

Procedures and functions of iseg[p/s]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_isegxcan_dll** Version: 2.68 Date: **22. April 2008 16:27**

Table of Contents

Procedures and functions of iseg[p/s]CAN DLL	1
1. Initialization of the CAN hardware	4
1.1. Initialization of non Plug-and-Play PEAK hardware.....	4
1.2. Initialization of Plug-and-Play PEAK or SYS TEC hardware	4
2. Re-initialization of isegnet, CAN-Client respectively of CAN hardware	5
3. Initialization a HV module in an application software	6
4. Procedure for a write access of message to CAN	6
5. Function for a write read access of a message to CAN	7
6. Function for a read access of a message to CAN.....	8
7. Function for read the release of iseg[p/s]can.dll	8
8. Function to change the time out of the internal thread.....	9
9. Function to read the status information of the PCAN driver	9
9.1. PEAK Hardware	9
9.2. SYS TEC Hardware	9
10. Overview of procedures and functions of isegCAN.DLL	10

1. Initialization of the CAN hardware

1.1. Initialization of non Plug-and-Play PEAK hardware

*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.
(implemented in isegpcan.dll only)

<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 "iseq HV16" ; from DLL to application, if initializing has been correctly returns "Client iseq 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"

1.2. Initialization of Plug-and-Play PEAK or SYS TEC hardware

*bool InitCan(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 (corresponds 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 RelnitCanSystem(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 0	for PCAN PCI for PCAN Dongle

*void RelnitCanClient(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 0	for PCAN PCI for PCAN Dongle
ClientName	was indicated while the initializing of the CAN	"iseg 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 logged on	0x01
Device class	reference of the device class of module	
0	EHQ 16 HV channels, standard resolution, HV <=2.55kV	
1	EHQ 8 HV channels, standard resolution , floating, HV<=1kV	
2	EHQ 8 HV channels, high precision, floating, HV<=1kV	
3	EHQ 8 channels, standard resolution, HV<=1kV, MIX	
4	CIO 8 HV channels, for special hardware with relay cards	
5	CIO up to 16 channels (ADC 2 x 8V / 8I), standard resolution, OPTION MIX	
6	EHQ 8 channels, standard resolution, OPTION MIX	
7	EHQ 8 channels, standard resolution, OPTION MIX	
8	CIO up to 16 channels (ADC 16 x 1V / 1I), standard resolution, OPTION MIX	
10	NHQ 1 or 2 HV channels, standard resolution	
11	NHQ 1 or 2 HV channels, high precision	
12	SHQ 1 or 2 HV channels, high precision	
13	EHQ 1 HV channel, standard resolution	
21	EDS 16 HV channels, standard resolution, distributor HV	
23	HPS2	
24	EHS/EMS/ELS 8 channels, standard resolution, common GND, OPTION MIX	
25	EHS/EMS/ELS 8 channels, standard resolution, floating GND, OPTION MIX	
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 filetype 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 iseg[p/s]can.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 for the hardware manufacture: "p" for PEAK CAN hardware and "s" for the SYS TEC USB CAN device.

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(unsigned char line)*

<i>argument</i>	<i>designation</i>	<i>example</i>
line	CAN line 0..15	0

9.1. PEAK Hardware

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

9.2. SYS TEC Hardware

USBCAN_ERR_MAXINSTANCES	returns "MAXINST"
USBCAN_ERR_ILLHANDLE	returns "ILLHNDL"
USBCAN_ERR_ILLPARAM	returns "ILLPARAM"
USBCAN_ERR_ILLHW	returns "ILLHW"
USBCAN_CANERR_OK	returns "OK"
USBCAN_CANERR_OVERRUN	returns "OVERRUN"
USBCAN_CANERR_BUSLIGHT	returns "BUSLIGHT"
USBCAN_CANERR_BUSHEAVY	returns "BUSHEAVY"
USBCAN_CANERR_BUSOFF	returns "BUSOFF"

10. Overview of procedures and functions of iseg[p/s]CAN.DLL

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

*bool InitCan(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(unsigned char line)*