 LEDs
0x01	8 Kurzhubtaster, 0 LEDs
0x02	4 Kurzhubtaster, 6 LEDs
0x03	8 Kurzhubtaster, 8 LEDs

4.11 Service SetLedState.request

Dieser Service fordert den Taster auf, den Zustand der LEDs zu ändern.

Der Taster ist mit 3-Farben-LEDs ausgestattet. Das heisst, jede LED kann eine der Farben rot, grün oder blau annehmen.
Weiter können die LEDs statisch (dauernd ein) oder blinkend (Tein = Taus, f = 1 Hz) betrieben werden.
Pro LED kann nur eine Farbe und Betriebsart gewählt werden. Werden mehrere Farben/Betriebsarten pro LED selektiert, dann gilt folgende Reihenfolge der Konfliktlösung:

- rot vor grün vor blau
- statisch vor blinkend

Mit der Frame-Variante 1 können die LEDs nur statisch angesteuert werden.
Der Frame ist wie folgt codiert:

Byte 0	Byte 1	Byte 2		
Frame Header	Service	Sub-Service, Daten[0]		
0xA5	0x30	0x11		
		Byte 3	Byte 4	Byte 5
		Daten[1]	Daten[2]	Daten[3]
		LEDs rot, statisch	LEDs grün, statisch	LEDs blau, statisch

Mit der Frame-Variante 2 können die LEDs statisch oder blinkend angesteuert werden.
Der Frame ist wie folgt codiert:

Byte 0	Byte 1	Byte 2		
Frame Header	Service	Sub-Service, Daten[0]		
0x28	0x30	0x12		
		Byte 3	Byte 4	Byte 5
		Daten[1]	Daten[2]	Daten[3]
		LEDs rot, statisch	LEDs grün, statisch	LEDs blau, statisch
		Byte 6	Byte 7	Byte 8
		Daten[4]	Daten[5]	Daten[6]
		LEDs rot, blinkend	LEDs grün, blinkend	LEDs blau, blinkend

Das LED-Byte ist wie folgt codiert:

D7	D6	D5	D4	D3	D2	D1	D0
L8	L7	L6	L5	L4	L3	L2	L1

Bezeichnung	Beschreibung
L8..L1	Zustände der LEDs L8..L1. 1 = LED eingeschaltet, blinkend 0 = LED ausgeschaltet

4.12 Service SetLedState.confirm

Über diesen Service wird bestätigt, dass der Zustand der LEDs geändert wurde.
Dies ist die Antwort auf den Service SetLedState.request.

Bei der Frame-Variante 1 wurden die LEDs nur statisch angesteuert.
Der Frame ist wie folgt codiert:

Byte 0	Byte 1	Byte 2
Frame Header	Service	Sub-Service, Daten[0]
0x22	0x31	0x11

Bei der Frame-Variante 2 wurden die LEDs statisch oder blinkend angesteuert.
Der Frame ist wie folgt codiert:

Byte 0	Byte 1	Byte 2
Frame Header	Service	Sub-Service, Daten[0]
0x22	0x31	0x12

4.13 Service GetLedState.request

Dieser Service fordert den Taster auf, den Zustand der LEDs zu senden.

Mit der Frame-Variante 1 werden nur die statisch angesteuerten LEDs angefordert.
Der Frame ist wie folgt codiert:

Byte 0	Byte 1	Byte 2
Frame Header	Service	Sub-Service, Daten[0]
0x22	0x32	0x11

Mit der Frame-Variante 2 werden die statisch oder blinkend angesteuerten LEDs angefordert.
Der Frame ist wie folgt codiert:

Byte 0	Byte 1	Byte 2
Frame Header	Service	Sub-Service, Daten[0]
0x22	0x32	0x12

4.14 Service GetLedState.confirm

Über diesen Service sendet der Taster den Zustand der LEDs.
Dies ist die Antwort auf den Service GetLedState.request.

Mit der Frame-Variante 1 werden nur die statisch angesteuerten LEDs gesendet. Ist eine blinkende LED gerade eingeschaltet, gilt sie nicht als statisch angesteuert und wird hier nicht angezeigt.
Der Frame ist wie folgt codiert:

Byte 0	Byte 1	Byte 2		
Frame Header	Service	Sub-Service, Daten[0]		
0xA5	0x33	0x11		
		Byte 3	Byte 4	Byte 5
		Daten[1]	Daten[2]	Daten[3]
		LEDs rot, statisch	LEDs grün, statisch	LEDs blau, statisch

Mit der Frame-Variante 2 werden die statisch oder blinkend angesteuerten LEDs gesendet.
Der Frame ist wie folgt codiert:

Byte 0	Byte 1	Byte 2		
Frame Header	Service	Sub-Service, Daten[0]		
0x28	0x33	0x12		
		Byte 3	Byte 4	Byte 5
		Daten[1]	Daten[2]	Daten[3]
		LEDs rot, statisch	LEDs grün, statisch	LEDs blau, statisch
		Byte 6	Byte 7	Byte 8
		Daten[4]	Daten[5]	Daten[6]
		LEDs rot, blinkend	LEDs grün, blinkend	LEDs blau, blinkend

Das LED-Byte ist wie folgt codiert:

D7	D6	D5	D4	D3	D2	D1	D0
L8	L7	L6	L5	L4	L3	L2	L1

Bezeichnung	Beschreibung
L8..L1	Zustände der LEDs L8..L1. 1 = LED eingeschaltet, blinkend 0 = LED ausgeschaltet

4.15 Service SetLedBrightness.request

Dieser Service fordert den Taster auf, die Helligkeit der LEDs zu ändern.

Der Frame ist wie folgt codiert:

Byte 0	Byte 1	Byte 2	Byte 3
Frame Header	Service	Sub-Service, Daten[0]	Daten[1]
0xA3	0x38	0x11	Helligkeit

Die Helligkeit der LEDs kann stufenlos eingestellt werden. So kann z.B. eine Nachtabsenkung realisiert werden.
Das Helligkeits-Byte ist wie folgt codiert:

Wert	Beschreibung
0	Die LEDs sind ausgeschaltet.
1..255	stufenloser Helligkeitswert der LEDs 1 = min. Helligkeit 255 = max. Helligkeit, Default

4.16 Service SetLedBrightness.confirm

Über diesen Service wird bestätigt, dass die Helligkeit der LEDs geändert wurde.
Dies ist die Antwort auf den Service SetLedBrightness.request.

Der Frame ist wie folgt codiert:

Byte 0	Byte 1	Byte 2
Frame Header	Service	Sub-Service, Daten[0]
0x22	0x39	0x11

4.17 Service GetLedBrightness.request

Dieser Service fordert den Taster auf, die Helligkeit der LEDs zu senden.

Der Frame ist wie folgt codiert:

Byte 0	Byte 1	Byte 2
Frame Header	Service	Sub-Service, Daten[0]
0x22	0x3A	0x11

4.18 Service GetLedBrightness.confirm

Über diesen Service sendet der Taster die Helligkeit der LEDs.
Dies ist die Antwort auf den Service GetLedBrightness.request.

Der Frame ist wie folgt codiert:

Byte 0	Byte 1	Byte 2	Byte 3
Frame Header	Service	Sub-Service, Daten[0]	Daten[1]
0xA3	0x3B	0x11	Helligkeit

Die Helligkeit der LEDs kann stufenlos eingestellt werden. So kann z.B. eine Nachtabsenkung realisiert werden.
Das Helligkeits-Byte ist wie folgt codiert:

Wert	Beschreibung
0	Die LEDs sind ausgeschaltet.
1..255	stufenloser Helligkeitswert der LEDs 1 = min. Helligkeit 255 = max. Helligkeit, Default

4.19 Service GetButtonState.request

Dieser Service fordert den Taster auf, den Zustand der Tasten zu senden.

Der Frame ist wie folgt codiert:

Byte 0	Byte 1
Frame Header	Service
0x21	0x40

4.20 Service GetButtonState.confirm

Über diesen Service sendet der Taster den Zustand der Tasten.
Dies ist die Antwort auf den Service GetButtonState.request.

Der Frame ist wie folgt codiert:

Byte 0	Byte 1	Byte 2
Frame Header	Service	Daten[0]
0x22	0x41	Tasten

Das Tasten-Byte ist wie folgt codiert:

D7	D6	D5	D4	D3	D2	D1	D0
T8	T7	T6	T5	T4	T3	T2	T1

Bezeichnung	Beschreibung
T8..T1	Zustände der Tasten T8..T1. Die Tasten sind bereits entprellt. 1 = Taste gedrückt 0 = Taste losgelassen

4.21 Service ButtonState.indication

Über diesen Service sendet der Taster den Zustand der Tasten automatisch bei jeder Änderung.
Dieser Service kann aktiviert werden über den Service SetSystemSettings.request.

Der Frame ist wie folgt codiert:

Byte 0	Byte 1	Byte 2
Frame Header	Service	Daten[0]
0x22	0x42	Tasten

Das Tasten-Byte ist wie folgt codiert:

D7	D6	D5	D4	D3	D2	D1	D0
T8	T7	T6	T5	T4	T3	T2	T1

Bezeichnung	Beschreibung
T8..T1	Zustände der Tasten T8..T1. Die Tasten sind bereits entprellt. 1 = Taste gedrückt 0 = Taste losgelassen

5 Anwendungen

5.1 Wechsel des Kommunikations-Modus

Die einfachste Art den Taster zu betreiben ist es, den Kommunikations-Modus und die Baud Rate mittels den Voreinstellungs-Widerständen einzustellen und diese während dem Betrieb nicht zu verändern.

Besteht keine Möglichkeit, die Voreinstellungs-Widerstände zu bestücken und sollen nicht die Default-Einstellungen verwendet werden, kann der Kommunikations-Modus auch per Software gändert werden. Der Ablauf dazu sieht wie folgt aus:

- Speisespannung anlegen
- Warten (ca. 250 ms) bis Taster 0xA0 sendet
- Service SetSystemSettings.request senden mit den Kommunikations-Einstellungen (Default Kommunikations-Einstellungen: 9600 Baud, asynchron)
- Warten bis Taster Service SetSystemSettings.confirm sendet (Default Kommunikations-Einstellungen: 9600 Baud, asynchron)
- Kommunikations-Einstellungen auf Busankoppler ändern
- Innerhalb von 10 s den Service GetSystemSettings.request senden (neue Kommunikations-Einstellungen)
- Warten bis Taster Service GetSystemSettings.confirm sendet (neue Kommunikations-Einstellungen)

5.2 Taster zurücksetzen

Neben dem Hardware Reset mittels neuem Anlegen der Speisespannung gibt es auch die Möglichkeit, den Taster mittels Software Reset zurückzusetzen. Der Ablauf dazu sieht wie folgt aus:

- Den Frame Header 0xA0 senden
- Warten (ca. 50 ms) bis Taster ebenfalls 0xA0 sendet (Reset Indikation)

6 Anhang

6.1 Tasten und LED Layout

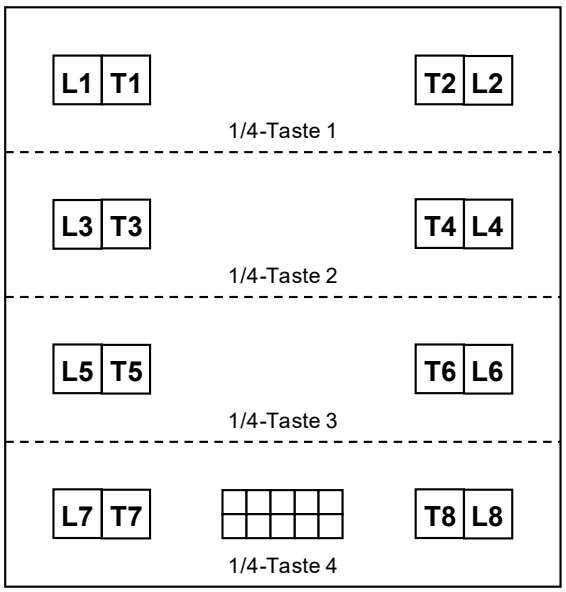
Nachfolgend ist die Zuordnung der LEDs (L1..8) und der Tasten (T1..8) auf dem Taster dargestellt. Die Zuordung variert je nach dem, welche Art von Kunststofftasten bestückt wird. Es existieren folgende Arten:

- 1/4-Taste
- 1/2-Taste
- 1/1-Taste

Es ist die Draufsicht gewählt. Die Stiftleiste der Schnittstelle zeigt vom Betrachter weg.

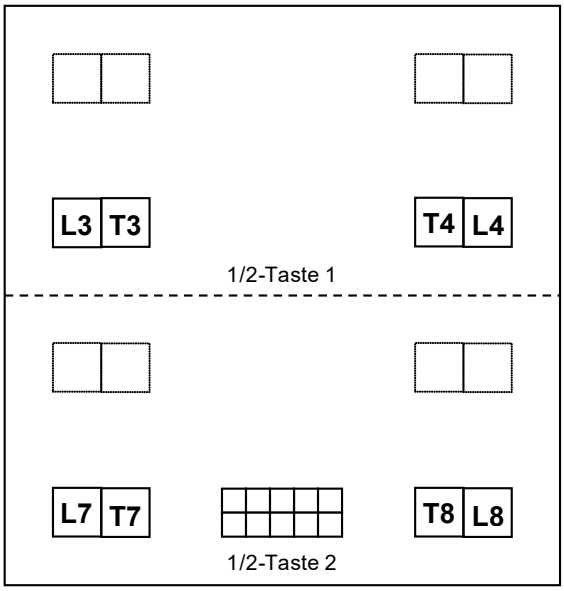
6.1.1 1/4-Taste

Bei Verwendung von 1/4-Tasten können folgende LEDs/Tasten angesteuert werden.



6.1.2 1/2-Taste

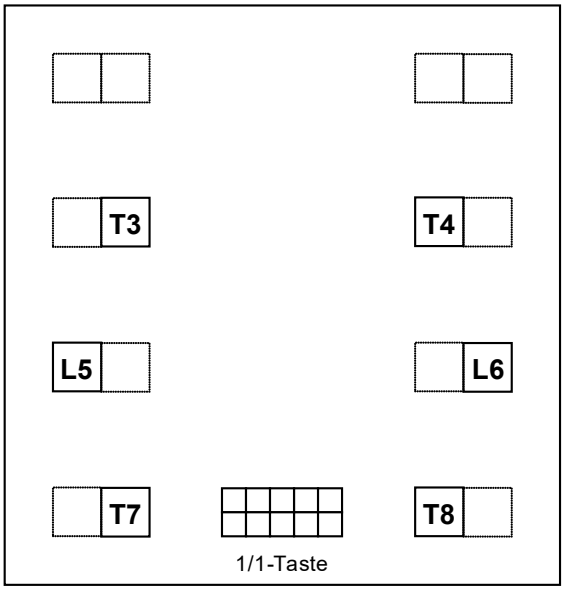
Bei Verwendung von 1/2-Tasten können folgende LEDs/Tasten angesteuert werden.



6.1.3 1/1-Taste

Bei Verwendung einer 1/1-Taste können folgende LEDs/Tasten angesteuert werden.

Hinweis: Aus mechanischen Gründen besitzt die 1/1-Taste je 2 Stössel links und rechts. Dies bedeutet, dass beim Betätigen der Taste links nur T3, nur T7 oder T3/T7 zusammen, beim Betätigen der Taste rechts nur T4, nur T8 oder T4/T8 zusammen angesteuert werden können.



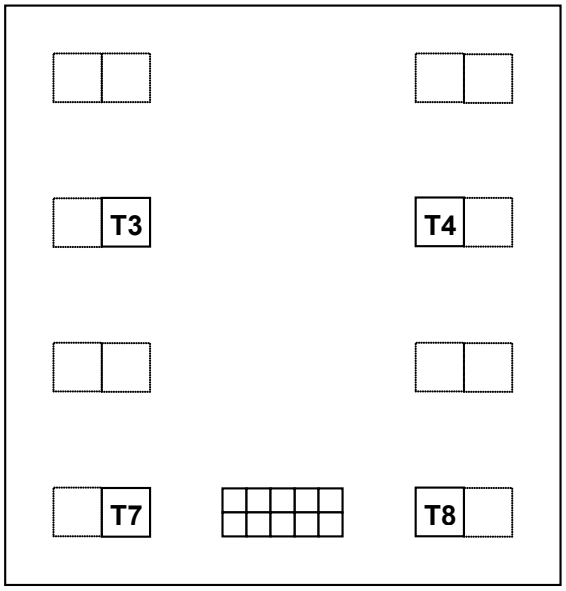
6.2 Bestückungsvarianten

Nachfolgend sind alle erhältlichen Bestückungsvarianten des Tasters aufgeführt. Dabei sind die bestückten LEDs (L1..8) und Tasten (T1..8) ersichtlich.

Es ist die Draufsicht gewählt. Die Stiftleiste der Schnittstelle zeigt vom Betrachter weg.

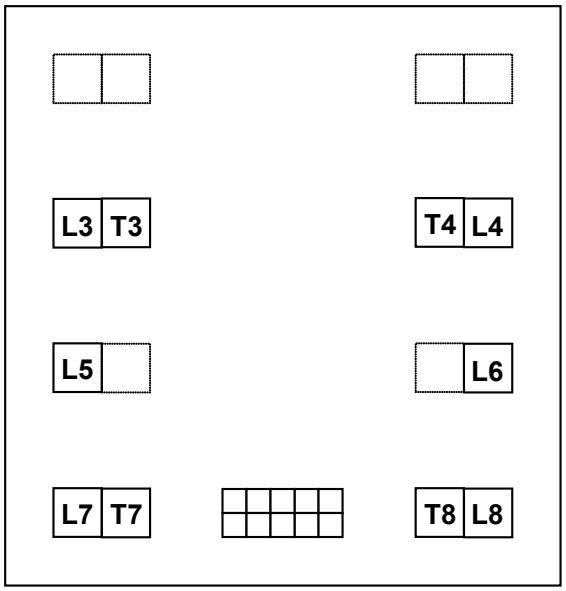
6.2.1 4 Kurzhubtaster, 0 LEDs

Der Artikel 900-3924.FMI.61 ist geeignet für 1/1 und 1/2 Tasten ohne LEDs.



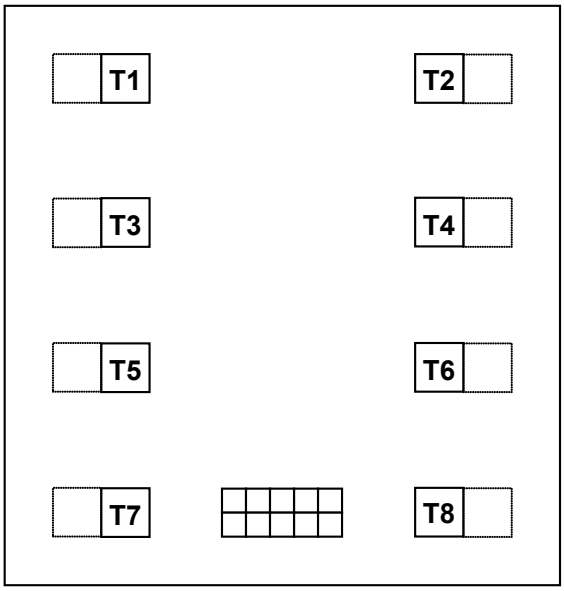
6.2.2 4 Kurzhubtaster, 6 LEDs

Der Artikel 900-3924.FMI.L.61 ist geeignet für 1/1 und 1/2 Tasten mit LEDs.



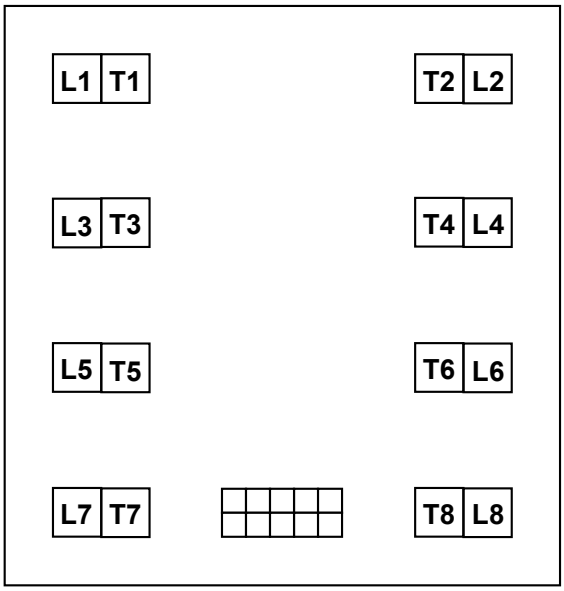
6.2.3 8 Kurzhubtaster, 0 LEDs

Der Artikel 900-3928.FMI.61 ist geeignet für 1/1, 1/2 und 1/4 Tasten ohne LEDs.



6.2.4 8 Kurzhubtaster, 8 LEDs

Der Artikel 900-3928.FMI.L.61 ist geeignet für 1/1, 1/2 und 1/4 Tasten mit LEDs.



6.3 Platzhalter für Voreinstellungs-Widerstände

Für die Voreinstellung des Kommunikations-Modus und der Baud Rate können Widerstände verwendet werden. In den meisten Fällen werden diese Widerstände direkt auf dem Busankoppler bestückt. Ist dies nicht möglich, gibt es noch eine Alternative.

Auf dem Taster hat es 2 Footprints für 0603-SMD-Widerstände (Platzhalter). Hier können die gwünschten Widerstände bestückt werden. Die Footprints sind wie folgt beschriftet:

- CT: Kommunikations-Modus (Com Type)
- BR: Baud Rate

6.4 Pull-Up/Down-Widerstände der Sendeleitungen

Die Sendeleitungen des Tasters (RxD, CTS) sind mit Pull-Up-Widerständen versehen. Das heisst, das die Pegel der Sendeleitungen bei einem Systemstart synchron mit der Speisespannung hochfahren.
Dies hat den Vorteil, dass der Busankoppler nicht versehentlich eine Daten-Startflanke (RxD) oder eine Sende-Anforerung (CTS) interpretiert, solange der Mikrokontroller des Tasters noch nicht hochgefahren ist.

Es wird empfohlen, dass die Sendeleitungen des Busankopplers (TxD, RTS) ebenfalls mit Pull-Up-Widerständen versehen werden.
Bei der Verwendung der Taktleitung (CLK) mit Clock Polarität Idle LOW (Default) soll ein Pull-Down-Widerstand verwendet werden.
Bei der Verwendung der Taktleitung (CLK) mit Clock Polarität Idle HIGH soll ein Pull-Up-Widerstand verwendet werden.

6.5 Legende

Nachfolgend werden einige Begriffe und Notationen in diesem Dokument erklärt:

Bezeichnung	Beschreibung
MSB	Most Significant Bit/Byte. Das höchstwertigste Bit/Byte. Beispiel einer typischen Bezeichnung: D7 (8bit Wert)
LSB	Least Significant Bit/Byte. Das niederwertigste Bit/Byte. Beispiel einer typischen Bezeichnung: D0 (8bit Wert)
Nibble	Ein Halbbyte (4 Bit)
0xA0	Hexadezimalzahlen beginnen immer mit „0x“ High Nibble / MSB: linksbündig Low Nibble / LSB: rechtsbündig
Frame	Ein komplettes Telegram des Protokolls.
Request	Anforderung Der Busankoppler sendet Request-Frames.
Confirmation	Bestätigung Der Taster sendet Confirmation-Frames.
Indication	Anzeige Der Taster sendet Indication-Frames.
RTS	Request to send, Anforderung für Handshake
CTS	Clear to send, Bestätigung des Handshake

Feller AG | Postfach | 8810 Horgen | 0844 72 73 74 | customercare.feller@feller.ch | www.feller.ch

Feller SA | Caudray 2| 1020 Renens | 0844 72 73 74 | customercare.feller@feller.ch | www.feller.ch

by Schneider Electric