# Application Note
# How to configure the application example

**Hilscher Gesellschaft für Systemautomation mbH**
www.hilscher.com

# Table of Contents

# 1 Introduction

## 1.1 About this Document

This document describes how to configure and to use the Application Example.

## 1.2 List of Revisions

| Rev | Date | Chapter | Revision |
|---|---|---|---|
| 1 | 2015-11-30 | all | Created |

*Table 1: List of Revisions*

# 2 Requirements

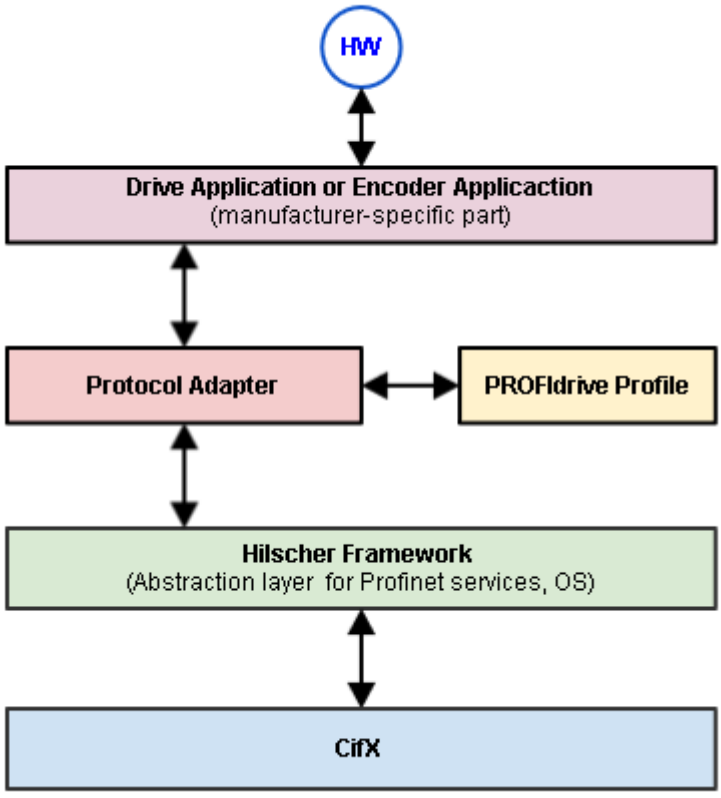
In order to run the Application you need the following components:

- 1 PC with 1 CifX Card
- Hilscher PROFINET Device stack Version 3.8.0.0 (or newer)
- PROFINET Controller (e.g Simotion D435 or CPU 1511-1)
- Microsoft Visual Studio .NET 2005 (or newer)

# 3 Description

This section describes the files used to configure the example application. This applies on both, the Profidrive application as well as on the Encoder application.

The figure below shows the internal structure of the components which together represent the example application:




The complete configuration is divided into several subjects. Excepting Profidrive profile configuration, all configurations will be posed by the Application. These configurations will be listed either in Drive_config.c or in Encoder_Config.c depending on used application.

The following configuration subjects are defined:

## 3.1 Device Configuration:

Device configuration provides a default modules, submodules configuration and device specific information, such as Device ID, Serial number and Order ID to PROFINET stack. The module configuration may be changed later at runtime if the application supports the expected submodule configuration, e.g. controller configures the Standard Telegram 3 but default application configuration is the Standard Telegram 1.

Submodule configuration is defined as follows:

```
PNS_SUBMODULES_T  g_submodules_Config [] =
{      /* Device access point */
       {
         /* .ulApi                 = */0,
         /* .ulSlot                = */0,
         /* .ulSubslot             = */1,
         /* .ulModulId             = */0x3D000001,
         /* .ulSubmoduleId         = */0x3D000010,
         /* .ulProviderDataLen     = */0,
         /* .ulConsumerDataLen     = */0,
         /* .ulConsumerDataOffset  = */0,
         /* .ulProviderDataOffset  = */0
       },
       /* Interface Submodule  */
       {
         /* .ulApi                 = */0,
         /* .ulSlot                = */0,
         /* .ulSubslot             = */0x8000,
         /* .ulModulId             = */0x3D000001,
         /* .ulSubmoduleId         = */0x3D000011,
         /* .ulProviderDataLen     = */0,
         /* .ulConsumerDataLen     = */0,
         /* .ulConsumerDataOffset  = */0,
         /* .ulProviderDataOffset  = */0
       },
       /* Port Submodule (Port 1) */
       {
         /* .ulApi                 = */0,
         /* .ulSlot                = */0,
         /* .ulSubslot             = */0x8001,
         /* .ulModulId             = */0x3D000001,
         /* .ulSubmoduleId         = */0x3D000012,
         /* .ulProviderDataLen     = */0,
         /* .ulConsumerDataLen     = */0,
         /* .ulConsumerDataOffset  = */0,
         /* .ulProviderDataOffset  = */0
       },
       /* Port Submodule (Port 1) */
       {
         /* .ulApi                 = */0,
         /* .ulSlot                = */0,
         /* .ulSubslot             = */0x8002,
         /* .ulModulId             = */0x3D000001,
         /* .ulSubmoduleId         = */0x3D000013,
         /* .ulProviderDataLen     = */0,
         /* .ulConsumerDataLen     = */0,
         /* .ulConsumerDataOffset  = */0,
```

```
        /* .ulProviderDataOffset  = */0
      },

      /* Parameter Access Point (Profile specific) */
      {
        /* .ulApi                 = */0x3D00,
        /* .ulSlot                = */1,
        /* .ulSubslot             = */1,
        /* .ulModulId             = */0x3D000101,
        /* .ulSubmoduleId         = */0x0000FFFF,
        /* .ulProviderDataLen     = */0,
        /* .ulConsumerDataLen     = */0,
        /* .ulConsumerDataOffset  = */0,
        /* .ulProviderDataOffset  = */0
      },
      /* Standard Telegram 81 (Profile specific) */
      {
        /* .ulApi                 = */0x3D00,
        /* .ulSlot                = */1,
        /* .ulSubslot             = */2,
        /* .ulModulId             = */0x3D000101,
        /* .ulSubmoduleId         = */STANDARD_TELEGRAM_81,
        /* .ulProviderDataLen     = */12,
        /* .ulConsumerDataLen     = */4,
        /* .ulConsumerDataOffset  = */0,
        /* .ulProviderDataOffset  = */0
      }
};
```

| Variable | Description |
|---|---|
| ulApi | The number of the API profile to be configured.<br>• 0 : default PROFINET profile<br>• 0x3A00 : Profidrive Profile<br>• 0x3D00 : Encoder Profile |
| ulSlot | The slot this submodule belongs to. |
| ulSubslot | The subslot this submodule belongs to. |
| ulModulId | The ModuleID of the module this submodule belongs to. |
| ulSubmoduleId | The SubmoduleID of this submodule. In case on no default Profile the<br>SubmoduleID describes the identification and the function of a submodule. For example :<br>• 1 : Std Telegram 1<br>• 0x51 : Std Telegram 81 |
| ulProviderDataLen | The length of data provided by this submodule.<br>(Device → Controller) |
| ulConsumerDataLen | The length of data consumed by this submodule.<br>(Controller → Device) |
| ulConsumerDataOffset | Offset in Input image where received data for the submodule shall be copied to. |
| ulProviderDataOffset | Offset in Output image where provided data for the submodule shall be taken from. |

The first four submodules are PROFINET specific, they define the general configuration of PROFINET device, its properties and the available options.

The last two submodules are application specific. They define the configuration of Drive or Encoder module. According to Profidrive specification a Drive or Encoder Object will be represented with two submodules:

- Module Access Point (MAP) to provide an access point for Base Mode Parameter Access with subslot = 1 and submodule ID = 0xFFFF.
- IO-Submodule describes the structure of the IO Data.

In the example above the Encoder Module will be plugged into Slot 1 and will support (by default) the Standard Telegram 81.

**Note**: Currently only one Module is supported

Device configuration is defined as follows:

```
DEVICE_CONFIG_DATA_T g_DeviceData  =
{
  /* .bPrefix         = */  'T',
  /* .bMinor          = */  0,
  /* .bMajor          = */  1,
  /* .bBuild          = */  0,
  /* .bRevision       = */  0,
  /* .breserved       = */  {0,0,0},
  /* .usDeviceId      = */  0xFFFB,
  /* .usVendorId      = */  0x011E,
  /* .usHWVersion     = */  1,
  /* .usProfileId     = */  0x3D00,
  /* .usProfileType   = */  1,
  /* .abDeviceType    = */  "Encoder",
  /* .abOrderId       = */  "Test123456",
  /* .abSerialNumber  = */  "Test123456",

  /* .usConsumerDataLength = */ CONSUMER_DATA_LENGTH,
  /* .usProviderDataLength = */ PROVIDER_DATA_LENGTH,
  /* .pbConsumerDataImg    = */ g_abConsumerDataArea,
  /* .pbProviderDataImg    = */ g_abProviderDataArea,

  /* .usNumberOfSubmodules = */  6,
  /* .patSubmodules        = */  &g_submodules_Config
};
```

| Variable | Description |
|---|---|
| bPrefix | Software Revision Prefix. Possible values and their meanings are:<br>‘V’: Released version |

| | |
|---|---|
| | 'R': Revision<br>'P': Prototype<br>'U': Under field test |
| bMajor<br>bMinor<br>bBuild<br>bRevision | Software version. |
| usDeviceId | The slot this submodule belongs to. |
| usVendorId | The subslot this submodule belongs to. |
| usHWVersion | Hardware Revision |
| usProfileId<br>usProfileType | Profile ID and Profile Specific Type Will be used for I&M Data response. |
| ulProviderDataLen | The length of data provided by this submodule (default 240 Bytes ) |
| ulConsumerDataLen | The length of data consumed by this submodule (default 240 Bytes ) |
| ulConsumerDataOffset | Offset in Input image where received data for the submodule shall be copied to. (Device → Controller) |
| ulProviderDataOffset | Offset in Output image where provided data for the submodule shall be Taken from. (Controller → Device) |
| usNumberOfSubmodules | Number of submodule-items this configuration contains. |
| patSubmodules | Address of first element in the submodule list. |

**Note:** device configuration must match the description file (GSDML) in order to establish the connection with IO Controller.

## 3.2 Profile Configuration:

The Profile configuration is listed in ProtocoAdapter_Config.c. This configuration describes the properties of Profidrive Profile. It contains information to the amount of resources to be used. At start up (Profile initialization) these resources will be allocated, it is the allowed maximum, no additional allocations in runtime.

Profile configuration structure is defined as follows:

```
const PROFILE_CONFIG_T g_Config_Profile =
{
   /*.bMaxNumOfObjectUnits        = */1,
   /*.bMaxNumOfObjects            = */1,
   /*.bMaxNumOfParameters         = */64,
   /*.usMaxParameterBlockLength   = */1024,
   /*.usMaxLatency                = */0
};
```

| Variable | Description |
|---|---|
| bMaxNumOfObjectUnits | Maximum amount of allowed Object Units |

| | |
|---|---|
| bMaxNumOfObjects | Maximum amount of allowed Object (Drive object or Encoder Object) |
| bMaxNumOfParameters | Max number of parameter requestes per multiple-parameter request |
| usMaxParameterBlockLength | Max parameter request / response length |
| usMaxLatency | Max latency time for the processing of a parameter request.<br>usMaxLatency = 0 means no latancy time is defined |

## 3.3 Object Unit Configuration:

Object Unit configuration structure is defined as follows:

```
const CONFIG_OBJECT_UNIT_T g_Config_ObjectUnit =
{
  /* .bUnitID     =  */ 1,
  /* .bUnitType   =  */ 1,

  /* .tVersion    = */
  {
    /* .bMinor    = */0,
    /* .bMajor    = */2
  },

  /* .tDate       =  */
  {
    /* .usYear    = */2015,
    /* .bMonth    = */10,
    /* .bDay      = */15
  },
};
```

| Variable | Description |
|---|---|
| bUnitID | Object unit identifier |
| bUnitType | Object unit type. The value is manufacturer specific |
| bMinor<br>bMajor | Object Unit firmware version |
| usYear<br>bMonth<br>bDay | Object Unit firmware date |

**Note:** Currently each Object Unit can consist logically out of only one Drive or Encoder Object.

## 3.4 Profile Object Configuration:

In addition to the Profile configuration each Profile Object must be configured separately. The Object configuration is also listed in ProtocoAdapter_Config.c. A Profile Object defines either an Encoder Object or a Drive Object.

Object configuration structure is defined as follows:

```
MODULE_CONFIG_DATA_T g_Module_Config =
{
  /* .bNumberOfModules        =  */1,
  /* .atModuleCfg             =  */
  {
   {
     /* .bObjectID            = */ 1,
     /* .usSlot               = */ 1,
     /* .ulModuleId           = */ 0x00000001,
     /* .usTelegramId         = */ STANDARD_TELEGRAM_81,
     /* .ulProperties         = */ PROFILE_EO_PROFPERTIES_SUPPORTED_APP_CLASS_3,
     /* .pbSetPointOffset     = */ 0,
     /* .pbActualValueOffset  = */ 0,
     /* .eModuleType          = */OBJECT_TYPE_ENCODER_INTERFACE,
     /* .tVersion             = */
    {
      /* .bMinor              = */0,
      /* .bMajor              = */1
    },
    /* .tDate                 = */
    {
      /* .usYear              = */2015,
      /* .bMonth              = */11,
      /* .bDay                = */15
    }
   }
  }
};
```

| Variable | Description |
|---|---|
| bNumberOfModules | Number of module-items this configuration contains. |
| bObjectID | unique ID of related Profile object |
| usSlot | The slot this module belongs to. |
| ulModuleId | Internal Module identification |
| usTelegramId | Default telegram number |
| ulProperties | The Properties of the object.<br>Bit 0-15 : supported application class<br>Bit 16-32 : reserved |
| pbSetPointOffset | Offset in Input image where setpoint telegram shall be copied to. |
| pbActualValueOffset | Offset in Output image where actual value telegram shall be taken from. |
| eModuleType | Object Type. Possible values and their meanings are:<br>1: OBJECT_TYPE_AXIS<br>5: OBJECT_TYPE_ENCODER_INTERFACE |
| bMinor | Object software version |

| | |
|---|---|
| bMajor | |
| usYear<br>bMonth<br>bDay | Object software date |

## 3.5 Manufacturer specific Parameters:

The application can define manufacturer specific parameter using the following structure

```
static const PARAMETER_DATA_T g_Manufacturer_Specific_Parameter [] =
{
  {
    /*  .usPnu;                    */   1,  /* Parameter number  1 */
    /*  .bFlags;                   */   0,
    /*  .bTextArrayLength;         */   0,
    /*  .usDataLength;             */   0,
    /*  .pvParameterValue;         */   0,
    /*  .usDescriptionID;          */   PARAMETER_DATATYPE_UINT32  |
                                        MSK_PARAMETER_NOT_WRITABLE |
                                        MSK_STANDARD_FACTOR_NOT_RELEVANT,
    /*  .usNumberOfElements;       */   0,
    /*  .flStandardisationFactor   */   0,
    /*  .usVariableAttribute;      */   0,
    /*  .pszName;                  */   "Test Parameter 1",
    /*  .tLowLimit;                */   {0},
    /*  .tHighLimit;               */   {0xFFFFFFFF},
    /*  .usDescIDExtension;        */   0,
    /*  .usIODataReferenceParam;   */   0,
    /*  .usIODataNormalisation;    */   0,
    /*  .pvDefaultValue;           */   0,
    /*  .pvParameterText;          */   0
  },
  {
    /*  .usPnu;                    */  2, /* Parameter number  1 */
    /*  .bFlags;                   */  MSK_READ_NOTIFY |
                                        MSK_PARAMETER_CONTROLED_BY_APPLICATION,
    /*  .bTextArrayLength;         */  0,
    /*  .usDataLength;             */  0,
    /*  .pvParameterValue;         */  0,
    /*  .usDescriptionID;          */  PARAMETER_DATATYPE_UINT16 |
                                        MSK_PARAMETER_ARRAY_TYPE |
                                        MSK_PARAMETER_NOT_WRITABLE |
                                        MSK_STANDARD_FACTOR_NOT_RELEVANT,
    /*  .usNumberOfElements;       */  2, /* array length */
    /*  .flStandardisationFactor   */  0,
    /*  .usVariableAttribute;      */  0,
    /*  .pszName;                  */  "Test Parameter 2",
    /*  .tLowLimit;                */  {0},
    /*  .tHighLimit;               */  {0xFFFF},
    /*  .usDescIDExtension;        */  0,
    /*  .usIODataReferenceParam;   */  0,
    /*  .usIODataNormalisation;    */  0,
    /*  .pvDefaultValue;           */  0,
    /*  .pvParameterText;          */  0
  }
};
```

| Variable | Description |
|---|---|
| usPnu | Parameter number |

| | |
|---|---|
| bFlags | Parameter flags :<br>Bit 0: the range of validity of parameter 0 = Global / 1 = Local<br>Bit 1-2: notify services 01 : Read notify / 10 : Write notify*/<br>Bit 3-7: reserved |
| bTextArrayLength | Length of additional Text array |
| usDataLength | length of Parameter value in Byte |
| pvParameterValue | pointer to Parameter Value |
| usDescriptionID | Parameter description Identifier.<br>Bit 0-7 : Parameter data type<br>Bit 9 : is set if the Standardization factor not relevant<br>Bit 10 : is set if the parameter is not writable<br>Bit 11 : is set if the parameter has a additional text array<br>Bit 14 : is set if the parameter is an Array<br><br>Other bits are reserved and shall be set to 0 |
| usNumberOfElements | Number of elements or length of String |
| flStandardisationFactor | The factor that converts the value into a standardized variable |
| usVariableAttribute | variable index + conversion index<br>The variable index represents the fixed coding of the physical variable of the parameter value with the conversion index, the unit may be converted into the base unit. |
| pszName | Symbolic name of parameter |
| tLowLimit | Low limit defines the valid value range of the parameter value |
| tHighLimit | High limit defines the valid value range of the parameter value |
| usDescIDExtension | Identifier extension. Shall be set to zero |
| usIODataReferenceParam | Reference parameter number |
| usIODataNormalisation | IO Data normalization |
| pvDefaultValue | Pointer to default Parameter Value. Shall be set to 0. |
| pvParameterText | Address of first text element in the Parameter text list. |

```
static const PARAMETER_BLOCK_T  g_Manufacturer_Specific_Parameter_Block =
{
     /* .bNumberOfParameters */ 2,
     /* .ptParameters        */ &g_Manufacturer_Specific_Parameter
};
```

| Variable | Description |
|---|---|
| bNumberOfParameters | Number of Parameter-items this list contains. |
| ptParameters | Pointer to the fist element in parameter list |

## 3.6 Module specific Configuration:

The module configurations of Encoder and Drive objects are significantly different. The Encoder example application use only Encoder configuration which defines sensor properties and scaling parameter. However in case of Profidrive application when the Standard Telegram 3 is supported the Encoder configuration shall also be defined.

### 3.6.1 Encoder configuration:

Encoder specific configuration structure is defined as follows:

```
const ENCODER_SCALING_PARAMETERS_T g_DefaultScalingParameters =
{
  /*.fScaling_Enabled          = */ 0,
  /*.ullMeasuringUnits         = */ 8192 ,
  /*.ullTotalMeasuringRange    = */ 8192,
  /*.ulPresetValue             = */ 0,
  /*.ulOffsetValue             = */ 0,
  /*.ulVelocityMeasuringUnit   = */ 0,
  /*.ulVelocityReferenceValue  = */ 0
};
```

| Variable | Description |
|---|---|
| fScaling_Enabled | Is set to true if scaling function control is supported |
| ullMeasuringUnits | **Rotary encoders**: The Singleturn resolution in measuring steps<br>**Linear encoders:** The measuring step in nm |
| ullTotalMeasuringRange | The total measuring range in measuring steps |
| ulPresetValue | Max parameter request / response length |
| ulVelocityMeasuringUnit | The coding of the velocity measuring units (not supported) |
| ulVelocityReferenceValue | 100% reference value for N2/N4 normalized v Encoder velocity output values (NIST_x). (not supported) |

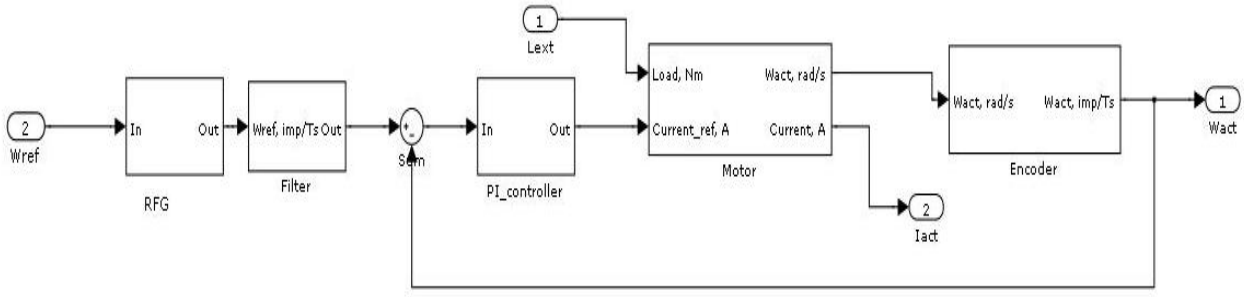
```
const ENCODER_CONFIG_T g_EncoderConfig =
{
  /*.bNumberOfSensors   = */ 1,
  /*.tSensorProp        = */
   {
     {
      /*.ulType            = */ ENCODER_TYPE_ROTARY_SINGLETURN,
      /*.ulResolution      = */ 0,
      /*.ulRevolution      = */ 0,
      /*.ulShiftFactor     = */ 0,
      /*.ulSupportedFaults = */ MSK_ENCODER_FAULTS_POSITON_ERROR,
       /*.fPresetEnabled    = */ 0,
     },
   },
};
```

| Variable | Description |
|---|---|
| bNumberOfSensors | Number of Sensor-items this list contains. Currently only one sensor is supported |
| ulType | Sensor Type. Possible values and their meanings are: |

| | |
|---|---|
| | 0 : ENCODER_TYPE_LINEAR_ABSOLUTE<br>1 : ENCODER_TYPE_LINEAR_INCREMENTAL<br>2 : ENCODER_TYPE_ROTARY_SINGLETURN<br>3 : ENCODER_TYPE_ROTARY_MULTITURN<br>4 : ENCODER_TYPE_ROTARY_INCREMENTAL |
| `ulResolution` | Rotary encoders: Number of pulses<br>Linear encoders: Length of a signal period in nm |
| `ulRevolution` | Max parameter request / response length |
| `ulShiftFactor` | how many bits defining the sum of quadrant information and fine resolution are displayed in Gx_XIST1 (not supported) |
| `ulSupportedFaults` | List of supported faults |
| `fPresetEnabled` | Is set to true if Preset function control is enabled (not supported) |

## 3.6.2 Drive Configuration

Profidrive example application implements a simple simulation of a DC Motor according to the following figure which shows the schematic of the closed loop control circuit.




Each block in the closed loop has a specific configuration structure and can be configured separately:

```
static MODEL_CONFIG_T g_ModelConfig =
{
      /* .ulCycleTime   = */   1000,
      /* .iMaxSpeed     = */   1800 / 60,
      /* .ulEncoderStep = */   4096

     /* .tController =  */
      {
      /* .ulControllerTs =  */ 0x00000106 ,
      /* .iIGain         =  */ 0x00000148 ,
      /* .iPGain         =  */ 0x0000051F ,
      /* .iImin          =  */ -655360
      /* .iImax          =  */ 655360
     },

      /* .tMotor = */
      {

      /* . ulMotorTs     =  */ 0x00000106,
```

```
      /* . ulElecConst   =  */ 0x000001C1 ,
      /* . iC_1_Gain     =  */ 0x0091FC43 ,
      /* . iC_2_Gain     =  */ 0x00000382 ,
      /* . iC_J_Gain     =  */ 0x0D610000 ,
      /* . iEMF_Gain     =  */ 0x000A0000 ,
      /* . iUmin         =  */ -1572864 ,
      /* . iUmax         =  */ 1572864 ,
     },

     /* .tEncoder  = */
     {
     /* .ulRadsToImpTs   =  */ 0x0000A72A ,
     },

    /* .tFilter = */
     {
     /* . FGAIN          =  */ 0x00190000 ,
     }
     /* .tRFG  = */
     {
     /* .ulRampupTime    =  */ 5000000 ,
     /* .ulRampDownTime  =  */ 8000000 ,
     }

};
```

**Global structure**

| Variable | Description |
|---|---|
| ulCycleTime | Cycle time in microseconds |
| iMaxSpeed | max speed 1800 RPM --> 30 Rotation per second |
| ulEncoderStep | Encoder steps of 4096 |

**Controller Block configuration**

| Variable | Description |
|---|---|
| ulControllerTs | Controller sample time<br>1ms -> 0.001s -> 0.001 * 65535 (dec) => 0x00000106 (Hex) |
| iIGain | controller I Gain<br>0.1 (dec) -> 0.1 * 66535 =  0x00000148 (Hex) |
| iPGain | Controller P Gain<br>0.02 (dec) -> 0.02 * 65535 ~= 0x0000051F (Hex) |
| iImin | Imin limit<br>-10 [Ampere] -> -10 * 65535 = -655350 |
| iImax | Imax limit<br>10 [Ampere] ->  10 * 65535 =  655350 |

**Motor Block configuration**

| Variable | Description |
|---|---|
| ulMotorTs | Motor sample  time |

| | |
|---|---|
| | 1ms -> 0.001s -> 0.001 * 65535 (dec) => 0x00000106 (Hex) |
| ulElecConst | Motor electrical constant C<br>0.00685 [N*m/A] -> 0.00685 * 65535 = 0x000001C1 |
| iC_1_Gain | 1 / ulElecConst -> 1/C<br>1/0.00685 ~= 145.98 (dec) -> 145.98 * 65535 = 0x0091FC43 |
| iC_2_Gain | 2 * ulElecConst -> 2 * C<br>2 * 0.00685 = 0.0137(dec)-> 0.0137 * 65535 = 0x00000382 |
| iC_J_Gain | C / J<br>0.00685/20e-7 ~= 3425 (dec) -> 3425 * 65535 ~= 0x0D610000 |
| iEMF_Gain | EMF Gain<br>10 -> 10 * 65535 = 655350 (dec) -> 0x000A0000 |
| iUmin | Umin limit<br>- 24 [Volt] -> -24 * 65535 = -1572840 |
| iUmax | Umax limit<br>24 [Volt] -> 24 * 65535 = 1572840 |

**Filter Block configuration**

| Variable | Description |
|---|---|
| ulFGain | FGAIN = 25<br>25 * 65535 = 1638400 (Dec) --> 0x00190000 |

**Encoder Block configuration**

| Variable | Description |
|---|---|
| ulRadsToImpTs | the step period<br>if Ts = 1ms and Nencoder = 4096 then<br>Gain = 4096 / 2 / 3.14 * 0.001 = ~0/.652 (dec) -><br>0.652 * 65535 = ~ 42794 (dec) -> 0x0000A72A |

**RFG Block configuration**

| Variable | Description |
|---|---|
| ulRampupTime | ramp up time in microseconds to reach the transmitted Setpoint |
| ulRampDownTime | ramp down time in microseconds to reach a standstill |