SYS TEC ELECTRONIC

SDO-Gateway Add-on

Software Manual

Auflage September 2013

Status / Änderungen

Status: Entwurf

Datum/ Version	Abschnitt	Änderung	Editor
10.07.2006/ 01		Erstanlage	Dietzsch
26.09.2013/ 02	alle	Überarbeitung nach CiA-302	Dietzsch

Im Buch verwendete Bezeichnungen für Erzeugnisse, die zugleich ein eingetragenes Warenzeichen darstellen, wurden nicht besonders gekennzeichnet. Das Fehlen der © Markierung ist demzufolge nicht gleichbedeutend mit der Tatsache, dass die Bezeichnung als freier Warenname gilt. Ebenso wenig kann anhand der verwendeten Bezeichnung auf eventuell vorliegende Patente oder einen Gebrauchsmusterschutz geschlossen werden.

Die Informationen in diesem Handbuch wurden sorgfältig überprüft und können als zutreffend angenommen werden. Dennoch sei ausdrücklich darauf verwiesen, dass die Firma SYS TEC electronic GmbH weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgeschäden übernimmt, die auf den Gebrauch oder den Inhalt dieses Handbuches zurückzuführen sind. Die in diesem Handbuch enthaltenen Angaben können ohne vorherige Ankündigung geändert werden. Die Firma SYS TEC electronic GmbH geht damit keinerlei Verpflichtungen ein.

Ferner sei ausdrücklich darauf verwiesen, dass SYS TEC electronic GmbH weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgeschäden übernimmt, die auf falschen Gebrauch oder falschen Einsatz der Hard- bzw. Software zurückzuführen sind. Ebenso können ohne vorherige Ankündigung Layout oder Design der Hardware geändert werden. SYS TEC electronic GmbH geht damit keinerlei Verpflichtungen ein.

Kontakt	**Direkt**	**Ihr Lokaler Distributor**
Adresse:	SYS TEC electronic GmbH Am Windrad 2 D-08468 Heinsdorfergrund GERMANY	Sie finden eine Liste unserer Distributoren unter http://www.systec-electronic.com/distributors
Angebots-Hotline:	+49 3765 38600-2110 info@systec-electronic.com	
Technische Hotline:	+49 3765 38600-2140 support@systec-electronic.com	
Fax:	+49 3765 38600-4100	
Webseite:	http://www.systec-electronic.com	

01. Auflage September 2013

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Referenzierte Dokumente

/1/ CANopen User Manual L-1020, SYS TEC electronic GmbH

/2/ CiA Draft Standard Proposal 302, Part 7 Multi-level networking, © CAN in Automation (CiA) e. V.

Abkürzungen

API	Application Programming Interface
CAN	Controller Area Network
CAN-ID	CAN Identifier, Arbitrierungsfeld des CAN-Telegramms. Es muss sichergestellt werden, dass nur ein Gerät zu einem Zeitpunkt mit ein und derselben CAN-ID sendet. Anderenfalls kann es zu Übertragungsstörungen bis zum Bus-Off kommen.
CCM	CANopen Controlling Modul, SYS TEC eigene CANopen Applikationsschnittstelle
CiA	CAN in Automation e.V.
COB	CANopen Message Object
COB-ID	CANopen Message Object ID, enthält die CAN-ID und zusätzliche Informationen die CANopen benötigt (z.B. Valid-Bit)
COP	CANopen
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
EMYC	CANopen Emergency Object (siehe CiA-301)
HBC	Heartbeat Consumer (siehe CiA-301)
HBP	Heartbeat Producer (siehe CiA-301)
LSS	Layer Setting Service (siehe CiA-301)
NMT	Network Management (siehe CiA-301)
OD	Object Dictionary
PDO	Process Data Object (siehe CiA-301)
SDO	Service Data Object (siehe CiA-301)

1 Einleitung

Dieses Handbuch beschreibt das Software-Modul zur Realisierung eines SDO-Gateways. Dieses Handbuch stellt eine Ergänzung zum CANopen User Manual L-1020 /1/ dar.

Das SDO-Gateway wird seit der CANopen Version 5.34 angeboten. Diese Version entsprach nicht dem CiA Standard 302 Part 7 /2/. Ab Version 5.56 wurde das SDO-Gateway nach diesem Standard neu implementiert und ist ab dieser Version verfügbar.

2 Das Prinzip des SDO-Gateway

Das Software-Modul realisiert den SDO-Transfer auf CANopen Geräten, die sich in anderen Netzwerken befinden als das SDO Client Gerät, nach dem CiA-Standard 302 Part 7 /2/.

Das SDO-Gateway wird in einem CANopen Gerät mit mindestens 2 CAN-Schnittstellen und damit mindestens 2 CANopen-Instanzen implementiert. In dessen Objektverzeichnis existieren Objekteinträge für die Konfiguration des SDO-Gateways.

Die Übertragung von Prozessdatenobjekten (PDO), Emergency-Nachrichten oder NMT-Kommandos in andere Netzwerke werden mit diesem Gateway nicht realisiert.

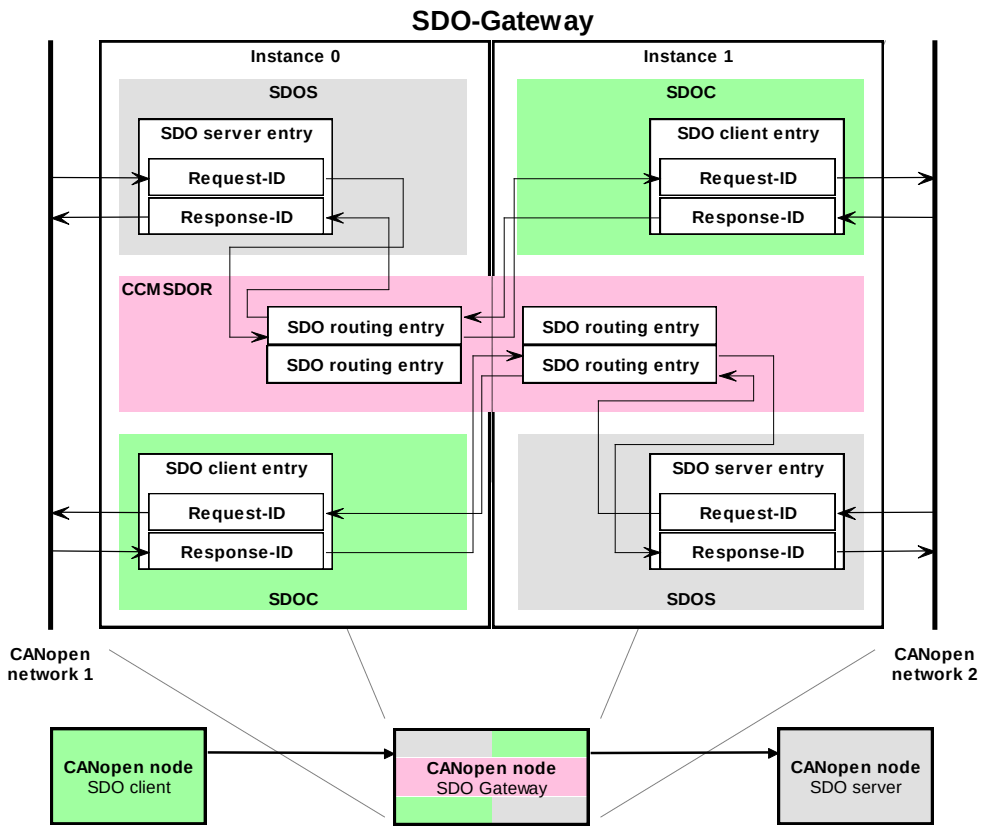

Abbildung 1: Schematische Darstellung des SDO-Gateway

Das SDO-Gateway nutzt den Standard SDO Client und Server jeder CANopen Instanz und vereinigt diese im Modul ***ccmsdor.c***. Das Routing

erfolgt über konfigurierbare Routing-Einträge im OD (*siehe Abbildung 1*) und ist in beiden Richtungen möglich.

Der CiA definiert, dass jeder CAN-Bus eine eigene Netzwerk-ID erhält. Jede CANopen-Instanz ist an ein CANopen-Netzwerk angeschlossen. Ein CANopen Gerät, das als SDO Client einen Zugriff auf ein anderes CANopen Gerät durchführen möchte, das sich in einem anderen CANopen-Netzwerk befindet, muss vor jedem SDO-Zugriff das SDO Network Indication Protocol an das SDO-Gateway senden (*siehe Abbildung 2*). Dies ist eine CAN-Nachricht in dem der SDO Client angibt, in welchem CANopen-Netzwerk sich das Zielgerät befindet. Das SDO-Gateway sucht in seiner SDO Routing-Tabelle nach der CANopen Netzwerk-ID. Wenn diese gefunden werden konnte, wird ein SDO-Kanal von einer Instanz zur anderen aufgebaut. Der folgende SDO-Zugriff, der (bezüglich der CAN-ID) an das SDO-Gateway gerichtet ist, wird nun über die andere Instanz in das andere CANopen-Netzwerk weitergeleitet, wobei sich die CAN-Identifier zwischen den CANopen-Netzwerken unterscheiden.

Es können mehrere SDO-Gateways kaskadiert werden. Sollen SDO-Zugriffe über mehrere SDO-Gateways geroutet werden, dann muss ein SDO Routing-Eintrag existieren, in dem die Node-ID des folgenden SDO-Routers hinterlegt ist (*siehe Abbildung 3*). Empfängt dabei das erste SDO-Gateway in der Kette bis zum SDO Server das SDO Network Indication Protocol, dann sendet dieses SDO-Gateway ebenfalls das SDO Network Inidcation Protocol zum nächsten SDO-Gateway, damit der SDO-Kanal bis zum Server freigeschaltet werden kann. Ein folgender SDO-Zugriff wird dann vom Client über mehrere SDO-Gateways bis zum Server ausgeführt.

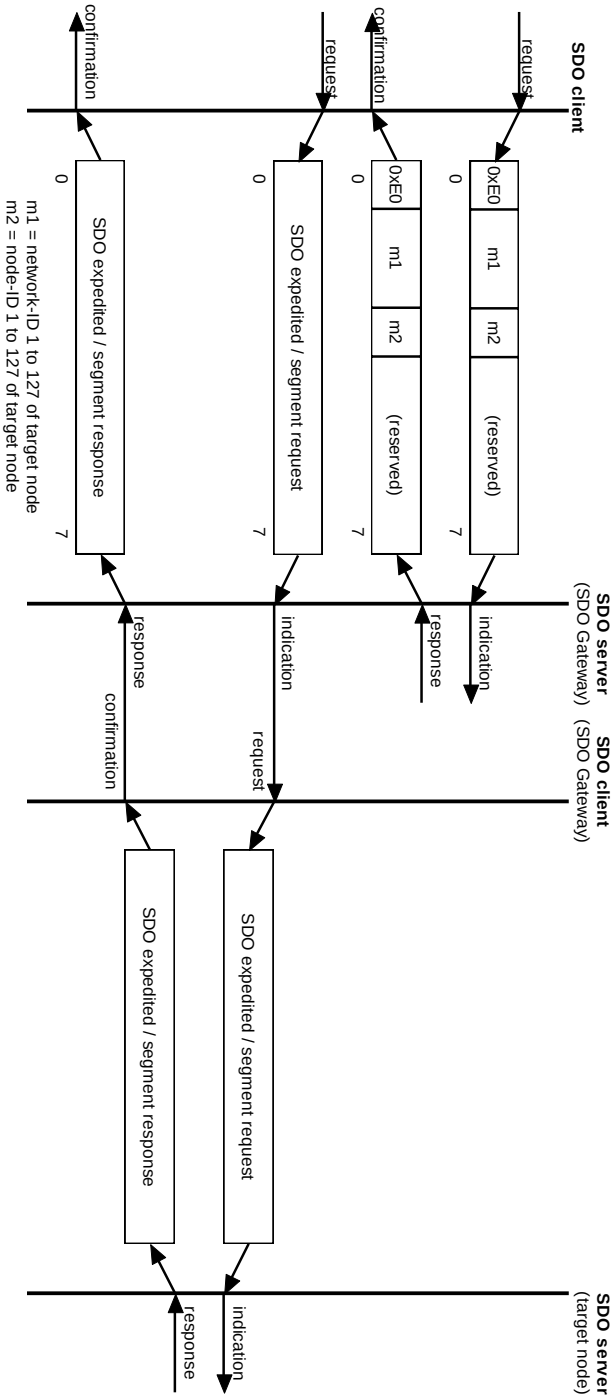

Abbildung 2: SDO Network Indication Protocol

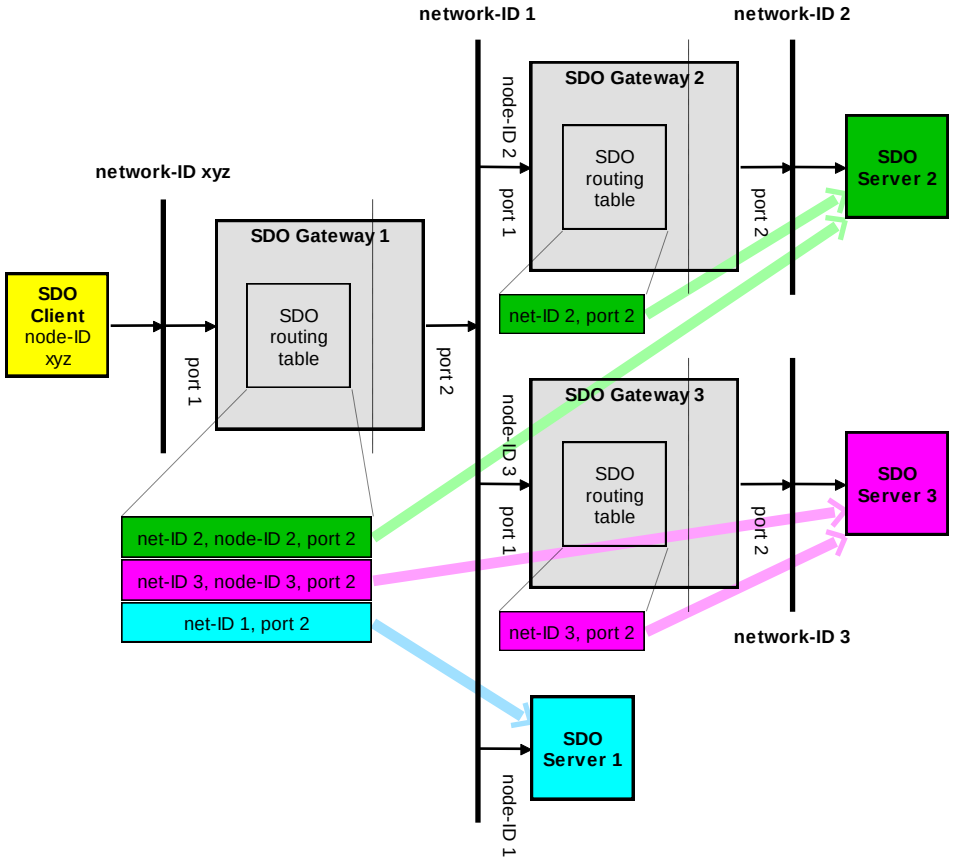

Abbildung 3: Beispiel einer Konfiguration mit mehreren SDO-Gateways

2.1 Einschränkungen des SDO-Gateway

Für das SDO-Gateway existieren folgende funktionelle Einschränkungen:

- In der aktuellen Version des SDO-Gateway werden nur SDO Expedited und Segmented Transfer unterstützt. Der SDO Blocktransfer nicht unterstützt.
- In der aktuellen Version arbeitet das SDO-Gateway nicht mit einem SDO-Manager zusammen. Die Vergabe der SDO-Clients und Server erfolgt statisch.

3 Installation des SDO-Gateway

Das Software-Modul für das SDO-Gateway wird über die Installation des CANopen Source Codes (SO-877) mitgeliefert. Setzen Sie bei der Installation des CANopen Source Codes die Komponente für die Installation der Erweiterung für das SDO-Gateway (SO-1078).

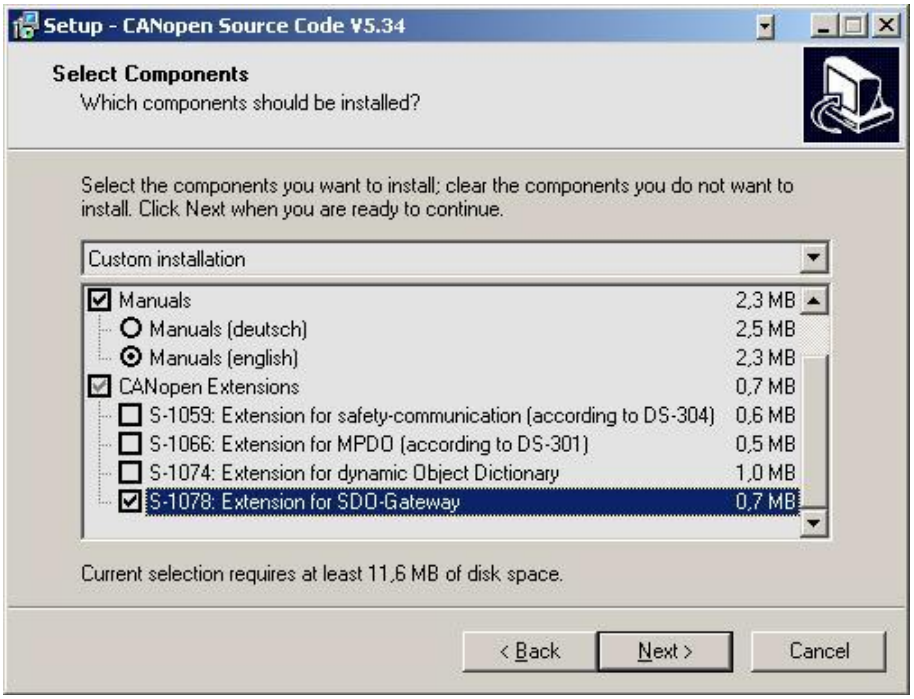

Abbildung 4: Installation des Software-Modul als Komponente

Während der Installation werden Sie nach dem Lizenzschlüssel für dieses Erweiterungsmodul gefragt (*siehe* *Abbildung 5*). Diesen Lizenzschlüssel erhalten Sie von der Firma SYS TEC electronic GmbH nach Bestellung und Bezahlung von SO-1078. War die Eingabe des Lizenzschlüssels korrekt, dann wird dieses Software-Modul zusätzlich zum CANopen Source Code in die gleiche Verzeichnisstruktur installiert.

Hatten Sie bereits den CANopen Stack SO-877 (ab Version V5.56) installiert und Sie haben nun das Erweiterungsmodul SO-1078 nachbestellt, dann können Sie dieses Erweiterungsmodul mit dem Aufruf von „C:\SYSTEC\COP.%Version%\Extension\SO-1078.exe“ nachinstallieren.

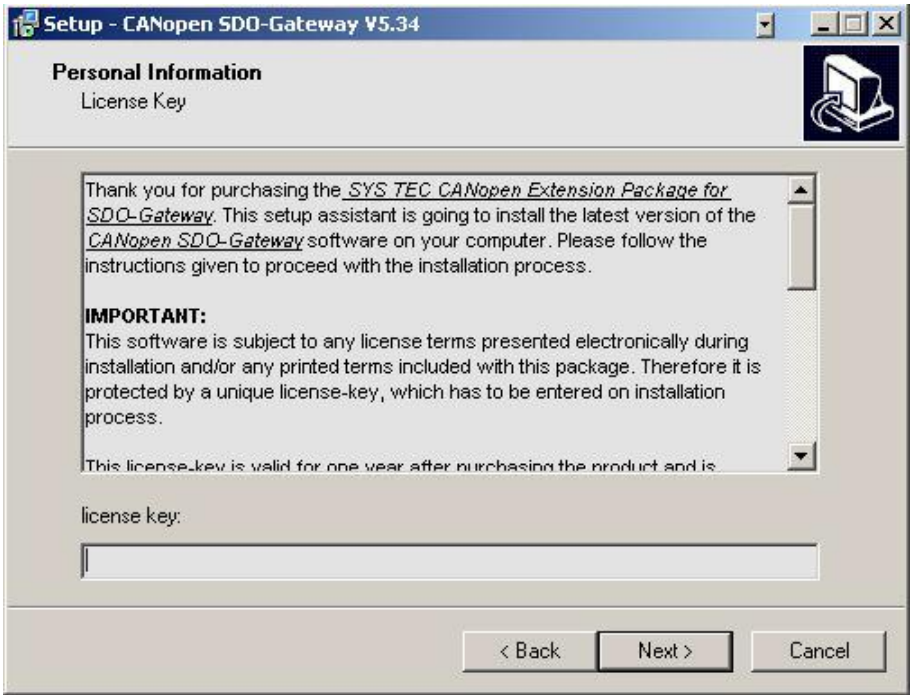

Abbildung 5: Eingabe des Lizenzschlüssels für das SDO-Gateway

Für die Installation unter dem Linux Betriebssystem benötigen Sie ein Windows-PC. Installieren Sie hier die Software-Pakete SO-877 und SO-1078 wie oben beschrieben. Setzen Sie dabei auch die Komponenten „ZIP: CANopen … (Unix format)“. Nach dieser Installation existieren im Verzeichnis „C:\SYSTEC\COP.%Version%.Linux“ die gepackten Dateien, die Sie auf Ihren Linux-PC transferieren und dort entpacken können.

4 Einbinden des SDO-Gateway

Das SDO-Gateway besteht aus einer einzigen Dateien: ***ccmsdor.c***. Um das SDO-Gateway nutzen zu können, muss die Datei ***ccmsdor.c*** dem Projekt hinzugefügt werden:

```
…\ccm\ccmsdor.c
```

Im Projekt müssen weiterhin mindestens folgende Dateien vorhanden sein (abgesehen vom CAN-Treiber, Target-Dateien, COB Modul, NMT Modul, OBD Modul und CCM Module):

```
…\copstack\sdoc.c
…\copstack\sdoscomm.c
```

In der Konstante CCM_MODULE_INTEGRATION in der Datei ***copcfg.h*** muss also mindestens das Bit 7 für den SDO Client gesetzt sein:

```
#define CCM_MODULE_INTEGRATION          (0x00000084L)
```

Das CANopen Projekt muss aus mindestens 2 CANopen Instanzen bestehen. Dazu müssen in der Datei ***copcfg.h*** folgende Konstanten angepasst werden:

```
#define COP_MAX_INSTANCES               2
#define CDRV_MAX_INSTANCES              2
```

Zusätzlich muss der SDO Router aktiviert werden, indem in der Datei ***copcfg.h*** die folgende Konstante mit TRUE angelegt wird:

```
#define SDO_USE_SDO_ROUTER              TRUE
```

4.1 Konfiguration des SDO-Gateway

Es gibt 3 weitere Konstanten für die Konfiguration des SDO-Gateway, die in die Datei ***copcfg.h*** hinzugefügt werden:

SDOR_MAX_ROUTING_ENTRIES:

Die Konstante SDOR_MAX_ROUTING_ENTRIES gibt an, wie viele Einträge die SDO Routing Tabelle beinhaltet. Sie benötigen jeweils einen Eintrag je Zielnetzwerk, in den ein SDO-Zugriff ausgeführt werden soll (*siehe dazu auch die Darstellung in Abbildung 3*). Da die SDO Routing-Einträge über das Objektverzeichnis im Objekt 0x1F2C konfigurierbar sind, gibt diese Konstante auch an, wie viele Subindizes im Objekt 0x1F2C maximal verwendet werden. Achten Sie bitte darauf, dass die Anzahl der Einträge (gezählt ab Subindex 1) des Objekts 0x1F2C mit dem Wert dieser Konstante korreliert. Die Konstante darf kleiner sein als die Anzahl der Subindizes, aber nicht größer! Da diese Konfiguration global gilt, gilt diese Einstellung auch für alle CANopen-Instanzen.

Wertebereich: 1 bis 254

Default-Wert: 1

```
#define SDOR_MAX_ROUTING_ENTRIES        2
```

SDOR_SDO_CLIENT_INDEX:

Die Konstante SDOR_SDO_CLIENT_INDEX gibt an, Welcher SDO Client Index aus dem OD der jeweils anderen CANopen Instanz verwendet werden soll, um einen SDO-Zugriff an das Zielgerät weiterzuleiten. Da diese Konfiguration global gilt, gilt diese Einstellung auch für alle CANopen-Instanzen.

Wertebereich: 0x1280 bis 0x12FF

Default-Wert: 0x1280

```
#define SDOR_SDO_CLIENT_INDEX        0x1281
```

SDOR_ROUTER_FORWARDING:

Die Konstante SDOR_ROUTER_FORWARDING gibt an, ob das SDO-Gateway auch in der Lage sein soll, SDO-Zugriffe über mehrere SDO-Gateways weiterzuleiten. Wie in *Abbildung 3* dargestellt, muss in diesem Fall im SDO Routing-Eintrag die Node-ID des folgenden SDO-Gateways eingetragen werden. Benötigt man diese Option nicht, dann wird mit der Einstellung FALSE, der Bedarf an Code-Speicher reduziert.

Wertebereich: FALSE / TRUE

Default-Wert: FALSE

```
#define SDOR_ROUTER_FORWARDING        TRUE
```

Die Konfiguration des SDO-Gateway über das Objektverzeichnis wird in *Kapitel 4.2* beschrieben.

4.2 Objekte für das SDO-Gateway

Im Objektverzeichnis werden neben den Objekten für SDO Client (0x1280 - 0x12FF) und evtl. auch Server (0x1200 – 0x127F) 4 weitere Objekte nach dem CiA-Standard 302 Part 7 /2/ benötigt. Das Objekt 0x1F2A dient zum Sperren der Schreibrechte auf das Objekt 0x1F2B. Über das Objekt 0x1F2B wird die eigene lokale Netzwerk-ID der entsprechenden CANopen-Instanz eingestellt. Die SDO Routing-Tabelle wird in Objekt 0x1F2C konfiguriert. Mit Objekt 0x1F2E wird anderen CANopen Geräten angezeigt, welche Funktionen das Gateway unterstützt.

Objekt 0x1F2A

Attribut	Wert
Index	0x1F2A
Name	Lock current configuration
Objekttyp	VAR
Datentyp	UNSIGNED32
Kategorie	Mandatory
Callback	CcmCbSdoRouterConfig()
Subindex	0x00
Zugriffsrechte	read-write
Wertebereich	0x65657266 = „free“ (Objekt 0x1F2C schreibbar) 0x6B636F6C = „lock“ (Objekt 0x1F2C gesperrt)
Default-Wert	0x65657266

Tabelle 1: Objekt 0x1F2A - Lock current configuration

Anlage dieses Objektes in der Datei ***objdict.h***:

```
OBD_BEGIN_INDEX_ROM(0x1F2A, 0x01, CcmCbSdoRouterConfig)
    OBD_SUBINDEX_RAM_VAR(0x1F2A, 0x00, kObdTypUInt32, kObdAccRW,
        tObdUnsigned32, LockCurrenConfiguration, 0x65657266)
OBD_END_INDEX(0x1F2A)
```

Objekt 0x1F2B

Attribut	Wert
Index	0x1F2B
Name	Local network-ID
Objekttyp	VAR
Datentyp	UNSIGNED16
Kategorie	Mandatory
Callback	CcmCbSdoRouterConfig()
Subindex	0x00
Zugriffsrechte	read-only (SDO); read-write (API)
Wertebereich	0 = ungültige Netzwerk-ID 0 - 127 = gültige Netzwerk-ID 128 - 65535 = reserviert
Default-Wert	Anwenderspezifisch

Tabelle 2: Objekt 0x1F2B - Local network-ID

Anlage dieses Objektes in der Datei ***objdict.h***:

```
OBD_BEGIN_INDEX_ROM(0x1F2B, 0x01, NULL)
    OBD_SUBINDEX_RAM_EXTVAR(0x1F2B, 0x00, kObdTypUInt16, kObdAccSR,
        tObdUnsigned16, CCM_LINK_OD_VARIABLE (OBD_LINKED_INSTANCE,
            m_wSdorLocalNetworkId), 0x0015)
OBD_END_INDEX(0x1F2B)
```

Objekt 0x1F2C

Attribut	Wert
Index	0x1F2C
Name	Remote network routing list
Objekttyp	ARRAY
Datentyp	UNSIGNED32
Kategorie	Mandatory
Callback	CcmCbSdoRouterConfig()
Subindex	0x00
Bedeutung	Highest sub-index supported
Zugriffsrechte	const
Wertebereich	0x01 bis 0xFE
Default-Wert	Anwenderspezifisch
Subindex	0x01 bis maximal 0xFE
Bedeutung	Routing n
Zugriffsrechte	read-write; read-only wenn 0x1F2A != 0x65657266
Wertebereich	*Siehe Abbildung 6*
Default-Wert	0x000000FF oder entsprechend vordefiniertes Routing.

Tabelle 3: Objekt 0x1F2C - Remote network routing list

Anlage dieses Objektes in der Datei ***objdict.h*** (als Beispiel):

```
OBD_BEGIN_INDEX_ROM(0x1F2C, 0x03, CcmCbSdoRouterConfig)
    OBD_SUBINDEX_ROM_VAR(0x1F2C, 0x00, kObdTypUInt8, kObdAccR,
        tObdUnsigned8, number_of_entries, 2)
```

```
    OBD_SUBINDEX_RAM_EXTVAR(0x1F2C, 0x01, kObdTypUInt32, kObdAccSRW,
        tObdUnsigned32, CCM_LINK_OD_ARRAY (OBD_LINKED_INSTANCE,
            m_adwSdorRoutingList, 0), 0x000000FF)
    OBD_SUBINDEX_RAM_EXTVAR(0x1F2C, 0x02, kObdTypUInt32, kObdAccSRW,
        tObdUnsigned32, CCM_LINK_OD_ARRAY (OBD_LINKED_INSTANCE,
            m_adwSdorRoutingList, 1), 0x000000FF)
OBD_END_INDEX(0x1F2C)
```

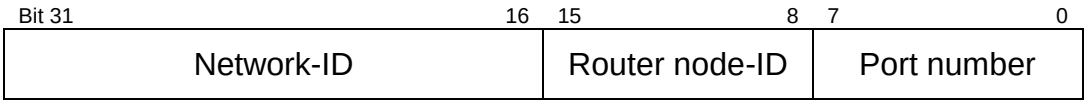

Abbildung 6: Objektstruktur für ein SDO Routing-Eintrag

Die ***Network-ID*** ist die ID, in der sich das Zielkgerät befindet. Sie muss einen Wert zwischen 1 und 127 besitzen. Die ***Router node-ID*** ist die Node-ID des folgenden SDO-Gateways, wenn der SDO-Zugriff über mehrere SDO-Gateways erfolgen muss. Hat diese ***Router node-ID*** den Wert 0, dann befindet sich das Zielgerät direkt in dem Netzwerk, der an der jeweiligen Portnummer angeschlossen ist. Die ***Port number*** ist ein Wert im Bereich 1 bis 32 und bezeichnet die jeweilige CANopen Instanz in die der SDO-Zugriff weitergeleitet werden soll. Der Wert 1 bezeichnet die erste CANopen Instanz (Instanz-Handle bzw. -Index 0), der Wert 2 bezeichnet die zweite CANopen Instanz (Instanz-Handle bzw. –Index 1), usw..

Objekt 0x1F2E

Attribut	Wert
Index	0x1F2E
Name	Functional elements and error state
Objekttyp	VAR
Datentyp	UNSIGNED8
Kategorie	Mandatory
Callback	NULL
Subindex	0x00
Zugriffsrechte	const
Wertebereich	0x80 (Bit 7 = Remote SDO FE implementiert)
Default-Wert	Anwenderspezifisch

Tabelle 4: Objekt 0x1F2E - Functional elements

Anlage dieses Objektes in der Datei ***objdict.h***:

```
OBD_BEGIN_INDEX_ROM(0x1F2E, 0x01, NULL)
    OBD_SUBINDEX_ROM_VAR(0x1F2E, 0x00, kObdTypUInt8, kObdAccR,
        tObdUnsigned8, FunctionalElements, 0x80)
OBD_END_INDEX(0x1F2E)
```

Achtung:

Abweichend vom CiA-Standard wird das Objekt 0x1F2E als „const“ definiert, da kein Routing von Emergency Nachrichten (Bit 6) und PDOs (Bit 5) unterstützt wird. Das Auftreten von Fehlern (Bit 4) wird in der Implementierung nicht in diesem Objekt abgelegt.

Hinweis:

In einem SDO-Gateway müssen die oben genannten Objekte in jeder CANopen Instanz angelegt sein. Ist es aber nicht erwünscht, dass von einer bestimmten CANopen Instanz aus SDO Routings eingestellt und ausgeführt werden, dann setzen Sie in dieser CANopen Instanz die Zugriffsrechte des Objekts 0x1F2C auf read-only oder den Default-Wert des Objektes 0x1F2A auf 0x6B636F6C („lock“).

Hinweis:

Der Einsatz des ODBuilders ist für den Flying Master (bzw. generell für den CANopen Manager) nicht möglich!

5 Änderungen im SDO Client Gerät

Der SDO Client, der einen SDO-Zugriff auf einen SDO Server in einem anderen CANopen Netzwerk starten soll, muss in der Lage sein, das SDO Network Indication Protocol zu senden. Andernfalls wird kein SDO-Kanal bis zum Server freigeschaltet.

Zunächst müssen im Projekt das SDOC Modul und dessen CCM-Datei eingebunden sein:

```
…\copstack\sdoc.c
…\ccm\ccmsdoc.c
```

In der Konstante CCM_MODULE_INTEGRATION in der Datei ***copcfg.h*** muss hier auch mindestens das Bit 7 für den SDO Client gesetzt sein:

```
#define CCM_MODULE_INTEGRATION          (0x00000084L)
```

In dieser Datei muss zusätzlich eine weitere Konstante mit dem Namen SDOC_USE_NETWORK_INIDICATION angelegt und auf TRUE gesetzt werden:

```
#define SDOC_USE_NETWORK_INIDICATION    TRUE
```

Im Objektverzeichnis wird nur das Objekt für den SDO Client benötigt (0x1280 – 0x12FF).

In der CANopen Applikation muss die Funktion ***CcmSdocIndicateNetwork()*** vor jedem SDO-Transfer gerufen werden (*siehe Kapitel 5.1 – und Abbildung 7*), der an einen SDO Server gerichtet ist, der sich in einem anderen CANopen Netzwerk befindet. Anschließend wird der SDO-Transfer mit der Funktion ***CcmSdocStartTransfer()*** gestartet (*siehe CANopen User Manual L-1020 /1/ im Kapitel „Funktion CcmSdocStartTransfer“*).

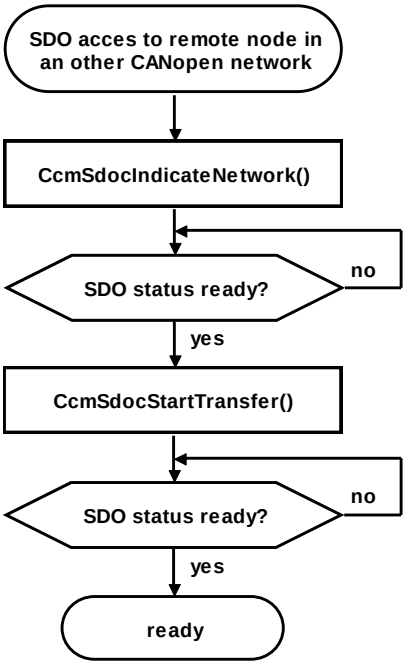

Abbildung 7: Ablauf bei einem SDO Zugriff über CANopen Netzwerke

5.1 Funktion CcmSdocIndicateNetwork

```
#include <cop.h>
tCopKernel PUBLIC CcmSdocIndicateNetwork (    CCM_DECL_INSTANCE_HDL_
                tSdocNetworkParam GENERIC* pSdoNetworkParam_p);
```

Parameter:

CCM_DECL_INSTANCE_HDL_:	Instanz-Handle
pSdoNetworkParam_p:	Pointer auf eine Strurktur vom Typ tSdocNetworkParam. Hier sind die Parameter für das SDO Network Indication Protocol abgelegt *(siehe Tabelle 5)*.

Return:

kCopSuccessful	0x00	Die Funktion wurde ohne Fehler ausgeführt.
kCopSdocInvalidParam	0x51	Ein einer Funktion übergebener Parameter ist ungültig (z.B. NULL für einen notwendigen Zeiger).

Weitere Fehlercodes siehe Funktion ***SdocIndicateNetwork()*** im *Kapitel 5.2*.

Beschreibung:

Die Funktion startet für den spezifizierten SDO Client das SDO Network Indication Protocol. Diese Funktion ist eine Wrapper-Funktion in der CCM-Schicht und ruft die Funktion ***SdocIndicateNetwork()*** auf.

Der Abschluss eines SDO-Transfers kann auf zwei Arten erkannt werden:

1. Dem Parameter ***m_pfnCbFinished*** in der Struktur ***tSdocNetworkParam*** wird ein Pointer auf eine SDO-Callback-Funktion zugewiesen. Diese Funktion wird nach Empfang der Antwort auf den *SDO Network Indication Protocol* oder nach einem SDO Abort aufgerufen, so dass die Applikation entsprechend reagieren kann.
2. Es wird zyklisch der Status des SDO Clients abgefragt. Hier muss der Parameter ***m_pfnCbFinished*** auf NULL gesetzt werden *(siehe Tabelle 5)*.

Der Aufbau der Parameterstruktur ***tSdocNetworkParam*** ist in Tabelle 5 beschrieben.

```
typedef struct _tagSdocNetworkParam
{
    WORD        m_wClientIndex;  // client index of the local OD
    WORD        m_wNetworkId;    // network-ID of the destination node
    BYTE        m_bNodeId;       // node-ID of the destination node
    tTime       m_TimeOut;       // timeout for the response from router

    tSdocCbFinished m_pfnCbFinished; // callback handler
}
tSdocNetworkParam;
```

Parameter	Bedeutung
m_wClientIndex	Objekt-Index des zu verwendenden SDO-Clients. Der Index muss sich in dem Bereich 0x1280 - 0x12FF befinden.
m_wNetworkId	Netzwerk-ID in der sich das Zielgerät befindet.
m_bNodeId	Node-ID des Zielgerätes.
m_TimeOut	Timeout für den SDO Client. Wird hier der Wert 0x00 übergeben, dann wird das beim Definieren des Clients bzw. durch einen vorher gestarteten Transfer eingestellte Timeout verwendet. Die Auflösung des Timeouts beträgt 100µs.
m_pfnCbFinished	Zeiger auf eine SDO-Callback-Funktion, die bei Abschluss des SDO Network Indication Protokolls (oder SDO-Abort) aufgerufen wird (*siehe CANopen User Manual L-1020 /1/ im Kapitel „Callback-Funktion für SDO-Transfer-Abschluss“*). Wird hier NULL angegeben, dann wird keine Callback-Funktion aufgerufen. Um den Abschluss des SDO Network Indication Protokolls zu erkennen, muss in diesem Fall zyklisch die Funktion ***CcmSdocGetState()*** aufgerufen werden (*siehe CANopen User Manual L-1020 /1/ im Kapitel „Funktion CcmSdocGetState“*).

Tabelle 5: Parameter der Struktur tSdocNetworkParam

5.2 Funktion SdocIndicateNetwork

```
#include <sdoc.h>
tCopKernel PUBLIC SdocIndicateNetwork (    MCO_DECL_INSTANCE_PTR_
                tSdocNetworkParam GENERIC* pSdoNetworkParam_p);
```

Parameter:

MCO_DECL_INSTANCE_PTR_:	Pointer auf die Instanz
pSdoNetworkParam_p:	Pointer auf eine Strurktur vom Typ tSdocNetworkParam. Hier sind die Parameter für das SDO Network Indication Protocol abgelegt *(siehe Tabelle 5)*.

Return:

kCopSuccessful	0x00	Die Funktion wurde ohne Fehler ausgeführt.
kCopIllegalInstance	0x01	Die parametrierte Instanz existiert nicht.
kCopSdocInvalidParam	0x51	Ein der Funktion übergebener Parameter ist ungültig (z.B. NULL für einen notwendigen Zeiger).
kCopSdocClientNotExist	0x52	Der parametrierte SDO-Client wurde nicht von der Applikation angelegt.
kCopSdocBusy	0x53	Der parametrierte SDO-Client ist gerade mit der Übertragung von Daten beschäftigt.

Es sind weitere Fehlercodes möglich (*siehe Funktion CobSend() in L-1020*).

Beschreibung:

Die Funktion startet für den spezifizierten SDO Client das *SDO Network Indication Protocol*. Für nähere Informationen über das Verhalten und über die Parameter lesen Sie das *Kapitel 5.1*.

6 Ein einfaches Demo des SDO-Gateway

Mit dem Software-Paket SO-1078 des SDO-Gateway wird eine einfache Demo mit 3 CANopen Geräten mitgeliefert. *Abbildung 8* zeigt eine schematische Darstellung des Geräteaufbaus.

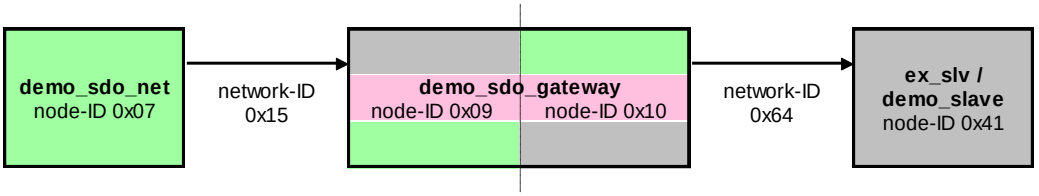

Abbildung 8: Schematische Darstellung des einfachen Demos

Die einzelnen CANopen Geräte können wahlweise als PC-Applikation (z.B. über ein SYS TEC USB-CANmodul) oder als Hardware-Platine (z.B. ein STM3210C-EVAL Board) realisiert werden. Die entsprechenden Projektdateien finden Sie z.B. in folgenden Verzeichnissen:

```
…\target\stm3210c-eval\no_os\keil.uv2\demo_sdo_gateway\
…\target\stm3210c-eval\no_os\keil.uv2\demo_sdo_net\
…\target\stm3210c-eval\no_os\keil.uv2\demo_slave\
…\target\x86\windows\vc9\demo_sdo_gateway\
…\target\x86\windows\vc9\demo_sdo_net\
…\target\x86\windows\vc9\demo_slave\
. . .
```

Übersetzen Sie die die entsprechenden Projekte in der Debug-Konfiguration, programmieren Sie sie gegebenenfalls in die entsprechende Hardware, verbinden Sie die Hardware über jeweils ein CAN-Kabel je CANopen Netzwerk (abgeschlossen mit den notwendigen Abschlusswiderständen von 120 Ohm an beiden Enden des CAN-Busses) und schalten Sie die Geräte ein.

Wenn Sie die Demos als PC-Applikation unter Windows mit USB-CANmodulen starten, dann benötigen Sie mindestens zwei USB-CANmodule mit der Gerätenummer 0 und ein USB-CANmodul mit der Gerätenummer 1. Statt zwei USB-CANmodule mit Gerätenummer 0 können Sie ein USB-CANmodul aber mit aktvierten USB-CANnetwork Treiber verwenden. Für das Demo ***demo_slave*** können Sie nach dem Start des Demo die CAN-Schnittstelle wählen.

Über die Ausgabe-Konsole (bzw. Terminal angeschlossen an der UART des Mikrocontrollers) können Sie im Demo ***demo_sdo_net*** den Erfolg oder Misserfolg der SDO-Übertragungen beobachten. Die Debug-Ausgaben des Demo ***demo_sdo_gateway*** informieren über den Zustand des SDO-Gateway.

Ausgaben bei Erfolg im Demo ***demo_sdo_net***:

```
device name of router node = 'CANopen SDO Gateway #0'
device name of remote node = 'CANopen Slave'
heartbeat time of remote node = 250 ms
```

Ausgaben bei Fehler der Übertragungen im Demo ***demo_sdo_net***:

```
ERROR code=0x0000 abort=0x05040000 state=0x03 for node=0x41 idx=0x1008 sub=0x00
ERROR code=0x0000 abort=0x05040000 state=0x02 for node=0x09 idx=0x1008 sub=0x00
```

Debug-Ausgaben im Demo ***demo_sdo_gateway***:

```
(0) CcmCbInitRouter(): net = 0x0064, node = 0x41
    -> network-ID has been found (0x00640002)
(0) CcmCbRemoteSdo():    msg-type = 0x40, abort = 0
(0) CcmCbSdocReceive(0): msg-type = 0x41
(0) CcmCbRemoteSdo():    msg-type = 0x60, abort = 0
(0) CcmCbSdocReceive(0): msg-type = 0x00
(0) CcmCbRemoteSdo():    msg-type = 0x70, abort = 0
(0) CcmCbSdocReceive(0): msg-type = 0x11
    -> SDO routing finished by a received client message
```

7 Hinweise für die Implementierung

Achten Sie auf die Konfigurationen der SDO Timeouts. Wenn der SDO Server auf den SDO-Zugriff nicht antworten kann (z.B. Software-Fehler, Hardware defekt, CAN-Kabel unterbrochen, …), dann wird das SDO-Gateway für die aktuelle SDO-Übertragung einen Timeout-Fehler feststellen und ihn an das SDO Client Gerät weiterleiten (*Node-ID 0x07 aus Abbildung 8*). Wegen der Software-Laufzeit und der Übertragungszeit auf dem CAN-Bus, wird die SDO-Abort-Nachricht verzögert beim SDO Client Gerät eintreffen. Ist der SDO Timeout Parameter beim SDO Client Gerät gleich groß oder sogar kleiner eingestellt, dann wird das SDO Client Gerät eine SDO-Abort-Nachricht noch vor dem SDO-Gateway senden. Unter Umständen senden sogar beide CANopen Knoten diesen SDO-Abort.

Um ein sauberes Verhalten zu erzielen, muss der SDO Timeout Parameter beim SDO Client Gerät größer eingestellt werden, als beim SDO-Gateway.

Werden mehrere SDO-Gateways hintereinander kaskadiert, dann sollten auch die SDO-Timeouts in den SDO-Gateways in Richtung des SDO Server Gerätes abnehmen (*siehe Abbildung 9*).

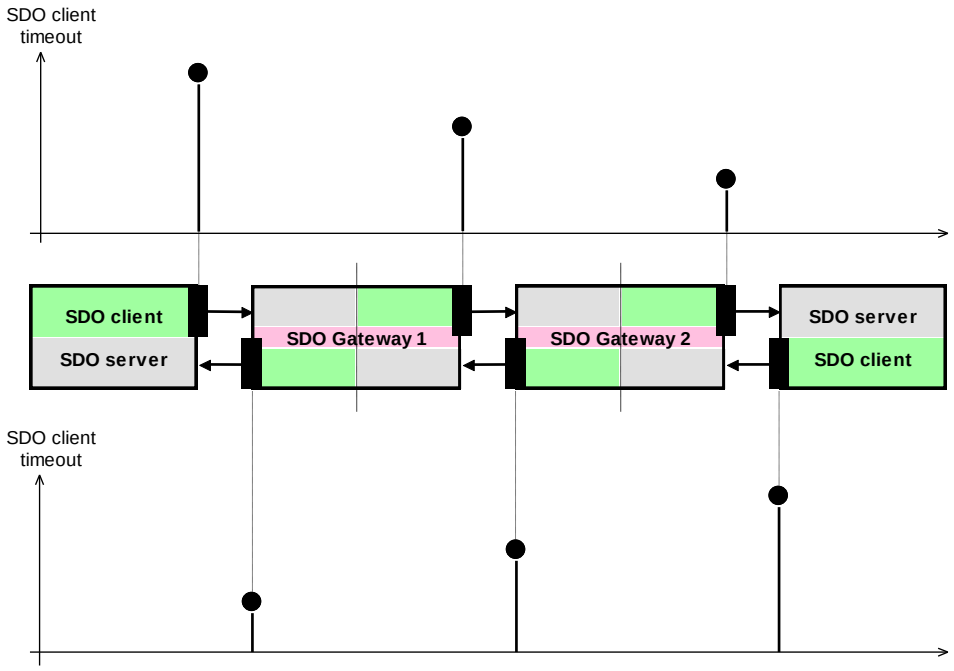

Abbildung 9: Konfiguration des SDO-Timeout Parameters

Nutzen Sie gegebenenfalls die Funktion ***CcmSdocDefineClientTab()*** (*siehe Kapitel „Funktion CcmSdocDefineClientTab“ in /1/*), um den SDO Timeout Parameter spezifisch für eine Instanz und SDO Client Index einzustellen, der abweichend vom Default-Timeout der Konstante SDOC_DEFAULT_TIMEOUT sein soll.

Indexverzeichnis

Dokument:	SDO-Gateway Add-on
Dokumentnummer:	L-1090d_01, Auflage September 2013

Wie würden Sie dieses Handbuch verbessern?

Haben Sie in diesem Handbuch Fehler entdeckt? Seite

Eingesandt von:

Kundennummer: _____

Name: _____

Firma: _____

Adresse: _____

Einsenden an:

SYS TEC electronic GmbH
Am Windrad 2
D-08468 Heinsdorfergrund
GERMANY
Fax : +49 3765 38600-4100

Veröffentlicht von

SYS TEC ELECTRONIC

Best.-Nr. L-1090d_01

Printed in Germany