18/07/2015

CRUSB_Spartan_1_0_0.dll
(Function list)

History:

CRUSB_Spartan_V1_0_0.dll	first working library

Library was compiled and linked in RAD Studio XE7
DIGA s.c. 07/2015

Functions:

CRUSBOpen	Open USB communication with CRUSB device
CRUSBClose	Close USB communication with CRUSB device
CRUSBInfo	Information about CRUSB
DLLInfo	Info about DLL
CAN_Init	Init CAN communication
CAN_Read	Read one CAN frame from Rx buffer
CAN_Write	Write one CAN frame to Tx buffer
CAN_Close	Close CAN communication
CAN_Status	Status of CAN
CAN_Rx_Clear	Clear Rx Buffer in DLL FIFO
CAN_Tx_Clear	Clear Tx Buffer in DLL FIFO

Using these functions, it is possible to make communication on CAN bus for 11bit and 29bit frames.

Service

DELPHI DLLinfo(DLL_INFO:DLL_INFO_)		Borlad C++ void DLLinfo(DLL_INFO* DLL_INFO_)
Type	Function	
Parameters	DLL_INFO: pointer to structure	unsigned char
	DLL_INFO.Version :widestring;	
Result	n/a	
Description	Return version of DLL.	

USB Communication

DELPHI CRUSBOpen(Device:Byte):Byte		Borlad C++ unsigned char CRUSBOpen(unsigned char Device)
Type	Function	
Parameters	Device:Byte	unsigned char
	0	Device number to connect (only one device)
Result	0 255	connected disconnected
Description	Calling this function is necessary to establish the connection via USB.	

CRUSBClose(Device:Byte):Byte		void CRUSBClose(unsigned char Device)
Type	Function	
Parameters	Device: Byte	unsigned char
	0	Device number to disconnect (only value 0 allowed)
Result	0	disconnected
Description	Calling this function is necessary to close USB communication with CRUSB.	

CRUSBInfo(CRUSB_info:pointer)			void CRUSBInfo(CRUSB_INFO* CRUSB_INFO_)
Type	Procedure		
Parameters	CRUSB_info: pointer to structure		
	CRUSB_info.Name: CRUSB_info.SN: CRUSB_info.HW: CRUSB_info.SW: CRUSB_info.DLL:	widestring; widestring; widestring; widestring; widestring;	'CRUSB Spartan' 'Hardware: x.xx' 'Software: x.xx' 'S/N:xxxxxxxx' 'DLL vx.x.x'
Result	n/a		
Description	Basic information about CRUSB (static info from USB driver).		

CAN Communication

CAN_Init(BTR0_BTR1:word;Par:word):byte		unsigned char CAN_Init(unsigned int BTR0_BTR1, unsigned int Par)
Type	Function	
Parameters	BTR0_BTR1: Word (CAN boudrate)	unsigned int
	 0x1C15 0x1C0A 0x1C1D 0x1C0E 0x1C0B 0x1C05 0x1C02 0x1B01 0x0502	PAR=0x18xx 10kb/s 20kb/s 50kb/s 100kb/s 125kb/s 250kb/s 500kb/s 800kb/s 1000kb/s
	Par: Word	unsigned int
	0xxx00 0xxx01 0xxx02 **0x0Cxx** **0x18xx**	Echo OFF Echo ON **listen ON (not implemented)** **12 MHz (not implemented)** **24 MHz (not implemented)**
Result	:byte 0 255	 connected disconnected
Description	This function connects to CAN bus, **Par** - parameter allows to get back sent CAN frames **(Do not use random values in BTR0_BTR1 because it can makes unexpected operation in CRUSB)** **For other values than listed above try to calculate according BTR0 and BTR1 for C_CAN Bosch)** **http://www.bittiming.can-wiki.info/#C_CAN**	

CAN_Close()		unsigned char CRUSBClose(void)
Type	Procedure	
Parameters	n/a	
Result	n/a	
Description	This function disconnect from CAN bus.	

CAN_Read(CAN_MSG:PCAN_MSG):longword	unsigned int CAN_Read(CAN_MSG_RX* CAN_MSG_RX_)		
Type	Function		
Parameters	CAN_MSG - Pointer to structure		
	CAN_MSG.ID_DW: Longword; CAN_MSG.DLC: Byte; CAN_MSG.DATA: Array [0..7] of Byte; CAN_MSG.MSG_TYPE: Byte; CAN_MSG.RX_TX: Boolean;	unsigned long unsigned char unsigned char[8] unsigned char bool	Frame ID Number of bytes in CAN frame Data 7 6 5 4 3 2 1 0 **ECHO** **RTR** **29bit** **DIR** 0 0 0 0 0 – Rx , 1 - TX
Result	longword	unsigned int	number of frames left in buffer (max 100 000 frames)
Description	Every call this function returns one CAN frame in structure CAN_MSG, and as a result the number of CAN frames in buffer. CAN_MSG.RX_TX is set when **Echo ON** (see CAN_Init).		

CAN_Write(CAN_MSG:TCAN_MSG)	void CAN_Write(CAN_MSG_TX _CAN_MSG_TX)		
Type	Procedure		
Parameters	CAN_MSG - structure		
	CAN_MSG.DATE_YY :Byte; CAN_MSG.DATE_MM :Byte; CAN_MSG.DATE_DD :Byte; CAN_MSG.TIME_HH :Byte; CAN_MSG.TIME_MM :Byte; CAN_MSG.TIME_SS :Byte; CAN_MSG.TIME_us :Word; CAN_MSG.ID_DW :Longword; CAN_MSG.DLC :Byte; CAN_MSG.DATA :array [0..7] of Byte; CAN_MSG.RTR :Boolean; CAN_MSG.EXT :Boolean; CAN_MSG.RX_TX :Boolean; CAN_MSG.MSG_TYPE :Byte;	unsigned char unsigned char unsigned char unsigned char unsigned char unsigned char unsigned int unsigned long unsigned char unsigned char[8] bool bool bool unsigned char	Year Month Day Hours Minutes Seconds Micro seconds Frame ID Number of bytes in CAN frame Data 1 - RTR 0 - Standard frame (11bit), 1- Extended frame (29bit) 0 – Rx, 1 – Tx Direction **not implemented**
Result	n/a		
Description	Every call send one can frame to Tx buffer.		

CAN_Status(CAN_INFO:PCAN_INFO)		void CAN_Status(CAN_STAT* CAN_STAT_)	
Type	Procedure		
Parameters	CAN_INFO - pointer to structure		
	CAN_INFO.w_Boudrate: Word; CAN_INFO.w_Busload: Word; CAN_INFO.b_Listen: Boolean; CAN_INFO.b_Rx_Error: Boolean; CAN_INFO. b_Tx_Error: Boolean; CAN_INFO.b_BusOFF: Boolean; CAN_INFO.b_Passive: Boolean; CAN_INFO.w_RX_Error_cnt: Word; CAN_INFO.w_TX_Error_cnt: Word; CAN_INFO.B_LEC: Byte; CAN_INFO.B_Buffer_Load:Byte;	unsigned int unsigned int bool bool bool bool bool unsigned int unsigned int unsigned char unsigned char	CAN baudrate Current CAN bus load **Not implemented** Rx Error appear Tx Error appear CAN Bus OFF appear CAN Bus paasive appear Number of Rx error frames Number of Tx error frames LEC error[1] **Not implemented**
Result	n/a		
Description	Returns the information about CAN controller. [1]		

CAN_Rx_Clear()	unsigned char CAN_Rx_Clear(void)	
Type	Function	
Parameters	n/a	
Result	Byte	unsigned char
Description	Function clears DLL Rx CAN messages FIFO	

CAN_Tx_Clear()	unsigned char CAN_Tx_Clear(void)	
Type	Function	
Parameters	n/a	
Result	Byte	unsigned char
Description	Function clears DLL Tx CAN messages FIFO	