SYS TEC
ELECTRONIC

CiA 402 Add-on

Software Manual

Edition August 2008

system house for distributed automation

In this manual are descriptions for copyrighted products which are not explicitly indicated as such. The absence of the trademark (©) symbol does not infer that a product is not protected. Additionally, registered patents and trademarks are similarly not expressly indicated in this manual

The information in this document has been carefully checked and is believed to be entirely reliable. However, SYS TEC electronic GmbH assumes no responsibility for any inaccuracies. SYS TEC electronic GmbH neither gives any guarantee nor accepts any liability whatsoever for consequential damages resulting from the use of this manual or its associated product. SYS TEC electronic GmbH reserves the right to alter the information contained herein without prior notification and accepts no responsibility for any damages which might result.

Additionally, SYS TEC electronic GmbH offers no guarantee nor accepts any liability for damages arising from the improper usage or improper installation of the hardware or software. SYS TEC electronic GmbH further reserves the right to alter the layout and/or design of the hardware without prior notification and accepts no liability for doing so.

	EUROPE	NORTH AMERICA
Address:	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
Ordering Information:	+49 (3661) 6279-0 info@systec-electronic.com	1 (800) 278-9913 info@phytec.com
Technical Support:	+49 (3661) 6279-0 support@systec-electronic.com	1 (800) 278-9913 support@phytec.com
Fax:	+49 (3661) 62 79 99	1 (206) 780-9135
Web Site:	http://www.systec-electronic.com	http://www.phytec.com

1st Edition August 2008

1 Introduction

This document describes the CiA 402 extenstion for the SYS TEC electronic GmbH CANopen Protocol Stack.
The software implements the state-machine according to CiA 402 [1] and contains all mandatory functions and objects. It supports single- and multi-axis operation.

2 The CiA 402 state machine

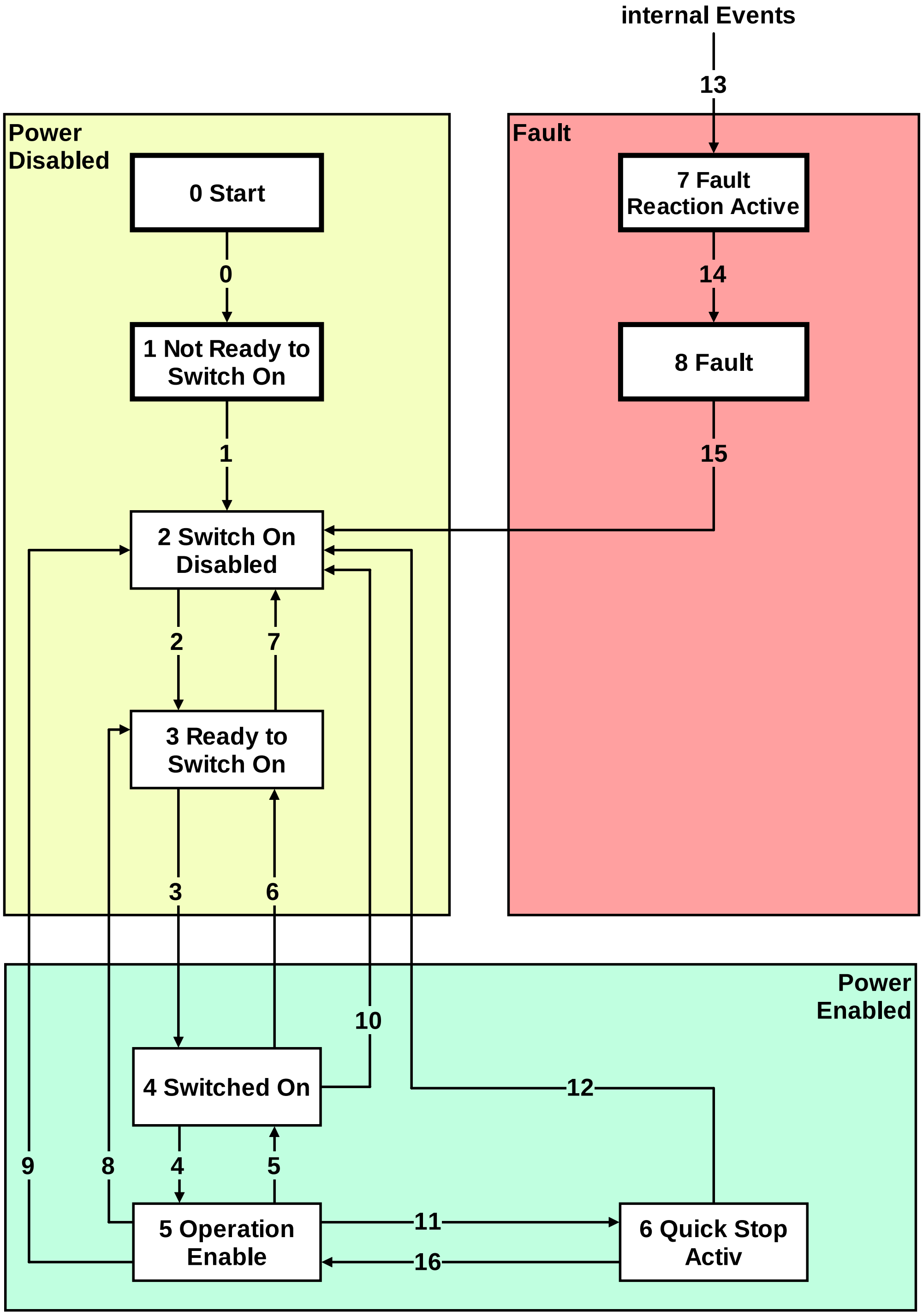

Figure 1: State-machine according to CiA 402 (see also [1])

3 API Description

The software contains the following files:

`CCM\Ccm402St.c` generic implementation of the CiA 402 state-machine.

`CCM\Ccm402Tp.c` contains function templates to be filled by the user. These functions are called from within the CiA 402 state-machine and act as an interface to the application (call-back functions).

`Include\Ccm402.h` Function prototypes and definitions of the CiA 402 extension.

`Examples\ex_slv402.c` Sample application to demonstrate the usage of the CiA 402 extension.

`Objdicts\DS402_1axis` Sample object dictionary for a single axis application. Includes the EDS file as well as the corresponding ODBilder project file.

`Objdicts\DS402_2axis` Sample object dictionary for a two-axis application. Includes the EDS file as well as the corresponding ODBilder project file.

3.1 Configuration of the CiA 402 state-machine

The configuration of the CiA 402 extension is independent from the configuration of the CANopen stack. The following configuration options are available in file `Ccm402.h`:

1. Number of axes (max. 8 according to CiA 402). The example source code demonstrate the use of two axes. In case only one axis is required by the user application the API get simplified and the runtime performance increases.

   ```
   #define CCM402_MAX_AXIS     1
   ```

   Note: The number of axis must match the object dictionary configuration. Hence, the object dictionary must support the number of axis configured with `CCM402_MAX_AXIS`.
   We provide example sources for single-axis and two-axis support. In case multiple axis are required, the API functions will require additional parameters which are activated by macro `CCM402_DECL_AXIS` and `CCM402_DECL_AXIS_`. The makro `COMMA` controls the usage of multiple axis in case multiple instances are used.

2. Selection of supported motion profiles.
   The user can deactivate not used profiles by setting the corresponding definition to `FALSE`.

   The following definitions are available:
   ```
   #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 Functions

3.2.1 Function Ccm402Init

Syntax:

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

Parameters:

CCM_DECL_INSTANCE_HDL_:	Instance handle
COMMA:	Macro used to control multiple axes and instances.
CCM402_DECL_AXIS:	Number of axis (if multiple axis are supported).

Returns:

kCopSuccessful	0	The function executed successfully.

Description:

This function initialises the CiA 402 state machine for the specified axis.

3.2.2 Function Ccm402SetState

Syntax:

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

Parameters:

CCM_DECL_INSTANCE_HDL_: Instance handle

NewState_p: The new CiA 402 state to be set.

*pwStatusword_p: Pointer to the status word of the axis to be updated with the new state.

CCM402_DECL_AXIS_: Number of axis (if multiple axis are supported).

Returns:

Nothing

Description:

This function will register a new state in the CiA 402 state-machine of the selected axis. Furthermore it registers the new state to the status word of the selected axis (e.g. object 6041h for axis 0). The status word must be defined in the application and connected to the corresponding object dictionary entry (*see also [3]*).

3.2.3 Function Ccm402GetState

Syntax:

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

Parameters:

CCM_DECL_INSTANCE_HDL_: Instance handle

COMMA: Macro used to control multiple axes and instances.

CCM402_DECL_AXIS: Number of axis (if multiple axis are supported).

Returns:

tDs402States

Description:

This function returns the current state of the selected axis. The following states are defined:

kDs402State_START	0	after Power-on or 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 Function Ccm402Process

Syntax:

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

Parameters:

CCM_DECL_INSTANCE_HDL_:	Instance handle
Controlword_p:	CiA 402 control word of the selected axis.
*pwStatusword_p:	Pointer to the status word of the selected axis to be updated.
CCM402_DECL_AXIS_:	Number of axis (if multiple axis are supported).

Returns:

kCopSuccessful	0	The function executed successfully.

Description:

This function processes the internals of the CiA 402 state-machine. It will process the control words, set the states accordingly and reference the registered call-back functions. The application must make sure to call this function cyclically. A old-state – new-state comparison is used to detect changes in the control word.

Which events may cause the state-machine to switch?

1. Control word change, pre-condition that the axis is in state “Remote-Mode” (it can be controlled via CANopen)

2. Events coming from the drive,
   i.e. this requires a function to directly switch the state-machine (*see Function Ccm402SetState*), .e.g. the drive is operated without CANopen running (Remote-Mode deactivated).
   There are functions (e.g. CDRVDLLEXPORT BOOL PUBLIC Ccm402OperationEnableAllowed() that are referenced by the state-machine whenever a state switch via CANopen occurs. Pre-condition, “Remote-Mode” is active and the control word has changed. This allows the application to bock or delay the switch if the internal state of the axis.

The old control word may be overwritten only after the state-machine has completed the state switch with its attached actions, i.e. the drive permits the change to “Operation enable” for example

The status word is sent via PDO directly. The application must be aware that there might be a time delay between the reception of the control word and the corresponding actions/reactions.

3.2.5 Function Ccm402OperationModeChanged

Syntax:

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

Parameters:

CCM_DECL_INSTANCE_HDL:	Instance handle
bNewOperationMode_p:	New operation mode received by CANopen
CCM402_DECL_AXIS_:	Number of axis (if multiple axis are supported).

Returns:

kCopSuccessful	0	The function executed successfully.

Description:

This function indicates if the Operation Mode has changed. This enables the application to react accordingly, e.g. to change a T/RPDO (*see also Section 4*)

3.3 SDO Callback Functions

SDO callback functions are called whenever a SDO read or write access to one of the following objects occurs (see also [3])

0x6040 Control Word
0x6060 Operation Mode

These objects also need to exist in file `objdict.h` (e.g. using the ODBuilder). The callback functions are implemented in file `Ccm402Tp.c` and the user need to modify them according to application requirements.

3.3.1 Function Ccm402CbSetControlword

Syntax:

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

Parameters:

CCM_DECL_INSTANCE_HDL_:	Instance handle
pParam_p:	Pointer to SDO parameter structure, see also [3]

Returns:

kCopSuccessful	0	The function executed successfully.

Description:
Callback function for object 0x6040. Within this function, for example, the application could implement that the control word is modified only if the drive is in operation mode “Remote-Node” , i.e. the drive is controlled via CANopen.

3.3.2 Function Ccm402CbSetOperationMode

Syntax:

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

Parameters:

CCM_DECL_INSTANCE_HDL_:	Instance handle
pParam_p:	Pointer to SDO parameter structure, see also [3]

Returns:

kCopSuccessful	0	The function executed successfully.

Description:

Callback function for object 0x6060. The user needs to populate this function according to application requirements.

3.4 PDO Callback Functions

PDO callback functions are called whenever a object was written via RPDO service (see also [3]).
The following objects feature a callback function for PDO access:

0x6040 Control word
0x6060 Operation mode

These callback functions need to be defined in the application (also refer to file `ex_slv402.c`). The user needs to adapt the callback functions according to application requirements.

3.4.1 Function AppCbVarControlword

Syntax:

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

Parameters:

pArg_p:	Pointer to argument of a callback function, see also [3]

Returns:

kCopSuccessful	0	The function executed successfully, to be set by the application, see also [3]

Description:

Callback function for object 0x6040. The control word is accepted or rejected depending on the operation mode. The user may modify the function according to application requirements.

3.4.2 Function AppCbVarOperationMode

Syntax:

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

Parameters:

pArg_p:	Pointer to argument of a callback function, see also [3]

Returns:

kCopSuccessful	0	The function executed successfully, to be set by the application, see also [3]

Description:

Callback function for object 0x6060. Upon reception of the corresponding variables the operaion mode is switched respectively and the function Ccm402OperationModeChanged() is referenced.

3.5 Callback functions of the CiA 402 state machine

The CiA 402 state machine references two kind of callback functions.

- Single-event referenced callback functions
- Cyclically referenced callback functions

3.5.1 Single-event referenced callback functions callded prior state change

Theses callback functions are referenced before executing a state change that was initiated via CANopen. The application can prevent or delay this state change by setting the return value to FALSE. Te functions are implemented in file `Ccm402Tp.c`. The user may modify the functions according to application requirements. The following sections describe the available functions.

3.5.1.1 Function Ccm402SwitchOnDisabledAllowed

Syntax:

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

Parameters:

CCM_DECL_INSTANCE_HDL:	Instance handle
COMMA:	Macro used to control multiple axes and instances.
CCM402_DECL_AXIS:	Number of axis (if multiple axis are supported).

Returns:

TRUE	Application allows state change to SWITCH_ON_DISABLED
FALSE	Application does not allow state change to SWITCH_ON_DISABLED

Description:
This callback function is called after a state change to state SWITCH_ON_DISABLED (Tansition 7, 9, 10, 12 or 15) was requested by CANopen. The application may reject or delay this state change by setting the return value to FALSE.

3.5.1.2 Function Ccm402ReadyToSwitchOnAllowed

Syntax:

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

Parameters:

CCM_DECL_INSTANCE_HDL:	Instance handle
COMMA:	Macro used to control multiple axes and instances.
CCM402_DECL_AXIS:	Number of axis (if multiple axis are supported).

Returns:

TRUE	Application allows state change to READY_TO_SWITCH_ON
FALSE	Application does not allow state change to READY_TO_SWITCH_ON

Description:
This callback function is called after a state change to state READY_TO_SWITCH_ON (Tansition 2, 6 or 8) was requested via CANopen. The application may reject or delay this state change by setting the return value to FALSE.

3.5.1.3 Function Ccm402SwitchedOnAllowed

Syntax:

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

Parameters:

CCM_DECL_INSTANCE_HDL:	Instance handle
COMMA:	Macro used to control multiple axes and instances.
CCM402_DECL_AXIS:	Number of axis (if multiple axis are supported).

Returns:

TRUE	Application allows state change to state SWITCHED_ON
FALSE	Application does not permit state change to SWITCHED_ON

Description:

This callback function is called after a state change request to state SWITCHED_ON (Tansition 3 or 5) was received via CANopen. The application may reject or delay this state change by setting the return value to FALSE.

3.5.1.4 Function Ccm402QuickStopAllowed

Syntax:

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

Parameters:

CCM_DECL_INSTANCE_HDL:	Instance handle
COMMA:	Macro used to control multiple axes and instances.
CCM402_DECL_AXIS:	Number of axis (if multiple axis are supported).

Returns:

TRUE	Application does not allow state change to QUICK_STOP_ACTIV
FALSE	Application does not permit state change to QUICK_STOP_ACTIV

Description:

This callback function is called upon reception of state change request to state QUICK_STOP_ACTIV (Tansition 11) via CANopen. The application may reject or delay this state change by setting the return value to FALSE.

3.5.1.5 Function Ccm402OperationEnableAllowed

Syntax:

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

Parameters:

CCM_DECL_INSTANCE_HDL:	Instance handle
COMMA:	Macro used to control multiple axes and instances.
CCM402_DECL_AXIS:	Number of axis (if multiple axis are supported).

Returns:

TRUE	Application allows state change to OPERATION_ENABLE
FALSE	Application does not permit state change OPERATION_ENABLE

Description:

This callback function is called upon reception of state change request to state OPERATION_ENABLE (Tansition 4 or 16) via CANopen. The application may reject or delay this state change by setting the return value to FALSE.

3.5.2 State-dependent cyclically referenced callback functions

These kind of state-dependent callback functions are referenced with every execution of function Ccm402Process() and the CiA 402 state machine is in certain states. This allows the application, for example, for executing application specific commands depending on the operation mode. These fuctions are implemented in file `Ccm402Tp.c` and can be adapted according to application requirements.

3.5.2.1 Function Ccm402Operation

Syntax:

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

Parameters:

CCM_DECL_INSTANCE_HDL_:	Instance handle
Controlword_p:	CiA 402 control word of the specified axis
*pwStatusword_p:	Pointer to status word variable
CCM402_DECL_AXIS_:	Number of axis (if multiple axis are supported).

Returns:

Nothing

Description:

This function is called cyclically in state OPERATION_ENABLE. Within this function the user-application must perform operation-mode specific commands (e.g. start/stop movement). The application needs to update the status word if necessary (e.g. demand speed/position reached).

3.5.2.2 Function Ccm402QuickStopOperation

Syntax:

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

Parameters:

CCM_DECL_INSTANCE_HDL_:	Instance handle
Controlword_p:	CiA 402 control word of the specified axis.
*pwStatusword_p:	Pointer to status word variable of the specified axis.
CCM402_DECL_AXIS_:	Number of axis (if multiple axis are supported).

Returns:

Nothing

Description:

This function is called cyclically in state QUICK_STOP_ACTIVE. Within this function the user-application needs to implement the quick stop functions and update the status word if necessary.

3.6 Operation Modes

CiA 402 defines different operation modes for a drive. Remote CANopen devices and the application may change the operation mode by writing the new operation mode to object 0x6060. Remote CANopen devices and the application may request the current operation mode by reading object 0x6061. The operation modes available depend on the drive and therefore on the user-application respectively. The CiA 402 state machine will always use object 0x6060 and 0x6061 for activating/altering the operation mode. Manufacturer-specific modes are supported.

The following modes are defined:

Value	Operation 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

Table 1: Operation Modes

What happens if the operation mode was requested to change?
The corresponding callback functions are called (*see also Sections 3.3 3.4 and 3.5*). Within these callback function the user may define application-specific actions (e.g. changing PDO configuration).

4 Object Dictionary

4.1 Communication parameters

The following section describes profile-independent part of the CiA 402 object dictionary definitions (*see also [1],[2]*). The user may create additional objects as required by the application (e.g. using the ODBuilder, [4]).

Index, Subindex	Name	Data type	Acces type	Default value	Cate-gory
0x1000	Device Type	u32	const	0x00020192	M
…					
0x1400	RPDO1, controls the state machine				M
0x1400,0	highest supported sub-index	u8	ro	2	
0x1400,1	COB-Id	u32	rw	0x200 + node-ID	
0x1400,2	transmission type	u8	ro	0xFF	
0x1402	RPDO3, controls the state machine in “Profile Position Mode”				O
0x1402,0	highest supported sub-index	u8	ro	2	
0x1402,1	COB-Id	u32	rw	(0x80000400 or 0x400)+node-ID	
0x1402,2	transmission type	u8	ro	FFh	
0x1403	RPDO4, controls the state machine in “Profile Velocity Mode”				O
0x1403,0	highest supported sub-index	u8	ro	2	
0x1403,1	COB-Id	u32	rw	(0x80000500 or 0x500)+node-ID	
0x1403,2	transmission type	u8	ro	FFh	
0x1404	RPDO5, controls the state machine in “Profile Torque Mode”				O
0x1404,0	highest supported sub-index	u8	ro	2	
0x1404,1	COB-Id	u32	rw	0x80000000	
0x1404,2	transmission type	u8	ro	0xFF	
0x1405	RPDO6, controls the state machine in “Velocity Mode”				O
0x1405,0	highest supported sub-index	u8	ro	2	
0x1405,1	COB-Id	u32	rw	0x80000000	
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	0x60400010	
0x1602	RPDO3 mapping parameter				M || O
0x1602,0	highest supported sub-index	u8	ro	2	
0x1602,1	1st application object	u32	ro	0x60400010	
0x1602,2	2nd application object	u32	ro	0x607A0020	
0x1603	RPDO4 mapping parameter				M || O

Index, Subindex	**Name**	**Data type**	**Acces type**	**Default value**	**Cate-gory**
0x1603,0	highest supported sub-index	u8	ro	2	
0x1603,1	1st application object	u32	ro	0x60400010	
0x1603,2	2nd application object	u32	ro	0x60FF0020	
0x1604	RPDO5 mapping parameter				M || O
0x1604,0	highest supported sub-index	u8	ro	2	
0x1604,1	1st application object	u32	ro	0x60400010	
0x1604,2	2nd application object	u32	ro	0x60710010	
0x1605	RPDO6 mapping parameter				M || O
0x1605,0	highest supported sub-index	u8	ro	2	
0x1605,1	1st application object	u32	ro	0x60400010	
0x1605,2	2nd application object	u32	ro	0x60420010	
0x1800	TPDO1, Status of state machine				M
0x1800,0	highest supported sub-index	u8	ro	5	
0x1800,1	COB-Id	u32	rw	0x40000180 + node-ID	
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 of state machine in “Profile Position Mode”				O
0x1802,0	highest supported sub-index	u8	ro	5	
0x1802,1	COB-Id	u32	rw	(0x40000380 or 0xC0000380) + node-ID	
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 of state machine in “Profile Velocity Mode”				O
0x1803,0	highest supported sub-index	u8	ro	5	
0x1803,1	COB-Id	u32	rw	(0x40000480 or 0xC0000480) + node-ID	
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 of state machine in „Profile Torque Mode”				O
0x1804,0	highest supported sub-index	u8	ro	5	
0x1804,1	COB-Id	u32	rw	0xC0000000	
0x1804,2	transmission type	u8	ro || rw	1	
0x1804,3	inhibit time	u16	ro || rw	0	
0x1804,5	event timer	u16	ro || rw	0	
0x1805	TPDO6, Status of state machine in „Velocity Mode”				O
0x1805,0	highest supported sub-index	u8	ro	5	
0x1805,1	COB-Id	u32	rw	0xC0000000	
0x1805,2	transmission type	u8	ro || rw	1	
0x1805,3	inhibit time	u16	ro || rw	0	

Index, Subindex	**Name**	**Data type**	**Acces type**	**Default value**	**Cate-gory**
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	0x60410010	
0x1A02	TPDO3 mapping parameter				M || O
0x1A02,0	highest supported sub-index	u8	ro	2	
0x1A02,1	1st application object	u32	ro	0x60410010	
0x1A02,2	2nd application object	u32	ro	0x60640020	
0x1A03	TPDO4 mapping parameter				M || O
0x1A03,0	highest supported sub-index	u8	ro	2	
0x1A03,1	1st application object	u32	ro	0x60410010	
0x1A03,2	2nd application object	u32	ro	0x606C0020	
0x1A04	TPDO5 mapping parameter				M || O
0x1A04,0	highest supported sub-index	u8	ro	2	
0x1A04,1	1st application object	u32	ro	0x60410010	
0x1A04,2	2nd application object	u32	ro	0x60770010	
0x1A05	TPDO6 mapping parameter				M || O
0x1A05,0	highest supported sub-index	u8	ro	2	
0x1A05,1	1st application object	u32	ro	0x60410010	
0x1A05,2	2nd application object	u32	ro	0x60440010	
0x6040	Controlword	u16	rw	0	M
0x6041	Statusword	u16	ro	0	M
0x6060	Operation mode	i8	rw	See Table 1	M
0x6061	Operation display	i8	ro	See Table 1	M

Table 2: CiA 402 specific object dictionary see also [1]

4.2 Homing Mode

The following objects are available for “Homing Mode”. The user may define additional objects as required by the application using the ODBuilder for example.

Index, Subindex	**Name**	**Data type**	**Acces type**	**Default value**	**Cate-gory**
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

Table 3: Object dictionary entries for "Homing Mode"

4.3 Profile Position Mode

The following objects are available for Profile Position Mode. The user may define additional objects as required by the application using the ODBuilder for example.

Index, Subindex	Name	Data type	Acces type	Default value	Cate-gory
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

Table 4: Object dictionary entries for “Profile Position Mode”

4.4 Profile Velocity Mode

The following objects are available for Profile Velocity Mode . The user may define additional objects as required by the application using the ODBuilder for example.

Index, Subindex	Name	Data type	Acces type	Default value	Cate-gory
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

Table 5: Object dictionary entries for “Profile Velocity Mode”

4.5 Profile Torque Mode

The following objects are available for Profile Torque Mode. The user may define additional objects as required by the application using the ODBuilder for example.

Index, Subindex	Name	Data type	Acces type	Default value	Cate-gory
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

Table 6: Object dictionary entries for "Profile Torque Mode"

4.6 Velocity Mode

The following objects are available for Velocity Mode. The user may define additional objects as required by the application using the ODBuilder for example.

Index, Subindex	Name	Data type	Acces type	Default value	Cate-gory
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

Table 7: Object dictionary entries for “Velocity Mode”

4.7 Interpolated Position Mode

There are no specific objects for this mode as they are application-specific and there are nor mandatory objects defined in CiA 402. However, the user may define additional objects if required by the application, using the ODBuilder for example.

4.8 Manufacturer-specific Objects, Example application

Index, Subindex	Name	Data type	Acces type	Default value	Cate-gory
0x5FFF	Activate “Remote Mode” and simulates errors	u8	rw	0	O

The above described object 0x5FFF is an application-specific object (`ex_slv402.c`) used for simulating various error conditions of a drive. It is also used to activate the Remote Mode (see also [1]).
This object is not part of CiA 402. It is for testing purpose only and should be removed in final applications.
This object enables to simulate events remotely via SDO access if no real application is present and therefore allows to trigger state machine transmissions. Of course, these events later need to be set/implemented by the application.
The following events are defined in file (`ex_slv402.c`) and are available via SDO:

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

`kFaultReaction_Activ:`
simulates an error that causes the state machine to transit to state „Fault Reaction Active“ (Tansition 13 in Figure 1).

`kFaultReaction_Completed:`
simulates the error reaction within a drive and causes the state machine to transit to state „Fault“ (Tansition 14 in Figure 1).

`kChange_Remote:`
simulates the state transition to state „Remote Mode“. Every time this value is written to Object 0x5FFF the “Remote Mode” toggles (ON/OFF).

Note: After power-on and reset the Remote Mode is set to OFF and the application cannot be controlled using the control word. If this is required the Remote Mode must be enabled first.

5 Abbreviations

hm	Homing Mode
i8	CANopen data type INTEGER8
i16	CANopen data type INTEGER16
i32	CANopen data type 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 data type UNSIGNED8
u16	CANopen data type UNSIGNED16
u32	CANopen data type UNSIGNED32
vl	Velocity Mode

6 References

[1] CANopen Device Profile Drives and Motion Control, CiA 402, Version 2.0 26.July 2002, CAN in Automation e.V.

[2] CANopen Device Profile Drives and Motion Control, Errata 1, CiA 402, 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

Document:	CiA 402 Add-on
Document number:	L-1096e_1, Edition August 2008

How would you improve this manual?

Did you find any mistakes in this manual? page

Submitted by:

Customer number: ____________________

Name: ____________________

Company: ____________________

Address: ____________________

Return to:

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

Published by

Ordering No. L-1096e_1
Printed in Germany