Software Dokumentation
ELA-IO Basic V3.5.12.1

elrest Automationssysteme GmbH
Leibnizstraße 10
73230 Kirchheim unter Teck
Germany
Telefon: +49 (0) 7021 / 92025-0
www.elrest.de

Inhaltsverzeichnis

Impressum

elrest Automationssysteme GmbH

Leibnizstraße 10
73230 Kirchheim unter Teck
Germany

Tel.: + 49 (0) 7021 / 92025-0
Fax: + 49 (0) 7021 / 92025-29

e-mail: vertrieb@elrest.de
Web: http://www.elrest.de

Technischer Support

Tel.: +49 (0) 7021 / 92025-33
Fax. +49 (0) 7021 / 92025-29
e-mail: support@elrest.de

Dieses Dokument wurde sorgfältig erstellt, um die Richtigkeit und Vollständigkeit der Dokumentation zu gewährleisten.
Da sich jedoch Fehler nie ausnahmslos vermeiden lassen, sind wir für ihre Anregungen und Mithilfe immer dankbar.

1 Allgemein

1.1 Vorwort

Dieses Handbuch enthält Texte, Abbildungen und Erläuterungen zur korrekten Installation und Bedienung. Vor der Installation und dem Einsatz der Geräte muss dieses Handbuch gelesen und beachtet werden.

Es wendet sich ausschließlich an ausgebildete Fachkräfte der Steuerungs- und Automationstechnik. Diese müssen mit den aktuellen Normen und Richtlinien vertraut sein.

Bei Fragen zur Installation, Anwendung und Bedienung wenden Sie sich bitte an die elrest-Kunden-Hotline:

Tel.:07021/92025-33

Fax:07021/92025-59

E-Mail: hotline@elrest.de

oder an Ihre zuständige Vertretung.

Dieses Handbuch wird vorbehaltlich etwaiger Änderungen herausgegeben. Änderungen können ohne Hinweis vorgenommen werden.

1.2 Haftungsbedingungen

Die Dokumentation wurde sorgfältig erstellt.

Alle Beispiele und Abbildungen in diesem Handbuch dienen nur als Hilfe zum Verstehen des Textes. Es können Änderungen ohne Hinweise vorgenommen werden. Für die Richtigkeit der dargestellten Bedienvorgänge kann keine Gewährleistung übernommen werden. An Hand von den Texten, Erläuterungen und Abbildungen in diesem Handbuch können keine Ansprüche auf schon gelieferte Produkte gemacht werden. elrest Automationssysteme GmbH übernimmt keine Verantwortung für eine Produktanwendung, die sich auf die dargestellten Beispiele (z.B. in eStudio Demo) bezieht.

elrest Automationssysteme GmbH übernimmt unter keinen Umständen die Haftung oder Verantwortung für Schäden, die aus einer unsachgemäßen Installation bzw. Anwendung der Geräte oder des Zubehörs entstanden sind.

Das Fachpersonal hat sicherzustellen, dass die Montage und die Anwendung der Produkte alle Sicherheitsanforderungen, Gesetzen, Bestimmungen und Normen entsprechen

Die nationalen Vorschriften und jeweils gültigen Sicherheitsbestimmungen sind zu beachten.

Eingriffe und Veränderungen an den Geräten führen zum Erlöschen des Garantieanspruches.

1.3 Sicherheitsrichtlinien und Schutzmaßnahmen

Dieses Handbuch wurde für geschultes und kompetentes Personal erstellt. Die Qualifizierung wird durch die europäischen Richtlinien für Maschinen, Niederspannungen und EMV definiert. Bei Spannungen, die grösser als die Schutzkleinspannung sind, muss die Montage der Geräte durch eine Elektrofachkraft erfolgen.

Die nationalen Vorschriften und jeweils gültigen Sicherheitsbestimmungen sind zu beachten. Eingriffe und Veränderungen an den Geräten führen zum Erlöschen des Garantieanspruches.

Aufgrund der großen Anzahl von verschiedenen Anwendungsmöglichkeiten dieser Geräte müssen Sie die Anpassung für Ihren speziellen Anwendungsfall selbst vornehmen.

Wenn Schaltungskomponenten ausfallen sollten, müssen entsprechende Sicherheitseinrichtungen dafür sorgen, dass die angeschlossene Peripherie angehalten wird.

Versuchen Sie nicht, die Geräte selbst zu reparieren oder elektrische Teile auszutauschen. Wenden Sie sich hierfür ausschließlich an die elrest Service Abteilung. Kontakt können Sie über die elrest-Hotline aufnehmen.

Beachten Sie bei Installation und Einsatz der Geräte die lokalen und nationalen Normen und Vorschriften

Die einschlägigen Vorschriften (VDE etc.) beim Umgang mit elektrischen Anlagen sind zu beachten:

- Freischalten
- Gegen Wiedereinschalten sichern
- Spannungsfreiheit feststellen
- Erden und Kurzschließen
- Keine Erdschleifen
- Benachbarte unter Spannung stehende Teile sind abzudecken oder abzuschranken

1.4 Copyright

1.5 Gewährleistung

Ein Gewährleistungsanspruch setzt eine fachgerechte Montage und Inbetriebnahme nach der für das Gerät gültigen Montage-, Inbetriebnahme- und Bedienungsanleitung voraus. Die erforderlichen Montage-, Inbetriebnahme- und Wartungsarbeiten dürfen nur von sachkundigen und autorisierten Personen durchgeführt werden. Sehen Sie hierzu unsere EULA Bestimmungen.

Hersteller	**Handelsmarke**	**Ursprungsland**
elrest Automationssysteme GmbH Leibnizstraße 10 D-73230 Kirchheim unter Teck		Germany

Telefon:+49 (0) 7021/92025-0
Fax: +49 (0) 7021/92025-29

1.6 Symbole

In diesem Handbuch werden zur Hervorhebung von bestimmten Informationen verschiedene Symbole verwendet. Hiermit erhält das Bedienpersonal notwendige Hinweise zu den Sicherheits- und Schutzmaßnahmen. Bei jedem Auftreten der Symbole muss der zugehörige Hinweis gelesen werden

HINWEIS

Wichtiger Hinweis!
Kennzeichnet eine mögliche Fehlfunktion, die aber keinen Sachschaden zur Folge hat, wenn sie nicht vermieden wird.

INFORMATION

Weitere Information
Weist auf weitere Informationen hin, die kein wesentlicher Bestandteil dieser Dokumentation sind (z. B. Internet).

Bezeichnet eine möglicherweise auftretende Gefahr, die zu einem Personen- oder Sachschaden führen kann

Bezeichnet Hinweise, damit die Handhabung einfacher wird.

2 Kommunikation CANopen

Der Funktionsbaustein „Ela_IO_Fb“ (FB) arbeitet auf Basis des CANopen-Protokoll.
Er bildet einen CANopen-Master nach welcher einfach und dynamisch zu programmieren ist.

Vorteil:
- Die Buskonfiguration ist nicht statisch.
  Bei Änderungen muss das Programm nicht neu übersetzt werden.

Nachteil:
- Es werden nur die implementierten Slaves unterstützt.

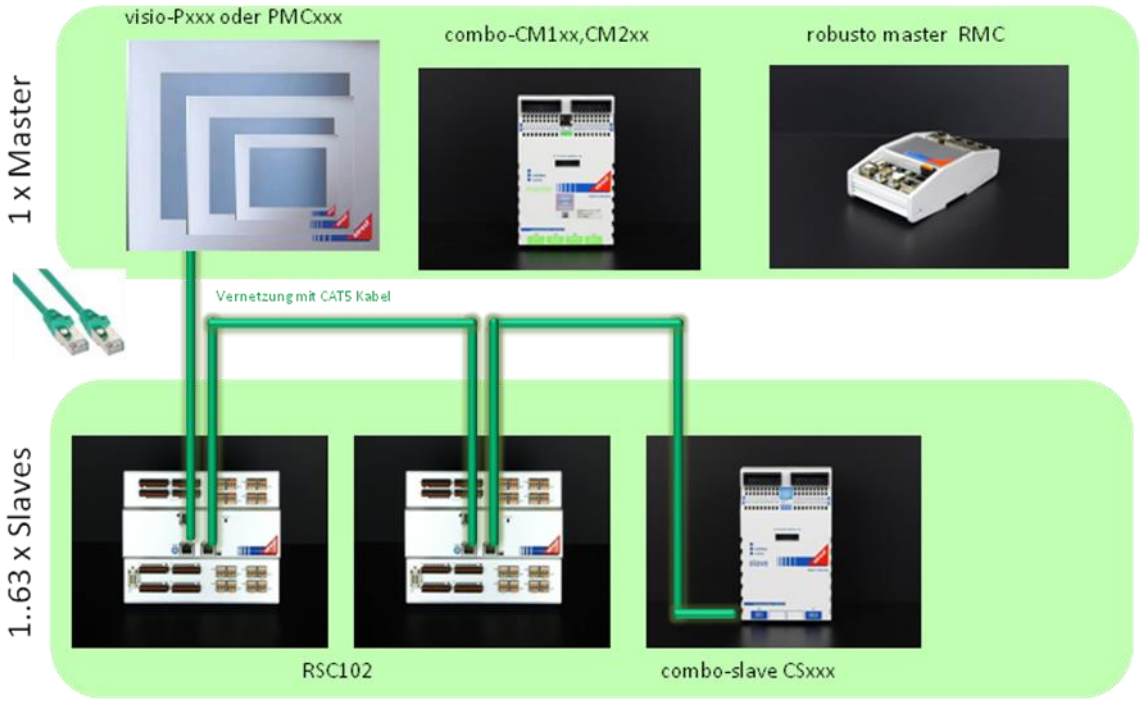
Derzeit sind dies:

Firma	Slave-Gerätereihe
elrest GmbH	Combo Slave 1xx
elrest GmbH	RCS 102 und RSC 123

CANopen ist ein Layer7 Protokoll für die Automatisierung.

Offenes Feldbusinterface zu beliebigen CAN Protokollen,
wie beispielsweise Truck-Norm J1939-based.

Verfügbarkeit
Die CAN Schnittstelle ist als CANopen Interface konzipiert. Verwendet werden können alle elrest PLCs mit CODESYS V3.



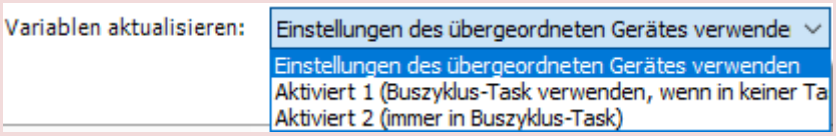
Ein CAN-Netzwerk kann aus insgesamt *maximal* 127 Teilnehmer bestehen. Ohne Repeater können 64 Teilnehmer verbunden werden.
Die CAN Physik erlaubt eine Leitungslänge von *maximal* 1000 m @ 50 kBaud

3 Funktionsbeschreibung

Der Funktionsbaustein „Ela_IO_Fb“ (FB) erlaubt den direkten Zugriff auf die Ein- und Ausgänge einer Mastersteuerung, sowie auf Slave-EA’s über eine CANopen-Kommunikation.

Alle EA’s werden hierbei über PLC-Code konfiguriert, geschrieben und gelesen.
Eine statische Projektierung innerhalb CODESYS entfällt.

Achtung:
Für das Lesen von Eingängen über die Funktion IO_Access() muss am Gerät unter E/A-Abbild die Variableneinstellung gesetzt werden auf:
Variable aktualisieren: → Einstellungen des übergeordneten Gerätes verwenden.



3.1 Merkmale allgemein:

- Betrieb von bis zu maximal 63 CANopen-Slaves vom Typ elrest RSC102, RSC123 und Combo Slave 1xx

3.2 Merkmale Pre-Operational:

- Sequenzieller Anlauf

3.3 Merkmale Operational:

3.3.1 Normalbetrieb

- Sofortiges setzten der Digitalausgänge bei Änderung
- Zusätzliches zyklisches Setzen der Digitalausgänge im Sekundentakt
- Sofortiges Setzen der Analogausgänge bei Änderung
- Zusätzliches zyklisches Setzen der Analogausgänge im Sekundentakt
- Lesen der Digitaleingänge bei Änderung
- Lesen der Analogeingänge bei Änderung
- Slaveüberwachung über Nodeguarding im 0,5 Sekundentakt
- Ändern des Analogkanaltyps während des Betriebs

3.3.2 Ausfall eines Slaves

- Erkennung des Slave-Verlustes und parametrierbarem Wiederanlauf bei erneuter Antwort auf Nodeguarding-Request.

4 Steuerfunktionen

4.1 Digitale Funktionen

4.1.1 Din() - Digitale Eingänge lesen

```
bDigitaleingang:= Din(Node, Channel);
```

Variable IN	**Typ: =Default**	**Bedeutung**
Node	BYTE	Nummer des abzufragenden Teilnehmers 0 = PLC Mastersteuerung 1..63 = Node ID des CanOpenSlave-Teilnehmers
Channel	UINT	Nummer des Digitalleingang 0..31
Variable OUT	**Typ: =Default**	**Bedeutung**
Rückgabewert	BOOL:=FALSE;	Zustand des angefragter Digitaleingang

Beispiel für das Setzen aller 32 Digitalausgänge auf Slavemodul „Node“ aus einem DWord.

Deklaration:

```
VAR
  Din : DWORD;      // Alle Digitaleingänge in einem DWORD
  i   : UINT;       // Nummer des Digitaleingang
END_VAR
```

Code:

```
FOR i:= 0 TO 31 BY 1 DO
  Din:=PUTBIT(Din,UINT_TO_BYTE(i),IO.Din(Node,i));
END_FOR
```

Im weiteren Verlauf kann auf den Inhalt des DWORD z.B. wie folgt zugegriffen werden.

```
SafetyDoorOpen  := Din.0;  // Digitaleingang 01
EmergencyStop   := Din.1;  // Digitaleingang 02
PressureOK      := Din.2;  // Digitaleingang 03
```

4.1.2 Dout() - Digitale Ausgänge setzen

Dout(Node, Channel, bValue);

Variable IN	**Typ: =Default**	**Bedeutung**
Node	BYTE	Nummer des anzusprechenden Teilnehmers 0 = PLC Mastersteuerung 1..63 = Node ID des CanOpenSlave-Teilnehmers
Channel	UINT	Nummer des Digitallausgang 0..31
bValue	BOOL:=FALSE;	Zustand des zu setzenden Digitalausgang
Variable OUT	**Typ: =Default**	**Bedeutung**
Rückgabewert	BOOL:=TRUE;	Ein Rückgabewert wird nicht verwendet, bzw. ist immer TRUE.

Beispiel für das Setzen aller 32 Digitalausgänge auf Slavemodul „Node“ aus einem DWord.

```
FOR i:= 0 TO 31 BY 1 DO
  IO.Dout(Node,i, EXTRACT(X:=Dout, N:=UINT_TO_BYTE(i));
END_FOR
```

4.1.3 DoutHandlingAfterStop() - Digitale Ausgänge setzen/rücksetzten für PLC = STOP

Beispiel:

IO.DoutHandlingAfterStop(Enable);

Hinweis:
Diese Option besteht aus 2 Anweisungen: DoutHandlingAfterStop() und DoutValueAfterStop()
Diese Option gilt derzeit nur für Mastersteuerungen.
Für Slave Module gilt das Default- Nodeguardingverhalten.

Variable IN	**Typ**	**Bedeutung**
Enable	BOOL	Aktivieren / Deaktivieren des Verhalten**:** Digitale Ausgänge setzen/rücksetzten für PLC = STOP

4.1.4 DoutValueAfterStop() - Digitale Ausgänge setzen/rücksetzten für PLC = STOP

Beispiel:

IO.DoutValueAfterStop(Node, Channel, xValue);

Hinweis:
Diese Option besteht aus 2 Anweisungen: DoutHandlingAfterStop() und DoutValueAfterStop()
Diese Option gilt derzeit nur für Mastersteuerungen.
Für Slave Module gilt das Default- Nodeguardingverhalten.

Variable IN	**Typ: =Default**	**Bedeutung**
Node	BYTE	Nummer des anzusprechenden Teilnehmers 0 = PLC Mastersteuerung 1..63 = Node ID des CanOpenSlave-Teilnehmers (in Vorbereitung. Aktuell wird der Nodeguarding- Default Wert der Slave Komponente verwendet)
Channel	UINT	Nummer des Digitallausgang 0..31
bValue	BOOL:=FALSE;	Sollzustand Digitalausgang (TRUE / FALSE) im Fehlerfall.

Variable OUT	Typ: =Default	Bedeutung
Rückgabewert	BOOL:=TRUE;	Ein Rückgabewert wird nicht verwendet,

4.2 Analoge Funktionen

4.2.1 AinConfigureType() - Analogeingangskonfiguration setzen

AInConfigureType(Node, Channel, eSensorType);

Variable IN	**Typ: =Default**	**Bedeutung**
Node		Nummer des abzufragenden Teilnehmers 0 = PLC Mastersteuerung 1..63 = Node ID des CanOpenSlave-Teilnehmers
Channel	UINT	Nummer des Analogeingang 1..16
eSensorType	INT	siehe nachstehende Enumeration
Variable OUT	**Typ: =Default**	**Bedeutung**
Rückgabewert	BOOL:=TRUE;	Ein Rückgabewert wird nicht verwendet, bzw. ist immer TRUE.

Enumeration für „eSensorType“ RSC102, RSC123 und CS1xx

Die mit „X“ gekennzeichneten Typen werden entsprechend unterstützt, mit (*) gekennzeichneten Sensortypen sind in Vorbereitung.

Enum Name	VAR	Enum Wert		RSC102 RSC123 Rev. 2	CM1xx
AIN_0_10VOLT	INT	0	voltage, 0/10V, res.150µV		
AIN_0_1VOLT	INT	1	voltage, 0/1V, res. 30µV		
AIN_0_0_1VOLT	INT	2	voltage, 0/0.1V, res. 10µV		
AIN_0_20mA	INT	3	current, 0/20mA, res.0.3µA		
AIN_4_20mA	INT	4	current, 4/20mA, res.0.3µA		
AIN_0_10VOLT_NORMED	INT	5	voltage, 0/10V, normed, 0..1		X
AIN_0_20mA_NORMED	INT	6	current, 0/20mA, normed		
AIN_4_20mA_NORMED	INT	7	current, 4/20mA, normed		
AIN_RESISTOR_2WIRE	INT	8	resistor, 4kOhm, 2 wire		
AIN_RESISTOR_3WIRE	INT	9	resistor, 4kOhm, 3 wire		
AIN_COMPENSATION_1	INT	10	compensation temperature 1		
AIN_COMPENSATION_2	INT	11	compensation temperature 2		
AIN_TC_TYPE_B	INT	12	TC Typ B, 250/1820°C res.1.9	*	
AIN_TC_TYPE_E	INT	13	TC Typ E, -100/800 °C res.0.2	*	
AIN_TC_TYPE_J	INT	14	TC Typ J, -100/1200°C res.0.3	*	
AIN_TC_TYPE_K	INT	15	TC Typ K, -150/1372°C res.0.4	*	
AIN_TC_TYPE_L	INT	16	TC Typ L, -150/900 °C res.0.3	*	
AIN_TC_TYPE_N	INT	17	TC Typ N, -150/1300°C res.0.5	*	
AIN_TC_TYPE_R	INT	18	TC Typ R, -50/1768 °C res.1.5	*	
AIN_TC_TYPE_S	INT	19	TC Typ S, -50/1768 °C res.1.6	*	
AIN_TC_TYPE_T	INT	20	TC Typ T, -150/400 °C res.0.4	*	
AIN_PT100_2WIRE	INT	21	Pt100 2 wire -30/500°C res3.00, -300..5000	X	X
AIN_PT200_2WIRE	INT	22	Pt200 2 wire -30/500°C res0.05		
AIN_PT500_2WIRE	INT	23	Pt500 2 wire -30/500°C res0.05	*	
AIN_PT1000_2WIRE	INT	24	Pt1000 2 wire -30/500°C res0.05, -300..5000	X	
AIN_NI100_2WIRE	INT	25	Ni100 2 wire -150/800°C res0.05		
AIN_NI120_2WIRE	INT	26	Ni120 2 wire -150/800°C res0.05		
AIN_NI200_2WIRE	INT	27	Ni200 2 wire -150/800°C res0.05		
AIN_NI500_2WIRE	INT	28	Ni500 2 wire -150/800°C res0.05		

Enum Name	VAR	Enum Wert		RSC102 Rev. 1	CM1xx
AIN_NI1000_2WIRE	INT	29	Ni1000 2 wire -150/800°C res0.05		
AIN_CU10_2WIRE	INT	30	Cu10 2 wire -150/800°C res0.05		
AIN_PT100_3WIRE	INT	31	Pt100 3 wire -150/800°C res0.05		
AIN_PT200_3WIRE	INT	32	Pt200 3 wire -150/800°C res0.05		
AIN_PT500_3WIRE	INT	33	Pt500 3 wire -150/800°C res0.05		
AIN_PT1000_3WIRE	INT	34	Pt1000 3 wire -150/800°C res0.05		
AIN_NI100_3WIRE	INT	35	Ni100 3 wire -150/800°C res0.05		
AIN_NI120_3WIRE	INT	36	Ni120 3 wire -150/800°C res0.05		
AIN_NI200_3WIRE	INT	37	Ni200 3 wire -150/800°C res0.05		
AIN_NI500_3WIRE	INT	38	Ni500 3 wire -150/800°C res0.05		
AIN_NI1000_3WIRE	INT	39	Ni1000 3 wire -150/800°C res0.05	*	
AIN_CU10_3WIRE	INT	40	Cu10 3 wire -150/800°C res0.05		
AIN_KTY_110_130	INT	41	KTY 110/130 -20/120 °C, res 0.05	*	
AIN_KTY_11_13_5	INT	42	KTY 11/13-5 -20/120 °C, res 0.05		
AIN_KTY_11_13_6	INT	43	KTY 11/13-6 -20/120 °C, res 0.05		
AIN_KTY_11_13_7	INT	44	KTY 11/13-7 -20/120 °C, res 0.05		
AIN_KTY_210_230	INT	45	KTY 210/230 -20/120 °C, res 0.05		
AIN_KTY_21_23_5	INT	46	KTY 21/23-5 -20/120 °C, res 0.05		
AIN_KTY_21_23_6	INT	47	KTY 21/23-6 -20/120 °C, res 0.05		
AIN_KTY_21_23_7	INT	48	KTY 21/23-7 -20/120 °C, res 0.05		
AIN_AIO8_OFF	INT	55	disable this channel	X	X
AIN_M10_10VOLT	INT	100	-10V/10V, -10000..10000	X	
AIN_500_OHM	INT	103	0/500Ohm, 0..500	X	
AIN_NTC_10KOHM	INT	104	NTC-10KOhm, -40..60°C, 400..600	X	
AIN_200KOHM	INT	105	0/200KOhm, 0..20000	X	

Die mit „X“ gekennzeichneten Typen werden entsprechend unterstützt, mit (*) gekennzeichneten Sensortypen sind in Vorbereitung.

4.2.2 AoutConfigureType() - Analogausgangskonfiguration setzen

AOutConfigureType(Node, Channel, eSensorType);

Variable IN	**Typ: =Default**	**Bedeutung**
Node	BYTE	Nummer des abzusprechenden Teilnehmers 0 = PLC Mastersteuerung 1..63 = Node ID des CanOpenSlave-Teilnehmers
Channel	UINT	Nummer des Analogausgang 1..16
eSensorType	INT	Siehe nachstehende Enumerationwerte
Variable OUT	**Typ: =Default**	**Bedeutung**
Rückgabewert	BOOL:=TRUE;	Ein Rückgabewert wird nicht verwendet, bzw. ist immer TRUE.

Die mit „X“ gekennzeichneten Typen werden entsprechend unterstützt, mit (*) gekennzeichneten Sensortypen sind in Vorbereitung.

Enum Name	VAR	Enum Wert		RSC102 Rev.1	CM1xx
AOUT_M10_10VOLT_NORMED	INT	51	voltage output, -10/10V normed	X	X
AOUT_0_20mA_NORMED	INT	52	current output, 0/20mA normed	*	
AOUT_4_20mA_NORMED	INT	53	current output, 4/20mA normed	*	
AOUT_0_10VOLT_NORMED	INT	54	voltage output, 0/10V normed		X
AOUT_AIO8_OFF	INT	55	disable this channel	X	X

HINWEIS

Bitte beachten Sie, dass bei Analogkanälen, die sowohl Ein- als auch Ausgang sein können, der Gegenkanal nicht gesetzt bzw. abgeschaltet werden sollte.

Sind beide gegensätzlichen Sensortypen gesetzt, so wird aus Sicherheitsgründen der Sensortyp „IN“ gewählt, um Anlagenschäden zu verhindern.

4.2.3 Ain() - Analoge Eingänge lesen

```
rAnalogwert := Ain(Node, Channel);
```

Variable IN	**Typ: =Default**	**Bedeutung**
Node	BYTE	Nummer des abzufragenden Teilnehmers 0 = PLC Mastersteuerung 1..63 = NodeID des CanOpenSlave-Teilnehmers
Channel	UINT	Nummer des abzufragenden Analogkanal
Variable OUT	**Typ: =Default**	**Bedeutung**
Rückgabewert	REAL:=0;	Analogwert

Beispiel:

Gerät: RSC102 NodeID: 1 Kanal: Alle 16

```
VAR
  IO    : ELA_IO_Fb;
  Ain   : ARRAY  [1..16] OF REAL;
  Node  : BYTE := 1;
END_VAR

FOR i:=1 TO SIZEOF(Ain)/SIZEOF(REAL) DO
   Ain[i] := IO.Ain(Node,i);
END_FOR
```

Sensoren nicht gesteckt:

Device.Application.PLC_PRG

Ausdruck	Datentyp	Wert	Vorbereiteter Wert
Ain	ARRAY [1..16] OF REAL		
Ain[1]	REAL	999	
Ain[2]	REAL	999	
Ain[3]	REAL	999	
Ain[4]	REAL	999	
Ain[5]	REAL	999	
Ain[6]	REAL	999	
Ain[7]	REAL	999	
Ain[8]	REAL	999	
Ain[9]	REAL	999	
Ain[10]	REAL	999	
Ain[11]	REAL	999	
Ain[12]	REAL	999	
Ain[13]	REAL	999	
Ain[14]	REAL	999	
Ain[15]	REAL	999	
Ain[16]	REAL	999	
Aout	ARRAY [1..16] OF REAL		
Aout[1]	REAL	1	
Aout[2]	REAL	0	

```
12
13  FOR i 16 :=1 TO SIZEOF(Ain)/SIZEOF(REAL) DO
14    Ain[i 16 ] 999  := IO.Ain(Node 1 ,i 16 );
15    IO.Aout(Node 1 ,i 16 ,Aout[i 16 ] 0 );
16  END_FOR
```

Beispiel:

Gerät: RSC102 NodeID: 2 Kanal: alle 16 Kanäle

4.2.4 Aout() - Analoge Ausgänge

Aout(Node, Channel, rValue);

Variable IN	**Typ: =Default**	**Bedeutung**
Node	BYTE	Nummer des anzusprechenden Teilnehmers 0 = PLC Mastersteuerung 1..63 = NodeID des CanOpenSlave-Teilnehmers
Channel	UINT	Nummer des zu setzenden Analogkanal
rValue	REAL:=0;	Skalierter Ausgangswert. Beispiel V3.5.1.4 und Sensortype 51 1.0 → 10 Volt 0.0 → 0 Volt -1.0 → -10 Volt
Variable OUT	**Typ: =Default**	**Bedeutung**
Rückgabewert	BOOL:=FALSE;	Ein Rückgabewert wird nicht verwendet, bzw. ist immer TRUE.

Beispiel: Zum setzen der Analogausgänge

Gerät: RSC102 SlaveNo:1 Kanal : alle 16

Deklaration:

```
VAR
  IO     : ELA_IO_Fb;
  Aout   : ARRAY [1..16] OF REAL;
  Node   : BYTE := 1;
END_VAR
```

Code:

```
FOR i:=1 TO SIZEOF(Ain)/SIZEOF(REAL) DO
   IO.Aout(Node,i,Aout[i]);
END_FOR
```

4.2.5 AoutHandlingAfterStop() - Analoge Ausgänge für PLC = STOP

Beispiel:

IO.AoutHandlingAfterStop(Enable);

Hinweis:
Diese Option besteht aus 2 Anweisungen: AoutHandlingAfterStop() und AoutValueAfterStop()
Diese Option gilt derzeit nur für Mastersteuerungen.
Für Slave Module gilt das Default- Nodeguardingverhalten.

Variable IN	**Typ**	**Bedeutung**
Enable	BOOL	Aktivieren / Deaktivieren des Verhalten**:** Wertevorgabe analoge Ausgänge für PLC = STOP

4.2.6 AoutValueAfterStop() - Analoge Ausgänge für PLC = STOP

Beispiel:

IO.AoutValueAfterStop(Node, Channel, Value);

Hinweis:
Diese Option besteht aus 2 Anweisungen: AoutHandlingAfterStop() und AoutValueAfterStop()
Diese Option gilt derzeit nur für Mastersteuerungen.
Für Slave Module gilt das Default- Nodeguardingverhalten.

Variable IN	**Typ: =Default**	**Bedeutung**
Node	BYTE	Nummer des anzusprechenden Teilnehmers 0 = PLC Mastersteuerung 1..63 = NodeID des CanOpenSlave-Teilnehmers (in Vorbereitung. Aktuell wird der Nodeguarding- Default Wert der Slave Komponente verwendet)
Channel	UINT	Nummer des zu setzenden Analogkanal
rValue	REAL:=0;	Analogausgangswert für den Fall PLC = STOP

4.3 Allgemeine Funktionen

4.3.1 NodeActive() - Abfrage Verfügbarkeit Slave

```
bNodeIsRunning := NodeActive(Node);
```

```
SlaveActive TRUE := io.NodeActive(1);
```

METHOD NodeActive

Abfrage ob Knoten bereit ist

NodeActive	BOOL	VAR_OUTPUT	*Rückgabewert siehe Tabelle unten*
Node	BYTE	VAR_INPUT	Input: Slave-NodeID

Die Funktion gibt Auskunft darüber, ob sich der Slave im „Operational Mode“ befindet und funktioniert.

Variable IN	**Typ :=Default**	**Bedeutung**
Node	BYTE	Slavenummer des abzufragenden Modul
Variable OUT	**Typ :=Default**	**Bedeutung**
Rückgabewert	BOOL :=FALSE;	TRUE: Slave befinden sich im Operational Mode und ist betriebsbereit. (RUN) FALSE: Slave ist nicht betriebsbereit (STOP / ERROR)

4.3.2 NodeState() - Zustand Slave

```
bySlaveStatus := SlaveState(Node);

SlaveState 5 := io.NodeState(1);
```

METHOD NodeState

Knoten Status: 0..3= Pre-Operational, 5= Operational (läuft) , 99 = Error (Knoten verloren)

NodeState	BYTE	VAR_OUTPUT	*Rückgabewert siehe Tabelle unten*
Node	BYTE	VAR_INPUT	Input: Slave-NodeID

Variable IN	**Typ: =Default**	**Bedeutung**
Node	BYTE	Slavenummer des abzufragenden Modul
Variable OUT	**Typ: =Default**	**Bedeutung**
Rückgabewert	BYTE:=0	0: Slave nicht vorhanden 3: Slave in Pre-Operational Mode (also in der Initialisierungsphase) 5: Slave in Operational Mode und betriebsbereit. 99: Slave verloren

4.3.3 SetNodeOptional() - Slave zwingend oder optional betreiben

SetNodeOptional(Node, TRUE);

```
IO.SetNodeOptional(Node:=1 , Optional:=FALSE );
```

METHOD SetNodeOptional

CAN-Slave ist optionaler Teilnehmer, Default = TRUE

SetNodeOptional	BOOL	VAR_OUTPUT	*Rückgabewert siehe Tabelle unten*
Node	BYTE	VAR_INPUT	NodeID
Optional	BOOL	VAR_INPUT	Slave ist marked as optional;

Variable IN	**Typ :=Default**	**Bedeutung**
Node	BYTE	Slavenummer des Modul
Optional	BOOL :=TRUE;	TRUE: Optional (Default) Ist der Slave verloren, bzw. bereits zu Beginn nicht vorhanden, so starten die restlichen Slaves dennoch. FALSE: Alle Slaves starten erst gemeinsam
Variable OUT	**Typ :=Default**	**Bedeutung**
Rückgabewert	BOOL :=TRUE;	Ein Rückgabewert wird nicht verwendet, bzw. ist immer TRUE.

4.3.4 UnforceCyclicSend() - Zyklisches Senden unterbinden.

UnforceCyclicSend(bUnforce)

```
IO.UnforceCyclicSend(Unforce:= xUnforce TRUE );
```

METHOD UnforceCyclicSend

UnforceCyclicSend	BOOL	VAR_OUTPUT	*Rückgabewert siehe Tabelle unten*
Unforce	BOOL	VAR_INPUT	No Circlic TxPDO- Telegramms allowed (Master --> Slave)

Digitale und analoge Zustandsänderungen werden nach Modifikation unmittelbar vom Master an den jeweiligen Slave gesendet.
Dies geschieht zusätzlich auch zyklisch im Sekundentakt. (Defaulteinstellung)
Hiermit soll die Sicherheit durch verloren gegangene CAN Bus Telegramme erhöht werden.
Wird die Option nicht benötigt, so kann diese auch abgeschaltet werden.

Variable IN	**Typ: =Default**	**Bedeutung**
Unforce	BOOL:=FALSE;	TRUE: Zusätzliches zyklisches Übertragen aller Rx-PDO-Telegramme vom Master an den Slave werden ausgeschaltet. FALSE: (Default) Zusätzliches zyklisches Übertragen aller Rx-PDO-Telegramme vom Master an den Slave sind eingeschaltet.
Variable OUT	**Typ: =Default**	**Bedeutung**
Rückgabewert	BOOL:=TRUE;	Ein Rückgabewert wird nicht verwendet, bzw. ist immer TRUE.

4.3.5 CanOpenInit() - CanConfiguration

Beispiel:
IO.CanOpenInit(CanNo:=CanNo,Baudrate:=Baudrate);

Ab Version 3.5.8.2 können Änderungen während der Laufzeit vorgenommen werden.

Variable IN	**Typ**	**Bedeutung**
CanNo	BYTE	Wahl des CAN-Controllers 0 := CAN0 (Default) 1 := CAN1 (>=2) := Kommunikationsbaustein ist deaktiviert
Baudrate	WORD	Vorwahl der Übertragungsrate in kb 0 := Kommunikationsbaustein ist deaktiviert 10 := 10 kb 20 := 20 kb 50 := 50 kb (Default) 100 := 100 kb 125 := 125 kb 500 := 250 kb 1000 := 1Mb (>=1001) := Kommunikationsbaustein ist deaktiviert

4.3.6 SetGuardTime() - Einstellungen Nodeguarding

Beispiel:
IO.SetGuardTime(Node :=1 ,GuardTime:= 500, Livetime := 6);

Werden Einstellungen abweichend der Defaultwerte benötigt, sollte vor dem ersten Instanzaufruf diese Methode angewandt werden. Dies ist jedoch nicht zwingend.
Diese Methode wird einzeln für jeden angeschlossenen Slave angewandt.
Wird die Methode nicht verwendet, so gilt:
Guardtime := 500;
Livetime := 10;

Variable IN	**Typ**	**Bedeutung**
Node	BYTE	Node ID des Slaves auf dem die Änderung vorgenommen werden soll.
GuardTime	WORD	GuardTime in Milisekunden. In diesem Abstand sendet der Master ein Nodeguarding Telegramm.
LiveTime	BYTE	Faktor. Wird über die Zeit Guardtime x LiveTime kein Nodeguarding Telegramm empfangen oder gesendet, so gilt der Partner als verloren.

5 Informationen

Folgende Variablenwerte geben Rückschluss über den Baustein

IO.CAN_Itf.FBVersion;

Gibt Auskunft über die Bausteinversion z.B. „3.5.8.2“

IO.CAN_Itf.FBState;

Gibt Auskunft über den Verbindungzustand der Kommunikation des CanOpen.

„DISABLE“ := Baustein ist deaktiviert
„RUN“ := Baustein läuft, ist aber nicht verbunden
„WAIT“ := Verbindet sich gerade mit einem Gerät
„CONNECTED := Ist mindestens mit einem Gerät verbunden

6 Analogkonfiguration

Die analogen Ein- und Ausgänge können zum Programmstart, als auch während des laufenden Betriebs geändert werden.

Funktionsweise mit Start-Konfiguration:

1.) Konfigurieren der Kanäle.

2.) Anhand der Kanalwahl wird der Slave im Pre-Operational-Mode bereits mit dieses Werten versorgt. (STOP- MODE)

3.) Start des Gerätes im Operational Mode (RUN-MODE)

Empfehlung für die einmalige Konfiguration zum Start.

Ab FB-Version 3.5.1.5 kann der FB ohne Analogkonfiguration gestartet werden. Dies kann erst zu einem späterem Zeitpunkt im Status „Operational“ erfolgen. In der Slaves RSCxx Variante werden hierzu zum Start bereits alle 5 Rx- und TxPDO’s aktiv geschalten.

Bitte konfigurieren Sie die Kanäle VOR dem eigentlichen Aufruf des Funktionsbaustein, wie folgt:

Deklaration:

```
VAR
    i  :UINT;       // Analogkanalnummer
END_VAR
```

Code:

```
IF xINIT = FALSE THEN
      FOR i:=1 TO 16 BY 1 DO
        CASE i OF

                  // Analog input
                2:    IO.AinConfigureType(Node, i, ELA_IO_ITF.AIN_NTC_10KOHM);
                3:    IO.AinConfigureType(Node, i, ELA_IO_ITF.AIN_PT1000_2WIRE);
                4:    IO.AinConfigureType(Node, i, ELA_IO_ITF.AIN_200KOHM);
                5..8: IO.AinConfigureType(Node, i, ELA_IO_ITF.AIN_0_10VOLT);

                  // Analog output
               1,9..16:   IO.AoutConfigureType(1, i, ELA_IO_ITF.AOUT_M10_10VOLT_NORMED);
                          IO.AoutConfigureType(4, i, ELA_IO_ITF.AOUT_M10_10VOLT_NORMED);
                          IO.AoutConfigureType(6, i, ELA_IO_ITF.AOUT_M10_10VOLT_NORMED);
         END_CASE
 END_FOR
          xINIT:= TRUE;  // INITIALISIERUNG ABGESCHLOSSEN

    ELSE
    IO();        // BAUSTEINAUFRUF CAN- KOMMUNIKATION
END_IF;
```

7 Anwendung

7.1 Aktivieren von Slavemodulen

Ein Slavemodul wird im KommunikationsFB zur Verwendung zugebucht, wenn vor dem ersten Bausteinaufruf IO() ein Zugriff auf das Modul erfolgt. Erfolgt dieser Zugriff nicht, wird der Slave nicht initialisiert und somit nicht verwendet. Dieses „Zubuchen“ kann durch die Benutzung einer der nachfolgenden Methoden erfolgen:

Din();
Dout(),
AinConfigureType()
AoutConfigureType()
Ain()
Aout()

Ein Minimalcode zum Starten von Slavemodul #1 könnte daher wie folgt aussehen:

```
;           // Code Anfang
Din(1,1);   // Abfrage auf Slave #1 Kanal #1
IO();       // Aufruf Kommunikationsbaustein.
;           // Code Ende
```

Taskpriorität:

Der Funktionsaufruf IO sollte mit einer der höchsten Prioritäten aufgerufen werden.
Zur Projektierung wurde folgende Einstellung benutzt.

Priorität : 1

Zykluszeit : 5 – 10 ms

ACHTUNG:
Ist wird die Zykluszeit größer gewählt, so benötigt der Slave mehr Zeit für den Anlauf.
Eine typische Zeit für eine Maximalkonfiguration aller Kanäle auf NTC_10KOHM und einer Zykluszeit von 10ms, ergibt eine Anlaufzeit von 4 Sekunden je Gerät.
Eine Zykluszeit von 100ms ergibt eine Anlaufzeit von ca 30 Sek.

Wählen Sie daher eine möglichst kleine Zykluszeit um eine schnelle CAN-Kommunikation zu gewährleisten.

Ab der Version 3.5.1.5 können alle Analogkanäle zu einem beliebigen Zeitpunkt konfiguriert oder geändert werden.

Ab der Version 3.5.1.6 wurde das Anlaufverhalten geändert und deutlich reduziert.
Relativ unabhängig von der eingestellten Zykluszeit konnte die Anlaufzeit unter 0,9 Sekunde verringert werden.

Ein weiteres Beispielcode könnte wie folgt aussehen:
4x digital Input, 4x digital Output, 1x Analog In (NTC 10k) , 1x analog Output (-10 +10Volt)

Deklaration:

```
VAR
        IO : RTH_IO_Fb;
        Ain:    ARRAY  [1..16] OF REAL;
        Aout:  ARRAY  [1..16] OF REAL;
        Din:    ARRAY  [1..32] OF BOOL;
        Dout:  ARRAY  [1..32] OF BOOL;
        bInit:   BOOL := FALSE;
        SlaveActive :BOOL;
        SlaveState  :BYTE;
END_VAR
```

Code:

```
// Config Analoge
IF bInit = FALSE THEN
  IO.AinConfigureType(1,1,  ELA_IO_ITF.AIN_NTC_10KOHM);
  IO.AoutConfigureType(1,2, ELA_IO_ITF.AOUT_M10_10VOLT_NORMED);
  bInit := TRUE;
END_IF

IF IO.NodeActive(Node:=1 ) = TRUE THEN          // Slave #1 is running

  // Read digital inputs
  Din[1] := IO.Din(1,0);                  // Read digital input #1
  Din[2] := IO.Din(1,1);                  // Read digital input #2
  Din[3] := IO.Din(1,2);                  // Read digital input #3
  Din[4] := IO.Din(1,3);                  // Read digital input #4

  // Read analog inputs
  Ain[2]:= IO.Ain(ChannelNo:= 1,1);          // Read analog  input #1 (Defined as NTC1000 )

  // Write digital output:
  IO.Dout(1,0, Dout[1]);                   // Write digital output #1
  IO.Dout(1,1, Dout[2]);                   // Write digital output #2
  IO.Dout(1,2, Dout[3]);                   // Write digital output #3
  IO.Dout(1,3, Dout[4]);                   // Write digital output #4

   // Write analog output
  IO.Aout(1,2,Aout[2]);                    // Write analog  output #2 ( Defined as -10V to +10 Volt normed)
END_IF;

// Funktion
IO();

// Diagnostic Funktion
SlaveActive := IO.NodeActive(Node:=1 );
SlaveState  := IO.NodeState(Node:=1 );
```

8 Funktionsweise mit Start- & Re-Konfiguration

Ein Aufruf der Analogkonfiguration mit den Befehlen

- AInConfigureType() oder
- AOutConfigureType()

bewirkt folgendes:

1.) Ein Aufruf diese Funktion im Initialschritt bindet den Slave zur Benutzung ein (zubuchen)

2.) Konfigurieren der Kanäle im Pre-Operational (Startphase)

3.) Re-Konfiguration im Operational-Mode (RUN-MODE)

Beispiel einer Änderung ALLER Kanäle zu Beginn und/oder während des laufenden Betrieb:

Fall 1:
Bei xTest = TRUE werden alle Analogkanäle als EINGANG für Fühler NTC_10KOHM betrieben.
Alle etwaigen gegensätzliche Korrespondierenden AUSGÄNGE werden ausgeschaltet. (rot)

Fall 2:
Bei xTest = FALSE werden alle Analogkanäle als AUSGANG M10_10VOLT betrieben.
Alle etwaigen gegensätzliche Korrespondierenden EINGÄNGE werden ausgeschaltet. (rot)

```
FOR i:= 1 TO 16 BY 1 DO
  IF xTest THEN
    IO.AinConfigureType(1,i, ELA_IO_ITF.AIN_NTC_10KOHM);                (* Fall 1 *)
    IO.AoutConfigureType(1,i, ELA_IO_ITF.AOUT_AIO8_OFF);
    ELSE
      IO.AoutConfigureType(1,i, ELA_IO_ITF.AOUT_M10_10VOLT_NORMED);    (* Fall 2 *)
      IO.AinConfigureType(1,i, ELA_IO_ITF.AIN_AIO8_OFF);
  END_IF
END_FOR;
IO();         // BAUSTEINAUFRUF CAN- KOMMUNIKATION
```

Da hier ein Wechsel von analog IN zu Analog OUT erfolgt, ist es wichtig, dass der jeweils gegensätzlich korrespondierende Kanal deaktiviert wird.
→ Markierung rot.

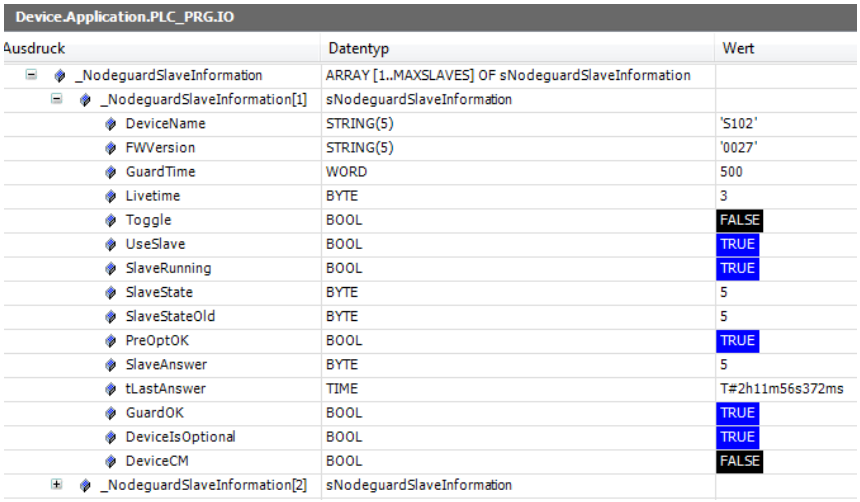
In Folge nimmt der FB eine Rekonfiguration der Kanäle im laufenden Betrieb vor.

*Dies erfolgt sequenziell und benötigt einige Zeit. Im oben dargestellten Beispiel müssen für die Rekonfiguration (4 RxSDO + 1x RxSDO + 5x TxSDO * 16 Kanäle) = 160 SDO- Telegramme zum bestehenden PDO- Verkehr hinzugefügt werden.*
Dies kann 1 bis 1,5 Sekunden Zeit in Anspruch nehmen.

Inhalt der Struktur Nodeguard Slave Information:
Nachfolgende Einträge sind Read-Only und dienen der Diagnose bzw. der Information.

9 Nodeguarding

Alle Einträge in der Struktur dienen der Information und sind Read-Only



Device.Application.PLC_PRG.IO

Ausdruck	Datentyp	Wert
_NodeguardSlaveInformation	ARRAY [1..MAXSLAVES] OF sNodeguardSlaveInformation	
_NodeguardSlaveInformation[1]	sNodeguardSlaveInformation	
DeviceName	STRING(5)	'S102'
FWVersion	STRING(5)	'0027'
GuardTime	WORD	500
Livetime	BYTE	3
Toggle	BOOL	FALSE
UseSlave	BOOL	TRUE
SlaveRunning	BOOL	TRUE
SlaveState	BYTE	5
SlaveStateOld	BYTE	5
PreOptOK	BOOL	TRUE
SlaveAnswer	BYTE	5
tLastAnswer	TIME	T#2h11m56s372ms
GuardOK	BOOL	TRUE
DeviceIsOptional	BOOL	TRUE
DeviceCM	BOOL	FALSE
_NodeguardSlaveInformation[2]	sNodeguardSlaveInformation	

Variablenname	**Wert**	**Erläuterung**
DeviceName (String(5))	‚S102‘	4-stelliger Gerätenamen ‚S102‘ → RSC102 ‚S123‘ → RSC123
FWVersion (String(5))	‚0027‘	4-stellige Firmwareversion des Slaves ‘0027’ → Version 0.27 ‚0100‘ → Version 1.00
GuardTime (Word)	Fix 500	GuardTime in ms, welche Master und Slave verwenden.
Livetime (Byte)	Fix 3	Anzahl der Guard-Telegramme die ausbleiben dürfen ehe der Slave als verloren gilt.
Toggle (Bool)		Der Zustand des höchste Bits im Nodeguarding Telegramm vom Master an die Slavesteuerung. Dieses muss mit jedem Guard-Telegramm den Zustand ändern.
UseSlave		Die Initialisierung und Benutzung dieses Slavemodul wird durch die Methoden AinConfigureType() bzw. AoutConfigureType() angefordert.
SlaveRunning		Der Slave ist im Status OPERATIONAL-MODE
SlaveState		0 = Init (Stop) 3 = Pre- Operational (Anlauf) 5 = Operational (Run) 99 = Slave Lost
SlaveAnswer		Operational: Wechsel zwischen 16#05 (5dez) und 16#85 (133dez)

		Slave verloren: Kein Wechsel. Slave nach Verlust wieder verfügbar: Wechsel zwischen 16#7F und 16#FF
tLastAnswer		Zeit der letzten Antwort seit PLC-Start. (Wird für die interne Berechnung benötigt)
GuardOK		Vom Slave kommen Nodeguarding-Telegramme.

Nodeguarding ist fix auf GuardTime =500ms, Timelifefaktor = 3.
Ein Nodeguarding ist für die Slave-Erkennung zwingend notwendig und entsprechend konfiguriert.

Im Operational-Mode (RUN-MODE) wird alle 500 ms zu jedem Slave ein Nodeguarding-Request-Telegramm ohne Inhalt gesandt, worauf dieser antwortet.

Das Antworttelegramm vom Slave zum Master toggelt mit dem Inhalt

0x05 / 0x85.

Abbildung : CAN-Mitschnitt im Analyzer.
Der Filter wurde so gesetzt, dass nur Nodeguarding Telegramme dargestellt werden.

TimeTick	ID	Dat...	Dir	L...	Interpreter
10:23:53.600	702hex		N002 (Master==>Slave)	0	Nodeguard:Request
10:23:53.600	702hex	05,	N002 (Master<==Slave)	1	Nodeguard:Response
10:23:53.600	701hex	05,	N001 (Master<==Slave)	1	Nodeguard:Response
10:23:54.100	701hex		N001 (Master==>Slave)	0	Nodeguard:Request
10:23:54.100	702hex		N002 (Master==>Slave)	0	Nodeguard:Request
10:23:54.100	701hex	85,	N001 (Master<==Slave)	1	Nodeguard:Response
10:23:54.100	702hex	85,	N002 (Master<==Slave)	1	Nodeguard:Response
10:23:54.614	701hex		N001 (Master==>Slave)	0	Nodeguard:Request
10:23:54.614	702hex		N002 (Master==>Slave)	0	Nodeguard:Request
10:23:54.614	701hex	05,	N001 (Master<==Slave)	1	Nodeguard:Response
10:23:54.614	702hex	05,	N002 (Master<==Slave)	1	Nodeguard:Response

9.1 Verbindungsunterbrechung:

Ist die Kommunikation zwischen Master und Slave für mehr als
Guardtime x Lifetimefaktor = 500ms x 3 = 1,5Sek
unterbrochen, so detektieren beide Seiten eine Störung.

Folgendes geschieht:

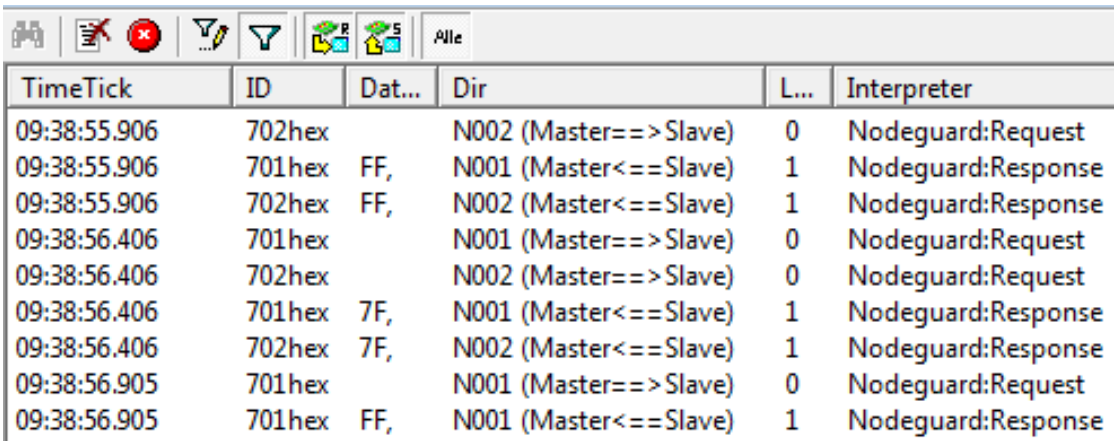
Master:
Der Slave wird als verloren erkannt.
→ NodeState = 99
→ NodeActive = FALSE

Slave:
Der Master wird als verloren erkannt.

→Alle digitalen Ausgänge werden abgeschalten.
→Alle analogen Ausgänge werden abgeschalten.

Erneute Verbindung / Anlauf:
Erhält der Slave erneut ein Nodeguarding Request von der Mastersteuerung, so teilt er über das Nodeguarding seinen Status mit.

Toggle:
0x7F / 0xFF → Eine Neukonfiguration ist notwendig
0x05 / 0x85 → Eine Neukonfiguration ist **nicht** notwendig.



TimeTick	ID	Dat...	Dir	L...	Interpreter
09:38:55.906	702hex		N002 (Master==>Slave)	0	Nodeguard:Request
09:38:55.906	701hex	FF,	N001 (Master<==Slave)	1	Nodeguard:Response
09:38:55.906	702hex	FF,	N002 (Master<==Slave)	1	Nodeguard:Response
09:38:56.406	701hex		N001 (Master==>Slave)	0	Nodeguard:Request
09:38:56.406	702hex		N002 (Master==>Slave)	0	Nodeguard:Request
09:38:56.406	701hex	7F,	N001 (Master<==Slave)	1	Nodeguard:Response
09:38:56.406	702hex	7F,	N002 (Master<==Slave)	1	Nodeguard:Response
09:38:56.905	701hex		N001 (Master==>Slave)	0	Nodeguard:Request
09:38:56.905	701hex	FF,	N001 (Master<==Slave)	1	Nodeguard:Response

Ist der Slave in der Mastersteuerung als verloren Markiert (Status = 99) und antwortet dieser erneut auf ein Requesttelegramm mit Inhalt 0x7F oder 0xFF, so wird der Slave neu konfiguriert.

Beispiel Abbildung:Slave #2 ist verloren und wird wieder verfügbar:

10:31:06.909	701hex		N001 (Master==>Slave)	0	Nodeguard:Request
10:31:06.909	702hex		N002 (Master==>Slave)	0	Nodeguard:Request
10:31:06.909	701hex	05,	N001 (Master<==Slave)	1	Nodeguard:Response
10:31:07.205	201hex	02,00,00,00,	N001 PDO-Recv (Master==>Slave)	4	RxPDO No[1]:
10:31:07.205	181hex	02,00,00,00,	N001 PDO-Send (Master<==Slave)	4	TxPDO No[1]:
10:31:07.408	701hex		N001 (Master==>Slave)	0	Nodeguard:Request
10:31:07.408	702hex		N002 (Master==>Slave)	0	Nodeguard:Request
10:31:07.408	702hex	FF,	N002 (Master<==Slave)	1	Nodeguard:Response
10:31:07.408	701hex	85,	N001 (Master<==Slave)	1	Nodeguard:Response
10:31:07.907	701hex		N001 (Master==>Slave)	0	Nodeguard:Request
10:31:07.907	702hex		N002 (Master==>Slave)	0	Nodeguard:Request
10:31:07.907	702hex	7F,	N002 (Master<==Slave)	1	Nodeguard:Response
10:31:07.907	701hex	05,	N001 (Master<==Slave)	1	Nodeguard:Response
10:31:07.907	000hex	80,00,	Broadcast	2	NMT:
10:31:08.001	000hex	81,02,	Broadcast	2	NMT:
10:31:08.001	602hex	40,00,10,00,	N002 SDO-Request (Master==>Slave)	4	Idx=1000,Sub=00,RD ==> Init, ccs=2,x=0
10:31:08.001	602hex	40,00,10,00,	N002 SDO-Request (Master==>Slave)	4	Idx=1000,Sub=00,RD ==> Init, ccs=2,x=0
10:31:08.001	602hex	40,00,10,00,	N002 SDO-Request (Master==>Slave)	4	Idx=1000,Sub=00,RD ==> Init, ccs=2,x=0
10:31:08.110	602hex	40,00,10,00,	N002 SDO-Request (Master==>Slave)	4	Idx=1000,Sub=00,RD ==> Init, ccs=2,x=0
10:31:08.110	602hex	40,00,10,00,	N002 SDO-Request (Master==>Slave)	4	Idx=1000,Sub=00,RD ==> Init, ccs=2,x=0
10:31:08.204	602hex	40,00,10,00,	N002 SDO-Request (Master==>Slave)	4	Idx=1000,Sub=00,RD ==> Init, ccs=2,x=0
10:31:08.204	602hex	40,00,10,00,	N002 SDO-Request (Master==>Slave)	4	Idx=1000,Sub=00,RD ==> Init, ccs=2,x=0
10:31:08.204	702hex	00,	N002 (Master<==Slave)	1	Nodeguard:Response
10:31:08.313	602hex	40,00,10,00,	N002 SDO-Request (Master==>Slave)	4	Idx=1000,Sub=00,RD ==> Init, ccs=2,x=0
10:31:08.313	582hex	43,00,10,00,91,01,0F,00,	N002 SDO-Response (Master<==Slave)	8	Idx=1000,Sub=00,RD <== Init expedited 983441 [F0191h] 4 Bytes
10:31:08.313	602hex	40,18,10,01,	N002 SDO-Request (Master==>Slave)	4	Idx=1018,Sub=01,RD ==> Init, ccs=2,x=0
10:31:08.313	582hex	43,18,10,01,32,00,00,00,	N002 SDO-Response (Master<==Slave)	8	Idx=1018,Sub=01,RD <== Init expedited 50 [32h] 4 Bytes
10:31:08.313	602hex	40,08,10,00,	N002 SDO-Request (Master==>Slave)	4	Idx=1008,Sub=00,RD ==> Init, ccs=2,x=0
10:31:08.313	582hex	43,08,10,00,53,31,30,32,	N002 SDO-Response (Master<==Slave)	8	Idx=1008,Sub=00,RD <== Init expedited 842019155 [32303153h] 4 Bytes
10:31:08.313	602hex	40,0A,10,00,	N002 SDO-Request (Master==>Slave)	4	Idx=100A,Sub=00,RD ==> Init, ccs=2,x=0
10:31:08.313	582hex	43,0A,10,00,30,30,32,37,	N002 SDO-Response (Master<==Slave)	8	Idx=100A,Sub=00,RD <== Init expedited 926036016 [37323030h] 4 Bytes

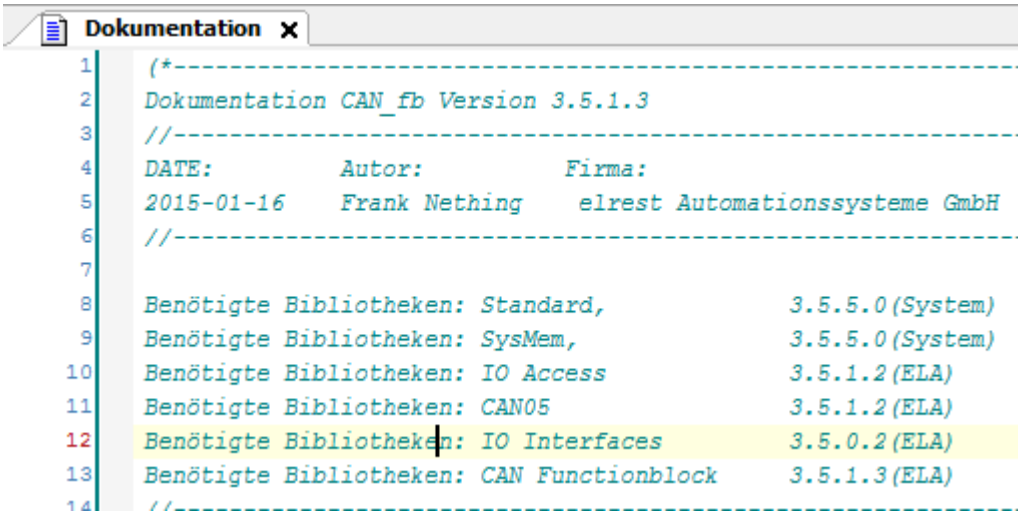
An der Markierung antwortet der Slave#2 erstmalig wieder auf eine Request mit 0xFF.
Nach einer weiteren Antwort mit 0x7F gilt der Slave wieder als vorhanden, jedoch als nicht konfiguriert.

Mit dem MNT Befehl 81,02 wird der Slave #2 in STOP gesetzt.
Dieser Betriebsartenwechsel bewirkt, dass er ca. 200ms nicht mehr auf Telegramme antworten kann.

Nachdem er auf Anfragen wieder antworten kann (hierfür wird das 1000er Objekt benutzt) startet die Konfiguration im Pre-Operational Mode.
Nach dessen Ende wird der Knoten Mit NMT 01,02 gestartet, und der Slave befindet sich im Operational Mode (RUN).

9.1.1 Dokumentation weiterer Komponenten

Eine Dokumentation weiterer Komponenten ist im Baustein „Dokumentation“ hinterlegt.



10 Hilfe bei Störungen

10.1 Service und Support

Hotline

Für zusätzliche Unterstützung und Informationen können Sie unsere Hotline zu folgenden Zeiten erreichen:

Mo-Do:8.30 - 12.00 und 13.00 - 16.30

Fr: 8.30 - 12.00

Tel.: +49 (0) 7021 / 92025-33

Außerhalb dieser Zeiten, können Sie uns per e-mail oder Fax erreichen:

Fax.: +49 (0) 7021 / 92025-29
e-mail: support@elrest.de

Training und Workshops

Wir bieten Ausbildung oder Projekt bezogene Workshops zu allen elrest Produkten an.

Für weitere Informationen kontaktieren Sie bitte unsere Vertriebsabteilung:

Telefon: +49 (0) 7021/92025-0
Fax: +49 (0) 7021/92025-29
E-mail: *vertrieb@elrest.de*

11 Historie

Datum	*Name*	*Kapitel*	*Änderung*
04.02.2015	Ne	V3.5.1.4	Neu erstellt
06.02.2015	Ne	V3.5.1.4	Formatierungen im Dokument
06.03.2015	Ne	V3.5.1.6	Schneller Anlauf, Bugfixes
16.04.2015	Ne	V3.5.1.6	Doc. Anpassung 1..16 → 0..15
19.04.2016	Ne	V3.5.1.7	Ausg.Ver.bei Stop, CAN0/CAN1
30.08.2016	Ne	V3.5.8.1	Bugfixes
18.11.2016	Ne	V3.5.8.1	Erweiterungen
14.03.2019	Ne	V3.5.12.1	Bugfix: AOut um einen Kanal versetzt.