

### MiniMon V3 Installation

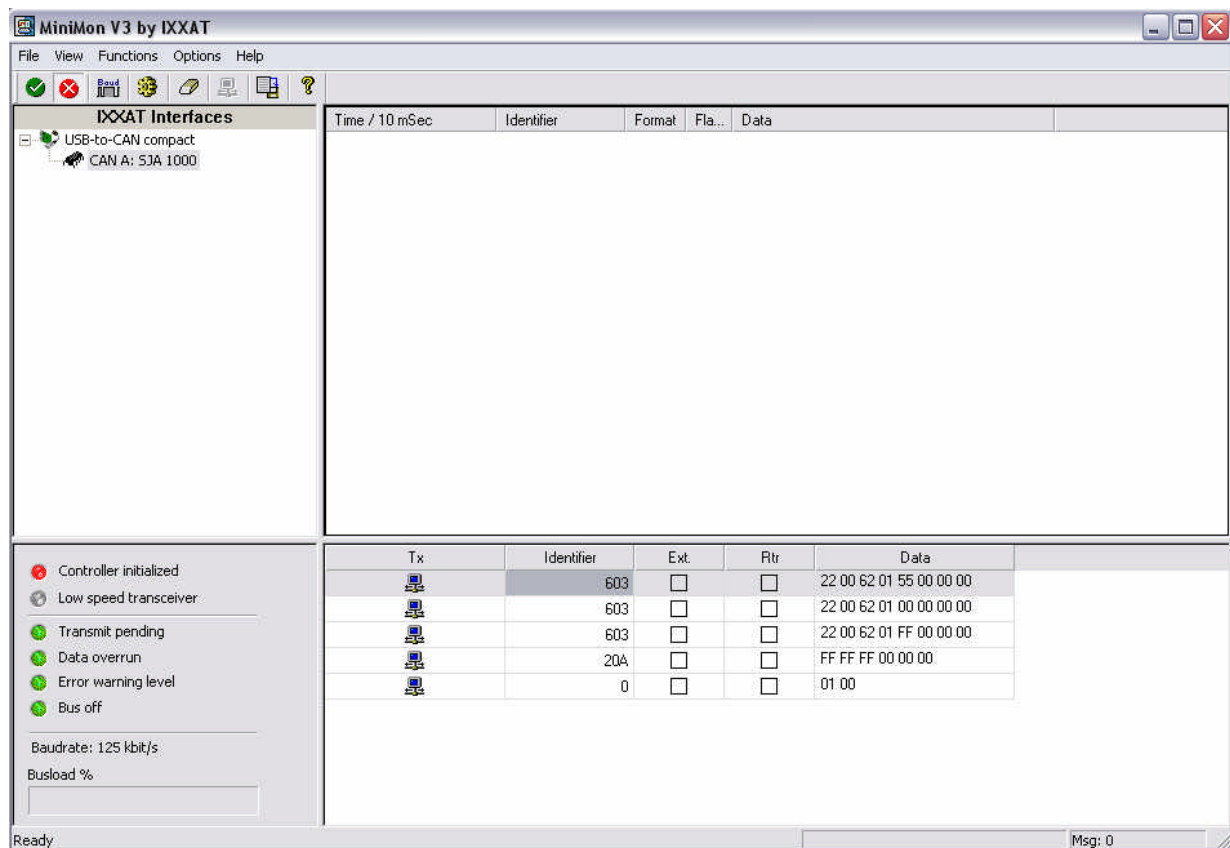
Die genaue Beschreibung zur Installation ist im Handbuch des IXXAT USB-to-CAN oder im Internet unter [http://www.ixxat.de/can\\_driver\\_for\\_windows\\_de.html](http://www.ixxat.de/can_driver_for_windows_de.html) zu finden.

### Das USB-to-CAN Tool verbinden

Ein bestehendes CAN-Verbindungskabel mit Sub-D9 Stecker kann einfach verwendet werden. Falls ein Adapterkabel angefertigt werden muss, ist die Pinbelegung wie folgt:

Pinnummer D9	Name	Funktion
2	CAN low	CAN low Signal bezogen auf CAN ground
3	CAN ground	CAN high Signal bezogen auf CAN ground
9	CAN high	CAN ground

### Minimon V3 Programm



Die Benutzeroberfläche gliedert sich grob in 4 Bereiche:

- linkes oberes Fenster: Auswahlfenster für das zu verwendende CAN-Interface
- rechtes oberes Fenster: Anzeige der Empfangenen Telegramme
- linkes unteres Fenster: Anzeige des Betriebszustandes
- rechtes unteres Fenster: Hier können bis zu 5 verschiedene Telegramme zum versenden vorbereitet werden. Durch klicken auf das blaue TX-Icon werden selbige dann versendet.

### MiniMon V3 Programm konfigurieren






Um die Nachrichten korrekt senden und empfangen zu können, müssen im MiniMon V3 Programm folgende Einstellungen getroffen werden:

**IXXAT Interface (linke Seite):** USB-to-CAN compact / CAN A: SJA1000 auswählen  
**Optionen -> Configuration: Baudrate:** entsprechend der auf dem CAN Bus verwendeten Baudrate

Unter dem Menüpunkt **Functions/Start** wird der gewählte Controller gestartet und ist somit bereit Telegramme zu empfangen und zu senden. Mit **Functions/Stop** wird der entsprechende Controller wieder gestoppt.

### Senden und Empfangen

Im rechten oberen Fenster werden sämtliche vom CAN Controller empfangene Telegramme mit einem Zeitstempel versehen und angezeigt. Auch die vom Controller selbst verschickten Telegramme werden hier angezeigt. Um Telegramme zu senden, muss im unteren rechten Fenster der entsprechende Identifier und die zugehörigen Daten eingetragen werden. Durch klicken auf das blaue Icon, wird das Telegramm verschickt. Es können bis zu 5 Telegramme in diesem Fenster vorbereitet und dann verschickt werden.

Tx	Identifier	Ext.	Rtr	Data
	603	<input type="checkbox"/>	<input type="checkbox"/>	22 00 62 01 55 00 00 00
	603	<input type="checkbox"/>	<input type="checkbox"/>	22 00 62 01 00 00 00 00
	603	<input type="checkbox"/>	<input type="checkbox"/>	22 00 62 01 FF 00 00 00
	20A	<input type="checkbox"/>	<input type="checkbox"/>	FF FF FF 00 00 00 00
	0	<input type="checkbox"/>	<input type="checkbox"/>	01 00

### Inbetriebnahme

Das Tutorial stützt sich hier auf die Verwendung eines CO4011B Eva Boards von frenzel + berg electronic. Alternativ können auch andere CANopen-Geräte verwendet werden. Da in den Beispielen Ein- und Ausgänge beschalten werden, muss sichergestellt sein, dass diese beim verwendeten Gerät auch vorhanden sind und gefahrlos geschaltet werden können. Auf herstellerspezifische Objekte wird hier nicht eingegangen.

Sobald das EVA-Board mit dem IXXAT verbunden wurde kann der MiniMon gestartet werden. Sobald das EVA-Board mit Spannung versorgt (oder resettet) wird, erscheint im Empfangsfenster die sog. Boot-up Message mit dem Identifier **703** und den Daten **00**. Mit dieser Nachricht meldet sich der Teilnehmer immer am Bus wenn er seinen Betrieb aufnimmt und ist dann im Pre-operational mode. Daraufhin kann der Teilnehmer mit einer Start node Nachricht in den Operational mode geschaltet werden. Dazu schickt man eine Nachricht mit Identifier 0 und den Daten **01XX** um ihn zu starten. Die **01** steht hier für das start node Kommando und **XX** für die Knotennummer. Alternativ kann anstatt der Kontennummer auch die **00** gesendet werden, was dann für alle Knoten gilt. Dieses Kommando bestätigt der Konten durch Übertragen aller seiner Sende PDOs. Der Operational mode ist nötig, um später auch PDOs versenden zu können. Im Pre-operational mode sind nur SDO und NMT Telegramme möglich.

Zu schickende Nachricht:

**0 01 03** Start Node Kommando für Konten 3

### CANopen Telegramme

(vgl. frenzel + berg electronic CANopen guideline Seite 12)

Identifizier 11-Bit (binär)	Identifizier (dezimal)	Identifizier (hexadezimal)	Funktion
00000000000	0	0	Netzwerkmanagement
00010000000	128	80h	Synchronisation
0001xxxxxxx	129 - 255	81h - FFh	Emergency
0011xxxxxxx	385 - 511	181h - 1FFh	PDO1 (tx)
0100xxxxxxx	513 - 639	201h - 27Fh	PDO1 (rx)
0101xxxxxxx	641 - 767	281h - 2FFh	PDO2 (tx)
0110xxxxxxx	769 - 895	301h - 37Fh	PDO2 (rx)
0111xxxxxxx	897 - 1023	381h - 3FFh	PDO3 (tx)
1000xxxxxxx	1025 - 1151	401h - 47Fh	PDO3 (rx)
1001xxxxxxx	1153 - 1279	481h - 4FFh	PDO4 (tx)
1010xxxxxxx	1281 - 1407	501h - 57Fh	PDO4 (rx)
1011xxxxxxx	1409 - 1535	581h - 5FFh	SDO senden
1100xxxxxxx	1537 - 1663	601h - 67Fh	SDO empfangen
1110xxxxxxx	1793 - 1919	701h - 77Fh	NMT Error Controll
xxxxxxx = Knotennummer 1 - 127			

**Im Programm Minimon V3 werden alle Werte in hexadezimaler Schreibweise eingegeben und angezeigt!**

Daraus lassen sich die entsprechenden Identifizier für die jeweiligen Telegramme wie folgt errechnen:

Telegramm	Identifizier
Emergency	80h + Knotennummer
Sende PDO1	180h + Knotennummer
Empfangs PDO1	200h + Knotennummer
...	...
SDO senden	580h + Knotennummer
SDO empfangen	600h + Knotennummer
NMT error controll	700h + Knotennummer

Die Empfangsrichtung der Telegramme wird immer aus Knotensicht gesehen. Soll dem Knoten beispielsweise per PDO1 ein Wechsel der Ausgangsbytes mitgeteilt werden, so wird ihm ein sog. Empfangs PDO (Identifizier 200h + Knotennummer) zugeschickt, da dieses Telegramm aus Kontensicht empfangen wird. Übermittelt der Konten beispielsweise einen Wert and den Master, so wird von diesem ein Sende PDO (180h + Knotennummer) empfangen.

### SDO Telegramm

Mit den SDO Telegrammen werden im CANopen Protokoll sogenannte Service Daten Objekte verschickt. Diese dienen dazu die Knoten zu parametrieren und die entsprechenden Einträge um Objektverzeichnis abzuändern oder auszulesen. (vgl. frenzel + berg electronic CANopen guideline Seite 13)

Wie aus dem CANopen Guide ersichtlich setzt sich ein SDO immer aus 8 Bytes zusammen:

- Byte 0: Domain Protocol
- Byte 1-3: Multiplexer
- Byte 4-7: Daten

### Aufbau eines Telegrammes

#### Identifizier:

Um dem Knoten ein SDO zu schicken ist ein sog. Empfangs SDO nötig. Aus der Tabelle ist ersichtlich, dass hierfür der Identifizier 600h + Knotennummer benötigt wird. Dieser wird im Sendefenster links unten im Feld Identifizier eingetragen. In den Beispielen wird hierfür die Knotennummer 3 verwendet. D.h. Identifizier = 603h

Die nun folgenden Werte werden im Feld Daten eingetragen

#### Byte 0 / Domain Protocol:

In diesem Byte wird festgelegt was mit dem SDO bezweckt werden soll. In diesem Tutorial wird nur kurz auf das lesen und schreiben eingegangen. Will man mit dem SDO etwas in ein Objekt schreiben, so muss dieses Byte den Wert 22h haben. Wenn ein Objekt ausgelesen werden soll, muss der Wert 40h sein.

Wert Byte 0 / Domain Protocol	Funktion
22h	schreiben
40h	lesen

#### Beispiel:

**603 22 XXXX XX DD DD DD DD**

*Empfangs SDO auf Knoten 3 / schreiben auf Objekt XXXX mit den Daten DD DD ....*

#### Byte 1 – 3 / Multiplexer

Der Multiplexer adressiert das Objekt. Beim Zusammenbauen des SDO ist darauf zu achten, dass bei Werten größer 8 Bit, also beispielsweise dem 16 Bit Index für das entsprechende Objekt im Objektverzeichnis die Reihenfolge der Bytes umgedreht wird. Konkret heißt das, wenn man auf das Objekt 6200h zugreifen will, dass man die Reihenfolge der Bytes tauscht und zuerst das low Byte und dann das high Byte in der Eingabezeile einträgt (siehe Beispiel). Byte 1 und 2 stellen das entsprechende Objekt in das man schreiben oder von dem man lesen will dar. Byte 3 stellt den Sub-Index des entsprechenden Objektes dar.

#### Beispiel:

**603 22 0062 01 DD DD DD DD**

*Empfangs SDO auf Knoten 3 / schreiben / auf Objekt 6200 / Subindex 01*

**Achtung: Wie oben beschrieben, muss die Reihenfolge Highbyte und Lowbyte getauscht werden. D.h. 6200h wird zu 0062h!**

### Byte 4 – 7 / Daten

In diesem Beispiel soll das Objekt 6200 „Write Output Byte“ geändert werden. In diesem Objekt stehen die Werte der zur Verfügung stehenden Ausgangsbytes. Der Wert dieses Objektes wird so an den Ausgängen ausgegeben. Im konkreten Fall soll für das Ausgangsbyte 1 des im Tutorial verwendeten CO4011B-EVA das Bitmuster 01010101 = 55h ausgegeben werden. Der Datentyp dieses Subobjektes ist Unsigned8, d.h. es kann der Wert 55h direkt und ohne die Reihenfolge zu ändern eingegeben werden. Die restlichen Bytes 5,6 und 7 müssen mit 0 aufgefüllt werden.

#### Beispiel:

**603 22 0062 01 55 00 00 00**

*Empfangs SDO auf Konten 3 / schreiben / auf Objekt 6200 / Subindex 01 / den Wert 55h*

Wenn das Telegramm vollständig eingegeben wurde, kann durch klicken auf das blaue Tx-Icon das Telegramm nun verschickt werden.

### Antworttelegramm

Da ein SDO Transfer immer beantwortet werden muss, sendet der Knoten immer ein Sende SDO (580h + Knotennummer) mit einer Antwort zurück. Diese Antwort enthält entweder einen Abbruch oder eine Bestätigung im Byte 0 (Domain Protocol) des SDOs und hat ansonsten bis auf die Daten den Selben Inhalt. 60h bedeutet in diesem Fall, dass die Änderung erfolgreich war. Die 42h bedeutet, dass das Lesen bzw. das Auslesen eines Objektes erfolgreich war und liefert im Datenanhang die entsprechenden Daten mit. Die 80h ist ein abort code, d.h. der Transfer ist fehlgeschlagen und es wird ein Code mitgesendet anhand dessen man die Ursache feststellen kann. (Siehe DS301 Kapitel 7.2.4.3.17 Protocol SDO abort transfer)

#### Für das Beispiel heißt das:

Bei Erfolg kommt folgende Antwort:

**583 22 0062 01 00 00 00 00**

*Sende SDO von Knoten 3 / schreiben erfolgreich / auf Objekt 6200 / Subindex 01*

Schlägt der Transfer fehl, hier beispielsweise wegen einem nicht vorhandenen Subindex 04:



gesendet: 603 22 0062 **04** 55 00 00 00 *Empfangs SDO/schreiben/ auf 6200 / Subindex 04 / Wert 55h*

empfangen: 583 80 0062 **04** 11 00 09 06 *Sende SDO/abort/Objekt 6200/Subindex 04 / Fehlercode 06090011*

**Achtung:** Der Fehlercode ist ein **UNSIGNED32-Wert, also 4 Bytes!!** Das heißt wie oben schon beschrieben, dass die Reihenfolge der nun 4 Bytes **gedreht werden muss!!** Also aus 11 00 09 06 wird nun 06 09 00 11 und dieser Fehlercode bedeutet laut DS301 „Sub-Index does not exist“.

Time / 10 mSec	Identifizier	Format	Flags	Data
00:18:33.75		603 Std	Self	22 00 62 01 55 00 00 00
00:18:33.76		583 Std		60 00 62 01 00 00 00 00
00:18:37.56		603 Std	Self	22 00 62 04 55 00 00 00
00:18:37.56		583 Std		80 00 62 04 11 00 09 06

Tx	Identifizier	Ext.	Rtr	Data
	603	<input type="checkbox"/>	<input type="checkbox"/>	22 00 62 01 55 00 00 00
	603	<input type="checkbox"/>	<input type="checkbox"/>	22 00 62 04 55 00 00 00

*Ein erfolgreicher SDO Transfer mit Bestätigung (22 ...) und ein Transfer mit falschem Sub-Index und Abbruch-Code (80 ...).*

Mit PDO Telegrammen werden wichtige Prozessdaten verschickt. Bei dem im Beispiel verwendeten CO4011B-Controller sind im Empfangs PDO1 die Werte für die digitalen Ausgänge, also das Objekt 6200h Subindex 1 und 2 gemapped. Dies bedeutet, dass die vom Knoten empfangenen Werte des Empfangs PDOs direkt in die entsprechenden Objekte im Verzeichnis geschrieben werden und somit an den Ausgangsbytes ausgegeben werden.

Auszug aus CO4011B Datenblatt:

Index	Entry	Explanation
1A00.00	3	There are 3 mapped objects in transmit PDO1
1A00.01	60000108h	First mapped object: Input Byte0
1A00.02	60000208h	First mapped object: Input Byte1
1A00.03	60000308h	First mapped object: Input Byte2

Die Tabelle zeigt die Mapping-Einträge im Objekt 1A00 für das erste Sende PDO. Hier wird festgelegt, welche Objekteinträge im Transmit PDO1 übertragen werden. Das erste Byte ist das Input Byte 0, das zweite Byte das Input Byte 1 und das dritte Byte das Input Byte 2. Ändert sich an den digitalen Eingängen des CO4011B etwas, so wird das Senden des PDOs veranlasst. Dies kann man beim EVA-Board durch einfaches Stecken des Jumper ausprobieren. Die daraufhin gesendeten PDOs werden dann im rechten oberen Fenster des MiniMon angezeigt.

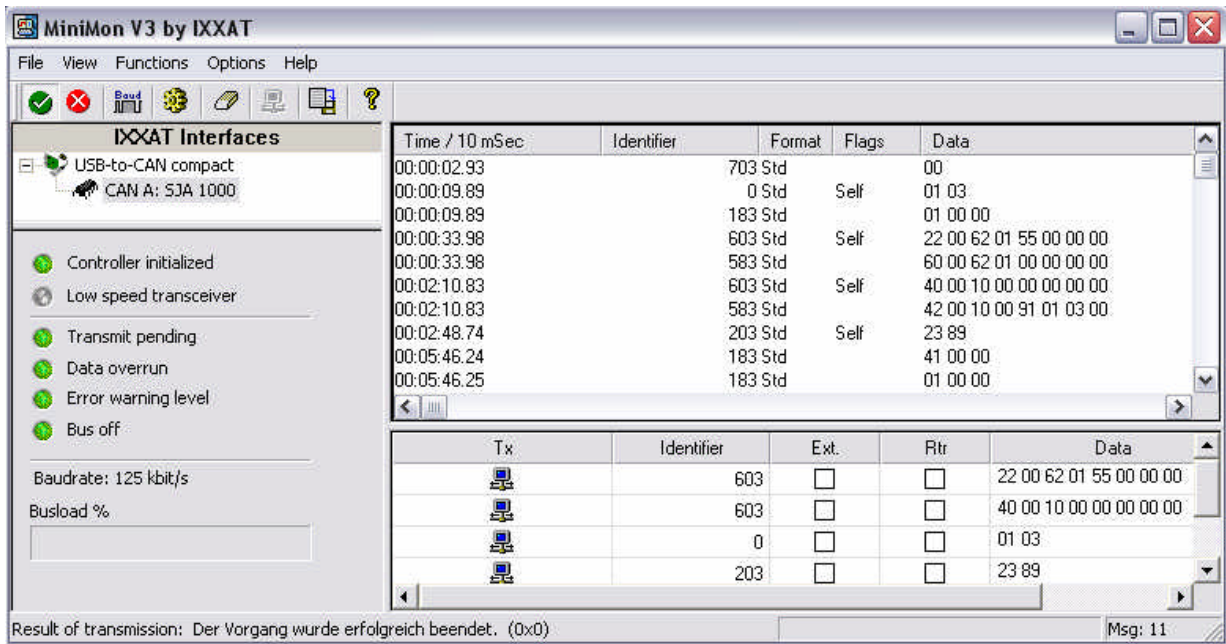
### Beispiel:

Es soll der Wert 23h an das Ausgangsbyte 0 und der Wert 89h an das Ausgangsbyte 1 geschickt werden.

**203 23 89**      *Receive PDO1 / Byte0 = 23h / Byte1 = 89h*

Je nach Mapping können nun so Daten empfangen oder versendet werden.

### Komplettes Beispiel



Telegramme von oben nach unten:

- 703 00** Boot up Message vom Knoten 3
- 0 01 03** Start Node vom Master an den Knoten 3
- 183 01 00 00** Transmit PDO1 als Antwort vom Knoten auf die Boot up Message
- 603 22 00 62 01 55 00 00 00** Empfangs SDO mit schreiben auf Objekt 6200 Subindex 1 den Wert 55h
- 583 60 00 62 01 00 00 00 00** Sende SDO mit Bestätigung, dass Schreiben auf Objekt 6200 erfolgreich war
- 603 40 00 10 00 00 00 00 00** Empfangs SDO Leseanforderung Objekt 1000
- 583 42 00 10 00 91 01 03 00** Sende SDO mit Bestätigung und Inhalt Objekt 1000: 0003 0191h (Unsigned 32)
- 203 23 89** Empfangs PDO1: der Wert 23h wird laut Mapping ins Objekt 6200 Sub. 01 geschrieben, der Wert 89h ins Objekt 6200 Sub. 02
- 183 41 00 00** Sende PDO: Laut Mapping hat das Objekt 6000 Sub. 01 den Wert 41h (Digitales Eingangsbyte 0)
- 183 01 00 00** Sende PDO: Laut Mapping hat das Objekt 6000 Sub. 01 den Wert 01h (Digitales Eingangsbyte 0)

Bei Fragen und Unklarheiten hilft das frenzel + berg electronic Support Team gerne weiter.

**Telefon: 0731 / 97057 – 42**  
**support@frenzel-berg.de**