AN0042 miCan-Stick Text-Kommandos

Thema	CAN Kommunikation über Text-Kommandos des miCan-Stick
Geräte	miCan-Stick mit Firmware ab V1.82.02.39 (Protokoll-Version 1.10)
Version	1.02
Datum	03.01.2008
Historie	-

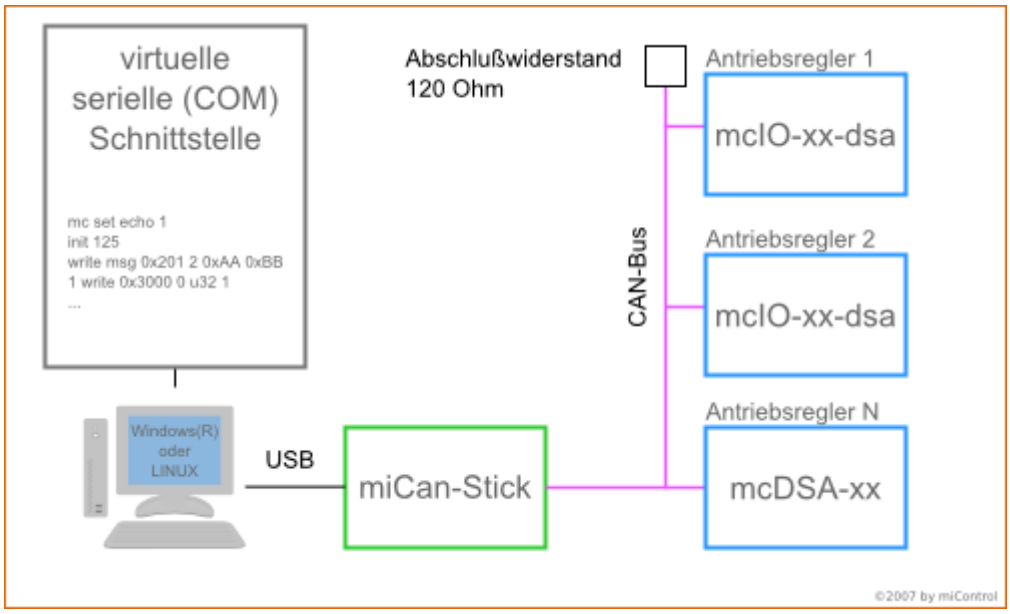
Beschreibung

Der miCan-Stick ist ein Interface zwischen einem USB (Universal Seriall Bus) und dem CAN-Bus (Controller Area Network).
Die Kommunikation mit dem miCan-Stick von der USB-Seite kann über folgende Protokolle erfolgen:

- Text-Kommandos - beschrieben in dieser ApplikationsNote
- mc5 - miControl binäres Protokoll (wird vom mPLC (miControl Entwicklungsumgebung) verwendet)

Bei der Installation von mPLC ab Version 2.62.00.00 wird ein Treiber für die virtuelle serialle Schnittstelle des miCan-Stick mitinstalliert. Über diese Schnittstelle kann eine Benutzerapplikation, mit Hilfe von den Text-Kommandos, den miCan-Stick ansprechen.

ACHTUNG: Wenn der miCan-Stick von dem mPLC verwendet wird, kann die virtuelle serielle Schnittstelle nicht geöffnet werden.



Die Text-Kommandos basieren auf der CiA-Spezifikation DS309-3. (CiA = CAN in Automation)

Jedes Text-Kommando besteht aus Schlüsselworten und Zahlen, die mit Leerzeichen getrennt sind. Jede Kommandoeingabe wird mit CRLF abgeschlossen. Alle Kommandos werden bestätig. Die Zahlen können entwerder in der dezimalen- oder hex-Notation (mit 0x - Präfix) angegeben werden. Um den Empfangspuffer von der USB-Seite zu löschen, kann ein Escape-Zeichen (=0x1B) an den miCan-Stick gesendet werden.
Wenn eine CAN-Nachricht empfangen wurde, wird eine Benachrichtigung über die serielle Schnittstelle unangefordet gesendet.

Syntax der Anforderung

```
"["sequence"]" [[net] node] command

wobei:

      sequence    - UNSIGNDE32 Nummer der Anforderung. Optional.
      net         - Netzwerknummer. Bei miCan-Stick wird dieser Nummer ignoriert.
      node        - Identifikationsnummer des Knotes nach CANopen (NodeId)
      command     - Kommando mit Parameter
```

```
Beispiele:

      [1] 1 24 write 0x3300 0 i32 1000    # Parameter Index=0x3300 Subindex=0
                                          # vom Gerät mit NodeId=24 wird
                                          # über SDO (Service Data Object)
                                          # mit dem Wert=1000 beschrieben.

      1 24 write 0x3300 0 i32 1000        # das Gleiche, aber ohne <sequence>
      24 write 0x3300 0 i32 1000          # das Gleiche, aber ohne <sequence>
                                          # und ohne <net>
```

Syntax der Antwort

```
"["sequence"]" response

wobei:

      sequence   - UNSIGNDE32 Nummer der Anforderung.
                   (Nur dann, wenn bei der Anforderung angegeben.)

      response   - Antworten:
                      OK
                      Error error_code
```

Syntax der Benachrichtigung

```
[net] MSG cob_id len_in_bytes value0 [..value7]

wobei:

      net        - Netzwerknummer.
                   Bei miCan-Stick wird dieser Nummer nicht ausgegeben.

Beispiele:

      MSG 0x00000181 3 0x11 0x22 0x33      # CAN-Nachricht mit ID=0x181
                                           #  und 3 DatenBytes=0x11 0x22 0x33
      MSG 0x00000182 1 0x56 0x78           # CAN-Nachricht mit ID=0x182
                                           #  und 2 DatenBytes=0x56 0x78
```

Kommandos

Der miCan-Stick unterstützt eine Untermenge der DS309-Kommandos und miControl-spezifische Kommandos:

DS309-3 Kommandos	Beschreibung

info version	Gibt Version-Informationen zurück
	
	Beispiel:
	`info version`
	`0x0139 0x15010001 1.82.02.39 0 128 1.10 0.0`
	
	wobei:
	- vendor_id = 0x0139
	- product_code = 0x15010001
	- firmware_version = 1.82.02.39
	- serial_number = 0
	- network_class = 128
	- protocol_version = 1.10
	- implementation_class = 0.0

[net] **init** baudrate	Setzt die CAN-Übertragungsgeschwindigkeit und aktiviert den CAN-Bus baudrate = 1000,800,500,250,125,100, 50,20 [kBit/s] oder -1 = CAN-Deaktivieren 0 = 1000 kBit/s 1 = 800 kBit/s 2 = 500 kBit/s 3 = 250 kBit/s 4 = 125 kBit/s 5 = 100 kBit/s 6 = 50 kBit/s 7 = 20 kBit/s
[net] **set network** value	Setzt ein Defaultwert für "net" *Beispiel:* `1 set network 2`
[net] **set node** value	Setzt ein Defaultwert für "node" *Beispiel:* `1 set node 0x21`
[[net] node] **start**	Setzt ein CANopen-Gerät in Operational-Zustand *Beispiel:* `5 start # NodeId=5`
[[net] node] **stop**	Setzt ein CANopen-Gerät in Stop-Zustand *Beispiel:* `5 stop # NodeId=5`
[[net] node] **preop**[**erational**]	Setzt ein CANopen-Gerät in PreOperational-Zustand *Beispiel:* `5 preop # NodeId=5`
[[net] node] **reset node**	Setzt ein CANopen-Gerät in Reset-Zustand *Beispiel:* `5 reset node # NodeId=5`
[[net] node] **reset comm**[**unication**]	Setzt ein CANopen-Gerät in ResetCommunication-Zustand *Beispiel:* `5 reset comm # NodeId=5`
[[net] node] **r**[**ead**] index subindex datatype	Liest ein Parameter über CANopen-Protokoll (SDO-Read) *Beispiel:* `5 r 0x3001 0 i32`
[[net] node] w[rite] index subindex datatype value	Beschreibt ein Parameter über CANopen-Protokoll (SDO-Write) *Beispiel:* `5 w 0x3300 0 i32 -1000`
[net] **set sdo_timeout** value	Setzt ein Timeout-Wert in Millisekunden für die read- und write-Kommandos (SDO-Transfer). Defaultwert ist auf 200 [ms] eingestellt. *Beispiel:* `set sdo_timeout 1000`

miControl Kommandos	Beschreibung
`[net] mc set echo value`	Echo: 1=aktivieren / 0=deaktivieren *Beispiel:* `mc set echo 1`
`[net] w[rite] m[sg] cob_id len_in_bytes value0[..value7]`	Sendet eine CAN-Nachricht mit ID=cob_id und Daten value0..value7 *Beispiel:* `w m 0x201 2 0x11 0x22`
`[net] r[ead] m[sg] cob_id [len_in_bytes]`	Sendet eine Remote-Frame mit ID=cob_id *Beispiel:* `r m 0x181`

Die Kommandos "write msg" und "read msg" sind unabhängig von den CANopen-Protokollen. Sie behandeln reine CAN-Nachrichten.
Wenn das Bit29 vom "cob_id" gesetzt ist, dann handelt es sich um ein 29-Bit langer Id-Nummer (extended Id) der CAN-Nachricht.

Beispiel:

```
w m 0x20000201 2 0x11 0x22
```

Datentypen

Der Datentyp (datatype) wird als ein String bei der Kommandos "read" und "write" angegeben.

```
datatype:
   i8    - Integer 8-Bit
   i16   - Integer 16-Bit
   i32   - Integer 32-Bit
   u8    - Unsigned Integer 8-Bit
   u16   - Unsigned Integer 16-Bit
   u32   - Unsigned Integer 32-Bit
```

Zusätzlich zu den Datentypen von der DS309-3 Spezifikation, können folgende Datentypen bei "read"-Kommando verwendet werden (miControl-Erweiterung):

```
   xi8    - Integer 8-Bit
   xi16   - Integer 16-Bit
   xi32   - Integer 32-Bit
   xu8    - Unsigned Integer 8-Bit
   xu16   - Unsigned Integer 16-Bit
   xu32   - Unsigned Integer 32-Bit
```

Dann erfolgt die Ausgabe des gelesenen Wertes in hex-Noatation.

Fehlerkodes

Wenn ein Befehl nicht interpretiert werden konnte oder nicht korrekt ausgeführt wurde, dann kommt folgende Antwort (siehe auch: "Syntax der Antwort"):

```
"["sequence"]" Error error_code
```

error_code	Beschreibung
100	Das Kommando wird nicht unterstützt
101	Fehler der Syntax - das Kommando konnte nicht interpretiert werden
-541	SdoWriteError - SDO-Objekt konnte nicht beschrieben werden.
-542	SdoReadError - SDO-Objekt konnte nicht gelesen werden.
-582	TxTimeout - Zeitüberschreitung beim Senden (z.B. Baudrate stimmt nicht überein, fehlendes Abschlusswiederstand oder defekte CAN-Bus-Leitung)
-583	ResponseTimeout - Zeitüberschreitung beim Warten auf eine Antwort (z.B. Gerät ist ausgeschaltet und Antwortet nicht)

Initialisierung (minimum)

<ESC>	Ein Byte = 0x1B an miCan-Stick senden (Eventuelle Zeichen im Empfangsbuffer löschen)
init 125	Baudrate setzen und CAN-Aktivieren

Beispiel

Informationsrichtung:

- `<` Anforderung an miCan-Stick
- `>` Antwort oder Benachrichtigung vom miCan-Stick

```
<     info version
>     0x0139 0x15010001 1.82.02.39 0 128 1.10 0.0
<     w m 0 2 2 0
>     OK
<     w m 0 2 1 0
>     OK
>     MSG 0x00000181 2 0x08 0x02
>     MSG 0x000001FF 2 0x08 0x02
>     MSG 0x00000281 0
>     MSG 0x000002FF 0
>     MSG 0x00000381 0
>     MSG 0x000003FF 0
>     MSG 0x00000481 0
>     MSG 0x000004FF 0
<     r m 0x181
>     OK
>     MSG 0x00000181 2 0x08 0x02
```