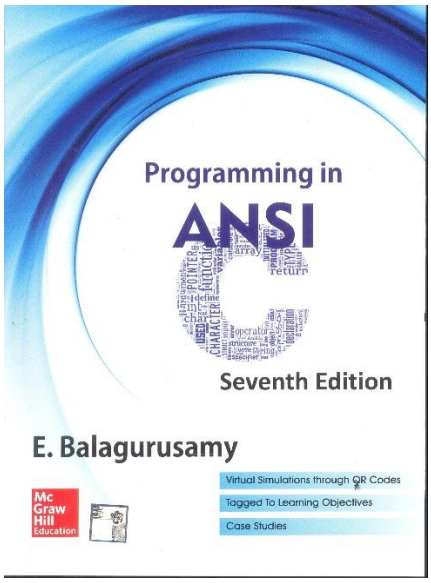
AN100631-001: Customizing KannMOTION drives, w. ANSI C-code

Introduction

This document shall enable Users to work with KannMOTION drives Generation-2. This Generation allows to customize drive function by USERS ANSI-C code.
KannMOTION is based on ISO/IEC 9899:1999 standard, this standard is commonly referred to as C99.

KannMOTION -adlos customizing approach, allows to implement own functionallity in a very code und runtime efficient way. As difference to other common known approaches, the user code must not been parsed by a special function like a PLC. The user code is directly converted/compiled into CPU machine code at compile time.
Fort hat, adlos offers 2-ways to implement own code:

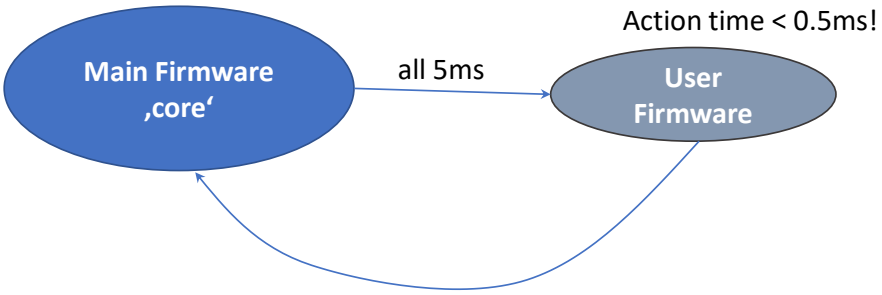
- Graphical -block programming (easier way)
- ANSI-C-code programming



How it works

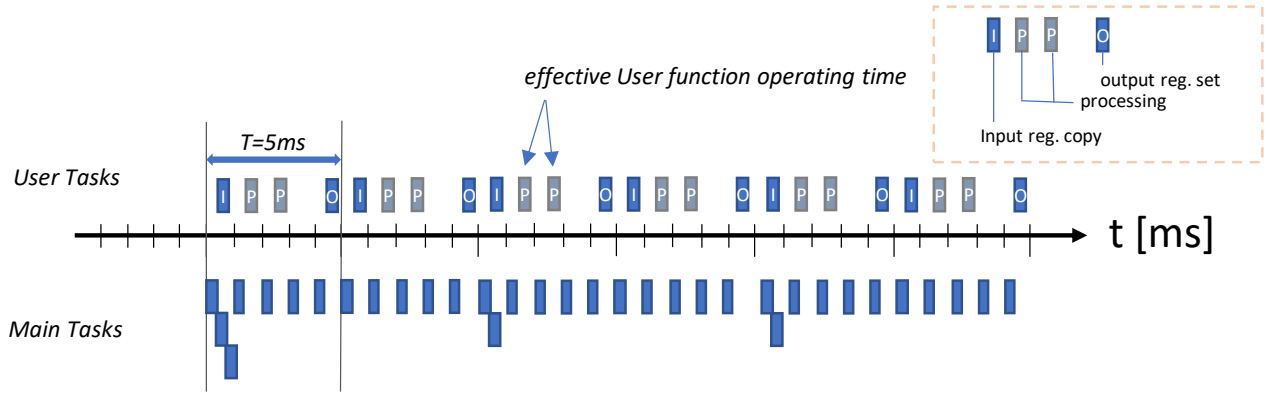
There is a base firmware running on our KannMOTION drivers, this firmware handles, communication, motor driving, positioning, error-management, io-scanning, …. means all base features of our drives.
As a very special extension, this base (core) firmware calls user code in user code section, if Checksumm and function pointers are valid.

Picture 1: basic call principle



The drivers firmware principle is based on 'cooperative multitaking', what means, that no task is allowed to block the operation, big working load needs tob e splitted in smaller working packages !

Picture 2: Timing



Write your code cooperative, means you are niot allowed to write time blocking code !
Your functions are not allowed to exeed more than 1ms operating time, for a stable system it's needed that your operation time/cycle is much less than 1ms, shall be in Range of <100us

```
While (stAppCSPS.SPSUserVar.u8_Din.b.bDI1==1)
{
}
```

Use stepchain Var or merker Var to store, hold, or wait on conditions….

```
if (stAppCSPS.SPSUserVar.u8_Din.b.bDI1==1)
{
   stAppCSPS.SPSUserVar.u8_StepChain[0]=1;
}
```

Working

Using a C-Editor

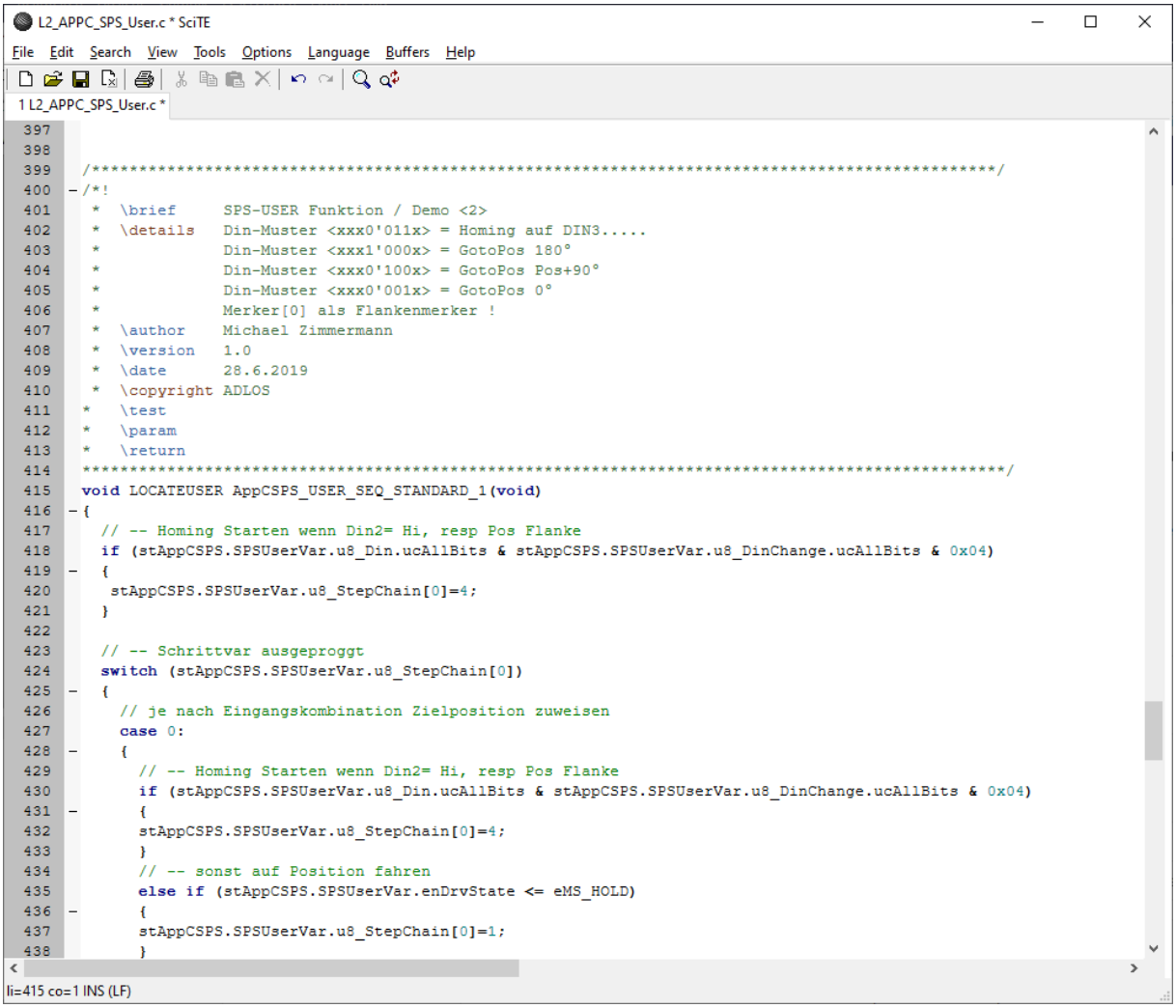
Actually we recommend to use a C-code highlighting editor, as our Pluged editor is not so nice to work with..
You can use:

- Any text editor
- SCiTE http://www.scintilla.org
- Eclipse…

Your User File

Place your code into the following functions:

Filename	Description
void LOCATEUSER AppCSPS_USER_SEQ_STANDARD_1(void)	Your sequence Part-1 - program here in what you need to do, splitt it maybe in a second part
void LOCATEUSER AppCSPS_USER_SEQ_STANDARD_2(void)	Your sequence Part-2 - second part maybe needed to meet cooperative multitasking needs
void LOCATEUSER AppCSPS_USER_SEQ_ERROR(void)	Your error Handler Maybe special thinks to do while going intio error mode




```
/*****************************************************************************************************/
/*!
 *  \brief     SPS-USER Funktion / Demo <2>
 *  \details   Din-Muster <xxx0'011x> = Homing auf DIN3.....
 *             Din-Muster <xxx1'000x> = GotoPos 180°
 *             Din-Muster <xxx0'100x> = GotoPos Pos+90°
 *             Din-Muster <xxx0'001x> = GotoPos 0°
 *             Merker[0] als Flankenmerker !
 *  \author    Michael Zimmermann
 *  \version   1.0
 *  \date      28.6.2019
 *  \copyright ADLOS
 *   \test
 *   \param
 *   \return
 *****************************************************************************************************/
void LOCATEUSER AppCSPS_USER_SEQ_STANDARD_1(void)
{
  // -- Homing Starten wenn Din2= Hi, resp Pos Flanke
  if (stAppCSPS.SPSUserVar.u8_Din.ucAllBits & stAppCSPS.SPSUserVar.u8_DinChange.ucAllBits & 0x04)
  {
   stAppCSPS.SPSUserVar.u8_StepChain[0]=4;
  }

  // -- Schrittvar ausgeproggt
  switch (stAppCSPS.SPSUserVar.u8_StepChain[0])
  {
    // je nach Eingangskombination Zielposition zuweisen
    case 0:
    {
      // -- Homing Starten wenn Din2= Hi, resp Pos Flanke
      if (stAppCSPS.SPSUserVar.u8_Din.ucAllBits & stAppCSPS.SPSUserVar.u8_DinChange.ucAllBits & 0x04)
      {
      stAppCSPS.SPSUserVar.u8_StepChain[0]=4;
      }
      // -- sonst auf Position fahren
      else if (stAppCSPS.SPSUserVar.enDrvState <= eMS_HOLD)
      {
      stAppCSPS.SPSUserVar.u8_StepChain[0]=1;
      }
```

Header Files

Filename	Description	Mode
L0_KannMotDrvTypes.h	KannMotion Types an Enumeration defines	Readonly-Info
L0_KannCSPS_exTypes.h	Special User Region types	Readonly-Info
L2_APPC_SPS_User.h	User-File Header	Readonly-Info

Adlos standard types

```
// ******* Unsigned Integer Types, abgeleitet von C99-Standard ******************************
typedef uint8_t   UI_8;                                  //!< 8 Bit -> [0..255]
typedef uint16_t UI_16;                                  //!< 16 Bit -> [0..65535]
typedef uint32_t UI_32;                                  //!< 32 Bit -> [0..4'294'967'295]
typedef uint64_t UI_64;                                  //!< 64 Bit

// ******* signed Integer Types *****************************
typedef int8_t   SI_8;                                   //!< 8 Bit -> [ -128  .. +127]
typedef int16_t SI_16;                                   //!< 16 Bit -> [-32768 .. +32767]
typedef int32_t SI_32;                                   //!< 32 Bit -> [-2147483648 .. +2147483647]
typedef int64_t SI_64;                                   //!< 64 Bit
```

Own functions

You might also define your own function, just take care to **do not**:

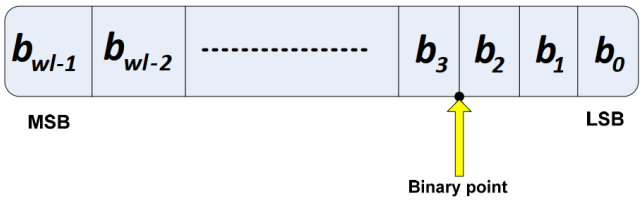
- Giving to much parameters (calling by value), due to Heap- and Stack usage
- Making recursive calls (Stack)

On the other side, do small an efficient, fast functions

Calculations

As the KannMOTION CPU does not have a floating point unit, avoid floting point calculations. Instead of floating point, use fixed point operations.

Figure 1: Fix Point representation



See
http://ee.sharif.edu/~digitalvlsi/Docs/Fixed-Point.pdf

Example:

```
// Goto-Befehl zu Pos i32_PosMerker[0]
case 1:
{
  // Positionswert rechnen
  stAppCSPS.SPSUserVar.i32_PosMerker[0] = 2063;
  stAppCSPS.SPSUserVar.i32_PosMerker[0] *= stAppCSPS.SPSUserVar.u16_Ain;
  stAppCSPS.SPSUserVar.i32_PosMerker[0] >>= 8;
  stAppCSPS.SPSUserVar.u16_Merker[1] = stAppCSPS.SPSUserVar.i32_PosMerker[0] & 0x0000FFFF;
  // genügend Änderung
  if (stAppCSPS.SPSUserVar.u16_Merker[1] >= stAppCSPS.SPSUserVar.u16_Merker[2])
  {
    if ((stAppCSPS.SPSUserVar.u16_Merker[1] - stAppCSPS.SPSUserVar.u16_Merker[2]) < 20)
    {
      break;
    }
  }
```

Flex-User Variables

USER-SPS / RAM / FlexUser Access

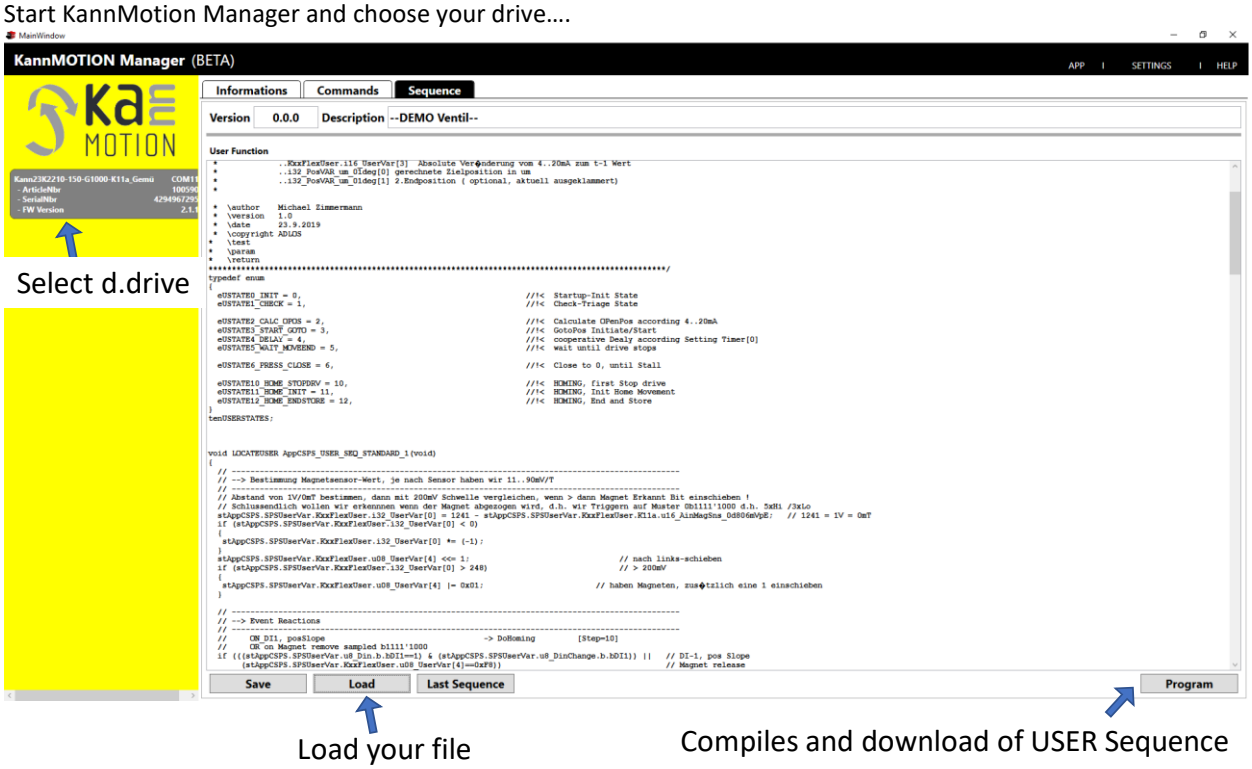
Total 20-Bytes FW 2.1 - xxx FW 2.1 - xxx

RAM Adress	Access type u08 / i08	u16 / i16	u32 / i32
Base+0	0	0	0
Base+1	1		
Base+2	2	1	
Base+3	3		
Base+4	4	2	1
Base+5	5		
Base+6	6	3	
Base+7	7		
Base+8	8	4	2
Base+9	9		
Base+10	10	5	
Base+11	11		
Base+12	12	6	3
Base+13	13		
Base+14	14	7	
Base+15	15		
Base+16	16	8	4
Base+17	17		
Base+18	18	9	
Base+19	19		

Controller type depending pre-allocation	
K11a	K17a
free to use	free to use
u16_Aout4to20mA_0d01mApE	
u16_AinMagSns_0d806mVpE	
u16_Ain4to20mA_0d016mApE	u16_Ain0to24_8d86mVpE

Program your Drive

Start KannMotion Manager and choose your drive….

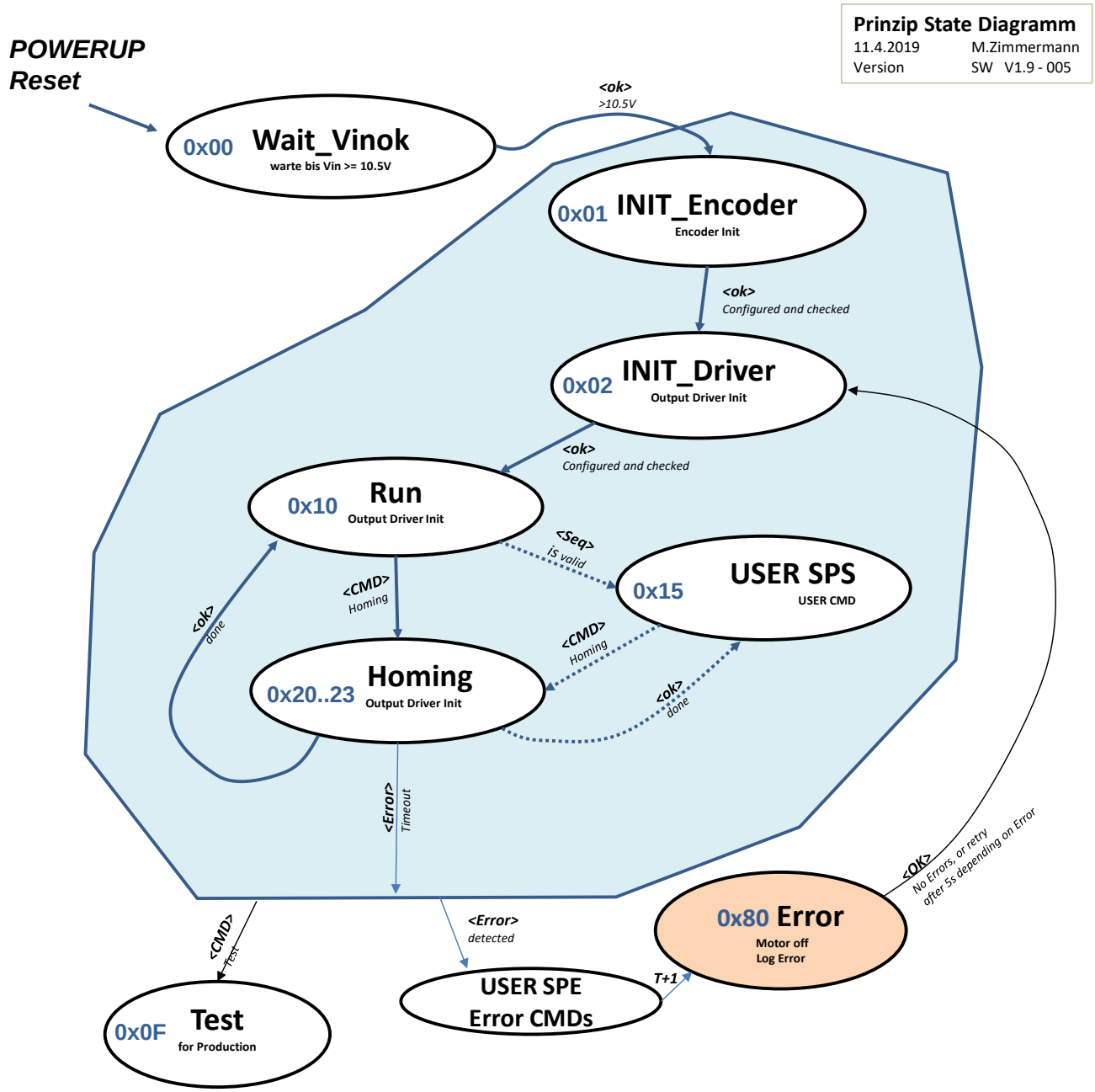


Download of Demo- C-Files:
http://kannmotion.adlos.com/download/ANs/AN100631_SampleCode.zip

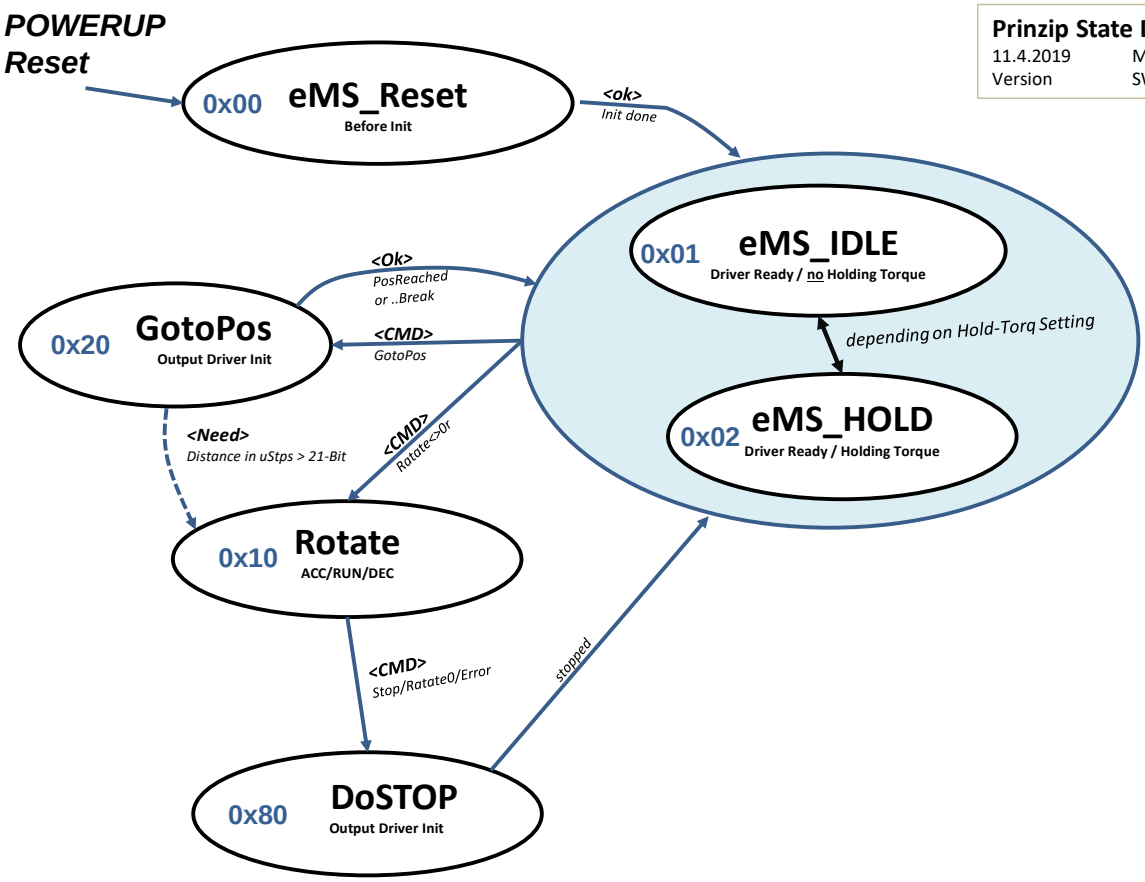
Download Communication Spec (including Param-List):
http://kannmotion.adlos.com/download/DOCs/100570_xxx_KannMotionProtokoll.pdf

Appendix

Drive Principal State Digram / Main States



Drive Principal State Digram / Drive Substates



Tools

Adlos Win32-APPs

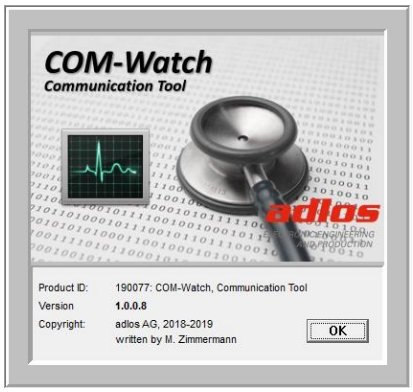
adlos offers for it's customers some Helping and Design-In Tools.

KannMotionManager Tool (190081), manage your drives

KannMOTION Manage is the general tool for our GEN2 drives. This tool comes with an integrated C-coder and a visual drag and drop User interface for customizing your drive.

https://kannmotion.adlos.com/download/kannmotionmanager/application/SetupKannMOTIONManager.zip

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

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