AN300107-000: Using Adlos RS485 W32 Bootloader GUI

Introduction

This document shall enable Users to work with KannMOTION RS485 Bootloader 190108.

This bootloader might be used for:

- Firmware Update of KannMOTION drives
- Download USER-Sequence into drive
- As production programmer of own sequence

The bootloader might work w. several RS485 interfaces connected to your windows computer. Adlos recommend to use an isolated interface like adlos 100731 USB/RS485 interface. Contact our sales department for more information.

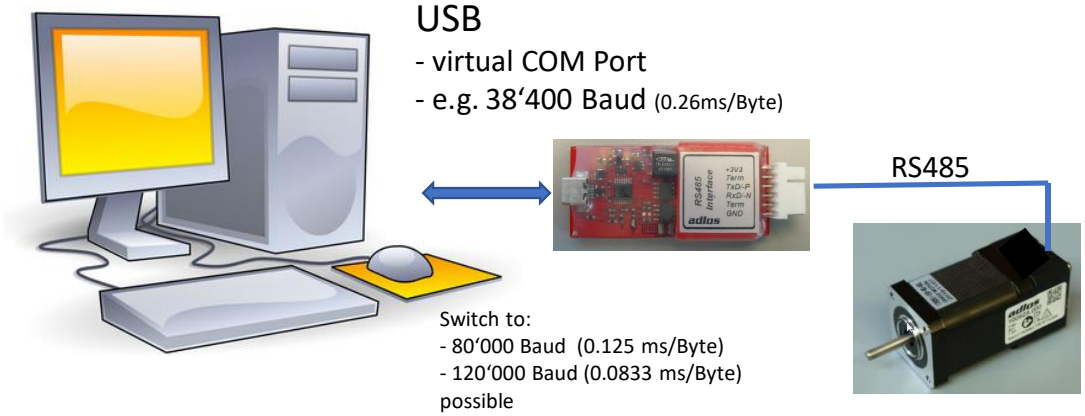
Interface: e.g. 100731 / USB – RS485 Converter isolated

Drive: e.g. 100703 Kann17H2061-150-K17e

Connection

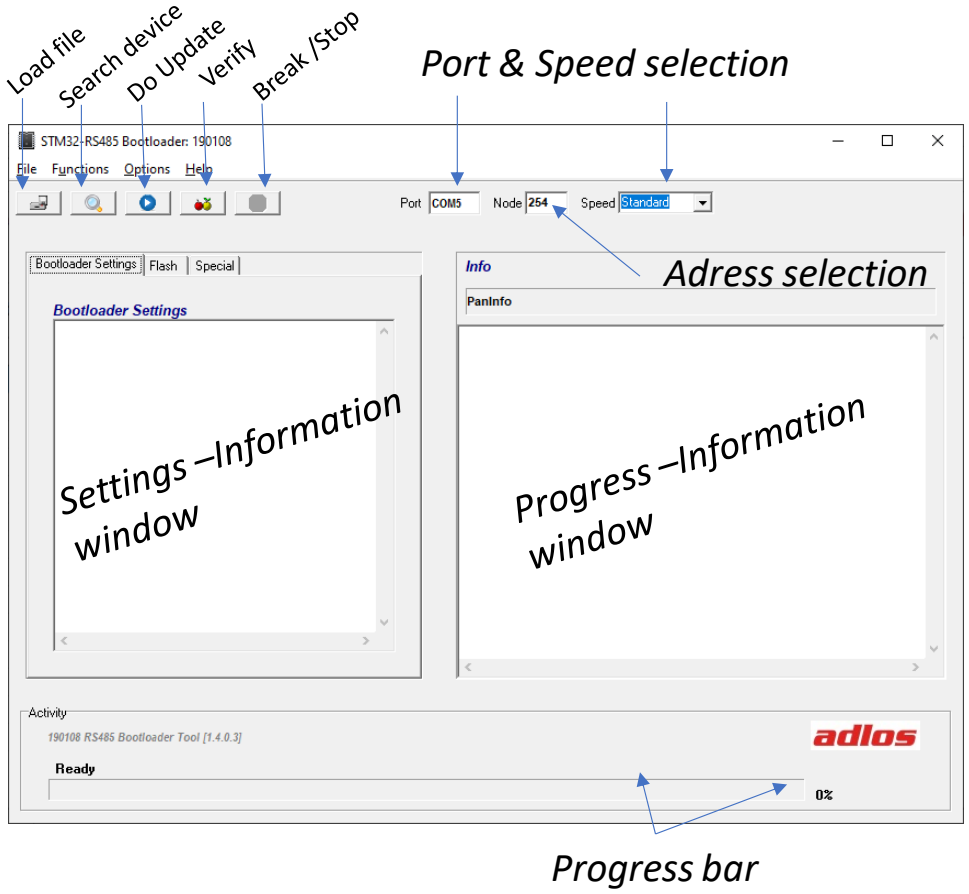
Basic principle /Data rates info

Adlos USC-CAN converter acts as an bridge between an Virtual COMport and CAN-bus, see Illustartor.



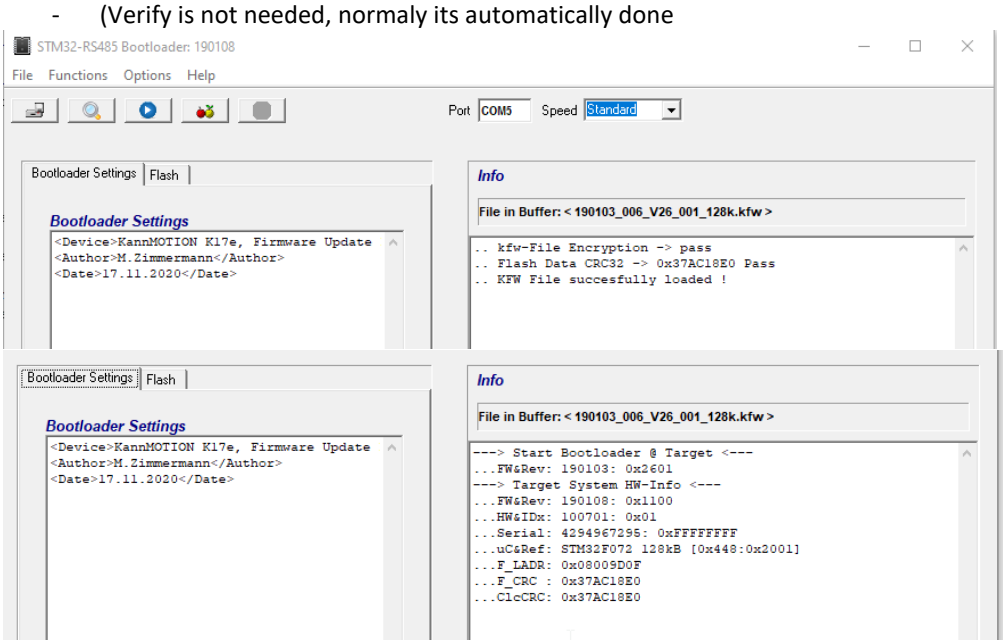
if you are not able to find your drive, mabe you need to have a look at windows Device-Manager. Maybe you also need admin roghts on your computer to install once the FTDI driver on your computer.

GUI operation

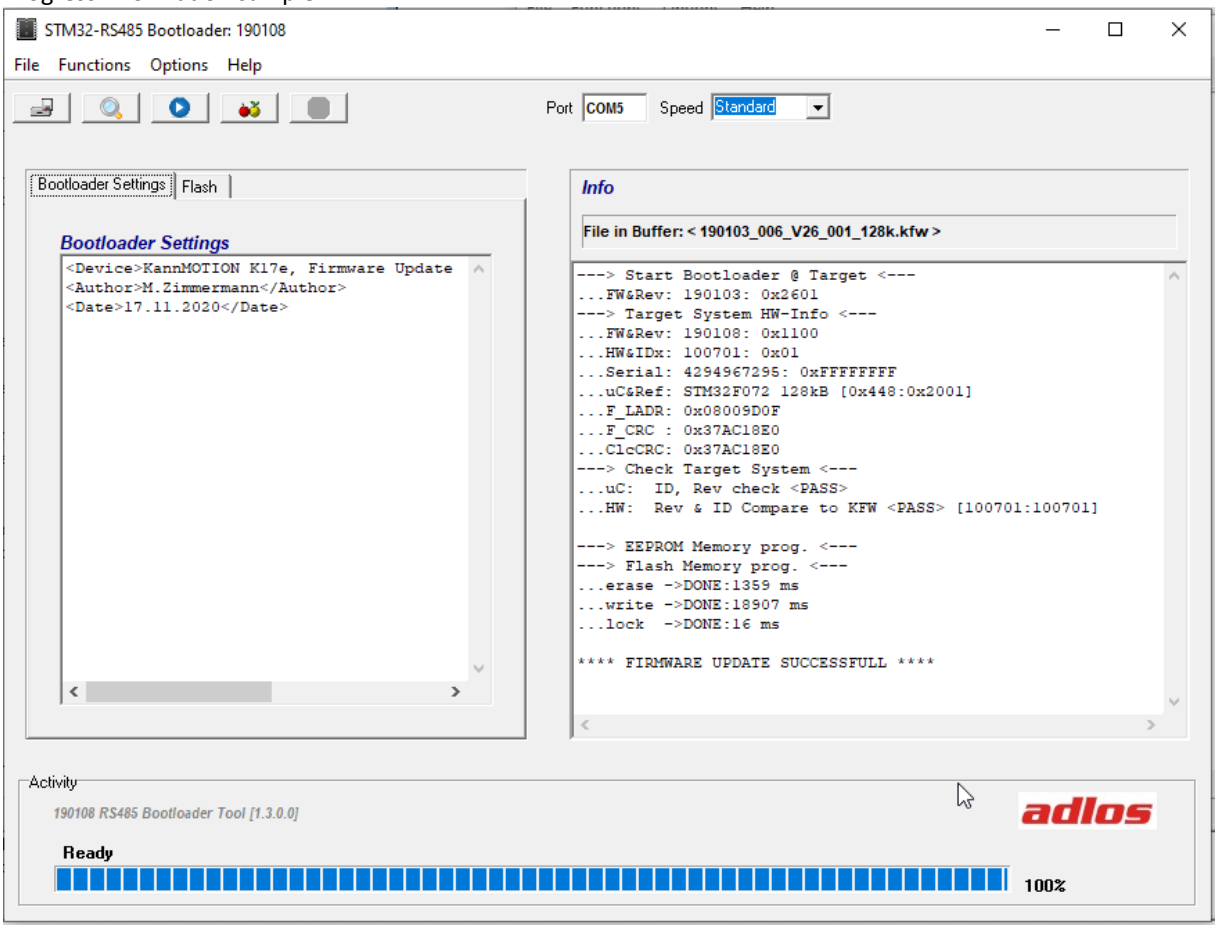


Normal steps:

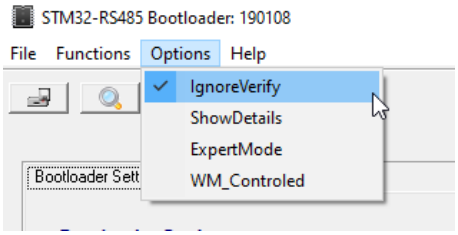
- Open file to download
- Search device
- Do Update
- (Verify is not needed, normaly its automatically done



Progress information sample



Options



IgnoreVerify: while checked no Cell by Cell verify is done, speeds up your update
Note: *APP-Flash is checked by CRC32, so nost of programming errors will be detected without donig an Cell by Cell verify*

ShowDetails: more information is written into progress information window during operation

ExpertMode: Some extra function may be available (only for Experts w. Key)

WM_Controled: Bootloder GUI has enabled Windows-Messaging Control

Using Bootloader in Production

You might use Bootloader GUI within an automatic environment like your production line, to setup up a special firmware.
Fort hat purpose you can CALL the bootloader App by your application.

	Param 1 *Port*	**Param 2** *Filename*	**Param 3 .. n** *Options*
BootloaderRS485_190108.exe	COM5	190082.kfw	

Option Tags

Tag	Meaning
-igVfy	*Ignore Verify, see Options at GUI*
-SpeedMe	*During download use 80 kBaud, recommended only while nice short wiring is done*
-SpeedHi	*During download use 120 kBaud, recommended only while nice short wiring is done*
-WH0xF1234567	*Enable Windows Messaging, the blue number on the right ist he window handle of the calling application (8-digit hexadecimal formated)*
-Hide	*Window is not visible after start*
-Node=3	*RS485 Node Adress, see Document 100578* *0: not addresses devices* *1..32 : valid address Range* *254 : compatibility mode, 'Ba' CMD is not used (for old Devices)* *255 : Broadcast (not recommended)*

Return codes

0: if operation was succesfull

The possible return codes are shon in next picture, some return codes might not be meaningfull on RS485, they are coded for an other application.

```
TErrors  = ( eMS_OK                    = 0,
             eMS_ERR_OUTofRange        = -1,
             eMS_ERR_ParamisWrProtected = -2,
             eMS_ERR_CMDnotAccepted    = -3,
             eMS_ERR_CMDnotKnown       = -4,
             eMS_ERR_ParamisNotKnown   = -5,
             eMS_ERR_ActionFailed      = -6,
             eMS_ERR_NoConnection      = -7,
             eMS_ERR_CheckSum          = -8,
             eMS_ERR_NOACK             = -9,
             eMS_ERR_COMPORT           = -10,
             eMS_ERR_CANDEVICE         = -11,
             eMS_ERR_ANSWER_LENGTH     = -12,
             eMS_ERR_uC_Rev_ID         = -13,
             eMS_ERR_App_Call_Param    = -14,
             eMS_ERR_NoFileinMem       = -15,
             eMS_ERR_HW_ID             = -16,
             eMS_ERR_Verify            = -17,
             eMS_ERR_FileNotFound      = -18,
             eMS_ERR_BootLoaderAtTarget = -19,
             eMS_ERR_USERBREAK         = -30,
             eMS_ERR_UNSPECIFIED       = -31
           );
```

Using Windows Messaging for Remote Control of application

There is a control posibility integrated, wich works on the windows messaging system. Details see:
https://docs.microsoft.com/en-us/windows/win32/dataxchg/wm-copydata

Windows API functions:

```
Tx: SendMessage(receiverHandle, WM_COPYDATA, Integer(xFrmHandle), Integer(@copyDataStruct));
Rx: WMCopyData(var Msg : TWMCopyData); message WM_COPYDATA;
```

Start bootloader:

		Param 1 *Port*	**Param 2** *Filename*	**Param 3 .. n** *Options*
1	BootloaderRS485_190108.exe	COM5	190082.kfw	**-igVfy -WH0xF1234567**
2	BootloaderRS485_190108.exe	**WH0xF1234567**	190082.kfw	**-igVfy**

1) will start bootloader, and Run Update progress
   Windows messaging might be used here to get more information in your own app, and you might Stop/break the updating procedure from your application
2) will start bootloader, and will wait for a Windows Messaging Command
   Windows messaging might control now connection of device, Update process, and also closing of the app

WM-Copydata definitions:

```
//! WM-Copydata Structure
typedef struct
{
  tenCMD enCMD;                    //!< Data-CMD identifier
  UI_8   u8_ValidDataCnt;          //!< number of databytes to use from u8_Data[32]
  UI_8   u8_Data[32];              //!< Data
  UI_8   u8_ResponseDataCnt;       //!< number of databytes expexted as answer
  UI_64  u64_Key;                  //!< Kommunication Key
}
tMyWMCopy;
```

U64Key = 13574684184 (fixed definition)

Command Byte definition

```
//! Commands
typedef enum
{
  // Direction Bootloader to Control APP
  eCMD_BL_GetWmEcho      = 0x00,          //!< Echo Message (Connection test), is sent at PowerUp
                                          //!< of Bootloader to identified Windowhandle at Start Param
                                          //!< Control App gets so the Window Handle of bootloader
  eCMD_BL_InitTunnel     = 0x01,          //!< Init of Comport Tunnel
  eCMD_BL_State_MSG      = 0x02,          //!< State-Message of bootloader
  eCMD_BL_Tunnel_TxData = 0x03,           //!< Data to send over Comport Tunnel, if ResponseDataCnt <> 0
                                          //!< an answer is expected
  // Direction Control App to Bootloader
  eCMD_TU_GetWmEcho      = 0x10,          //!< Echo Message (Connection test)
  eCMD_TU_Tunnel_Timeout= 0x11,           //!< Comport Tunnel Timeout
  eCMD_TU_State_MSG      = 0x12,          //!< State-Message of Cntrl APP
  eCMD_TU_Tunnel_RxData = 0x13,           //!< Data got from Comport Tunnel
  eCMD_TU_BL_Contr       = 0x1F,          //!< Bootloader Control CMD e.g. ( break, start, close )
}
tenCMD;
```

Finding Out Bootloaders Window-Handle

If you start Bootloader with +WH0xnnnnnnn the Bootloader will send *eCMD_BL_GetWmEcho* to your Window, if your Window Handle is nnnnnnnn !
In that Message you will see transmitters window handle (Bootloaders window handle), store taht handle and you might now control Bootloader with this handle.

An other possibility ist to use Windows API, Findwindowhandle…

```
receiverHandle := FindWindow(PChar('TFrmBootloader'),PChar('STM32-RS485 Bootloader: 190108'));
```

Command Details

eCMD_BL_State_MSG

ValidDataCnt = 4 to 28 (depending on ErrorMSG length);
ResponseDataCnt = 0;

the ***tMyWMCopy.u8_Data[32]*** is formatted as follows:

```
//! Bootloader APP-State Data Formatting
typedef struct
{
  UI_8  u8Progress;                    //!< State Progress (0-200) in 0.5%
  tenBLState enState;                  //!< State
  SI_16 i16ErrorCode;                  //!< Error Code
  char Infotext[28];                   //!< Error oder Infotext ASCI string
}
tBLStateData;
```

enState meaning interopretation as followed

```
//! BL State definition
typedef enum
{
  eBLState_ok = 0x00,                 //!< BL is ok, no Error
  eBLState_Error = 0x01,              //!< BL is not ok, Error
}
tenBLState;
```

eCMD_TU_State_MSG

ValidDataCnt = 4 to 28 (depending on ErrorMSG length);
ResponseDataCnt = 0;

the ***tMyWMCopy.u8_Data[32]*** is formatted as follows:

```
//! Control APP-State Data Formatting
typedef struct
{
  UI_8 ACKNACK;                          //!< 6 = ACK // 7 = NACK if NACK then ERRORMSG = set
  tenTUState enState;                    //!< State
  UI_16 u16_TunnelMaxTimeout_ms;         //!< TunnelTimeout in [ms]
  char ErrorMSG[24];                     //!< Error Text ASCI string
}
tTUStateData;
```

enState meaning interopretation as followed

```
//! CNTRL APP State definition
typedef enum
{
  eTUStateClosed = 0x01,                //!< Comport Tunnel is closed
  eTUStateSettingUp = 0x02,             //!< Comport Tunnel is on construction
  eTUStateIDLE = 0x03,                  //!< Comport Tunnel is Ready
  eTUStateTransmitting = 0x04,          //!< Comport Tunnel is working (TX7RX data)
}
tenTUState;
```

eCMD_TU_BL_Contr

Command to Control Bootloader by WM.
At this Message there is only One Byte for Control needed, for that purpose data[0] has following meaning:

```
// Bootloader Control CMDs (1.Datenbyte) bei eCMD_TU_BL_Contr -CMD
TBOOTCMD_CNTRL = (
  eCNTR_CMD_RD_Target_BT_Click = $01,       // Tastendruck für Target uc und ID drücken
  eCNTR_CMD_StartUpdate_BT_Click = $10,     // Tastendruck Start-Update (mit Verify)
  eCNTR_CMD_StartUpdate_woV_BT_Click = $11, // Tastendruck Start-Update (ohne Verify)
  eCNTR_CMD_BREAK_BT_Click = $81,           // Tastendruck abbrechen
  eCNTR_CMD_Close_BT_Click = $F0,           // Tastendruck BL beeenden/schliessen
  eCNTR_CMD_RS232_BRIDGE_OFF = $40,         // RS232 wird direkt gesteuert
  eCNTR_CMD_RS232_BRIDGE_ON = $41,          // RS232 wird umgelaitet auf WM
  eCNTR_CMD_BL_HIDE = $46,                  // App Window will be hidden (not shown)
  eCNTR_CMD_BL_SHOW = $47                   // App Window will be shown
);
```

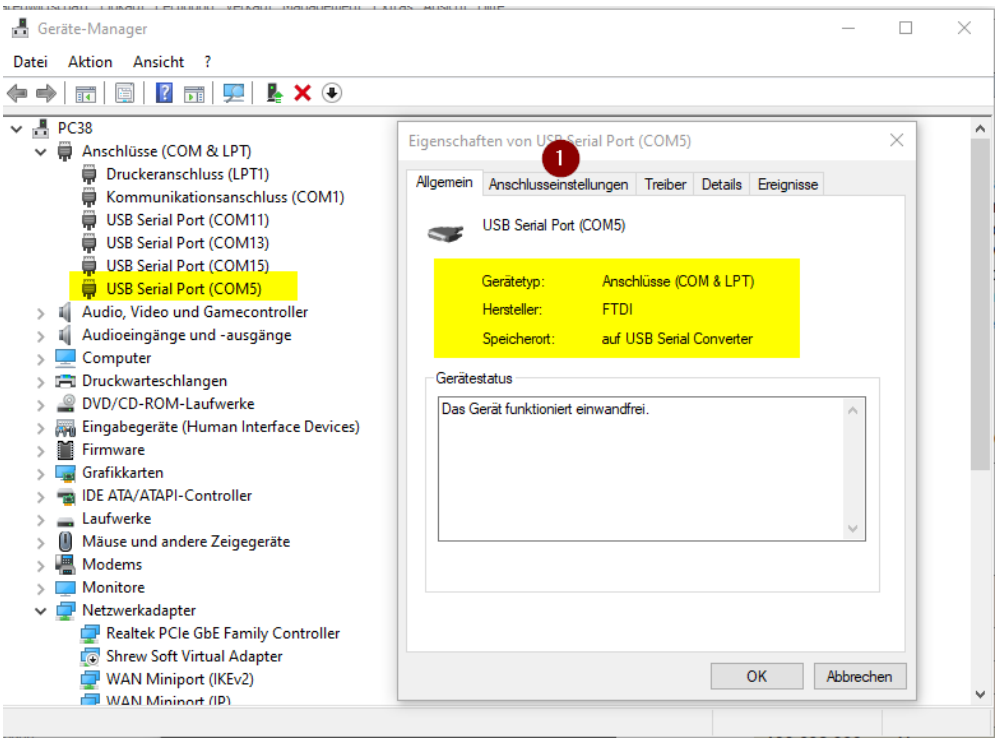
Automatic COM Tunnel procedure example

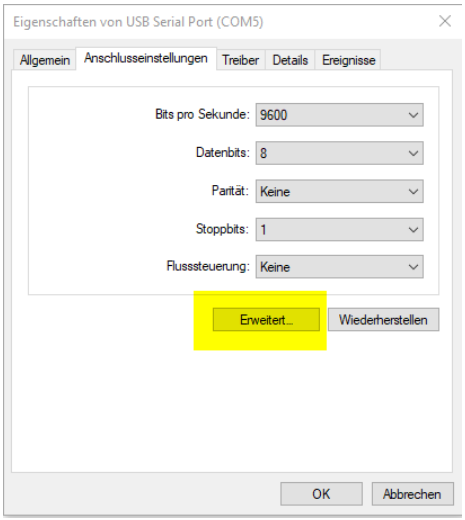
1) StartBootloader from your App
   BootloaderRS485_190108.exe -WH0x00110882 "C:\SW_Works_Git\190103_K17e_GCC\Distribution\190103_006_V26_001_128k.kfw"
2) You will get back an 'InitTunnel' and 'StateMsg'
   initialize your Comport Tunnel
3) Control Bootloader App by eCMD_TU_BL_Contr … e.g. set STartUpdate_woV_Bt
4) Close Bootloader APP

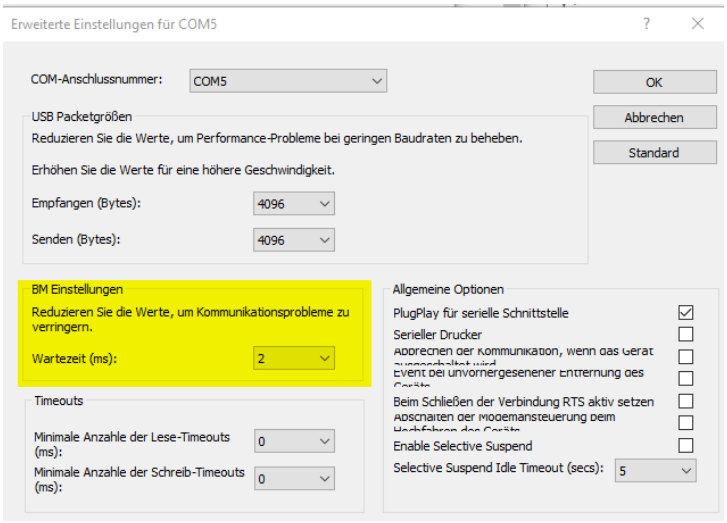
Appendix

Optimize Interface settings

Bootloader might set a message to optimize your interface settings. If that occur, start your windows device manager, and select your COM-port device. After that proceed as followed:





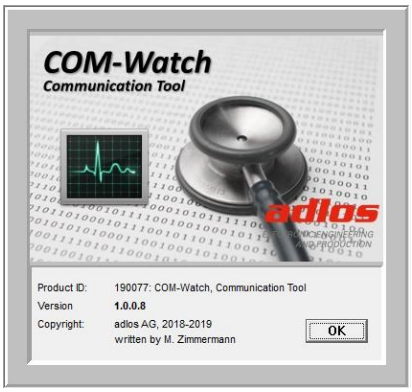


Tools

Adlos Win32-APPs

adlos offers for it's customers some Helping and Design-In Tools.

ComWatch Communication Tool (190077), for Life values



ComWatch is a helping tool for engineers and technicians to explore device specific parametes, read out tracking data and settings and doing firmware updates.
The software is as it is, and in principle for free for adlos customers, the software is not made for a broad range of standard users, it's made in principle for technical engineers which are used in working w. windows based software and have some understanding of technical things.

https://kannmotion.adlos.com/download/comwatchtool/ComWatchSetup.zip

API (190073/190074/190080)

If you want to creat your own Windows based application, adlos offers to use its API, maeans some usefull DLLs (DLL: .net Assemblies) to allow quick an save acces to our devices.

Contact information

Adlos AG
Föhrenweg 14
FL-9496 Balzers

Thomas Vogt
Thomas.Vogt@adlos.com
Tel: +423 263 63 63

Countries: CH, A, LI, SK, IT
www.adlos.com

KOCO MOTION GmbH
Niedereschacher Straße 54
D-78083 Dauchingen

Olaf Kämmerling
O.Kaemmerling@kocomotion.de
Tel: +49 7720/995858-0

Countries: DE, BE, NL, LU
www.kocomotion.de