emtas™

Your embedded solution partner.

<< Reference Manual <<

CANopen Profile

CANopen®

CANopen Profile 401

2.7.3

Generated by Doxygen 1.8.11

Contents

Chapter 1

CANopen Stack p401 Modul

This a complete implementation of the CiA device profile for generic IO devices. Nevertheless not all optional objects and there behaviour are supported. The current implementation lacks for example floating point object for analog IO. Digital process IO is done via 8-bit ports, 1bit, 16-, and 32bit are not supported. Analog process IO is 16bit wide. 8- and 32bit are not supported (yet).

To use the functionality provided in the 401 module, It first has to be initialized calling co401Init(), which typically is done after calling coCanOpenStackInit(). The next step is telling the module which functions are provided to server the real hardware by registering hardware access functions using coEventRegister_401(). Once the hardware can be accessed by the module, all output ports have to be set up, whether with the default value defined in the object dictionary by calling co401DigOutDef(void) and co401AnOutDef(void) or with the error value defined in the object dictionary by calling co401DigOutErr(void) and co401AnOutErr(void);

Once this is done, the co401Task() must be called regularly and cyclic.

If currently not implemented objects are essential for your project, please contact service@emtas.de

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

File Documentation

3.1 co_p401.c File Reference

device profile implementation according CiA 401

Functions

- RET_T coEventRegister_401 (CO_EVENT_401_DI_T pDI, CO_EVENT_401_DO_T pDO, CO_EVENT_↩ 401_AI_T pAI, CO_EVENT_401_AO_T pAO)
  - *coEventRegister_401 register all user provided HW API functions*
- void co401DigOutDef (void)
  - *co401DigOutDef Prepares the default output value for a digout port*
- void co401DigOutErr (void)
  - *co401DigOutErr Prepares the error output value for a digout port*
- void co401AnOutDef (void)
  - *co401AnOutDef Prepares the default output value for a digout port*
- void co401AnOutErr (void)
  - *co401AnOutErr prepares the error output value for a digout port*
- RET_T co401Init (void)
  - *Initializes the modules internal functionalities*
- void co401Task (void)
  - *co401Task implements the CiA profile 401 functionality*

3.1.1 Detailed Description

device profile implementation according CiA 401

3.1.2 Function Documentation

3.1.2.1 void co401AnOutDef (void)

- co401AnOutDef Prepares the default output value for a digout port

Calculates for all analog out ports the default value obtained from 0x6411 and calls the HW API to set the value.

3.1.2.2 void co401AnOutErr (void)

- co401AnOutErr prepares the error output value for a digout port

Calculates for all analog out ports the error value obtained from 0x6444 and calls the HW API to set the value.

Real available analog output ports, object 0x6411, have to be used, because the standard says object 0x6443 is optional, and if not available, the default value, which is 1 (TRUE), should be used.

An error value is only set, if error mode is defined AND an error value is specified. Otherwise no value is set, the port stays with its old value.

The function is called automatically in case the device is going into an communication error, e.g. a CAN Bus-Off condition or it detects a communication error according object 0x1029.

After device initialization the function *can* be called to set the error value at device start-up.

References co401AnOutErr(), and co401DigOutErr().

Referenced by co401AnOutErr().

3.1.2.3 void co401DigOutDef (void)

- co401DigOutDef Prepares the default output value for a digout port

Calculates for all digital out ports the default value obtained from 0x6200 and calls the HW API to set the value.

3.1.2.4 void co401DigOutErr (void)

- co401DigOutErr Prepares the error output value for a digout port

Calculates for all digital out ports the error value obtained from 0x6207 and calls the HW API to set the value.

Real available output ports, object 0x6200, have to be used, because the standard says object 0x6206 is optional, and if not available, the default value, which is 0xFF, should be used.

An error value is only set, if error mode is defined AND an error value is specified.

The function is called automatically in case the device is going into an communication error, e.g. a CAN Bus-Off condition or it detects a communication error according object 0x1029.

After device initialization the function *can* be called to set the error value at device start-up.

Referenced by co401AnOutErr().

3.1.2.5 RET_T co401Init (void)

- Initializes the modules internal functionalities

The main task of this function is registering itself all CANopen indications handled by the module at the CANopen stack.

3.1.2.6 void co401Task (void)

- co401Task implements the CiA profile 401 functionality

This function handles all process input ports (anin, digin) and should be called cyclically to poll the inputs. Typically, in non OS applications, it is called in the main application loop, as often as possible.

This function calls the HW access functions for dig in and analog in. Take care how efficient, time consuming these functions are.

A typical sequence

```
1   while (1)  {
2       coCommTask();
3       co401Task();
4 }
```

3.1.2.7 RET_T coEventRegister_401 (CO_EVENT_401_DI_T *pDI*, CO_EVENT_401_DO_T *pDO*, CO_EVENT_401_AI_T *pAI*, CO_EVENT_401_AO_T *pAO*)

- coEventRegister_401 register all user provided HW API functions

the function pinter is take into internal pointers. If one of the basic functionalities **digital in**, **digital out analog in** or **analog out** is not provided, NULL should be used as argument.

Parameters

pDI	pointer to the digital in HW API function
pDO	pointer to the digital out HW API function
pAI	pointer to the analog in HW API function
pAO	pointer to the analog out HW API function

Returns

RET_OK - always, no checks are possible. It is assumed, that all pointers are pointing to the correct functions.

3.2 co_p401.h File Reference

defines for ciA 401 profile implementation

Functions

- RET_T coEventRegister_401 (CO_EVENT_401_DI_T, CO_EVENT_401_DO_T, CO_EVENT_401_AI_T, CO_EVENT_401_AO_T)
  - *coEventRegister_401 register all user provided HW API functions*
- RET_T co401Init (void)
  - *Initializes the modules internal functionalities*

- void co401Task (void)
  - *co401Task implements the CiA profile 401 functionality*
- void co401DigOutDef (void)
  - *co401DigOutDef Prepares the default output value for a digout port*
- void co401DigOutErr (void)
  - *co401DigOutErr Prepares the error output value for a digout port*
- void co401AnOutDef (void)
  - *co401AnOutDef Prepares the default output value for a digout port*
- void co401AnOutErr (void)
  - *co401AnOutErr prepares the error output value for a digout port*

3.2.1 Detailed Description

defines for ciA 401 profile implementation

co_401 profile specific CANopen header

3.2.2 Function Documentation

3.2.2.1 void co401AnOutDef (void)

- co401AnOutDef Prepares the default output value for a digout port

Calculates for all analog out ports the default value obtained from 0x6411 and calls the HW API to set the value.

3.2.2.2 void co401AnOutErr (void)

- co401AnOutErr prepares the error output value for a digout port

Calculates for all analog out ports the error value obtained from 0x6444 and calls the HW API to set the value.

Real available analog output ports, object 0x6411, have to be used, because the standard says object 0x6443 is optional, and if not available, the default value, which is 1 (TRUE), should be used.

An error value is only set, if error mode is defined AND an error value is specified. Otherwise no value is set, the port stays with its old value.

The function is called automatically in case the device is going into an communication error, e.g. a CAN Bus-Off condition or it detects a communication error according object 0x1029.

After device initialization the function *can* be called to set the error value at device start-up.

References co401AnOutErr(), and co401DigOutErr().

Referenced by co401AnOutErr().

3.2.2.3 void co401DigOutDef (void)

- co401DigOutDef Prepares the default output value for a digout port

Calculates for all digital out ports the default value obtained from 0x6200 and calls the HW API to set the value.

3.2.2.4 void co401DigOutErr (void)

- co401DigOutErr Prepares the error output value for a digout port

Calculates for all digital out ports the error value obtained from 0x6207 and calls the HW API to set the value.

Real available output ports, object 0x6200, have to be used, because the standard says object 0x6206 is optional, and if not available, the default value, which is 0xFF, should be used.

An error value is only set, if error mode is defined AND an error value is specified.

The function is called automatically in case the device is going into an communication error, e.g. a CAN Bus-Off condition or it detects a communication error according object 0x1029.

After device initialization the function *can* be called to set the error value at device start-up.

Referenced by co401AnOutErr().

3.2.2.5 RET_T co401Init (void)

- Initializes the modules internal functionalities

The main task of this function is registering itself all CANopen indications handled by the module at the CANopen stack.

3.2.2.6 void co401Task (void)

- co401Task implements the CiA profile 401 functionality

This function handles all process input ports (anin, digin) and should be called cyclically to poll the inputs. Typically, in non OS applications, it is called in the main application loop, as often as possible.

This function calls the HW access functions for dig in and analog in. Take care how efficient, time consuming these functions are.

A typical sequence

```
1   while (1)  {
2       coCommTask();
3       co401Task();
4 }
```

3.2.2.7 RET_T coEventRegister_401 (CO_EVENT_401_DI_T *pDI,* CO_EVENT_401_DO_T *pDO,* CO_EVENT_401_AI_T *pAI,* CO_EVENT_401_AO_T *pAO*)

- coEventRegister_401 register all user provided HW API functions

the function pinter is take into internal pointers. If one of the basic functionalities **digital in**, **digital out analog in** or **analog out** is not provided, NULL should be used as argument.

Parameters

pDI	pointer to the digital in HW API function
pDO	pointer to the digital out HW API function
pAI	pointer to the analog in HW API function
pAO	pointer to the analog out HW API function

Returns

RET_OK - always, no checks are possible. It is assumed, that all pointers are pointing to the correct functions.

3.3 usr_401.c File Reference

Implements hardware access to fulfill co_p401.c requirements.

Functions

- void byte_out_printfInd (UNSIGNED8 port, UNSIGNED16 outVal)
  - *set an 8 bit digital output - printf()*
- void byte_out_piInd (UNSIGNED8 port, UNSIGNED8 outVal)
  - *set an 8 bit digital output - process image*
- UNSIGNED8 byte_in_piInd (UNSIGNED8 port, UNSIGNED8 filter)
  - *read an 8 bit digital input - process image*
- void analog_out_printfInd (UNSIGNED8 port, INTEGER16 outVal)
  - *set an 16bit analog output - printf()*
- void analog_out_piInd (UNSIGNED8 port, INTEGER16 outVal)
  - *set an 16bit analog output - process image*
- INTEGER16 analog_in_piInd (UNSIGNED8 port)
  - *read an 16 bit analog input - process image*

3.3.1 Detailed Description

Implements hardware access to fulfill co_p401.c requirements.

These functions are called by the CANopen library module co_p401 which implements the CiA 401 profile behaviour. The user has to register the functions to the CANopen stack before the stack is activated.

3.3.2 Function Documentation

3.3.2.1 INTEGER16 analog_in_piInd (UNSIGNED8 *port*)

- read an 16 bit analog input - process image

This version uses a virtual file system where anin ports are files named like the port number under the directory anin. The port value is read there as decimal value.

Note

All analog channels are 16 bit integer. If hardware has a lower resolution, use the most left aligned bits.

This function has to be registered using coEventRegister_401()

```
1 coEventRegister_401(NULL, NULL, analog_in_piInd, NULL)
```

Returns

void

Parameters

port	selected 8bit port

3.3.2.2 void analog_out_piInd (UNSIGNED8 *port*, INTEGER16 *outVal*)

- set an 16bit analog output - process image

This version uses a virtual file system where anout ports are files named like the port number under the directory anout. The port value is written there as decimal value.

Note

All analog channels are 16 bit integer. If hardware has a lower resolution, use the most left aligned bits.

This function has to be registered using coEventRegister_401()

```
1 coEventRegister_401(NULL, NULL, NULL, analog_out_piInd)
```

Returns

void

Parameters

port	selected 16bit analog channel
outVal	output value

3.3.2.3 void analog_out_printfInd (UNSIGNED8 *port*, INTEGER16 *outVal*)

- set an 16bit analog output - printf()

This simple version uses only printf() to print the information to the stdout of the process.

This function has to be registered using coEventRegister_401()

```
1 coEventRegister_401(NULL, NULL, NULL, analog_out_printfInd)
```

Returns

void

Parameters

port	selected 16bit analog channel
outVal	output value

3.3.2.4 UNSIGNED8 byte_in_piInd (UNSIGNED8 *port*, UNSIGNED8 *filter*)

- read an 8 bit digital input - process image

This version uses a virtual file system where digin ports are files named like the port number under the directory digin. The port value is read there as decimal value.

This function has to be registered using coEventRegister_401()

```
1 coEventRegister_401(byte_in_piInd, NULL, NULL, NULL)
```

Returns

void

Parameters

port	selected 8bit port
filter	port filter value

3.3.2.5 void byte_out_piInd (UNSIGNED8 *port*, UNSIGNED8 *outVal*)

- set an 8 bit digital output - process image

This version uses a virtual file system where digout ports are files named like the port number under the directory digout. The port value is written there as decimal value.

This function has to be registered using coEventRegister_401()

```
1 coEventRegister_401(NULL, byte_out_piInd, NULL, NULL)
```

Returns

void

Parameters

port	selected 8bit port
outVal	output value

3.3.2.6 void byte_out_printfInd (UNSIGNED8 *port*, UNSIGNED16 *outVal*)

- set an 8 bit digital output - printf()

This simple version uses only printf() to print the information to the stdout of the process.

This function has to be registered using coEventRegister_401()

```
1 coEventRegister_401(NULL, byte_out_printfInd, NULL, NULL)
```

Returns

void

Parameters

port	selected 8bit port
outVal	output value

Index