SYS TEC ELECTRONIC

CiA 402 CANopen-Erweiterung

Software Manual

Auflage August 2008

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.

	EUROPA	NORDAMERIKA
Adresse:	SYS TEC electronic GmbH August-Bebel-Str. 29 D-07973 Greiz GERMANY	PHYTEC America LLC 203 Parfitt Way SW, Suite G100 Bainbridge Island, WA 98110 USA
Angebots-Hotline:	+49 (0) 36 61 / 62 79-0 info@systec-electronic.com	+1 (800) 278-9913 order@phytec.com
Technische Hotline:	+49 (0) 36 61 / 62 79-0 support@systec-electronic.com	+1 (800) 278-9913 support@phytec.com
Fax:	+49 (0) 36 61 / 6 79 99	+1 (206) 780-9135
Webseite:	http://www.systec-electronic.com	http://www.phytec.com

2. Auflage August 2008

1 Einleitung

Das Dokument beschreibt die CiA 402 Erweiterungen zum SYS TEC CANopen Stack. Die Software enthält die Implementierung eines Zustandsautomaten nach CiA 402 [1].
Es sind alle Mandatory- Funktionen und Objekte implementiert.
Die Erweiterung unterstützt eine oder mehrer Achse in einer CANopen Instanz.

2 Der CiA 402 Zustandsautomat

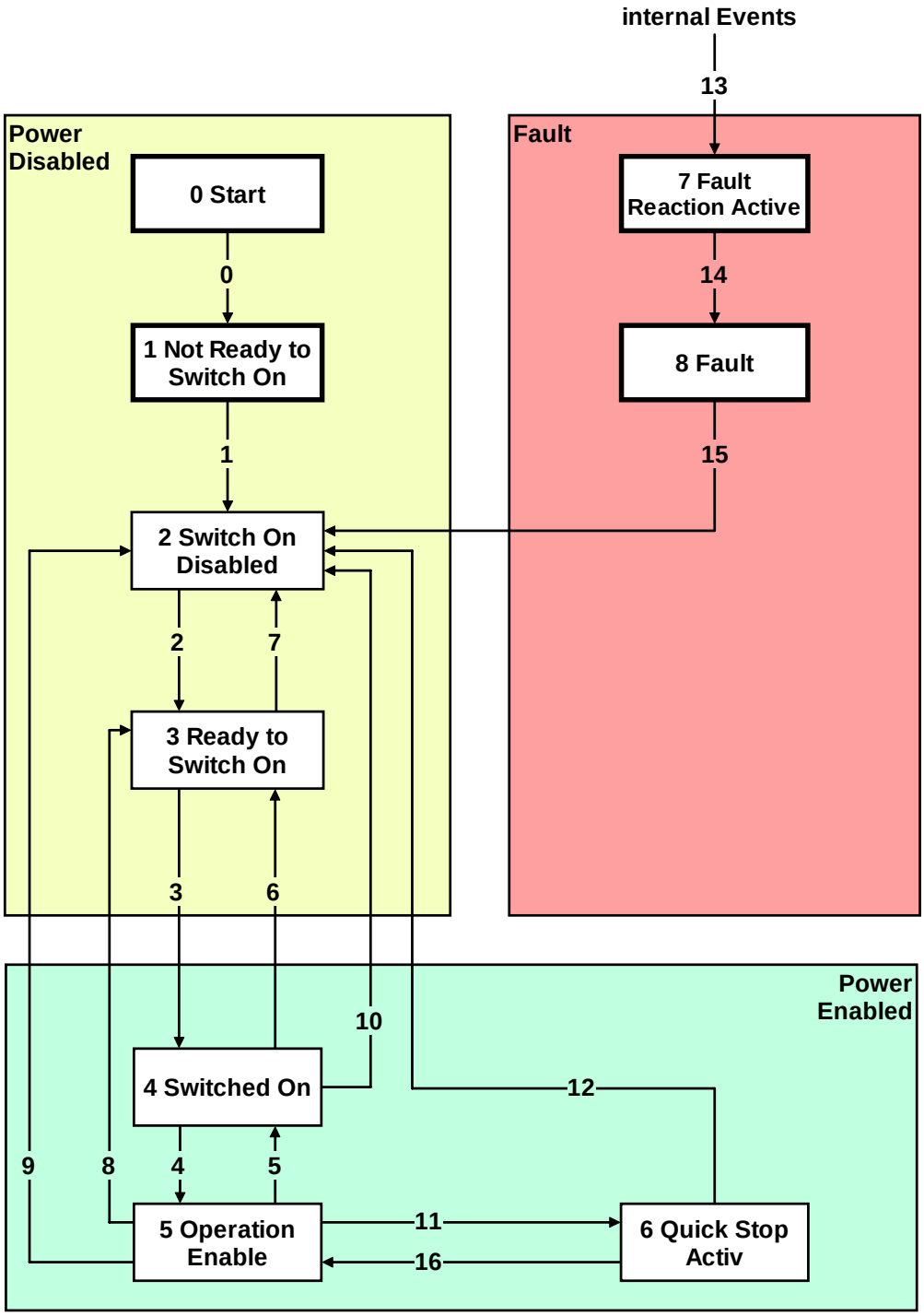

Abbildung 1: Zustandsautomat nach CiA 402, siehe [1]

3 API- Funktionen

Die CiA 402 Implementierung besteht aus folgenden Files:

`CCM\Ccm402St.c`	enthält den grundlegenden CiA 402 Zustandsautomat
`CCM\Ccm402Tp.c`	enthält Funktionstemplates, die der Anwender ausfüllen muss; dies sind Funktionen, die aus dem CiA 402 Zustandsautomat heraus gerufen werden müssen, um in der Applikation etwas zu bewirken.
`Include\Ccm402.h`	Funktionsprototypen, Definitionen und Konfiguration der CiA 402 Erweiterung
`Examples\ex_slv402.c`	Beispielapplikation, zeigt die Anwendung der CiA 402 Erweiterung
`Objdicts\DS402_1axis`	Beispiel eines Objektverzeichnis für eine Achse, incl. EDS-File und Projektdatei für den SYS TEC ODBuilder
`Objdicts\DS402_2axis`	Beispiel eines Objektverzeichnis für zwei Achsen, incl. EDS-File und Projektdatei für den SYS TEC ODBuilder

3.1 Konfiguration des CiA 402 Zustandsautomaten

Die Konfiguration erfolgt unabhängig vom restlichen CANopen Stack im File `Ccm402.h`.
Folgendes kann eingestellt werden:

1. Anzahl der verwendeten Antriebe (Achsen), lt. CiA 402 sind max. 8 Achsen einstellbar. Im Source- Beispiel werden bis zu zwei Antriebe unterstützt. Wird nur eine Achse verwendet, dann vereinfacht sich das Funktionsinterface der API- Funktionen und die Laufzeit erhöht sich. Die Einstellung erfolgt mit dem folgendem Define:

   ```
   #define CCM402_MAX_AXIS     1
   ```

   Hinweis: Werden mehrere Achsen betrieben, dann ist auch das Objektverzeichnis entsprechend anzupassen (z.B. mittels ODBuilder). Es stehen Beispiele für eine Achse (`\objdicts\DS402_1axis\`) und zwei Achsen (`\objdicts\DS402_2axis\`) zur Verfügung.
   Bei Verwendung mehrerer Achsen erhalten die Funktionen weitere Parameter, die mittels Makro `CCM402_DECL_AXIS` bzw. `CCM402_DECL_AXIS_` aktiviert werden können. Das Makro `COMMA` steuert die Verwendung mehrerer Achsen mit der Verwendung mehrerer Instanzen.

2. Auswahl der verwendeten Antriebsprofile.
   Profile, die nicht unterstützt werden sollen, können entsprechend abgeschaltet werden. Dazu stehen folgende Defines zur Verfügung:

   ```
   #define CCM402_USE_PROFILE_POSITION_MODE        FALSE
   #define CCM402_USE_VELOCITY_MODE                TRUE
   #define CCM402_USE_PROFILE_VELOCITY_MODE        TRUE
   #define CCM402_USE_TORQUE_PROFILE_MODE          TRUE
   #define CCM402_USE_HOMING_MODE                  FALSE
   #define CCM402_USE_INTERPOLATED_POSITION_MODE   FALSE
   ```

3.2 API- Funktionen

3.2.1 Funktion Ccm402Init

Syntax:

```
#include <ccm402.h>
COPDLLEXPORT tCopKernel PUBLIC
Ccm402Init (                    CCM_DECL_INSTANCE_HDL_
                                COMMA
                                CCM402_DECL_AXIS)
```

Parameter:

CCM_DECL_INSTANCE_HDL_:	Instanz-Handle
COMMA:	Makro zur Verwendung bei mehren Achsen und Instanzen
CCM402_DECL_AXIS:	Nummer der Achse, wenn mehr als eine Achse verwendet wird.

Return:

kCopSuccessful	0	Die Funktion wurde ohne Fehler ausgeführt

Beschreibung:
Die Funktion initialisiert den Zustandsautomat nach CiA 402 für den angegebenen Antrieb (Achse).

3.2.2 Funktion Ccm402SetState

Syntax:

```
#include <ccm402.h>
COPDLLEXPORT    void    PUBLIC
Ccm402SetState (                CCM_DECL_INSTANCE_HDL_
                                tDs402States NewState_p,
                                WORD MEM *pwStatusword_p
                                CCM402_DECL_AXIS_)
```

Parameter:

CCM_DECL_INSTANCE_HDL_:	Instanz-Handle
NewState_p:	CiA 402 State, der eingestellt werden soll.
*pwStatusword_p:	Adresse auf das Statusword des Antriebes. Diese wird entsprechend des neuen Status von der Funktion aktualisiert.
CCM402_DECL_AXIS_:	Nummer der Achse, wenn mehr als eine Achse verwendet wird.

Return:
ohne

Beschreibung:
Die Funktion setzt die CiA 402 Zustandsmaschine für die angegebene Achse in den neuen Sollzustand. Weiterhin wird in das Statusword (z.B. Objekt 0x6041 für die Achse 0) der neue Zustand eingetragen. Die Variable Statusword muss in der Applikation vorhanden sein und mit dem Objektverzeichnis verknüpft sein, siehe [3].

3.2.3 Funktion Ccm402GetState

Syntax:

```
#include <ccm402.h>
COPDLLEXPORT        tDs402States
PUBLIC Ccm402SetState (        CCM_DECL_INSTANCE_HDL_
                               COMMA
                               CCM402_DECL_AXIS)
```

Parameter:

CCM_DECL_INSTANCE_HDL_:	Instanz-Handle
COMMA:	Makro zur Verwendung bei mehren Achsen und Instanzen
CCM402_DECL_AXIS:	Nummer der Achse, wenn mehr als eine Achse verwendet wird.

Return:
tDs402States

Beschreibung:
Die Funktion gibt den aktuelle Status der CiA 402 Zustandsmaschine für die angegebene Achse zurück.

Folgende CiA 402 Stati sind definiert:

kDs402State_START	0	nach Power On oder Reset
kDs402State_NOT_READY_TO_SWITCH_ON	1	
kDs402State_SWITCH_ON_DISABLED	2	
kDs402State_READY_TO_SWITCH_ON	3	
kDs402State_SWITCHED_ON	4	
kDs402State_OPERATION_ENABLE	5	
kDs402State_QUICK_STOP_ACTIV	6	
kDs402State_FAULT_REACTION_ACTIV	7	
kDs402State_FAULT	8	

3.2.4 Funktion Ccm402Process

Syntax:

```
#include <ccm402.h>
COPDLLEXPORT tCopKernel  PUBLIC
Ccm402Process (                 CCM_DECL_INSTANCE_HDL_
                                WORD Controlword_p,
                                WORD MEM *pwStatusword_p
                                CCM402_DECL_AXIS_)
```

Parameter:

CCM_DECL_INSTANCE_HDL_:	Instanz-Handle
Controlword_p:	CiA 402 Steuerwort der entsprechenden Achse.
*pwStatusword_p:	Adresse auf das Statusword des Antriebes. Diese wird entsprechend des neuen Status von der Funktion aktualisiert.
CCM402_DECL_AXIS_:	Nummer der Achse, wenn mehr als eine Achse verwendet wird.

Return:

kCopSuccessful	0	Die Funktion wurde ohne Fehler ausgeführt

Beschreibung:

Diese Funktion wird zyklisch gerufen und führt die interne Bearbeitung des CiA 402 Zustandsautomat aus. Es werden die empfangenden Steuerworte ausgewertet, die Zustände entsprechend angepasst und Callback-Funktionen in die Applikation gerufen bzw. die Funktionen direkt gerufen.
Es erfolgt ein Alt-Neu- Vergleich des Steuerwortes am Anfang der Funktion, um Änderungen zu erkennen.

Was führt zum Schalter innerhalb des Zustandsautomaten?

1. Steuerwort (Controlword), Voraussetzung, der Antrieb befindet sich im „Remote- Mode“, d.h. er darf via CANopen gesteuert werden.

2. Event’s vom Antrieb,
   d.h. es wird eine Funktion benötigt, die den Zustandsautomat direkt weiterschalten kann (siehe Ccm402SetState()) , z.B. wenn der Antrieb ohne CANopen läuft (Remote- Mode ist deaktiviert)

   Es gibt Funktionen (z.B. Ccm402OperationEnableAllowed (..)), die bei einem Zustandswechsel via CANopen gerufen werden. Voraussetzung ist, das der „Remote- Mode“ aktiv ist und sich das Controlword geändert hat.
   Dadurch wird es möglich in Abhängigkeit des internen Zustandes des Antriebes das Weiterschalten des Zustandsautomation zu verhindern oder zu verzögern.

Das alte Controlword darf mit dem neuen nur überschrieben werden, wenn der Zustandsautomat die Änderungen auch ausgeführt hat, d.h. z.B. das der Antrieb mittels der Funktion den Wechsel nach „Operation enable“ erlaubt hat.

Das Statuswort wird mittels PDO direkt gesendet? ACHTUNG! Es ist denkbar, das es eine Zeitverzögerung zwischen dem Empfang eines Steuerwortes und der zugehörigen Reaktion/Auswirkung gibt.

3.2.5 Funktion Ccm402OperationModeChanged

Syntax:

```
#include <ccm402.h>
COPDLLEXPORT tCopKernel  PUBLIC
Ccm402OperationModeChanged (        CCM_DECL_INSTANCE_HDL
                                    tObdInteger8 bNewOperationMode_p
                                    CCM402_DECL_AXIS_)
```

Parameter:

CCM_DECL_INSTANCE_HDL:	Instanz-Handle
bNewOperationMode_p:	neu Betriebsmode, via CANopen empfangen
CCM402_DECL_AXIS_:	Nummer der Achse, wenn mehr als eine Achse verwendet wird.

Return:

kCopSuccessful	0	Die Funktion wurde ohne Fehler ausgeführt

Beschreibung:

Die Funktion zeigt an, das sich der Operation Mode geändert hat. Hier kann in der Applikation darauf reagiert werden. So ist es möglich z.B. ein TPDO bzw. RPDO zu verändern (siehe Abschnitt 1.1).

3.3 SDO Callback-Funktionen

SDO Callback-Funktionen werden nur gerufen, wenn ein Objekt mittels SDO gelesen oder geschrieben wird (siehe [3]).

Folgende Objekte erhalten eine Callback-Funktion für SDO- Zugriffe:

0x6040 Steuerwort (Controlword)
0x6060 Betriebsmode (operation mode)

Diese müssen im `objdict.h` eingetragen sein (z.B. mit dem ODbuilder). Die Callback-Funktionen sind Bestandteil des Files `Ccm402Tp.c` und müssen je nach Applikation vom Anwender angepasst werden.

3.3.1 Funktion Ccm402CbSetControlword

Syntax:

```
#include <ccm402.h>
COPDLLEXPORT tCopKernel  PUBLIC
Ccm402CbSetControlword (          CCM_DECL_INSTANCE_HDL_
                                  tObdCbParam MEM* pParam_p)
```

Parameter:

CCM_DECL_INSTANCE_HDL_:	Instanz-Handle
pParam_p:	Zeiger zur SDO Parameterstruktur, siehe [3]

Return:

kCopSuccessful	0	Die Funktion wurde ohne Fehler ausgeführt

Beschreibung:
Callback-Funktion für das Objekt 0x6040. In dieser Callback-Funktion kann z.B. gesteuert werden, das das Controlword nur verändert wird, wenn sich der Antrieb im „Remote- Mode“ befindet, d.h. via CANopen gesteuert werden darf.

3.3.2 Funktion Ccm402CbSetOperationMode

Syntax:

```
#include <ccm402.h>
COPDLLEXPORT tCopKernel PUBLIC
Ccm402CbSetOperationMode(       CCM_DECL_INSTANCE_HDL_
                                tObdCbParam MEM* pParam_p)
```

Parameter:

CCM_DECL_INSTANCE_HDL_:	Instanz-Handle
pParam_p:	Zeiger zur SDO Parameterstruktur, siehe [3]

Return:

kCopSuccessful	0	Die Funktion wurde ohne Fehler ausgeführt

Beschreibung:
Callback-Funktion für das Objekt 0x6060. Ist in Abhängigkeit der Applikation bei Wechsel des Betriebsmode auszufüllen.

3.4 RPDO- Callback-Funktionen

PDO- Callback-Funktionen werden nur gerufen, wenn ein Objekt mittels RPDO geschrieben wurde (siehe [3]).
Folgende Objekte erhalten eine Callback-Funktion für PDO- Zugriffe:

0x6040 Steuerwort (Controlword)
0x6060 Betriebsmode (operation mode)

Diese müssen in der Applikation definiert werden (siehe Bsp.: `ex_slv402.c`). Die Callback-Funktionen müssen je nach Applikation vom Anwender angepasst werden.
Beispiele:

3.4.1 Funktion AppCbVarControlword

Syntax:

```
#include <ccm402.h>
tCopKernel PUBLIC
AppCbVarControlword (                    void GENERIC * pArg_p)
```

Parameter:

pArg_p:	Zeiger auf ein Argument der Callback-Funktion, siehe [3]

Return:

kCopSuccessful	0	Die Funktion wurde ohne Fehler ausgeführt, ist von der Applikation zu setzen, siehe [3]

Beschreibung:
Callback-Funktion für das Objekt 0x6040. Das Controlword wird in Abhängigkeit des „Remote-Mode“ des Antriebes akzeptiert oder abgelehnt.

3.4.2 Funktion AppCbVarOperationMode

Syntax:

```
#include <ccm402.h>
tCopKernel PUBLIC
AppCbVarOperationMode(          void GENERIC * pArg_p)
```

Parameter:

pArg_p:	Zeiger auf ein Argument der Callback-Funktion, siehe [3]

Return:

kCopSuccessful	0	Die Funktion wurde ohne Fehler ausgeführt, ist von der Applikation zu setzen, siehe [3]

Beschreibung:

Callback-Funktion für das Objekt 0x6060. Bei Empfang der zugehörigen Variablen wird der Betriebsmode entsprechend umgeschaltet und dazu die Funktion Ccm402OperationModeChanged() gerufen.

3.5 Callback-Funktionen des CiA 402 Zustandsautomaten

Es gibt zwei Gruppen von Callback-Funktionen, der aus dem CiA 402 Zustandsautomat gerufen werden.

3.5.1 Einmaliger Aufruf einer Callback-Funktion vor dem Zustandswechsel

Diese Callback-Funktionen werden vor einem Zustandswechsel, der mittels Steuerwort via CANopen aktiviert wird, aufgerufen. Die Applikation kann des Zustandswechsel verhindern oder verzögern, indem der Return-Wert der Funktion FALSE gesetzt wird. Diese Funktionen befinden sich im File `Ccm402Tp.c` und können vom Anwender bei Bedarf erweitert werden.
Folgende Funktionen stehen zur Verfügung.

3.5.2 Funktion Ccm402SwitchOnDisabledAllowed

Syntax:

```
#include <ccm402.h>
CDRVDLLEXPORT BOOL PUBLIC
Ccm402SwitchOnDisabledAllowed(   CCM_DECL_INSTANCE_HDL
                                 COMMA
                                 CCM402_DECL_AXIS)
```

Parameter:

CCM_DECL_INSTANCE_HDL:	Instanz-Handle
COMMA:	Makro zur Verwendung bei mehren Achsen und Instanzen
CCM402_DECL_AXIS:	Nummer der Achse, wenn mehr als eine Achse verwendet wird.

Return:

TRUE	Wechsel nach SWITCH_ON_DISABLED durch die Applikation erlaubt
FALSE	Wechsel nach SWITCH_ON_DISABLED gesperrt

Beschreibung:
Aufruf beim Wechsel nach SWITCH_ON_DISABLED (Tansition 7, 9, 10, 12 oder 15). Die Applikation kann den Wechsel in den Status SWITCH_ON_DISABLED verhindern, durch den Returnwert FALSE.

3.5.3 Funktion Ccm402ReadyToSwitchOnAllowed

Syntax:

```
#include <ccm402.h>
CDRVDLLEXPORT BOOL PUBLIC
Ccm402ReadyToSwitchOnAllowed(   CCM_DECL_INSTANCE_HDL
                                COMMA
                                CCM402_DECL_AXIS)
```

Parameter:

CCM_DECL_INSTANCE_HDL:	Instanz-Handle
COMMA:	Makro zur Verwendung bei mehren Achsen und Instanzen
CCM402_DECL_AXIS:	Nummer der Achse, wenn mehr als eine Achse verwendet wird.

Return:

TRUE	Wechsel nach READY_TO_SWITCH_ON durch die Applikation erlaubt
FALSE	Wechsel nach READY_TO_SWITCH_ON gesperrt

Beschreibung:
Aufruf beim Wechsel nach READY_TO_SWITCH_ON (Tansition 2, 6 oder 8).
Die Applikation kann den Wechsel in den Status READY_TO_SWITCH_ON verhindern, durch den Returnwert FALSE.

3.5.4 Funktion Ccm402SwitchedOnAllowed

Syntax:

```
#include <ccm402.h>
CDRVDLLEXPORT BOOL PUBLIC
Ccm402SwitchedOnAllowed (       CCM_DECL_INSTANCE_HDL
                                COMMA
                                CCM402_DECL_AXIS)
```

Parameter:

CCM_DECL_INSTANCE_HDL:	Instanz-Handle
COMMA:	Makro zur Verwendung bei mehren Achsen und Instanzen
CCM402_DECL_AXIS:	Nummer der Achse, wenn mehr als eine Achse verwendet wird.

Return:

TRUE	Wechsel nach SWITCHED_ON durch die Applikation erlaubt
FALSE	Wechsel nach SWITCHED_ON gesperrt

Beschreibung:

Aufruf beim Wechsel nach SWITCHED_ON (Tansition 3 oder 5).
Die Applikation kann den Wechsel in den Status SWITCHED_ON verhindern, durch den Returnwert FALSE.

3.5.5 Funktion Ccm402QuickStopAllowed

Syntax:

```
#include <ccm402.h>
CDRVDLLEXPORT BOOL PUBLIC
Ccm402QuickStopAllowed (        CCM_DECL_INSTANCE_HDL
                                COMMA
                                CCM402_DECL_AXIS)
```

Parameter:

CCM_DECL_INSTANCE_HDL:	Instanz-Handle
COMMA:	Makro zur Verwendung bei mehren Achsen und Instanzen
CCM402_DECL_AXIS:	Nummer der Achse, wenn mehr als eine Achse verwendet wird.

Return:

TRUE	Wechsel nach QUICK_STOP_ACTIV durch die Applikation erlaubt
FALSE	Wechsel nach QUICK_STOP_ACTIV gesperrt

Beschreibung:

Aufruf beim Wechsel von nach QUICK_STOP_ACTIV (Tansition 11). Die Applikation kann den Wechsel in den Status QUICK_STOP_ACTIV verhindern, durch den Returnwert FALSE.

3.5.6 Funktion Ccm402OperationEnableAllowed

Syntax:

```
#include <ccm402.h>
CDRVDLLEXPORT BOOL PUBLIC
Ccm402OperationEnableAllowed(   CCM_DECL_INSTANCE_HDL
                                COMMA
                                CCM402_DECL_AXIS)
```

Parameter:

CCM_DECL_INSTANCE_HDL:	Instanz-Handle
COMMA:	Makro zur Verwendung bei mehren Achsen und Instanzen
CCM402_DECL_AXIS:	Nummer der Achse, wenn mehr als eine Achse verwendet wird.

Return:

TRUE	Wechsel nach OPERATION_ENABLE durch die Applikation erlaubt
FALSE	Wechsel nach OPERATION_ENABLE gesperrt

Beschreibung:

Aufruf beim Wechsel nach OPERATION_ENABLE (Tansition 4 oder 16). Die Applikation kann den Wechsel in den Status OPERATION_ENABEL verhindern, durch den Returnwert FALSE.

3.5.7 Wiederholter Aufruf einer Callback-Funktion in einem Zustand

Diese Callback-Funktionen werden bei jedem Aufruf der Funktion Ccm402Process() gerufen, wenn sich der CiA 402 Zustandsautomat in einem bestimmten Zustand befindet. Damit wird es möglich, bestimmte Kommandos z.B. betriebsmodeabhängig auszuführen. Diese Funktionen befinden sich im File `Ccm402Tp.c` und müssen vom Anwender erweitert werden.

3.5.7.1 Funktion Ccm402Operation

Syntax:

```
#include <ccm402.h>
CDRVDLLEXPORT void PUBLIC
Ccm402Operation(                CCM_DECL_INSTANCE_HDL_
                                WORD Controlword_p,
                                WORD MEM *pwStatusword_p
                                CCM402_DECL_AXIS_)
```

Parameter:

CCM_DECL_INSTANCE_HDL_:	Instanz-Handle
Controlword_p:	CiA 402 Steuerwort der entsprechenden Achse.
*pwStatusword_p:	Adresse auf das Statusword des Antriebes.
CCM402_DECL_AXIS_:	Nummer der Achse, wenn mehr als eine Achse verwendet wird.

Return:
ohne

Beschreibung:
Funktion wird zyklisch im Zustand OPERATION_ENABLE gerufen. Hierin müssen in Abhängigkeit des Betriebsmodus die Kommandos ausgeführt werden (z.B. starten oder stoppen der Verfahrbewegung usw.). Das Statusword ist entsprechend des Zustandes des Antriebs zu aktualisieren (z.B. Sollgeschwindigkeit erreicht, Position erreicht usw.).

3.5.7.2 Funktion Ccm402QuickStopOperation

Syntax:

```
#include <ccm402.h>
CDRVDLLEXPORT void PUBLIC
Ccm402QuickStopOperation(       CCM_DECL_INSTANCE_HDL_
                                WORD Controlword_p,
                                WORD MEM *pwStatusword_p
                                CCM402_DECL_AXIS_)
```

Parameter:

CCM_DECL_INSTANCE_HDL_:	Instanz-Handle
Controlword_p:	CiA 402 Steuerwort der entsprechenden Achse.
*pwStatusword_p:	Adresse auf das Statusword des Antriebes.
CCM402_DECL_AXIS_:	Nummer der Achse, wenn mehr als eine Achse verwendet wird.

Return:
ohne

Beschreibung:
Funktion wird zyklisch im Zustand QUICK_STOP_ACTIVE gerufen. Hier werden die Quick- Stopp Funktionen realisiert und das Ergebnis, bzw. der aktuelle Zustand der Funktion im Statuswort hinterlegt.

3.6 Betriebsmodi (Operation Mode)

Der CiA 402 kennt verschiedene Betriebsmodi, in denen ein Antrieb arbeiten kann. Diese werden über das Objekte 0x6060 eingestellt. Der aktuelle Betriebsmode, indem sich das Gerät befindet, steht im Objekt 0x6061. Es können lt. CiA 402 nicht mehrere Modi parallel betrieben werden. Ein Umschalten ist jedoch möglich. Das ist von den Möglichkeiten des Gerät abhängig.

Der Zustandsautomat greift stets auf die Objekte 0x6060 und 0x6061 zu, um den jeweiligen Mode zu aktivieren.

Der Betriebsmode kann von außen oder durch die Applikation mittels Objekt 0x6060 eingestellt werden.

Es sind auch herstellerspezifische Modi möglich.

Wert	**Mode**
-1 … -128	manufacture specific modes of operation
0	reserved
1	Profile Position Mode (pp)
2	Velocity Mode (vl)
3	Profile Velocity Mode (pv)
4	Torque Profile Mode (tq)
5	reserved
6	Homing Mode (hm)
7	interpolated Position Mode
8… 127	reserved

Tabelle 1: Operation Modes

Was passiert bei der Änderung des Operation Mode? Es wird in diesem Fall eine Callback-Funktion. in die Applikation gerufen. Darin können bei Bedarf Änderungen z.B. an der PDO- Konfiguration vorgenommen werden (z.B. mit der Funktion CcmDefinePdoTab(), siehe [3]).

4 Objekt-Verzeichnis

4.1 Kommunikationsparameter

Dieser Teil enthält Objekt, die unabhängig vom Profile des Antriebs sind (siehe [1] und [2]). Weiter Objekte können bei Bedarf z.B. mit dem SYS TEC ODBuilder entsprechend [4] angelegt werden.

Index, Subindex	Name	Daten typ	Zugriffs typ	Standard-wert	Kate-gorie
0x1000	Device Type	u32	const	0x0002 0192	M
0x1400	RPDO1, steuert den Zustandsautomat				M
0x1400,0	highest supported sub-index	u8	ro	2	
0x1400,1	COB-Id	u32	rw	0x200 + NodeID	
0x1400,2	transmission type	u8	ro	0xFF	
0x1402	RPDO3, steuert den Zustandsautomat “Profile Position Mode”				O
0x1402,0	highest supported sub-index	u8	ro	2	
0x1402,1	COB-Id	u32	rw	(0x8000 0400 or 0x400) + NodeID	
0x1402,2	transmission type	u8	ro	0xFF	
0x1403	RPDO4, steuert den Zustandsautomat “Profile Velocity Mode”				O
0x1403,0	highest supported sub-index	u8	ro	2	
0x1403,1	COB-Id	u32	rw	(0x8000 0500 or 0x500) + NodeID	
0x1403,2	transmission type	u8	ro	0xFF	
0x1404	RPDO5, steuert den Zustandsautomat “Profile Torque Mode”				O
0x1404,0	highest supported sub-index	u8	ro	2	
0x1404,1	COB-Id	u32	rw	0x8000 0000	
0x1404,2	transmission type	u8	ro	0xFF	
0x1405	RPDO6, steuert den Zustandsautomat “Velocity Mode”				O
0x1405,0	highest supported sub-index	u8	ro	2	
0x1405,1	COB-Id	u32	rw	0x8000 0000	
0x1405,2	transmission type	u8	ro	0xFF	
0x1600	RPDO1 mapping parameter				M
0x1600,0	highest supported sub-index	u8	ro	1	
0x1600,1	1st application object	u32	ro	0x6040 0010	
0x1602	RPDO3 mapping parameter				M || O
0x1602,0	highest supported sub-index	u8	ro	2	
0x1602,1	1st application object	u32	ro	0x6040 0010	

Index, Subindex	Name	Daten typ	Zugriffs typ	Standard-wert	Kate-gorie
0x1602,2	2nd application object	u32	ro	0x607A 0020	
0x1603	RPDO4 mapping parameter				M || O
0x1603,0	highest supported sub-index	u8	ro	2	
0x1603,1	1st application object	u32	ro	0x6040 0010	
0x1603,2	2nd application object	u32	ro	0x60FF 0020	
0x1604	RPDO5 mapping parameter				M || O
0x1604,0	highest supported sub-index	u8	ro	2	
0x1604,1	1st application object	u32	ro	0x6040 0010	
0x1604,2	2nd application object	u32	ro	0x6071 0010	
0x1605	RPDO6 mapping parameter				M || O
0x1605,0	highest supported sub-index	u8	ro	2	
0x1605,1	1st application object	u32	ro	0x6040 0010	
0x1605,2	2nd application object	u32	ro	0x6042 0010	
0x1800	TPDO1, Status des Zustandsautomaten				M
0x1800,0	highest supported sub-index	u8	ro	5	
0x1800,1	COB-Id	u32	rw	0x4000 0180 + NodeID	
0x1800,2	transmission type	u8	ro	0xFF	
0x1800,3	inhibit time	u16	ro || rw	0	
0x1800,5	event timer	u16	ro || rw	0	
0x1802	TPDO3, Status des Zustandsautomaten “Profile Position Mode”				O
0x1802,0	highest supported sub-index	u8	ro	5	
0x1802,1	COB-Id	u32	rw	(0x4000 0380 or 0xC000 0380) + NodeID	
0x1802,2	transmission type	u8	ro || rw	1	
0x1802,3	inhibit time	u16	ro || rw	0	
0x1802,5	event timer	u16	ro || rw	0	
0x1803	TPDO4, Status des Zustandsautomaten “Profile Velocity Mode”				O
0x1803,0	highest supported sub-index	u8	ro	5	
0x1803,1	COB-Id	u32	rw	(0x4000 0480 or 0xC000 0480) + NodeID	
0x1803,2	transmission type	u8	ro || rw	1	
0x1803,3	inhibit time	u16	ro || rw	0	
0x1803,5	event timer	u16	ro || rw	0	
0x1804	TPDO5, Status des Zustandsautomaten „Profile Torque Mode”				O
0x1804,0	highest supported sub-index	u8	ro	5	
0x1804,1	COB-Id	u32	rw	0xC000 0000	
0x1804,2	transmission type	u8	ro || rw	1	
0x1804,3	inhibit time	u16	ro || rw	0	
0x1804,5	event timer	u16	ro || rw	0	

Index, Subindex	Name	Daten typ	Zugriffs typ	Standard-wert	Kate-gorie
0x1805	TPDO6, Status des Zustandsautomaten „Velocity Mode”				O
0x1805,0	highest supported sub-index	u8	ro	5	
0x1805,1	COB-Id	u32	rw	0xC000 0000	
0x1805,2	transmission type	u8	ro || rw	1	
0x1805,3	inhibit time	u16	ro || rw	0	
0x1805,5	event timer	u16	ro || rw	0	
0x1A00	TPDO1 mapping parameter				M
0x1A00,0	highest supported sub-index	u8	ro	1	
0x1A00,1	1st application object	u32	ro	0x6041 0010	
0x1A02	TPDO3 mapping parameter				M || O
0x1A02,0	highest supported sub-index	u8	ro	2	
0x1A02,1	1st application object	u32	ro	0x6041 0010	
0x1A02,2	2nd application object	u32	ro	0x6064 0020	
0x1A03	TPDO4 mapping parameter				M || O
0x1A03,0	highest supported sub-index	u8	ro	2	
0x1A03,1	1st application object	u32	ro	0x6041 0010	
0x1A03,2	2nd application object	u32	ro	0x606C 0020	
0x1A04	TPDO5 mapping parameter				M || O
0x1A04,0	highest supported sub-index	u8	ro	2	
0x1A04,1	1st application object	u32	ro	0x6041 0010	
0x1A04,2	2nd application object	u32	ro	0x6077 0010	
0x1A05	TPDO6 mapping parameter				M || O
0x1A05,0	highest supported sub-index	u8	ro	2	
0x1A05,1	1st application object	u32	ro	0x6041 0010	
0x1A05,2	2nd application object	u32	ro	0x6044 0010	
0x6040	Controlword	u16	rw	0x0	M
0x6041	Statusword	u16	ro	0x0	M
0x6060	Modes of operation	i8	rw	siehe Tabelle 1	M
0x6061	Modes of operation display	i8	ro	siehe Tabelle 1	M

Tabelle 2: Objektverzeichnis, spezifisch für CiA 402, siehe [1]

4.2 Homing Mode

Folgende Objekte wurden für den Homing Mode
angelegt. Weiter Objekte können bei Bedarf z.B. mit dem SYS TEC ODBuilder angelegt werden.

Index, Subindex	Name	Daten-typ	Zugriffs typ	Standard wert	Kate-gorie
0x6098	Homing methode	i8	rw	0	M
0x6099	Homing speeds				M
0x6099,0	highest supported sub-index	u8	ro	2	M
0x6099,1	Speed during search for switch	u32	rw	0	M
0x6099,2	Speed during search for zero	u32	rw	0	M

Tabelle 3: Objekteinträge für den „Homing Mode“

4.3 Profile Position Mode

Folgende Objekte wurden für den Profile Position Mode angelegt. Weiter Objekte können bei Bedarf z.B. mit dem SYS TEC ODBuilder angelegt werden.

Index, Subindex	Name	Daten-typ	Zugriffs typ	Standard wert	Kate-gorie
0x6064	Position actual value, see TPDO3	i32	ro	no	M
0x607A	Target position, see RPDO3	i32	rw	no	M
0x6081	Profile velocity	u32	rw	no	M
0x6083	Profile acceleration	u32	rw	no	M
0x6084	Profile deceleration	u32	rw	no	O
0x6086	Motion profile type	i16	rw	0	M
0x6093	Position factor	u32	rw		M
0x6094	Velocity encoder factor	u32	rw		M
0x6095	Velocity factor 1	u32	rw		M
0x6097	Acceleration factor	u32	rw		M

Tabelle 4: Objekteinträge für den „Profile Position Mode“

4.4 Profile Velocity Mode

Folgende Objekte wurden für den Profile Velocity Mode angelegt. Weiter Objekte können bei Bedarf z.B. mit dem SYS TEC ODBuilder angelegt werden.

Index, Subindex	Name	Daten-typ	Zugriffs typ	Standard wert	Kate-gorie
0x6069	Velocity sensor actual value	i32	ro		M
0x606A	Sensor selection Code	i16	rw	0	O
0x606B	Velocity demand value,	i32	ro		M
0x606C	Velocity actual value, see TPDO4	i32	ro		M
0x6081	Profile Velocity	u32	rw	no	
0x6083	Profile acceleration	u32	rw	no	M
0x6084	Profile deceleration	u32	rw	no	O
0x6086	Motion profile type	i16	rw	0	M
0x60FF	Target Velocity, see RPDO4,	i32	rw		M

Tabelle 5: Objekteinträge für den „Profile Velocity Mode“

4.5 Profile Torque Mode

Folgende Objekte wurden für den Profile Torque Mode angelegt. Weiter Objekte können bei Bedarf z.B. mit dem SYS TEC ODBuilder angelegt werden.

Index, Subindex	Name	Daten-typ	Zugriffs typ	Standard wert	Kate-gorie
0x6071	Target torque, see RPDO5	i16	rw	0	M
0x6073	max current	u16	rw		O
0x6077	torque actual value	i16	ro		O
0x6087	Torque slope	u32	rw	0	M
0x6088	Torque profile type, gibt die Art der Rampe an, 0 lineare Rampe (trapezförmig) 1 sin² Rampe 0x8000.. 0xFFFF herstellerspezifisch	i16	rw	0	M

Tabelle 6: Objekteinträge für den „Profile Torque Mode“

4.6 Velocity Mode

Folgende Objekte wurden für den Velocity Mode angelegt. Weiter Objekte können bei Bedarf z.B. mit dem SYS TEC ODBuilder angelegt werden.

Index, Subindex	Name	Daten-typ	Zugriffs typ	Standard wert	Kate-gorie
0x6042	target velocity “Velocity Mode” (vl), see RPDO6	i16	rw		M
0x6043	velocity demand (vl)	i16	ro		M
0x6044	control effort (vl), see TPDO6,	i16	ro		M
0x6046	velocity min max amount Drehzahlbegrenzung	u32 array			M
0x6046,0	number of entries	u8	ro	2	M
0x6046,1	velocity min amount,	u32	ro	0	M
0x6046,2	velocity max amount	u32	rw	no	M
0x6048	velocity acceleration, Anstieg der Beschleunigungsrampe, = Delta Speed/Delta Time	record			M
0x6048,0	number of entries	u8	ro	2	M
0x6048,1	Delta Speed	u32	ro	1000	M
0x6048,2	Delta Time	u16	rw	no	M
0x6049	velocity deceleration Bremsrampe, = Delta Speed/Delta Time	record			M
0x6049,0	number of entries	u8	ro	2	M
0x6049,1	Delta Speed	u32	ro	1000	M
0x6049,2	Delta Time	u16	rw	no	M

Tabelle 7 Objekteinträge für den „Velocity Mode

4.7 Interpolated Position Mode

Für diesen Mode wurden keine zusätzlichen Objekte angelegt. Diese sind von der Applikation abhängig und es wurden im CiA 402 keine mandatory Objekte definiert. Bei Bedarf können Objekte z.B. mit dem SYS TEC ODBuilder definiert werden.

4.8 Herstellerspezifische Objekte, Beispielapplikation

Index, Subindex	Name	Daten typ	Zugriffs typ	Standard wert	Kate-gorie
0x5FFF	Aktivierung des „Remote Mode“ und Simulation von Fehlern	u8	rw	0	O

Das Objekt 0x5FFF ist kein Bestandteil des CiA 402 und muss in der eigentliche Applikation entfernt werden. Es dient im Beispiel `ex_slv402.c` zur Simulation der Fehlerzustände eines Antriebs und zum Aktivieren/ Deaktivieren des „Remote Mode“ (siehe [1]). Dadurch wird es möglich Events via SDO zu simulieren, die in der späteren Applikation direkt vom Antrieb beeinflusst werden (interne Events).
Folgende Events sind im File `ex_slv402.c` definiert und können via SDO geschrieben werden:

```
typedef enum
{
     kFaultReaction_Activ     = 1,
     kFaultReaction_Completed = 2,
     kChange_Remote           = 4
}
tAppDriveFault;
```

`kFaultReaction_Activ:`
simuliert eine Fehler und führt zum Wechsel im den Zustand „Fault Reaction Active“ (Tansition 13 in Abbildung 1).

`kFaultReaction_Completed:`
simuliert die Fehlerbehebung im Antrieb und für zum Wechsel im den Zustand „Fault“ (Tansition 14 in Abbildung 1).

`kChange_Remote:`
simuliert den Wechsel des Antriebes in den „Remote Mode“. Jedes Schreiben des Wertes via SDO wechselte des „Remote Mode“ (ON/OFF).
Hinweis: Im Grundzustand der Beispiel- Applikation befindet sich der Antrieb im „Remote Mode“ = OFF, d.h. er kann nicht via CAN mittels Controlword gesteuert werden.

5 Abkürzungen

hm	Homing Mode
i8	CANopen Datentyp INTEGER8
i16	CANopen Datentyp INTEGER16
i32	CANopen Datentyp INTEGER32
M	mandatory
O	optional
pp	Profile Position Mode
pv	Profile Velocity Mode
rw	read write
ro	read only
tq	Profile Torque Mode
u8	CANopen Datentyp UNSIGNED8
u16	CANopen Datentyp UNSIGNED16
u32	CANopen Datentyp UNSIGNED32
vl	Velocity Mode

6 Literatur

[1] CANopen Device Profile Drives and Motion Control, CiA DSP402, Version 2.0 26.July 2002, CAN in Automation e.V.

[2] CANopen Device Profile Drives and Motion Control, Errata 1, CiA DSP402, Version 0.9 ??.February 2005, CAN in Automation e.V.

[3] CANopen User Manual, Software Manual, SYS TEC electronic GmbH, Greiz, Doku-Nr.: L-1020

[4] CANopen Application Layer and Communication Profile, CiA DSP301, Version 4.1 21.February 2006, CAN in Automation e.V.

Index

Dokument:	CiA 402 CANopen- Erweiterung
Dokumentnummer:	L-1096d_2, Auflage August 2008

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
August-Bebel-Str. 29
D-07973 Greiz
GERMANY
Fax : +49 (0) 36 61 / 62 79 99

Veröffentlicht von

SYS TEC ELECTRONIC

Best.-Nr. L-1096d_2
Printed in Germany