

Procedures and functions of iseg CAN DLL

Note

The information in this manual is subject to change without notice. We take no responsibility for any error in the document. We reserve the right to make changes in the product design without reservation and without notification to the users.

Document: **Functions of isegcanv dll** Version: 2.48 Date: **12. Januar 2005 17:18**

Table of Contents

Procedures and functions of iseg CAN DLL.....	1
1. Initialization of CAN hardware, creating a isegnet and CAN-Client and connecting together.....	4
2. Re-initialization of isegnet, CAN-Client respectively of CAN hardware.....	5
3. Initialization a HV module in an application software.....	5
4. Procedure for a write access of message to CAN.....	6
5. Function for a write read access of a message to CAN.....	6
6. Function for a read access of a message to CAN.....	7
7. Function for read the release of isegcanv.dll.....	7
8. Function to change the time out of the internal thread.....	8
9. Function to read the status information of the PCAN driver.....	8
10. Overview of procedures and functions of isegCAN.DDL.....	9

1. Initialization of CAN hardware, creating a isegnet and CAN-Client and connecting together

*bool InitCanSystem(int HwNbr, unsigned short PortAdr, unsigned short Intr, unsigned char BitRate, HWND Handle, char * ClientName);*

Function returned a true, if it has been all correctly initialized.

<i>argument</i>	<i>designation</i>	<i>example</i>
HwNbr	hardware number	1 = PEAK ISA-CAN-Card 2 = PEAK CAN-Dongle 3 = PEAK CAN-Dongle EPP 4 = PHYTEC ISA Card
PortAdr	port address	0x100, .. , 0x378,..
Intr	interrupt	3, 4, 5, 7, 9, 10, 11, 12, 15
BitRate	CAN bit rate	20, 50, 100, 125, 250
Handle	Windows handle	&Application
ClientName	pointer to a string with call by reference	gives the client name from application to DLL for instance "iseg HV16" ; from DLL to application, if initializing has been correctly returns "Client iseg HV16", If there has been an error, returns one of the following messages: "Error while registering Hardware !" "Error while registering Net!" "Error while registering Client" "Error while connecting to Net" "Error while registering Message"

*bool InitCanPCI(unsigned short CANLine, unsigned char BitRate, HWND Handle, char * ClientName);*

Function returned a true, if it has been all correctly initialized.

<i>argument</i>	<i>designation</i>	<i>example</i>
CANLine	CAN line (coresponds the controller on PCAN card)	0, 1 .. 8
BitRate	CAN bit rate	20, 50, 100, 125, 250
Handle	Windows handle	&Application
ClientName	pointer to a string with call by reference (see above)	

2. Re-initialization of isegnet, CAN-Client respectively of CAN hardware

void ReInitCanSystem(unsigned short CANLine)

Reinitializes the CAN connection by removing of all connected clients and closing the internal network.

<i>argument</i>	<i>designation</i>	<i>example</i>
CANLine	CAN port on the CAN interface card	0, 1 .. 8 for PCAN PCI 0 for PCAN Dongle

*void ReInitCanClient(uw16 CANLine, char *ClientName)*

Reinitializes the CAN connection by removing the client which is indicated without a dismounting of all other connected clients.

<i>argument</i>	<i>designation</i>	<i>example</i>
CANLine	CAN port on the CAN interface card	0, 1 .. 8 for PCAN PCI 0 for PCAN Dongle
ClientName	was indicated while the initializing of the CAN	"iseq HV16"

3. Initialization a HV module in an application software

*bool CanLogOnOff(unsigned short *ID, unsigned char *Stat, unsigned char *Typ, char *TimeStamp)*

Function returned a true, if a iseg HV-module has been log on.

<i>argument</i>	<i>designation</i>	<i>example</i>
ID	reference to CAN identifier from HV-module	0x2d0
Stat	reference to status of high voltage module which has been loged on	0x01
Device class	reference of the device class of module	0 EHQ 16 channel (standard, HV <=2.55kV) 1 EHQ 8 channel (high precision, floating, HV<=1kV) 2 EHQ 8 channel (standard, floating, <=1kV) 3 EHQ 8 channel standard multi-voltage 4 CIO (for HV with relay cards) 5 CIO standard (2 x 8V / 8I) 6 EHQ 8 channel (standard, HV>1kV) 7 EHQ 8 channel (standard, floating, HV>1kV) 8 CIO standard (16 x 1V / 1I) 10 NHQ standard 11 NHQ high precision 12 SHQ high precision 13 EHQ 1 channel standard 254 CIO standard with separate I/O instruction
TimeStamp	pointer to a string with the time of the receiving message format "hh:mm:ss.ms"	hh hours mm minuets ss seconds ms milliseconds

4. Procedure for a write access of message to CAN

*void Write2Can(unsigned short ID, unsigned char *CanBuff, char *TimeStamp)*

<i>argument</i>	<i>designation</i>	<i>example</i>
ID		CAN identifier of HV-module 0x2d0
CanBuff	Reference to a stream of unsigned char, first element including the length of message and at the following see iseg device	0x03, 0xa0, 0x01, 0x00 Write a set voltage with a value of 256 to channel 0. (channels are coded from 0x0 to 0xf)
TimeStamp	pointer to a string with the send time of message format "hh:mm:ss.ms"	hh hours mm minuets ss seconds ms milliseconds

5. Function for a write read access of a message to CAN

*bool WriteReadfCan(unsigned short ID, unsigned char *CanBuff, char *TimeStamp)*

Function returned a true, if it has been read a message from CAN with the same identifier which has been hand over.

<i>argument</i>	<i>designation</i>	<i>example</i>
ID	CAN identifier must corresponding with ID of HV-module	0x2d1 R/W bit is set to 1
CanBuff	Reference to a read stream of unsigned char, first element including the length of message and at the following see iseg device	0x01, 0xa0 Read the set voltage from channel 0. (channels are coded from 0x0 to 0xf)
TimeStamp	pointer to a string with the send time of message - time of the receiving message format "hh:mm:ss.ms"	hh hours mm minuets ss seconds ms milliseconds

*bool WriteReadfCanFTime(unsigned short ID, unsigned char *buff, FILETIME *time)*

It corresponds to the function above but the time is handle in filetime format (using for OPC only)

*bool WriteReadRTRfCan(unsigned short ID, unsigned char *buff, char *TimeStamp);*

It corresponds to the function above but in the buff is the entry of length zero and it will be sent the RTR bit.

6. Function for a read access of a message to CAN

*bool ReadfCan(unsigned short *ID, unsigned char *CanBuff, char *TimeStamp)*

Function returned a true, if it has been read a message from CAN.

<i>argument</i>	<i>designation</i>	<i>example</i>
ID	Reference to CAN identifier of read message.	0x0d0
CanBuff	Reference to a read stream of unsigned char, first element including the length of message and the following see iseg device protocol.	0x02, 0xc0, 0x01 Read the active message of general status.
TimeStamp	pointer to a string with the time of the receiving message format "hh:mm:ss.ms"	hh hours mm minuets ss seconds ms milliseconds

*bool ReadfBuff(unsigned short *ID, unsigned char *buff, char *TimeStamp)*

Load CAN message from internal Buffer of the isegcanv.dll. If a CAN message is reseived between the part of make a blank buffer and the function call WriteReadfCAN() the DLL will copy this in here internal buffer. It is important to evaluate fast error messages.

Function returned a true, if it has been read a message from CAN.

<i>argument</i>	<i>designation</i>	<i>example</i>
ID	Reference to CAN identifier of read message.	0x0d0
CanBuff	Reference to a read stream of unsigned char, first element including the length of message and the following see iseg device protocol.	0x02, 0xc0, 0x01 Read the active message of general status.
TimeStamp	pointer to a string with the time of the receiving message format "hh:mm:ss.ms"	hh hours mm minuets ss seconds ms milliseconds

7. Function for read the release of isegcanv.dll

*char *GetDLLVer(void)*

Function returned reference to a stream of characters which included the release of DLL. The release consist of three numbers as software release and one character: "h" – for high speed applications or "s" for slow speed applications.

8. Function to change the time out of the internal thread

void SetCanTimeOut(unsigned short tout)

<i>argument</i>	<i>designation</i>	<i>example</i>
tout	value in milliseconds	100

appropriate for the functions *WriteReadfCan()* and *WriteReadfCanFTime*;

9. Function to read the status information of the PCAN driver

*char *GetCanStatus(void)*

CAN_ERR_OK	returns "OK";	
CAN_ERR_NOVXD	returns "NOVXD";	// VxD not loaded, license not up to date
CAN_ERR_BUSOFF	returns "BUSOFF";	// CAN controller is switched off from the bus
CAN_ERR_ILLHW	returns "ILLHW";	// hardware handle invalid
CAN_ERR_BUSHEAVY	returns "BUSHEAVY";	// bus error – error counter exceeding the limit
CAN_ERR_OVERRUN	returns "OVERRUN";	// to many CAN messages in to short time
CAN_ERR_BUSLIGHT	returns "BUSLIGHT";	// errors were detected on the bus

10. Overview of procedures and functions of isegCAN.DDL

*bool InitCanSystem(int HwNbr, unsigned short PortAdr, unsigned short Intr, unsigned char BitRate, HWND Handle, char *string)*

*bool InitCanPCI(unsigned short CANLine, unsigned char BitRate, HWND Handle, char *string)*

void ReInitCanSystem(unsigned short CANLine)

*void ReInitCanClient(uw16 CANLine, char *ClientName)*

*bool CanLogOnOff(unsigned short *ID, unsigned char *Stat, char *TimeStamp)*

*void Write2Can(unsigned short ID, unsigned char *CanBuff, char *TimeStamp)*

*bool WriteReadfCan(unsigned short ID, unsigned char *CanBuff, char *TimeStamp)*

*bool WriteReadfCanFTime(unsigned short ID, unsigned char *buff, FILETIME *time)*

*bool WriteReadRTRfCan(unsigned short ID, unsigned char *buff, char *TimeStamp)*

*bool ReadfCan(unsigned short *ID, unsigned char *CanBuff, char *TimeStamp)*

*bool ReadfBuff(unsigned short *ID, unsigned char *buff, char *TimeStamp)*

*char *GetDLLVer(void)*

void SetCanTimeOut(unsigned short tout)

*char *GetCanStatus(void)*