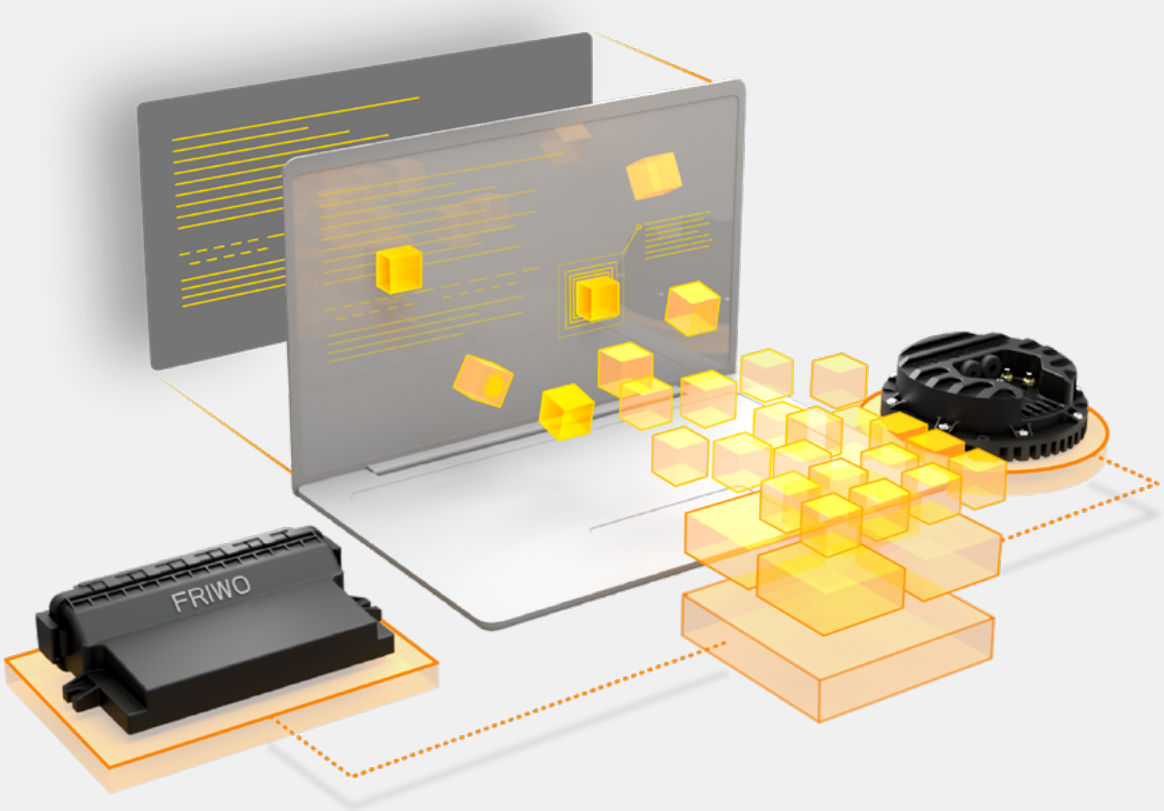
FRIWO | SDK

SOFTWARE DEVELOPMENT KIT

End-to-end Development Environment Setup Solution

APPLICATION GUIDE – HILL ASSIST



ABOUT

The **FRIWO SDK** enables the user to integrate own functionalities into a fully developed software environment for FRIWO motor controllers.

This document demonstrates how to implement a **Hill Assist** function by creating a customized TRQ_DES-module. Basically, the Hill Assist prevents the vehicle from rolling backwards for a parameterizable time when starting on hill and releasing the brake pedal.

This guide gives a step-by-step overview of the implementation workflow from project creation to flashing and testing of the generated firmware on hardware. Therefore, it assumes the **FRIWO SDK** Tool Environment to be set up already. For a guidance of the basic setup please refer to our **Quickstart Guide**.

https://friwo.link/ag/quickstart-guide

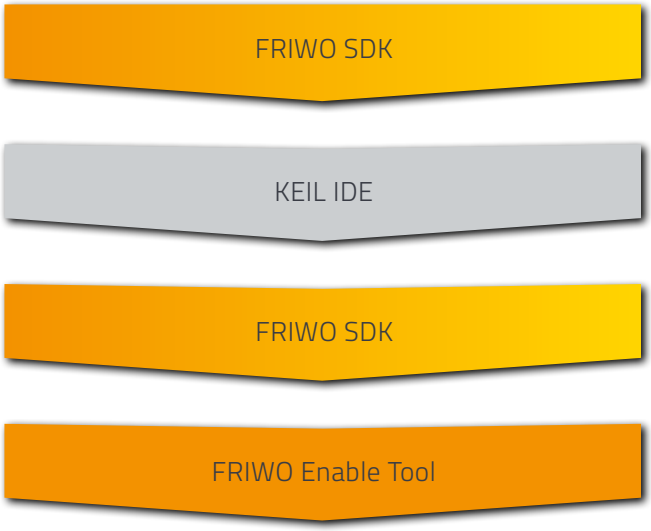
If you need a detailed description of the variable naming scheme, have a look at the **Software Manual**.

https://friwo.link/ag/manual

For a detailed description of the used module, please download our **Module Description**.

https://friwo.link/ag/module-description

THE BASIC WORKFLOW OF THE FRIWO SDK



We start, by creating a new project inside the **FRIWO SDK**. All necessary basic software gets pulled from the FRIWO Servers and made ready for usage.

In order to write your software, you need an IDE. We recommend **Keil IDE** or Visual Studio Professional.

After the software is written inside the IDE, we switch back to the **FRIWO SDK**. We start the compilation process and receive the customized firmware inside our workspace folder.

With the **FRIWO Enable Tool**, we can flash the customized firmware on the motor controller.

CREATE A NEW PROJECT

First, a new project is created using the following steps.

- Open FRIWO SDK
- In main view press **Select Project**
- **Name** the project (i.e. HillAssist)
- Choose your **workspace folder** path (default: C:\Users\USERNAME\Documents\SDK_Workspace)
- Choose framework **MCU FRIWO Standard V1.0**
- Press **Create**

Next, we define the module to be customized.

- In main view press **Select Module**
- Select module **TRQ_DES** to be customized
- Confirm the dialog window
- Press **OK**

A new project folder is created in your workspace folder. The project's subfolder .\module_TRQ_DES contains the c-files TRQ_DES_custom.c and TRQ_DES_custom.h, as well as a variable description file TRQ_DES_variables.xml.

PREPARE SOURCE FILES

Now the functionality of the Hill Assist can be put into practice by adjusting the generated c-files of the TRQ_DES-module. This is done by using the ANSI C IDE of your choice. The following procedure is explained using Keil µVision4 IDE.

For the first step we are having a look at the header-file TRQ_DES_custom.h.

- Open **Keil µVision4 IDE**
- Select **File->Open**
- Navigate to the project's workspace and there inside the subfolder .\module_TRQ_DES
- Select **TRQ_DES_custom.h**
- Press **Open**

To give you an overview, the header-file TRQ_DES_custom.h shows all interface variables of class SDK_Modul_Interfaces from basic software. These variables can be used inside the module TRQ_DES:

```
/*****************************************************************************\
   General SDK module interface variables | Width: 32
\*****************************************************************************/

/* Inputs */
extern SDK_Modul_Interfaces Float32 AIN1_Throttle /*
   Description: Desired driver torque, combined from all input-signal sources [%] */;
extern SDK_Modul_Interfaces Float32 AIN2_Throttle /*
   Description: Desired driver torque, combined from all input-signal sources [%] */;
extern SDK_Modul_Interfaces Float32 APP_Brake_Signal_Channel /*
   Description: Desired driver torque, combined from all input-signal sources [%] */;
extern SDK_Modul_Interfaces Float32 APP_Reverse_Gear_Signal_Channel /*
   Description: Desired driver torque, combined from all input-signal sources [%] */;
extern SDK_Modul_Interfaces Float32 APP_Throttle_Signal_Channel /*
   Description: Desired driver torque, combined from all input-signal sources [%] */;
extern SDK_Modul_Interfaces Float32 CAN_EXT_Reverse_Gear /*
   Description: Desired driver torque, combined from all input-signal sources [%] */;
extern SDK_Modul_Interfaces Float32 CAN_EXT_Torque_Request /*
   Description: Desired driver torque, combined from all input-signal sources [%] */;
extern SDK_Modul_Interfaces Float32 DIN_DIN1_Signal /*
   Description: Desired driver torque, combined from all input-signal sources [%] */;
extern SDK_Modul_Interfaces Float32 DIN_DIN2_Signal /*
   Description: Desired driver torque, combined from all input-signal sources [%] */;
extern SDK_Modul_Interfaces Float32 PWMI_Throttle /*
   Description: Desired driver torque, combined from all input-signal sources [%] */;
extern SDK_Modul_Interfaces Float32 MO_Rotor_Speed /*
   Description: Mechanical rotor-speed [1/s] */;
extern SDK_Modul_Interfaces Float32 SM_Trq_Control_status /*
   Description: Status of torque-control;StateList;0=Torque-control
deactivated;1=Torque-control active */;
extern SDK_Modul_Interfaces Float32 APP_Disp_Ride_Mode /*
   Description: Selected ride-mode */;


/* Outputs */
extern SDK_Modul_Interfaces Float32 TRQ_DES_Driver_Brake /*
   Description: Brake signal used for desired driver torque calculation [%] */;
extern SDK_Modul_Interfaces Float32 TRQ_DES_Driver_Reverse_Gear /*
   Description: Selected driving direction;StateList;0 = Forward gear selected;1 =
Reverse gear selected */;
extern SDK_Modul_Interfaces Float32 TRQ_DES_Driver_Throttle /*
   Description: Accelerator-pedal signal used for desired driver torque calculation
[%] */;
extern SDK_Modul_Interfaces Float32 TRQ_DES_Trq_Req_Rel /*
   Description: Desired driver torque, combined from all input-signal sources [%] */;
```

Note: The header-file does NOT have to be changed.

Next, we open the c-file TRQ_DES_custom.c in order to start programming.

- Select **File->Open**
- Navigate to the project's subfolder .\module_TRQ_DES
- Select **TRQ_DES_custom.c**
- Press **Open**

The commented part at the top of the c-file again lists all in- and output variables of the module, defined in TRQ_DES_custom.h as a default.

For our use case we define additional global variables which are accessible by FRIWO EnableTool and can be divided into two types:

- **Display** variables: Read only access (EMERGE_DISP_RAM)
- **Calibration** variables: Read and write access (EMERGE_NV_RAM_PAGE1)

The differentiation is done by using the keyword __attribute__ to provide the defined variables with the property to be stored in a specific RAM page or section.

- Define the following custom display variables using syntax __attribute__((section("EMERGE_DISP_RAM")))

```
__attribute__((section("EMERGE_DISP_RAM")))
volatile Float32 TRQ_DES_Throttle_Input
/* Description: Throttle input value after selection of input channel [%]; */;

__attribute__((section("EMERGE_DISP_RAM")))
volatile Float32 TRQ_DES_Brake_Input
/* Description: Brake input value after selection of input channel [%]; */;

__attribute__((section("EMERGE_DISP_RAM")))
volatile Float32 TRQ_DES_Reverse_Gear
/* Description: Shows if reverse gear is selected;  0 = forward gear selected; 1 =
reverse gear selected */;

__attribute__((section("EMERGE_DISP_RAM")))
volatile Float32 TRQ_DES_Flag_Upper_Lim
/* Description: Shows if desired torque has reached the upper bound of allowed
operational range */;

__attribute__((section("EMERGE_DISP_RAM")))
volatile Float32 TRQ_DES_Flag_Lower_Lim
/* Description: Shows if desired torque has reached the lower bound of allowed
operational range; */;

__attribute__((section("EMERGE_DISP_RAM")))
volatile UInt32 TRQ_DES_Hill_Assist_Disp_State
/* Description: Shows the actual state of hill-assist algorithm; 0 = Hill-Assist
not active; 1 = Entry state;
2 = Prepare for Throttle-Priorization; 3 = Accelerate with Brake on; 4 =
Accelerate with Brake released */;

__attribute__((section("EMERGE_DISP_RAM")))
volatile UInt32 TRQ_DES_Hill_Assist_Disp_Counter
/* Description: Shows the actual value of hill-assist counter to priorize
throttle; */;
```

- Define the following custom calibration variables using syntax __attribute__((section("EMERGE_NV_RAM_PAGE1")))

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
/* Define variables to be calibrated with FRIWO EnableTool                  */
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
/* Section EMERGE_NV_RAM_PAGE1:  Standart application data which will be stored in
Flash when writing a snapshot*/
/* All data types which can be chosen:  Int8, Int16, Int32, UInt8, UInt16, UInt32,
Bool and Float32  */

__attribute__((section("EMERGE_NV_RAM_PAGE1")))
volatile Float32 TRQ_DES_C_Test_Torque_Request_tmpl = 0.F /*
      Description: Template parameter for test torque request via custom variable
[%]; Limits: -100...100 */;

__attribute__((section("EMERGE_NV_RAM_PAGE1")))
volatile UInt32 TRQ_DES_C_Test_Reverse_Gear_tmpl = 0 /*
      Description: Template parameter to set reverse gear via custom variable
[%]; Limits: 0...1 */;

__attribute__((section("EMERGE_NV_RAM_PAGE1")))
volatile UInt32 TRQ_DES_C_ThrottleBrakeComb_Cut_Time_tmpl = 10000 /*
      Description: Template parameter for time in milliseconds after which the
accelerator-pedal signal will be
      cut-off on parallel use of throttle and brake [ms]; Limits: 0...4294967295
*/;

__attribute__((section("EMERGE_NV_RAM_PAGE1")))
volatile Float32 TRQ_DES_C_ThrottleBrakeComb_Max_Speed_tmpl = 2.F /*
      Description: Template parameter to define a rotor speed threshold at which
the accelerator-pedal signal
      will always be cut-off by using the brake in parallel [1/s]; Limits:
-1...2000 */;
```

Next create a local function used as saturation for signals which are limited in range.

- Define a function called “Saturation”

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
/* Define local functions                                                                 */
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/

Float32 Saturation(Float32 lower_lim, Float32 upper_lim, Float32 sig_in){
	Float32 sig_out = 0.F;

	if (sig_in > upper_lim) {
		sig_out = upper_lim;
	}
	else {
		if (sig_in < lower_lim) {
			sig_out = lower_lim;
		}
		else {
			sig_out = sig_in;
		}
	}
	return sig_out;
}
```

IMPLEMENT HILL ASSIST

At this point we're done defining global variables and local functions. Now we can dive into the function **TRQ_DES_custom** and focus on the Hill Assist itself.

- Define local static variables used for SR-Flip-Flop and State-Machine

```
void TRQ_DES_custom(void){

/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
/* Define local variables                                                    */
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
static int SR_ff_Q = 0;
static int SR_ff_Qn = 1;
static int SR_ff_S = 0;
static int SR_ff_R = 0;

struct state_machine {
      int state[10];
};

static struct state_machine TRQ_DES_Hill_Assist;
static UInt32 TRQ_DES_Hill_Assist_ctr = 0;
```

- Assign the throttle and brake signals from the analog input interface variables and saturate them from 0 to 100

```
 TRQ_DES_Throttle_Input = AIN1_Throttle;

/* Saturation */
   TRQ_DES_Throttle_Input = Saturation(0.F, 100.F, TRQ_DES_Throttle_Input);


 TRQ_DES_Brake_Input = AIN2_Throttle;

/* Saturation */
   TRQ_DES_Brake_Input = Saturation(0.F, 100.F, TRQ_DES_Brake_Input);
```

At this point we also need to feed the cross connections to other modules such as the State Manager.
This step is important as the State Manager checks these signals to enable a safe system startup.

- Assign the module's internal variables to output interface variables connected to other modules such as the State Manager

```
/* Cross connections of "raw" input values for state management and system
startup */
TRQ_DES_Driver_Throttle = TRQ_DES_Throttle_Input;
TRQ_DES_Driver_Brake = TRQ_DES_Brake_Input;
TRQ_DES_Driver_Reverse_Gear = 0;
```

Note: For simplicity of this application guide the signal for reverse gear selection TRQ_DES_Driver_Reverse_Gear is set to zero implying only forward gear.

- Invert the brake signal TRQ_DES_Brake_Input

```
/* Invert brake signal for desired torque calculation */
TRQ_DES_Brake_Input = -TRQ_DES_Brake_Input;
```

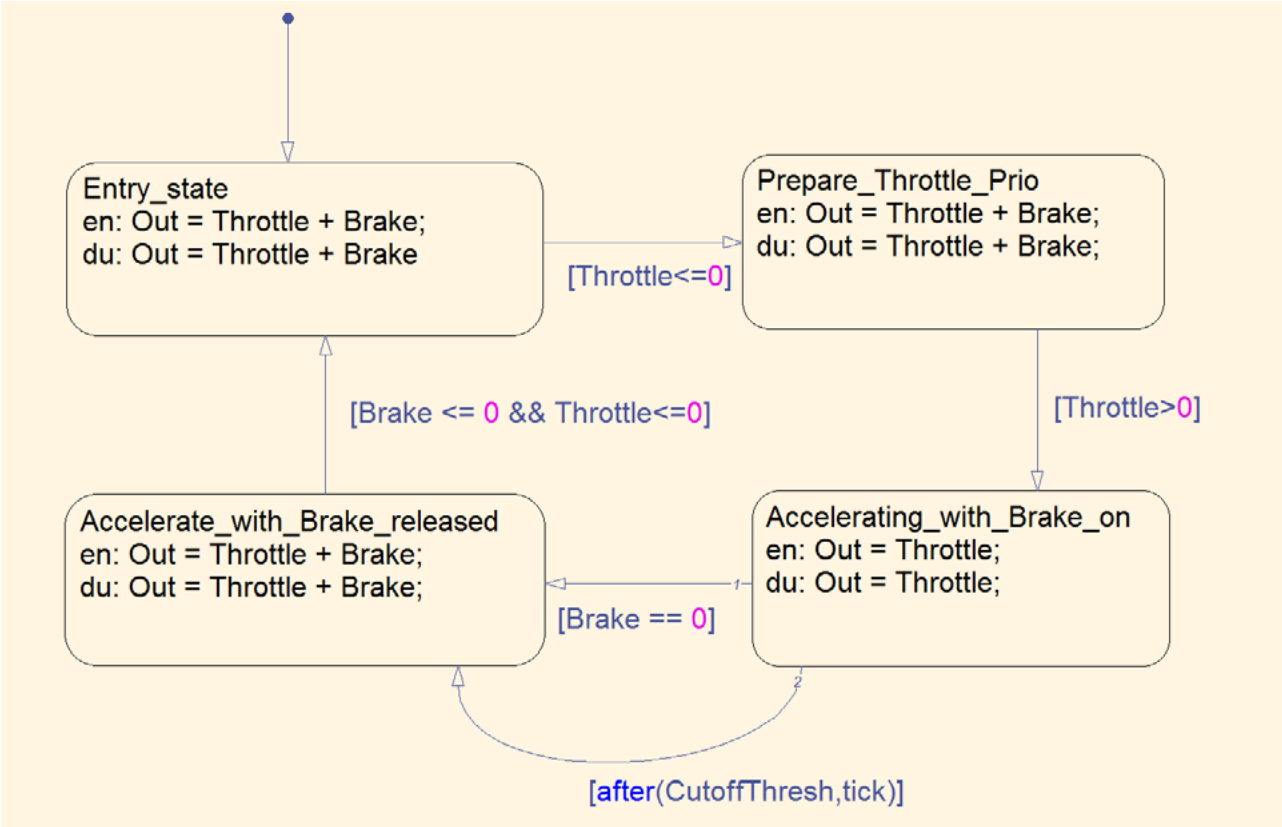
In the next step we define the conditions, when to activate and deactivate the Hill Assist. Activation should be done, if throttle input is lower than or equal to zero and motor speed is below a certain threshold TRQ_DES_C_Throttle-BrakeComb_Max_Speed_tmpl. Once activated, Hill Assist should stay active as long as motor speed is below the mentioned threshold, independent of the throttle input. To realize the state memory, a SR-Flip-Flop is implemented.

- Set up the conditions for Hill Assist activation using a SR-Flip-Flop

```
/* Check if motor speed is below speed threshold and throttle input is zero to
activate hill assist */

if(abs(MO_Rotor_Speed) > TRQ_DES_C_ThrottleBrakeComb_Max_Speed_tmpl){
      SR_ff_S = 1;
}
else{
      SR_ff_S = 0;
}

if(TRQ_DES_Throttle_Input <= 0 && !SR_ff_S){
      SR_ff_R = 1;
}
else{
      SR_ff_R = 0;
}

/* S-R flip flop to activate hill assist */
 if (SR_ff_S && (!(SR_ff_R))) {
SR_ff_Qn = 0;
SR_ff_Q = 1;
 }
 else {
if ((!(SR_ff_S)) && SR_ff_R) {
   SR_ff_Qn = 1;
   SR_ff_Q = 0;
}
else {
   if (SR_ff_S && SR_ff_R) {
      SR_ff_Qn = 1;
      SR_ff_Q = 0;
   }
   else {
      SR_ff_Q = !(SR_ff_Qn);
   }
}
}
```

The Hill Assist is then called, if the condition SR_ff_Qn == 1 holds. Otherwise, throttle and brake inputs are simply added up.

The Hill Assist itself is divided into four states, illustrated in the following state-flow chart.



As soon as the Hill Assist is activated, the state machine passes the state "Entry state" and switches to state "Prepare_Throttle_Prio", where it waits for throttle signal to be bigger than zero.

If the driver requests a positive torque, the state machine jumps to the next state "Accelerating_with_Brake_on". Here the throttle signal gets prioritized over the brake signal, so the requested torque is as high making acceleration on hill possible.

The state machine remains in this state as soon as either a specified time value TRQ_DES_C_ThrottleBrakeComb_Cut_Time_tmpl is reached by the counter TRQ_DES_Hill_Assist_ctr, or the driver releases the brake pedal.

If both, the signal for throttle and the one for brake are lower than or equal to zero, the state machine jumps back into the state "Entry state".

- Implement the Hill Assist using the condition SR_FF_Qn == 1 and the previously defined state machine structure TRQ_DES_Hill_Assist with the states 0 to 3

```
if(SR_ff_Qn){
/* Hill Assist */
if (TRQ_DES_Hill_Assist.state[0]) {
        TRQ_DES_Hill_Assist_Disp_State = 1;
  if (TRQ_DES_Throttle_Input <= 0.F) {
        TRQ_DES_Hill_Assist.state[0] = 0;
        TRQ_DES_Hill_Assist.state[1] = 1;
        TRQ_DES_Trq_Req_Rel = TRQ_DES_Throttle_Input + TRQ_DES_Brake_Input;
  }
  else {
        TRQ_DES_Trq_Req_Rel = TRQ_DES_Throttle_Input + TRQ_DES_Brake_Input;
  }
}
else {
  if (TRQ_DES_Hill_Assist.state[1]) {
        /* Prepare for Throttle-Priorization */
        TRQ_DES_Hill_Assist_Disp_State = 2;
        if (TRQ_DES_Throttle_Input > 0.F) {
               TRQ_DES_Hill_Assist.state[1] = 0;
               TRQ_DES_Hill_Assist_ctr = 0;
               TRQ_DES_Hill_Assist.state[2] = 1;
               TRQ_DES_Trq_Req_Rel = TRQ_DES_Throttle_Input;
        }
        else {
               TRQ_DES_Trq_Req_Rel = TRQ_DES_Throttle_Input + TRQ_DES_Brake_Input;
        }
  }
  else {
        if (TRQ_DES_Hill_Assist.state[2]) {
               /* Accelerate with Brake on */
               TRQ_DES_Hill_Assist_Disp_State = 3;
               TRQ_DES_Hill_Assist_ctr++;
               if (TRQ_DES_Brake_Input == 0.F) {
                  TRQ_DES_Hill_Assist.state[2] = 0;
                  TRQ_DES_Hill_Assist_ctr = 0;
                  TRQ_DES_Hill_Assist.state[3] = 1;
                  TRQ_DES_Trq_Req_Rel = TRQ_DES_Throttle_Input + TRQ_DES_Brake_Input;
               }
               else {
                  if (TRQ_DES_Hill_Assist_ctr >= TRQ_DES_C_ThrottleBrakeComb_Cut_Time_tmpl) {
                              TRQ_DES_Hill_Assist.state[2] = 0;
                              TRQ_DES_Hill_Assist_ctr = 0;
                              TRQ_DES_Hill_Assist.state[3] = 1;
                              TRQ_DES_Trq_Req_Rel = TRQ_DES_Throttle_Input + TRQ_DES_Brake_Input;
                  }
                  else {
                         TRQ_DES_Trq_Req_Rel = TRQ_DES_Throttle_Input;
                  }
               }
               TRQ_DES_Hill_Assist_Disp_Counter = TRQ_DES_Hill_Assist_ctr;
        }
        else {
               if (TRQ_DES_Hill_Assist.state[3]) {
                     /* Accelerate with Brake released */
                     TRQ_DES_Hill_Assist_Disp_State = 4;
                  if ((TRQ_DES_Throttle_Input <= 0.F) && (TRQ_DES_Brake_Input <= 0.F)) {
                              TRQ_DES_Hill_Assist.state[3] = 0;
                              TRQ_DES_Hill_Assist.state[0] = 1;
                              TRQ_DES_Trq_Req_Rel = TRQ_DES_Throttle_Input + TRQ_DES_Brake_Input;
                  }
                  else {
                              TRQ_DES_Trq_Req_Rel = TRQ_DES_Throttle_Input + TRQ_DES_Brake_Input;
                  }
               }
               else {
                     /* Entry state */
                     TRQ_DES_Hill_Assist_Disp_State = 1;
                     TRQ_DES_Hill_Assist.state[0] = 1;
                     TRQ_DES_Trq_Req_Rel = TRQ_DES_Throttle_Input + TRQ_DES_Brake_Input;
               }
        }
  }
}
}
else{
TRQ_DES_Hill_Assist_ctr = 0;
TRQ_DES_Hill_Assist_Disp_State = 0;
TRQ_DES_Trq_Req_Rel = TRQ_DES_Throttle_Input + TRQ_DES_Brake_Input;
}
```

- Finally, saturate the requested torque output and use flags to show if the signal reached a specific limit

```
        /* Saturation */
        if (TRQ_DES_Trq_Req_Rel > 100.F) {
                TRQ_DES_Flag_Upper_Lim = 1;
                TRQ_DES_Trq_Req_Rel = 100.F;
        }
        else {
                if (TRQ_DES_Trq_Req_Rel < -100.F) {
                TRQ_DES_Flag_Lower_Lim = 1;
                TRQ_DES_Trq_Req_Rel = -100.F;
                }
        }
}
```

- In Keil menu bar select **File->Save all** to save your file inside the project workspace

PREPARE VARIABLE DESCRIPTION

To display the previously defined global variables in FRIWO EnableTool, we need to describe these variables in the generated file TRQ_DES_variables.xml.

- Navigate to the project's subfolder .\module_TRQ_DES
- Open TRQ_DES_variables.xml with a text editor of your choice
- Add all global variables defined at the beginning of the source file TRQ_DES_custom.c (see also section "Prepare Source Files") using the following format for each.

```
<ddObj Name="TRQ_DES_Throttle_Input" Kind="Variable">
   <ddProperty Name="Description">Throttle input value after selection of input
channel [%]; Limits: 0...100</ddProperty>
   <ddProperty Name="Type">Float32</ddProperty>
   <ddProperty Name="Scaling">./LocalScaling</ddProperty>
   <ddProperty Name="Value"></ddProperty>
   <ddProperty Name="Min">0</ddProperty>
   <ddProperty Name="Max">100</ddProperty>
   <ddProperty Name="Address"></ddProperty>
   <ddObj Name="LocalScaling" Kind="Scaling">
      <ddProperty Name="LSB">1</ddProperty>
      <ddProperty Name="Unit">s</ddProperty>
   </ddObj>
</ddObj>
```

Note: All variables used in this application guide are of type Float32 and UInt32.

GENERATE FIRMWARE

At this point we have successfully prepared the c- and variable description files for implementation of the Hill Assist. In the following we use FRIWO SDK to compile our individual module TRQ_DES_custom and integrate it into the basic software framework.

- Open FRIWO_SDK (if not already open)
- In main view press **Select Project**
- Press **Load Project**
- **Navigate** to the project folder in your SDK workspace (i.e. .\SDK_Workspace\HillAssist)
- Select the project file (i.e. HillAssist.sdkproj) and press **open**

Now the default project settings are loaded. We can check if the right module (TRQ_DES_custom) is selected by pressing **Select Module** in the main view.

- Press **Compile** to build the firmware

The compile process is finished if the process bar is fully loaded and the status shows "Finished – Click here to view your firmware!". By clicking on the text, you are navigated directly to the output build folder of the firmware file (*.eef). This folder is generated inside the project path (i.e. .\HillAssist\FirmwareRelease\RELEASEID)

If an error occurs during the compilation process, refer to our **SDK Manual** using the indicated error code.

https://friwo.link/ag/manual

UPDATE MOTOR CONTROLLER

For updating the Motor Controller with the generated firmware we have to switch from FRIWO SDK to **FRIWO EnableTool**.

For further information about updating the Motor Controller please refer to our **Enable Tool manual**.

https://friwo.link/ag/et-manual

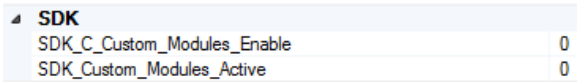
CALIBRATE HILL ASSIST

At first, make sure the following requirements are given:

- System is in safe conditions, i.e. motor is in standstill
- FRIWO EnableTool is running
- Motor Controller with customized Hill Assist firmware is connected via USB
- Correct .xml-file for variable configuration is loaded in FRIWO EnableTool
- Throttle pedal is connected to AIN1-Interface
- Brake pedal is connected to AIN2-Interface

Now we have a look at the SDK setup:

- In FRIWO EnableTool variable list on the right select the SDK dropdown



- Set SDK_C_Custom_Modules_Enable to 1 to activate the TRQ_DES_custom module **Note:** Activation/Deactivation of SDK modules is only possible, if the motor is in standstill.

TRQ_DES_custom	
TRQ_DES_Brake_Input	0
TRQ_DES_C_Test_Reverse_Gear_tmpl	0
TRQ_DES_C_Test_Torque_Request_tmpl	0
TRQ_DES_C_ThrottleBrakeComb_Cut_Time_tmpl	10000
TRQ_DES_C_ThrottleBrakeComb_Max_Speed_tmpl	2
TRQ_DES_Flag_Lower_Lim	0
TRQ_DES_Flag_Upper_Lim	0
TRQ_DES_Hill_Assist_Disp_Counter	0
TRQ_DES_Hill_Assist_Disp_State	2
TRQ_DES_Reverse_Gear	0
TRQ_DES_Throttle_Input	0
TRQ_DES_default	
TRQ_DES_Trq_Req_Rel	0

The variable SDK_Custom_Modules_Active = 1 indicates that the TRQ_DES_custom module is executed on the Motor Controller.

- In the EnableTool variable list on the right select the TRQ_DES_custom dropdown to show the previously defined variables

TRQ_DES_custom	
TRQ_DES_Brake_Input	-1
TRQ_DES_C_Test_Reverse_Gear_tmpl	0
TRQ_DES_C_Test_Torque_Request_tmpl	-1
TRQ_DES_C_ThrottleBrakeComb_Cut_Time_tmpl	10000
TRQ_DES_C_ThrottleBrakeComb_Max_Speed_tmpl	2
TRQ_DES_Flag_Lower_Lim	0
TRQ_DES_Flag_Upper_Lim	0
TRQ_DES_Hill_Assist_Disp_Counter	8141
TRQ_DES_Hill_Assist_Disp_State	3
TRQ_DES_Reverse_Gear	0
TRQ_DES_Throttle_Input	8.271
TRQ_DES_default	
TRQ_DES_Trq_Req_Rel	8.271

In initial operation, when TRQ_DES_Throttle_Input and TRQ_DES_Brake_Input both equal zero, the Hill Assist state machine dwells in state "Prepare_Throttle_Prio", indicated by TRQ_DES_Hill_Assist_Disp_State = 2 (Range: 1...4).

- Simultaneously hold the brake pedal and accelerate while watching the two variables TRQ_DES_Hill_Assist_Disp_Counter and TRQ_DES_Hill_Assist_Disp_State

The state machine jumps into state "Accelerating_with_Brake_on", indicated by TRQ_DES_Hill_Assist_Disp_State = 3. During this time, the requested torque TRQ_DES_Trq_Req_Rel equals the signal TRQ_DES_Throttle_Input, which is prioritized. The variable TRQ_DES_Hill_Assist_Disp_Counter shows the actual value of the Hill Assist counter and starts when entering the state "Accelerating_with_Brake_on".

If this counter reaches the parameterizable value of TRQ_DES_C_ThrottleBrakeComb_Cut_Time_tmpl = 10000 (in milliseconds), or if the brake pedal is released, the state machine jumps into state "Accelerate_with_Brake_released". In order to get back into initial state, both the throttle and brake pedal must be released.

Apart from the general Hill Assist state-flow functionality, the state machine will be deactivated, if the motor speed exceeds the parameterizable value of TRQ_DES_C_ThrottleBrakeComb_Max_Speed_tmpl = 2 (in 1/s). If this condition holds, both signals TRQ_DES_Throttle_Input and TRQ_DES_Brake_Input are added up in this application.

Have fun programming!

Feedback

We are working very hard to improve our products and therefore **feedback** is indispensable! Please send us your valuable feedback as contact form or via Mail to feedback@friwo.com

https://friwo.link/ag/feedback