BTR CAN-Bus Library
V1.01

Blank Page

Index

Blank Page

Introduction

For hardware information please see the following datasheets:
DATA_Sheet_DV-CANFDE4-01_EN.pdf DATA_Sheet_DV-CANFAA4-01_EN.pdf
DATA_Sheet_DV-CANFAE4-01_EN.pdf DATA_Sheet_DV-CANFRAS4-01_EN.pdf

The BTR CAN-Bus-Library simplifies the use of the BTR CAN-Bus modules. Its subroutines permit the use of all communication functions of the modules with just a few calls.

Each subroutine of the BTR CAN-Bus Library returns a result about its operation. It helps you debugging your program. The return value reports an invalid handling of subroutines. If an element in your program does not work as you expect, please check this return value.

In the following you will find a short description how to use the BTR CAN-Bus-Library.

First of all you have to include the library files.

```
#include BTR_CAN_BUS_LIB.INC
```

Inside of your task main you have to initialize the BTR CAN-Bus-Library and all modules you will use in your application.

```
  call vLCLInit( )
  call bLCLInitModule( FAE4, 3, Return ) ' module type, module address
                                         ' (as set on module), result
  call bLCLInitModule( FRAS4_21, 4, Return )
```

Before calling any subroutine of the BTR CAN-Bus Library call *vLCLInit* for initialization.

Call the function bLCLInitModule for every module you are going to use.

Never use the same module address more than once. Possible module addresses are all integer values from 1 to 99. The address which you use in your program must be the same address as set on your BTR CAN-Bus module.

Now you are able to call functions of the library as described in chapter "Subroutines". But make yourself sure that your module stays active by using the functions bLCLSendControlMsg for every input module, bLCLAnaOutProcData for every analogue output module and bLCLDigOutProcData for every digital output module at least once a second.

```
  call bLCLSendControlMsg( 1, Return )
'  module address (as set on module), result
  call bLCLAnaOutProcData( 2, val1, val2, val3, val4, Return )
'  module address (as set on module), 4 analogue values, result
  call bLCLDigOutProcData( 4, dig1, dig2, dig3, dig4, Return )
'  module address (as set on module), 4 digital values, result
```

! Every input module needs a control message to stay active (see bLCLSendControlMsg). Output modules need an output message every second to stay active (see bLCLAnaOutProcData or bLCLDigOutProcData).

One small but complete example for the BTR CAN-Bus Library is BTR_CAN_BUS_LIB_EXAMPLE_FRAS4_21.TIG .
This example also includes the use of the Tiger Graphic Library.

```
' only needed if the Tiger Graphic Library is not saved in the installation
directory:
#project_path "..\TGL\TigerGraphicLibrary"

'***************************************************************************
'       Choose your LCD type
'***************************************************************************
#define LCD_INVERSION_MODE      1       ' 0 = normal    for "white" LCD
'                                       ' 1 = inversion for "blue"  LCD

'***************************************************************************
'       Include the files for the TGL and your application
'**************************************************************************
#include TigerGraphicLibrary.INC
#include BTR_CAN_Bus_LIB.INC

'*************************************************************************
'       Create your defines and declare your variables
'*************************************************************************
#define WINDOW 0

word wgBtn1, wgBtn2, wgBtn3, wgBtn4
byte bgFont

task main

  word wlElementId, wlIbuFill
  byte blReturn, blKeycode, blFontId
  byte blDig1, blDig2, blDig3, blDig4

  '*************************************************************************
  '     Install the device drivers for the TP and the LCD
  '*************************************************************************
  #include TGL_DEVICE_DRIVERS_TP1000.INC

  '*************************************************************************
  '     Initialize Tiger Graphic Library, variables, BTR_CAN_Library
  '*************************************************************************
  call vTglInit()
  call vLCLInit()   ' CAN device driver installation here
  blReturn = LCL_OK
  wlElementId = 0
  blFontId = 0
  wlIbuFill = 0
  blKeycode = 0
  blDig1 = 0
  blDig2 = 0
  blDig3 = 0
  blDig4 = 0

  '*************************************************************************
  '     Initialize your modules
  '*************************************************************************
  call bLCLInitModule( DIGITAL_OUT, 4, blReturn ) ' module address 4 e.g.

  '*************************************************************************
  '     Create your fonts
  '*************************************************************************
  call bTglCreateFontParams( blFontId, "Valencia", 10, "normal", "center", &
  "center", "prop", 0, SPACING_CHAR_DEFAULT, 0, "imm", "word", blReturn )
  bgFont = blFontId
  blFontId = blFontId + 1

  '*************************************************************************
  '     Initialize your elements
  '*************************************************************************

  call bTglCreateTextButtonWnd( 60, 30, "1", bgFont, 3, &
  TP_KEY_NO_AUTOREPEAT, wlElementId, WINDOW, 25, 200, 1, blReturn )
  wgBtn1 = wlElementId
  wlElementId = wlElementId + 1
  call bTglCreateTextButtonWnd( 60, 30, "2", bgFont, 3, &
  TP_KEY_NO_AUTOREPEAT, wlElementId, WINDOW, 95, 200, 2, blReturn )
  wgBtn2 = wlElementId
  wlElementId = wlElementId + 1
  call bTglCreateTextButtonWnd( 60, 30, "3", bgFont, 3, &
  TP_KEY_NO_AUTOREPEAT, wlElementId, WINDOW, 165, 200, 3, blReturn )
  wgBtn3 = wlElementId
  wlElementId = wlElementId + 1
  call bTglCreateTextButtonWnd( 60, 30, "4", bgFont, 3, &
  TP_KEY_NO_AUTOREPEAT, wlElementId, WINDOW, 235, 200, 4, blReturn )
  wgBtn4 = wlElementId
  wlElementId = wlElementId + 1

  '*************************************************************************
  '     your ideas here
  '*************************************************************************
  call bTglShowWindow( WINDOW, blReturn )
```

```
  while 1=1
    get #TP, #0, #UFCI_IBU_FILL, 0, wlIbuFill ' get TouchPanel buffer length
    if wlIbuFill > 0 then                     ' check input length of buffer
      get #TP, #0, 1, blKeycode               ' get keycode
      switchi blKeycode
      case 1: blDig1 = mod( (blDig1 + 1), 2 )
      case 2: blDig2 = mod( (blDig2 + 1), 2 )
      case 3: blDig3 = mod( (blDig3 + 1), 2 )
      case 4: blDig4 = mod( (blDig4 + 1), 2 )
      endswitch
    endif
    call bLCLDigOutProcData( 4, blDig1, blDig2, blDig3, blDig4, blReturn )
  endwhile

end
```

General Settings

As in each other Tiger-BASIC™ application you can make some global settings for your project. For details see the programming manual of the Tiger-BASIC™ language.

For using the Tiger Graphic Library and its components please see the TigerGraphicLibrary manual.

The sizes for *user_string_size* ,*user_tempstr_size* and *user_stack_size* are default values.

```
'' initializes all variables for program start
'' ==> should be deactivated for development!!
user_var_init
'' enforces declarations of variables
'' ==> helps avoiding erors caused by wrong types of variable!!
user_var_strict
'' default string size for declarations
'' can be 0 if string length is given for each string declaration
'' defining each string length by declaration saves stack memory
user_string_size    64
'' size of temporary strings in some string operations
'' combined operations or logical terms could cause errors,
'' if this value is smaller than the used string
user_tempstr_size   9600
'' memory size for all variables and subs in one task
'' a program require this size of ram for each task
'' the tgl runs with additionally 2 tasks
'' graphic fonts   require a minimum stack size of 300 bytes
'' bitmap elements require a minimum stack size of 400 bytes
'' text elements   require a minimum stack size of 700 bytes
user_stack_size  2k
```

Subroutines

Subroutine for the initialization of the BTR CAN-Bus Library

- *vLCLInit*

Subroutine for the initialization of a module

- *bLCLInitModule*

Subroutines for communication with modules

- *bLCLSendControlMsg*
- *bLCLDigOutProcData*
- *bLCLAnaOutProcData*
- *bLCLSendServRTRMsg*
- *bLCLSendProcRTRMsg*
- *bLCLGetServiceData*
- *bLCLGetProcessData*
- *bLCLGetProcessData10*

Subroutines for the conversion of analogue values

- *rLCLConvertAnalogueValToReal*
- *lLCLConvertRealToAnalogueVal*

vLCLInit

`call vLCLInit( )`

Function: Initializes all internal variables of the BTR CAN-Bus Library and runs internal tasks. Also installs the CAN device driver.

! Before calling any subroutine of the BTR CAN-Bus Library call *vLCLInit* for initialization.

bLCLInitModule

call bLCLInitModule (ModuleType, ModuleAddr, Result)

Function: Initializes the module's buffer and the address/type array.

Call this function for every module you are using.

Never use the same address more than once.

Possible module addresses are all integer values from 1 to 99. The address which you use in your program must be the same address as set on your BTR CAN-Bus module.

Parameters:

	B	W	L	S	F	
ModuleType	●	-	-	-	-	the kind of module using the given address FDE4 FAA4 FAE4 FRAS4_21
ModuleAddr	●	-	-	-	-	unique address of module
						Return Values:
Result	●	-	-	-	-	error code, for details see table of error codes 0 ok ›0 error

bLCLSendControlMsg

call bLCLInitModule (ModuleAddr, Result)

Function: This function sends a control message which every input module needs each second to stay active.

Parameters:

	B	W	L	S	F	
ModuleAddr	●	-	-	-	-	unique address of module
Result	●	-	-	-	-	**Return Values:** error code, for details see table of error codes 0 ok ›0 error

bLCLDigOutProcData

`call bLCLDigOutProcData ( ModuleAddr, Digit1, Digit2, Digit3, Digit4, Result )`

Function: This function sends a process data message to the module with the given module address. This message must be sent every second to keep the digital output module (FRAS4_21) active.

! Use this method only for digital output modules (FRAS4_21).

Parameters:

	B	W	L	S	F	
ModuleAddr	●	-	-	-	-	unique address of module
Digit1	●	-	-	-	-	first digital value (0 or 1)
Digit2	●	-	-	-	-	second digital value (0 or 1)
Digit3	●	-	-	-	-	third digital value (0 or 1)
Digit4	●	-	-	-	-	fourth digital value (0 or 1)
						Return Values:
Result	●	-	-	-	-	error code, for details see table of error codes 0 ok ›0 error

bLCLAnaOutProcData

call bLCLAnaOutProcData (ModuleAddr, Value1, Value2, Value3, Value4, Result)

Function: This function sends a process data message to the module with the given module address. This message must be sent every second to keep the analogue output module (FAA4) active.

! Use this method only for analogue output modules (FAA4).

Parameters:

	B	W	L	S	F	
ModuleAddr	●	-	-	-	-	unique address of module
Value1	-	●	-	-	-	first analogue value (0,...,1023)
Value2	-	●	-	-	-	second analogue value (0,...,1023)
Value3	-	●	-	-	-	third analogue value (0,...,1023)
Value4	-	●	-	-	-	fourth analogue value (0,...,1023)
						Return Values:
Result	●	-	-	-	-	error code, for details see table of error codes 0 ok >0 error

bLCLSendServRTRMsg

`call bLCLSendServRTRMsg ( ModuleAddr, Result )`

Function: This function requests service data of module with given address. After using this function you can get these service data by calling the function bLCLGetServiceData.

Parameters:

	B	W	L	S	F	
ModuleAddr	●	-	-	-	-	unique address of module
						Return Values:
Result	●	-	-	-	-	error code, for details see table of error codes 0 ok ›0 error

bLCLSendProcRTRMsg

call bLCLSendProcRTRMsg (ModuleAddr, Result)

Function: This function requests process data of module with given address. After using this function you can get these process data by calling the function bLCLGetProcessData.

Parameters:

	B	W	L	S	F	
ModuleAddr	●	-	-	-	-	unique address of module
						Return Values:
Result	●	-	-	-	-	error code, for details see table of error codes 0 ok >0 error

bLCLGetServiceData

call bLCLGetServiceData (ModuleAddr, ModuleType, Version, Switches, Result)

Function: This function returns service data of module with given address. Service data are: module type, version and switches (only for FRAS4_21)

Parameters:

	B	W	L	S	F	
ModuleAddr	●	-	-	-	-	unique address of module
						Return Values:
ModuleType	●	-	-	-	-	the kind of module using the given address
						10 for FDE4
						13 for FAA4
						12 for FAE4
						11 for FRAS4_21
Version	●	-	-	-	-	version of module
Switches	●	-	-	-	-	only for FRAS4_21: state of switches
						bit 4-7: switch 1-4 manual mode
						bit 0-3: switch 1-4 ON
Result	●	-	-	-	-	error code, for details see table of error codes
						0 ok
						›0 error

bLCLGetProcessData

`call bLCLGetProcessData ( ModuleAddr, Value1, Value2, Value3, Value4, Result )`

Function:	This function returns process data of module with given address. These process data contain 4 analogue 10bit values or 4 digits. Do not use this function for FDE10 module!

Parameters:

	B	W	L	S	F	
ModuleAddr	●	-	-	-	-	unique address of module
						Return Values:
Value1	-	-	●	-	-	first value
Value2	-	-	●	-	-	second value
Value3	-	-	●	-	-	third value
Value4	-	-	●	-	-	fourth value
Result	●	-	-	-	-	error code, for details see table of error codes 0 ok ›0 error

bLCLGetProcessData10

```
call bLCLGetProcessData(    ModuleAddr, Value1, Value2, Value3, Value4, &
                            Value5, Value6, Value7, Value8, Value9, &
                            Value10, Result )
```

Function: This function returns process data of module with given address. These process data contain 10 digits. This function is designed for FDE10 module use only.

Parameters:

	B	W	L	S	F	
ModuleAddr	●	-	-	-	-	unique address of module
						Return Values:
Value1	-	-	●	-	-	first value
Value2	-	-	●	-	-	second value
Value3	-	-	●	-	-	third value
Value4	-	-	●	-	-	fourth value
Value5	-	-	●	-	-	fifth value
Value6	-	-	●	-	-	sixth value
Value7	-	-	●	-	-	seventh value
Value8	-	-	●	-	-	eighth value
Value9	-	-	●	-	-	ninth value
Value10	-	-	●	-	-	tenth value
Result	●	-	-	-	-	error code, for details see table of error codes 0 ok ›0 error

rLCLConvertAnalogueValToReal

`call rLCLConvertAnalogueValToReal ( ValueIn, ValueType, ValueOut, Result )`

Function: This function converts 10-bit analogue values sent from module into floating point values. Be sure to state the correct value type.

Parameters:

	B	W	L	S	F	
ValueIn	-	-	●	-	-	10-bit analogue value for conversion
ValueType	●	-	-	-	-	type of conversion, LCL_TEMPERATURE_50_150 for temperature from -50°C to 150°C (Ni1000/Pt1000) LCL_TEMPERATURE_0_400 for temperature from 0°C to 400°C (Pt1000) LCL_VOLTAGE for voltagefrom 0V to 10V DC
						Return Values:
ValueOut	-	-	-	-	●	converted value
Result	●	-	-	-	-	error code, for details see table of error codes 0 ok ›0 error

lLCLConvertRealToAnalogueVal

`call lLCLConvertRealToAnalogueVal ( ValueIn, ValueType, ValueOut, Result )`

Function: This function converts a floating point value into an analogue value. This analogue value could be sent to a FAE4 module. Be sure to state the correct value type.

Parameters:

	B	W	L	S	F	
ValueIn	-	-	-	-	●	floating point value for conversion
ValueType	●	-	-	-	-	type of conversion, LCL_TEMPERATURE_50_150 for temperature from -50°C to 150°C (Ni1000/Pt1000) LCL_TEMPERATURE_0_400 for temperature from 0°C to 400°C (Pt1000) LCL_VOLTAGE for voltage from 0V to 10V DC
						Return Values:
ValueOut	-	-	●	-	-	converted value
Result	●	-	-	-	-	error code, for details see table of error codes 0 ok ›0 error

Error Codes

Each subroutine of the BTR CAN-Bus Library returns a result about its operation. It helps you debugging your program. The return value informs you about the following details:

- validity of used parameters
- correctness of usage

No.	Name	Description
0	LCL_OK	OK Exit
100	LCL_INVALID_DATA_LENGTH	too many data bytes to fit into standard frame
101	LCL_INVALID_RTR_BIT	value for RTR bit is too high (only use 0 or 1)
102	LCL_INVALID_MODULE_TYPE	you can only only use 4 different module types (1=FDE4, 2=FAA4, 3=FAE4, 4=FRAS4_21)
103	LCL_WRONG_MODULE_TYPE	type definition in message of module is not equal to initial module configuration
104	LCL_INVALID_MODULE_ADDR	given module address exceeds maximum address (only 1,...,99 are allowed)
105	LCL_INVALID_ORDER_TYPE	there are 3 different order types (0=process data, 1=service data, 2=control)
106	LCL_INVALID_DATA_BIT	one or more incorrect digital output values (only use 0 or 1)
107	LCL_INVALID_ANALOGUE_VALUE	one or more analogue values exceed 1023 (10bit) (only use 0,...,1023)
108	LCL_MODULE_ADDRESS_USED	module must be initialised with another address; this address is already used
109	LCL_INVALID_VALUE_TYPE	this is not a valid value type for the conversion functions (only use 0=LCL_TEMPERATURE_50_150, 1=LCL_TEMPERATURE_0_400, 2=LCL_VOLTAGE)
110	LCL_VALUE_OUT_OF_RANGE	the value given to a conversion function is out of range

Overview of Example Programs

Name	Module	Description
BTR_CAN_BUS_LIB_EXAMPLE_FAA4.TIG	FAA4	This example shows 4 sliders (Tiger Graphic Library) which can be used to change 4 analogue values. These values are sent to an FAA4 (analogue output) module. FAA4 module uses module address 2 in this example.
BTR_CAN_BUS_LIB_EXAMPLE_FAE4.TIG	FAE4	This example reads 4 analogue values from one FAE4 (analogue input) module. These values are shown in 4 labels (Tiger Graphic Library). FAE4 module uses module address 3 in this example.
BTR_CAN_BUS_LIB_EXAMPLE_FDE4.TIG	FDE4	This example reads 4 digital values from one FDE4 (digital input) module. These values are shown by some graphics (Tiger Graphic Library). FDE4 module uses module address 1 in this example.
BTR_CAN_BUS_LIB_EXAMPLE_FRAS4_21.TIG	FRAS4/21	This example shows 4 buttons (Tiger Graphic Library) which can be used to change 4 digital values. These values are sent to an FRAS4/21 (digital output) module. FRAS4/21 module uses module address 4 in this example.
BTR_CAN_BUS_LIB_EXAMPLE_FDE10.TIG	FDE10	This example reads 10 digital values from one FDE10 (digital input) module. These values are shown by some graphics (Tiger Graphic Library). FDE10 module uses module address 5 in this example.

Overview of applications

Name	Description
CAN_Bus_Pumpmonitor_V_1_01.TIG	This application uses one FAE4 (address 3) module and one FRAS4/21 (address 4) module.

Overview of Include Files

Name	Description
BTR_CAN_BUS_LIB.INC	file inclusions
BTR_CAN_BUS_LIB_CONF.INC	user configurations
BTR_CAN_BUS_LIB_DEFS.INC	definitions and error codes
BTR_CAN_BUS_LIB_GLOBS.INC	global variables
BTR_CAN_BUS_LIB_SUBS.INC	subroutines

Documentation History

Version of	Description / Changes
1.00	first version
1.01	added FDE10 module example and functionality, bugfixes (service data functions), CAN_Bus_Pumpmonitor_V_1_01.TIG application