Sensorless ACIM Field-Oriented Control on DSC 56F837xx

1. Introduction

This application note describes the implementation of the sensorless motor control reference application software for a 3-phase AC Induction Machine (ACIM) on the DSC MC56F83783 devices. The sensorless control software and ACIM control theory in general is described in *Sensorless ACIM Field-Oriented Control* (document DRM150). The High-Voltage Motor-Control Platform, consisting of the HVP-MC3PH power stage and the HVP-56F83783 controller card (described at the beginning of this document), is used as the hardware platform for the ACIM control reference solution. The hardware-dependent part of the sensorless control software, which includes the peripheral setup and the Motor Control Peripheral Drivers (MCDRV), is described as well. The last part of the document describes the user interface represented by the Motor Control Application Tuning (MCAT) page based on the FreeMASTER run-time debugging tool. These tools represent a simple and user-friendly way of algorithm tuning, software control, debugging, and diagnostics.

Contents

2. High-Voltage Motor-Control Platform

The ACIM reference application is available only for the 3-phase High-Voltage Motor-Control Platform (HVP), which is a 115/230-VAC, 1-kW power stage and a part of the HVP-MC3PH kit. In combination with the HVP-56F83783 controller card, it provides a software development platform for more than one-horse-power high-voltage motors. The block diagram of the complete HVP with the HVP-56F82748 card is shown in Figure 1.

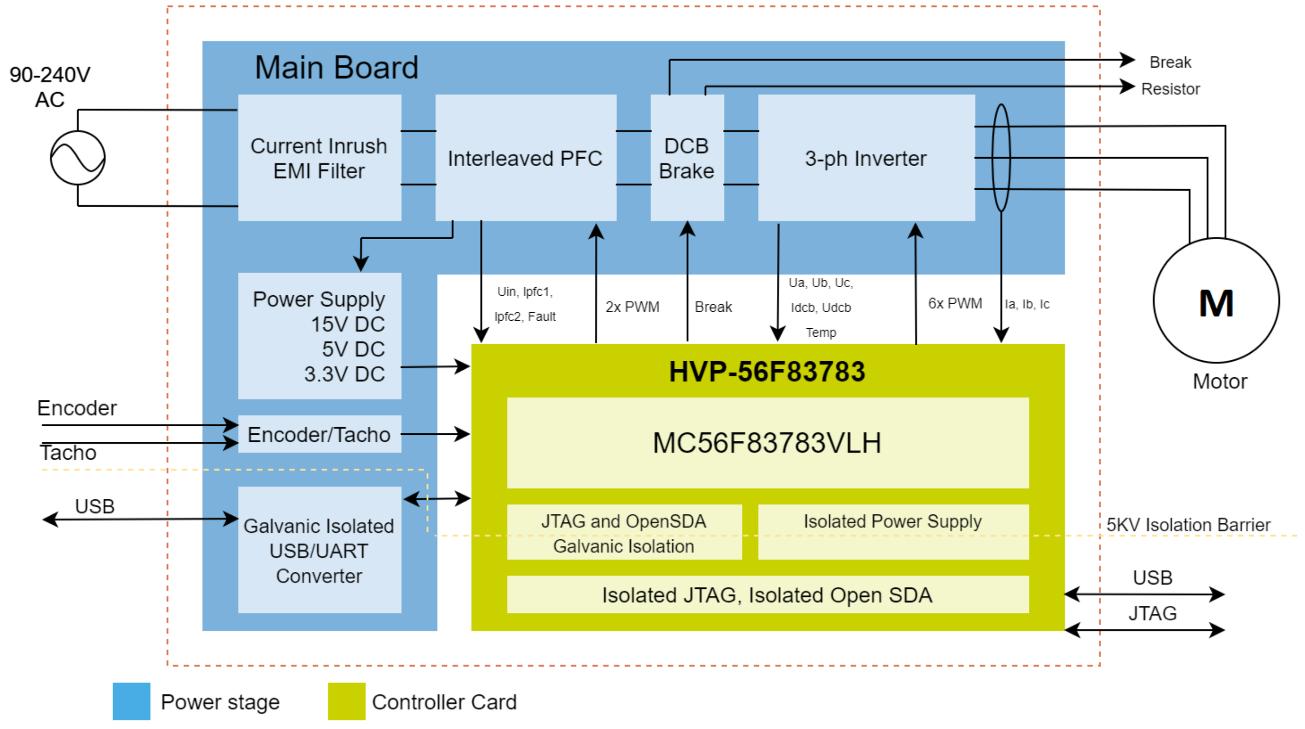


Figure 1. HVP block diagram

The HVP power stage setup is easy and straightforward. See the *High-Voltage Motor-Control Platform User's Guide* (document HVPMC3PHUG) and *DSC ACIM Control Reference Application Package User's Guide* (document UM11206) for more information about the HVP setup.

CAUTION

Due to the presence of high voltage, the HVP represents a safety risk when not used properly. For more information about the HVP, see www.nxp.com.

3. MCU peripheral settings

This section focuses on the hardware-dependent part of code for all supported MCUs, which includes the peripheral initialization and explanation of the application timing. The description is valid for the DSC ACIM version 1.0.0.

3.1. DSC56800EX Quick Start tool

The MCU pins, clocks, and peripherals used are configured by the Graphical Configuration Tool which is a component of the DSC56800EX Quick Start tool. The MCU configuration is saved in the *appconfig.h* file that is included by the application source code. For more information, see the DSC56800EX Quick Start User's Guide (document DSC56800EXQSUG).

3.2. DSC MC56F837xx family

MC56F823xx/7xx is a low-power DSP MCU family, offering outstanding power consumption at run time in a compact 5 x 5 mm package with exceptional performance, precision, and control for high-efficiency digital power conversion (MC56F837xx) and advanced motor control (MC56F837xx) applications. MC56F837xx includes advanced high-speed and high-accuracy peripherals, such as high-resolution Pulse Width Modulation (PWM) with 312-picosecond resolution, dual high-speed 12-bit Analog-to-Digital Converters (ADCs) with built-in PGA sampling of up to 1.25 Mega Samples Per Second (MSPS) at 12 bits. Faster application-specific control loops are driven via a 32-bit DSP core with single-cycle math computation, fractional arithmetic support, and parallel moves. For more information, see the *MC56F837xx Reference Manual* (document MC56F83XXXRM).

The HVP-56F83783 controller card is based on the MC56F83783VLH device. The controller card is equipped with the JTAG open-standard serial and debug interface. For more information about the HVP-56F83783 controller card, see the HVP-56F83783 page.

The peripherals (whose setup is detailed later on in this chapter) used by the ACIM motor-control software on MC56F83783 are:

- 12-bit cyclic Analog-to-Digital Converter (ADC12) for the phase currents, DC-bus voltage, and IPM temperature measurement.
- eFlexPWM module (PWM) for the 6-channel PWM generation.
- Periodic Interrupt Timer (PIT) for the slow control loop timing.
- XBARA multiplexer for the over-current fault and ADC12 trigger routing.
- Serial Interface (QSCI0) for the FreeMASTER communication.
- General-Purpose Input/Output (GPIO) pins for the inrush relay and brake circuit control.

The application timing diagram is shown in Figure 2. All tasks are handled using these interrupt service routines:

- ADC_EOS_isr ()—level-two priority interrupt triggered when the conversion of all enabled samples is completed by the ADCA. It handles the fast control loop of the FOC and the FreeMASTER recorder feature.
- PIT_ISR ()—level-one priority interrupt triggered by the overflow of the PIT. It handles the slow control loop of the FOC.

The fast and slow control loop ISRs are described in more detail in *Sensorless ACIM Field-Oriented Control* (document DRM150).

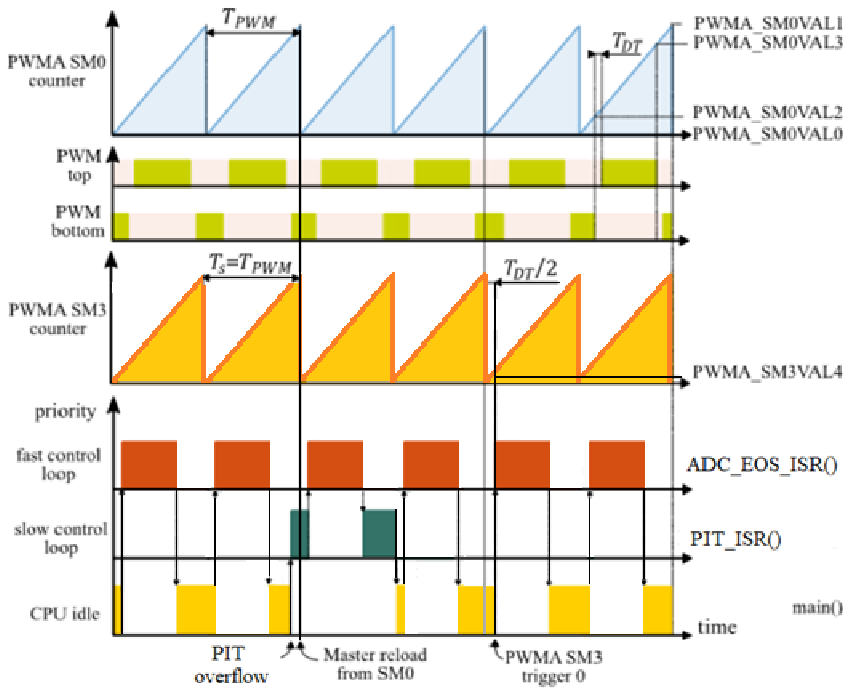


Figure 2. Example of application timing on MC56F83783

The PWM sub-module 0 (SM0) timer internal counter counts from the PWM_SM0VAL0 value to the PWM_SM0VAL1 value with the T_{PWM} period. The switching of the transistors on each motor phase is determined by the PWMA_SM[0..2]VAL2 and PWMA_SM[0..2]VAL3 register pair on PWMA SM0, SM1, and SM2. The dead time, which delays the rising edge of the transistor control signals by T_{DT}, is inserted to avoid short circuit on the DC-bus.

The selection of the PWM switching frequency affects the switching power losses (lower frequency is better) and audible noise (higher frequency is better). This reference solution offers the possibility to easily increase the ratio between the FOC sampling period T_s and the PWM period T_{PWM} (see Section 4.1, “MCDRV initialization and configuration”). The example in Figure 2 shows the case when T_s equals T_{PWM}.

The ADCA and ADCB (because both ADC12s run in the triggered parallel mode) are triggered by the PWM SM0 trigger 0 signal, which is connected to the ADC via XBARA. The trigger is issued when the SM0 internal counter reaches the PWMA_SM0VAL4 value, which is set to $T_{DT}/2$ by default (this value ensures correct ADC sampling even at a very high duty cycle). The internal counter of SM0 is reloaded by the master reload trigger event from SM0. ADC12 converts a total of four samples at the beginning of sampling period T_s:

- The first two samples on the ADCA (channel 1 for phase A or 6 for phase C) and ADCB (channel 2 for phase B or 7 for phase C) are the samples of phase currents.
- The DC-bus voltage is sampled second by the ADCA channel 3.
- The IPM temperature is sampled second by the ADCB channel 0.

When all the samples are converted, processing of the ADC_EOS_isr () high-priority ISR starts.

The CPU load and memory usage for the ACIM FOC reference software (see the *DSC ACIM Control Reference Application Package User's Guide* (document UM11206) for more details) is shown in Table 1. The results apply to the application built using the CodeWarrior 11.x IDE with the maximum speed optimization. The memory usage is calculated from the linker *.map* file, including the 2-KB FreeMASTER recorder buffer (allocated in RAM).

Table 1. MC56F83783 CPU and memory usage

—	MC56F83783
CPU clock [MHz]	100
Fast Control Loop (%)	41.12
Slow Control Loop (%)	1.08
Total CPU load [%]	42.2
Flash usage [B]	18 718
RAM usage [B]	4 460

3.2.1. On-Chip Clock Synthesis (OCCS)

The MC56F837xx DSC uses the OCCS and SIM modules to configure and distribute the clock across the peripheral modules. The OCCS module provides several clock-source options for the MCU. The System Integration Module (SIM) provides system control and chip configuration. The OCCS module configuration is as follows:

- The 8-MHz clock from the 48-MHz/6 internal oscillator is used as the reference clock source.
- The PLL is used to generate the 100-MHz OCCS core output clock.
- The peripheral clock frequency is set to 100 MHz.

3.2.2. Periodic Interrupt Timer (PIT)

The PIT peripheral module is used for the slow control loop timing. The PIT module is configured as follows:

- The input clock is set to 195,312 KHz (1/512 of the peripheral clock frequency).
- The interrupt with a level-one priority is enabled on the counter reaching the modulo value (195).
- The modulo is set so that the overflow interrupt occurs at the slow control loop period (1.00352 ms).

3.2.3. 12-bit cyclic Analog-to-Digital Converter (ADC12)

The ADC12 module is used to measure the phase currents, DC-bus voltage, the IPM temperature (a total of four samples are taken each sampling period). It consists of two converters (ADCA and ADCB).

The ADC12 module is configured as follows:

- The input clock is set to 20 MHz (1/5 of the peripheral clock frequency).
- The end-of-scan interrupt with a level 2 priority is enabled on the ADCA.
- The single-ended, 12-bit conversion with the hardware trigger from the PWMA is selected. The triggered parallel conversion is used on both ADCA and ADCB.
- Only the SAMPLE0, SAMPLE1, SAMPLE8, and SAMPLE9 samples are enabled.

3.2.4. Pulse Width Modulator A (PWMA)

The first three sub-modules of the eFlexPWM periphery PWMA are used to generate the 6-phase PWM for motor control with this setup:

- The input clock is set to $f_{PWMin} = 100$ MHz (fast peripheral clock frequency).
- The output PWM frequency is set to $f_{PWM} = 1/T_{PWM} = 10$ kHz. The PWMA_SM[0..2]INIT and PWMA_SM[0..2]VAL1 registers are used to define the PWM period and the PWMA_SM[0..2]VAL2 and PWMA_SM[0..2]VAL3 registers specify the current duty cycle.
- The counters at SM1 and SM2 are synchronized with the master sync signal from sub-module 0.
- The center-aligned, complementary PWM is generated only with a full cycle reload.
- A dead time of $T_{DT} = 1.5$ µs is inserted. This value is recommended by the manufacturer of the IPM used on the HVP-MC3PH board. The dead time counter modulo is set to PWMA_SM[0..2]DTCNT0 = PWMA_SM[0..2]DTCNT1 $= T_{DT} f_{PWMin} = 150.$
- Channels A and B at SM0, SM1, and SM2 are disabled on fault number 0 active, with automatic clearing (the PWM outputs are re-enabled at the first PWM reload after the fault disappears). Fault number 0 (connected to the IPM fault pin via GPIO, active in low) is enabled.

Sub-module 0 VAL4 is used for the ADC12 triggering with this setup:

- The trigger is issued when the PWMA_SM0VAL4 value is reached ($T_{DT}/2$ by default).

The eFlexPWM module is a dedicated peripheral enabling the generation of 3-phase PWM signals connected to the IPM H-bridge driver. The three PWM submodules used in the application are configured using the Graphical Configuration Tool (GCT), as listed here:

- PWM_0:
  - IPBus clock source of 100 MHz.
  - Running frequency of 10 kHz with 100-µs period.
  - INIT register—5000, VAL1 4999—13-bit resolution.
  - Complementary mode with 1.5-µs dead time.
  - PWM reload and synchronization signals generated at every opportunity from this module.
  - Trigger 4 enabled to provide synchronization with the ADC module via XBAR.
  - High-side and low-side PWM_A and PWM_B outputs in positive (active high) polarity.

- PWM_1 and PWM_2:

  - PWM_0 clock source.
  - Running frequency of 10 kHz with 100-µs period.
  - INIT register—5000, VAL1 4999—13-bit resolution.
  - Complementary mode with 1.5-µs dead time.
  - PWM reload and synchronization signals generated at every opportunity from this module.
  - High-side and low-side PWM_A and PWM_B outputs in positive (active high) polarity.
- PWM FAULT:
  - Fault 0 pin with a low fault level is connected via XBAR to fixed 10,5-A hardware over-current protection of the power module.
  - Fault 1 pin with a low fault level is connected via XBAR to a comparator output that creates adjustable over-current protection (the comparator compares the DC-bus current with an adjustable level of the DAC).
  - Fault input filter is disabled.

3.2.5. Inter-Peripheral Crossbar Switch A (XBARA)

The XBARA module is used to route the over-current fault signal from the IPM fault pin to the PWMA and to route the trigger signal from the PWMA to the ADC12. The XBARA module is set up as follows:

- The XBARA input IN6 (GPIO_C16/GPIO_F0) is connected to output OUT29 (PWMA_FAULT0).
- The XBARA input IN14 (CMPC_OUT) is connected to output OUT30 (PWMA_FAULT1).
- The XBARA input IN20 (PWMA0_MUX_TRIG0/PWMB0_OUT_TRIG0 signal) is connected to output OUT12 (ADCA_TRIG).

3.2.6. High-Speed Comparator C (HSCMP_C)

The high-speed comparator module is used for adjustable over-current fault detection. The output of comparator C is connected to the PWMA fault 1 input. The HSCMP_C module is configured as follows:

- The negative comparator input is connected to the CMPC_IN0 (GPIO_B3) – DC-bus current signal.
- The positive comparator input is connected to the 6-bit DAC. The DAC reference output voltage level can be set so that the over-current fault is activated when the DC-bus current ranges from 0,063 A to 7,937 A with a resolution of 63 mA.
- The comparator output polarity is set to normal: output high is when the positive input is higher than the negative input.

3.2.7. Universal Asynchronous Receiver and Transmitter (SC0)

The SCI0 module is used for the FreeMASTER communication between the MCU board and the PC. The module configuration is as follows:

- The baud rate is set to 19200 Bd.
- Both the receiver and the transmitter are enabled.
- Other settings are set to default.

3.2.8. General-Purpose Input/Output (GPIO)

These GPIO pins are used:

- Inrush relay control on GPIOF7.
- Braking circuit control on GPIOF6.
- LED state indication on GPIOC0.

4. Motor-Control Peripheral Drivers

The Motor-Control Peripheral Drivers (MCDRV) are a simple way of peripheral initialization and access for 3-phase ACIM or PMSM control. The features provided by the MCDRV library include the 3-phase PWM generation using Space Vector Modulation (SVM) and measurement of the 3-phase current, DC-bus voltage, and IPM temperature (or one general user-defined auxiliary quantity). The principles of both the 3-phase current measurement and the SVM are described in *Sensorless ACIM Field-Oriented Control* (document DRM150).

The MCDRV consist of peripheral driver library modules for each supported periphery. All the ADC and PWM periphery drivers share the same API within their class. This enables the higher-level code to be platform-independent, because the peripheral driver function calls are replaced by universally-named macros. The list of supported peripherals and APIs of their drivers is provided in Section 4.2, “MCDRV application interface”.

4.1. MCDRV initialization and configuration

The MCDRV initialization module consists of a set of MCU peripheral-initialization functions, as well as all the options that can be defined by the user. The functions are in the device-specific *mcdrv_hvp-<device>.c* source and *mcdrv_hvp-<device>.h* header files. Out of all functions in the MCDRV initialization module, it is only necessary to call the *MCDRV_Init_M1()* function once during MCU startup before calling any other MCDRV functions. All the peripherals used by the given device for motor-control purposes are then initialized within this function.

The *mcdrv_hvp-<device>.h* header file provides several options that you can define:

- M1_MCDRV_ADC—this macro specifies the ADC periphery used.
- M1_MCDRV_PWM3PH—this macro specifies the PWM periphery used.
- M1_MCDRV_TMR_SLOWLOOP—this macro specifies the timer for the slow control loop timing.

- M1_MCDRV_CMP—this macro specifies the comparator periphery for the over-current fault detection.
- M1_PWM_FREQ—the value of this definition sets the PWM frequency f_{PWM} in Hz.
- M1_FOC_FREQ_VS_PWM_FREQ—enables you to select a ratio between the sampling period T_s and the PWM period T_{PWM} (where $T_s = \text{M1\_FOC\_FREQ\_VS\_PWM\_FREQ} \times T_{PWM}$). This is convenient when the PWM frequency must be higher than the maximum fast-loop interrupt length due to the CPU performance restrictions.
- M1_SLOW_LOOP_FREQ—the value of this definition sets the slow loop period frequency in Hz.
- M1_PWM_PAIR_PH[A..C]—these macros enable a simple assignment of the physical motor phases to the PWM periphery channels or sub-modules. Alter the order of the motor phases this way. Only the values of 0, 1, and 2 can be assigned to these macros.
- OVER_CURRENT_THRESHOLD—adjustable over-current fault threshold detected by the comparator. This value can be set in the range from 250 to 7750 with a resolution of 250 (corresponds to the current threshold in mA).
- M1_BRAKE_[SET, CLEAR]—DC-bus brake circuit control macro.
- M1_ADC[0,1]_PH_[A..C]—these macros serve to assign the ADC channels for the phase-current measurement (the unassigned ADC channels are set to the ADC_NO_CHAN value). The general rule is that at least one of the phase currents must be measurable on both ADC converters and the remaining two phase currents must be measurable on different ADC converters. If this rule is broken, a pre-processor error is issued. The reason for this rule is that, to ensure a proper ADC measurement in a wide range of the PWM duty cycle, the selection of the phase-current pair to measure depends on the current SVM sector. For more information about the 3-phase current measurement, see *Sensorless ACIM Field-Oriented Control* (document DRM150).
- ADC[0,1]_UDCB and ADC[0,1]_AUX—these defines are used to select the ADC channel for the measurement of the DC-bus voltage and one user-defined auxiliary quantity, which is not used directly for motor control (the IPM temperature is measured by default). The rule for the ADC channel assignment is that the DC-bus voltage and the auxiliary quantity must be measurable on different ADC converters, so that the measurement can be done simultaneously. If this rule is broken, a pre-processor error is issued during the software build.

4.2. MCDRV application interface

The ADC and PWM motor-control drivers share the same API within their class. To ensure device independency on the MCDRV API, all driver functions are accessible through universally-named macros in the *mcdrv_hvp-<device>.h* file.

4.2.1. ADC control API description

The initialization macros are used to assign I/O variables (for example, to store the measurement results to the variables in your application). These macros are defined:

- M1_SET_PTR_I_ABC(*var*)—assigns a pointer to the GMCLIB_3COOR_T_F16 structure variable *var*, in which you want to store the phase current measurement results. The GMCLIB_3COOR_T_F16 datatype is defined in the Real-Time Control Embedded Software Libraries (RTCESL). For more information, see www.nxp.com/rtcesl.
- M1_SET_PTR_U_DC_BUS(*var*)—assigns a pointer to the 16-bit fractional variable *var*, in which you want to store the DC-bus voltage measurements.
- M1_SET_PTR_AUX_CHAN(*var*)—assigns a pointer to the 16-bit fractional variable *var*, in which you want to store the auxiliary quantity measured values.
- M1_SET_PTR_SECTOR(*var*)—assigns a pointer to the 16-bit unsigned integer variable *var* that contains the number of the current SVM sector.

NOTE

These macros must be executed before calling any MCDRV ADC functions. Otherwise, your application goes to a hard fault.

These functions are available:

- *bool_t* M1_MCDRV_CURR_3PH_CHAN_ASSIGN(MCDRV_ADC_T*)—calling this function assigns proper ADC channels for the next 3-phase current measurement based on the SVM sector. This function always returns *true*.
- *bool_t* M1_MCDRV_CURR_3PH_CALIB_INIT(MCDRV_ADC_T*)—this function initializes the phase current channel offset measurement. This function always returns *true*.
- *bool_t* M1_MCDRV_CURR_3PH_CALIB(MCDRV_ADC_T*)—this function reads the current information from the unpowered phases of a standstill motor and filters them using moving average filters. The goal is to obtain the value of the measurement offset. The length of the window for moving average filters is set to eight samples by default. This function always returns *true*.
- *bool_t* M1_MCDRV_CURR_3PH_CALIB_SET(MCDRV_ADC_T*)—this function asserts the phase current measurement offset values to the internal registers. Call it after a sufficient number of M1_MCDRV_CURR_3PH_CALIB() calls. This function always returns *true*.
- *bool_t* M1_MCDRV_ GET(MCDRV_ADC_T*)—this function reads and calculates the actual values of the 3-phase currents, DC-bus voltage, and auxiliary quantity and stores them in the variables defined by the user in the initialization macros (see above). This function always returns *true*.

4.2.2. PWM control API description

The initialization macros are used to assign the I/O variables (for example, to set the required duty cycles from your application). These macros are defined:

- M1_SET_PTR_DUTY(*var*)—sets the pointer to the GMCLIB_3COOR_T_F16 structure variable *var*, in which you define the required phase PWM duty cycles. The GMCLIB_3COOR_T_F16 datatype is defined in RTCESL.

NOTE

This macro must be executed before calling any MCDRV PWM functions. Otherwise, your application goes to a hard fault.

These functions are available:

- *bool_t* M1_MCDRV_PWM3PH_SET(M1_MCDRV_PWM_T*)—this function updates the PWM phase duty cycles based on the required values stored in the variable defined by the user in the initialization macros (see above). This function always returns *true*.
- *bool_t* M1_MCDRV_PWM3PH_EN(M1_MCDRV_PWM_T*)—calling this function enables all PWM channels. This function always returns *true*.
- *bool_t* M1_MCDRV_PWM3PH_DIS(M1_MCDRV_PWM_T*)—calling this function disables all PWM channels. This function always returns *true*.
- *bool_t* M1_MCDRV_PWM3PH_FAULT_GET(M1_MCDRV_PWM_T*)—this function returns and automatically clears the state of the over-current fault flags. This function returns *true* when an over-current event occurs. Otherwise, it returns *false*.

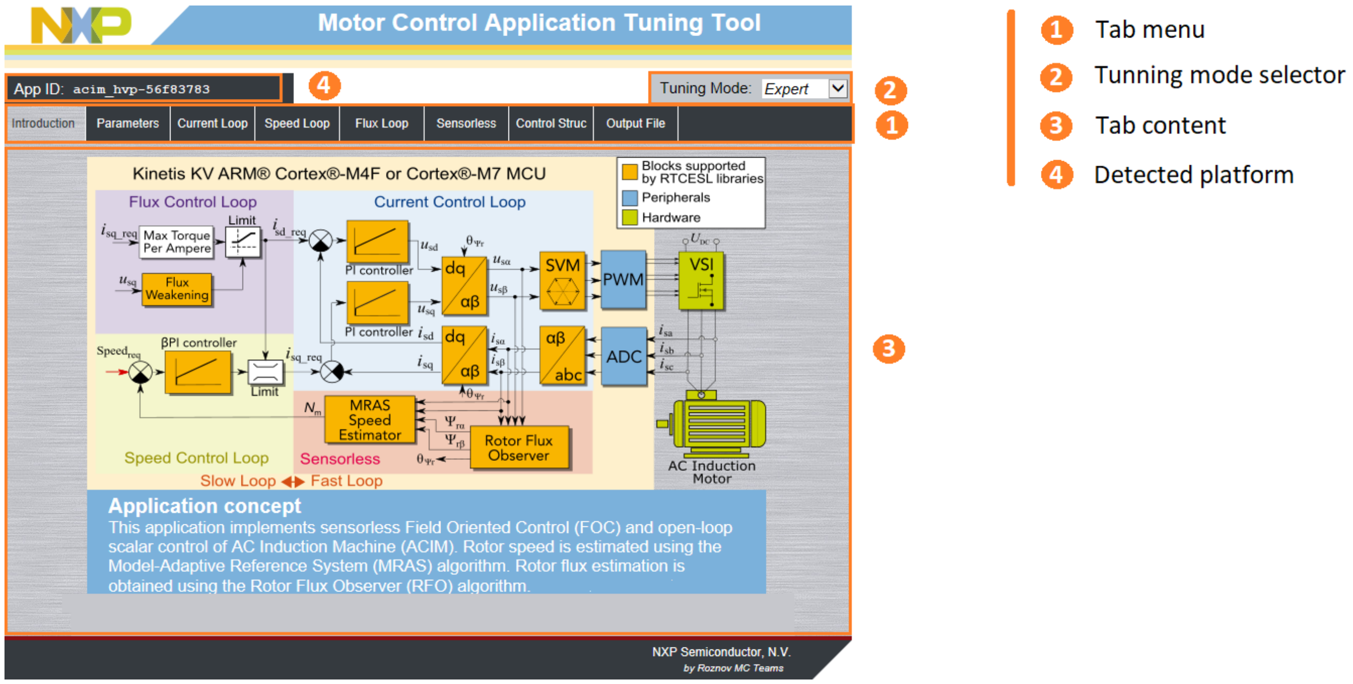
5. Tuning and controlling the application

This section provides information about the tools and recommended procedures for controlling the ACIM sensorless application. As the primary means of communication, the application contains an embedded-side driver of the FreeMASTER real-time debug monitor and a data visualization tool for communication with the PC. FreeMASTER supports non-intrusive monitoring, as well as modifying of target variables in real time, which is very useful for algorithm tuning. Besides the target-side driver, FreeMASTER requires installing the PC application as well. For more information, see www.nxp.com/freemaster.

The ACIM sensorless FOC application can be easily tuned using the Motor Control Application Tuning (MCAT) page for ACIM. The MCAT for ACIM is a user-friendly modular page, which runs within the FreeMASTER PC application. To launch it, execute the **.pmp* file located next to your ACIM sensorless project. When the communication with the MCU side of the application is established, the MCU platform is detected and a proper MCAT setup is used. Without a connection, many features are disabled and the pertinent files are generated next to the **.pmp* file. See the user's guide for your version of ACIM sensorless application for more information about the FreeMASTER communication setup (document UM11206). Figure 3 shows the MCAT for ACIM welcome page. The tool consists of the tab menu (point 1), the tuning experience level selector (point 2), the detected platform (point 3) and the tab content itself (point 4).

Each tab represents one sub-module, which enables you to tune and control different aspects of the application:

- “Introduction”—welcome page with the ACIM sensorless FOC diagram and a short description of the application.
- “Parameters”—this page enables you to modify the motor parameters, specification of hardware and application scales, and fault limits. For more information, see Section 5.2.1, “Input Application Parameters tab”.
- “Current loop”—specify the current loop PI controller gains, output limits, and default *d*-axis stator current reference here. For more information, see Section 5.2.3, “Current loop tuning”.
- “Speed loop”—this tab contains fields to specify the speed controller proportional and integral gains, as well as the output limits, parameters of the speed ramp, and startup procedure. For more information, see Section 5.2.4, “Speed loop tuning”.
- “Flux loop”—this tab is used to set up the *d*-axis current control, which includes the Max-Torque Per Ampere (MTPA) and Field-Weakening (FW) algorithm settings. For more information, see Section 5.2.5, “Flux loop tuning”.
- “Sensorless”—this page enables you to tune the parameters of the Rotor Flux Observer (RFO) for the rotor flux position estimator and the Model-Reference Adaptive System (MRAS) speed observer. For more information, see Section 5.2.2, “Sensorless rotor flux position and speed estimation”.
- “Control Struc”—the application control page enables you to choose between the scalar control (also known as Volt per Hertz or V/Hz) and FOC, where you can disable parts of the FOC cascade structure for tuning purposes. This tab enables you to set the required speed, stator currents, and stator voltage. It also provides information about the application state. For more information, see Section 5.1, “Application control using MCAT”.
- “Output file”—this tab enables you to view all the calculated constants that are required by the ACIM sensorless FOC control algorithms and generate a new *m1_acim_appconfig.h* application configuration header file. For more information, see Section 5.2.6, “MCAT output file generation”.
- “Control page”—this tab contains graphical elements, such as the speed gauge, DC-bus voltage measurement bar, and a variety of switches that enable simple, quick, and user-friendly application control. You may turn on the demo mode, which sets various pre-defined rotor speeds over time. For more information, see the user’s guide distributed with your version of ACIM sensorless application (document UM11206).



Application concept

This application implements sensorless Field Oriented Control (FOC) and open-loop scalar control of AC Induction Machine (ACIM). Rotor speed is estimated using the Model-Adaptive Reference System (MRAS) algorithm. Rotor flux estimation is obtained using the Rotor Flux Observer (RFO) algorithm.

Figure 3. MCAT for ACIM welcome page

Most of the tabs offer the possibility to immediately load the parameters specified in MCAT into the target using the “Update target” button, and save them to (or restore them from) the hard drive file using the “Store Data” (or “Reload Data”) button. The data stored using the “Store Data” button are automatically loaded the next time the MCAT is launched and the MCU communication is established. For more information about the application states, see *Sensorless ACIM Field-Oriented Control* (document DRM150).

The “Basic” and “Expert” tuning modes are available. Selecting the latter one grants you the access to modify all parameters and fields available in the MCAT. Using the “Expert” mode is not recommended for inexperienced users. When the MCAT operates in the offline mode, the “App Id” line reads “offline”. When the communication with the target MCU is established using a correct software, the “App Id” line displays the correct platform name and all stored parameters for the given MCU are loaded.

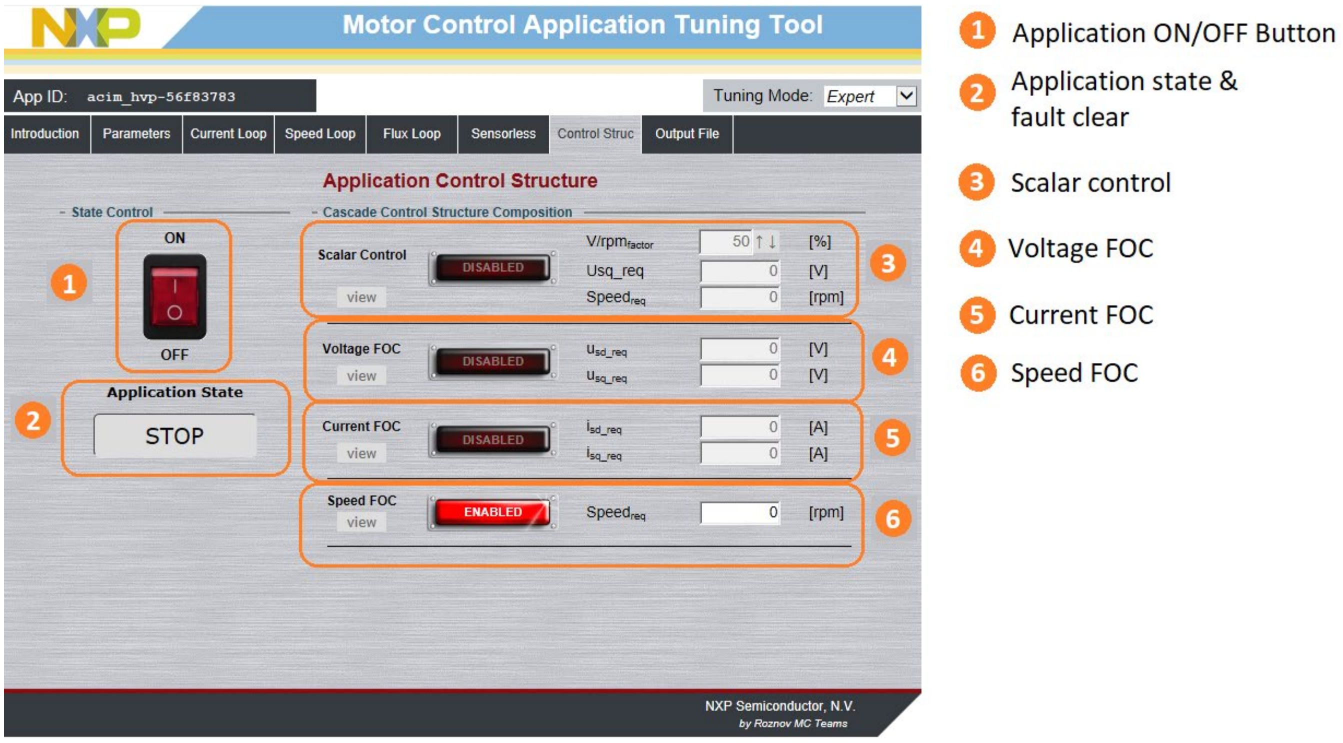
Besides the MCAT page for ACIM, several scopes, recorders, and variables in the “Variable watch” window are pre-defined in the FreeMASTER project file to further simplify the motor parameters tuning and debugging.

The following sections provide simple instructions on how to change the parameters of a connected ACIM and the parameters of the application and how to tune the application.

5.1. Application control using MCAT

Control the application using the “Control Struc” tab, which is shown in Figure 4. The application state control area on the left-hand side of the screen (points 1 and 2) shows the current application state and enables switching the main application switch on or off (turning the running application off disables all PWM outputs). The “Cascade Control Structure” area is placed on the right-hand side of the screen (points 3 to 6). Here you can choose between the scalar and FOC control using the appropriate buttons.

Enable the selected parts of the FOC cascade structure by selecting “Voltage FOC”, “Current FOC”, or “Speed FOC”. This is useful for application tuning and debugging.



App ID: acim_hvp-56f83783

Tuning Mode: Expert

Scalar Control	DISABLED	V/rpm$_{factor}$	50	[%]
		Usq_req	0	[V]
		Speed$_{req}$	0	[rpm]
Voltage FOC	DISABLED	U_{sd_req}	0	[V]
		U_{sq_req}	0	[V]
Current FOC	DISABLED	I_{sd_req}	0	[A]
		I_{sq_req}	0	[A]
Speed FOC	ENABLED	Speed$_{req}$	0	[rpm]

Figure 4. MCAT for ACIM control page

The scalar control diagram is shown in Figure 5. It is the simplest type of ACIM control strategy. The ratio between the magnitude of the stator voltage and the frequency (frequency information is contained in the *Speed$_{req}$* value) is kept at a nominal value, which results in a nominal flux amplitude. This control method is sometimes called Volt per Hertz (V/Hz). The position-estimation Rotor Flux Observer (RFO) algorithm runs in the background to enable the RFO tuning.

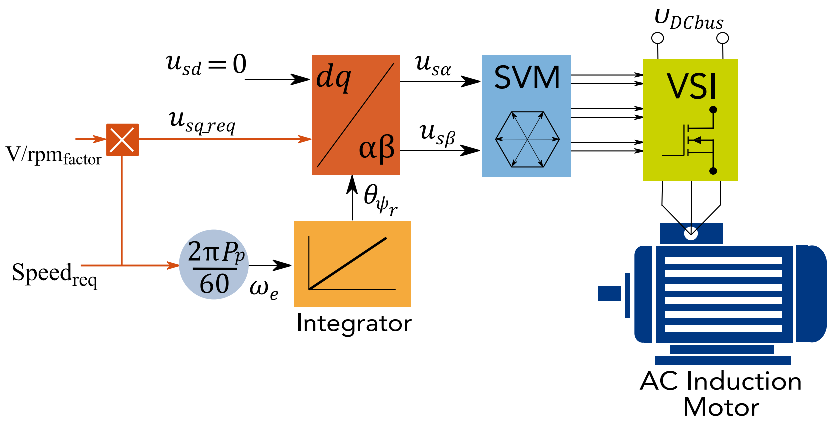


Figure 5. Scalar control mode

The block diagram of the Voltage FOC is shown in Figure 6. Unlike the V/Hz, the position feedback is closed using the RFO algorithm and the stator voltage magnitude is not dependent on the motor speed.

Specify both the d-axis and q-axis stator voltages using the u_{sd_req} and u_{sq_req} fields. This control method is useful for the RFO tuning as well.

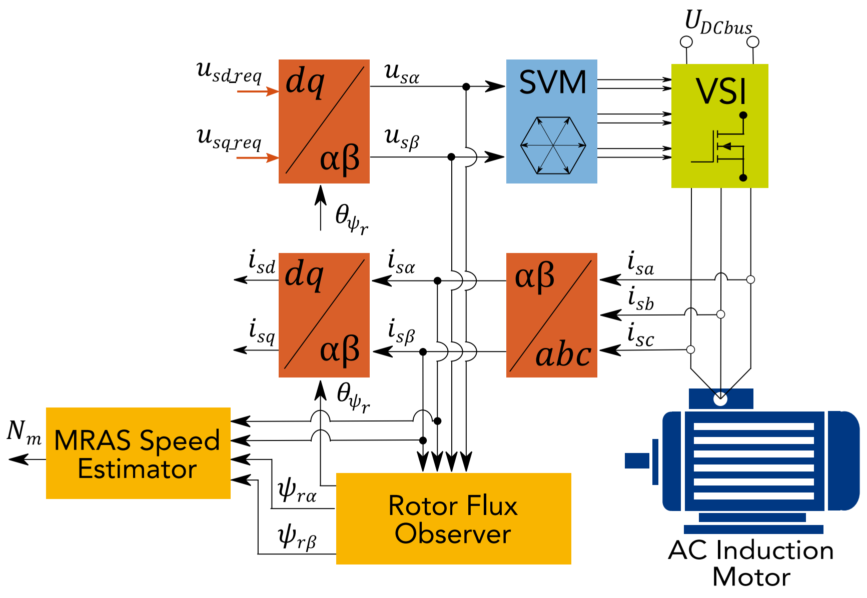


Figure 6. Voltage FOC control mode

The Current FOC (torque) control requires the rotor position feedback and currents to be transformed into the rotor flux frame. Control the motor using the reference variables i_{sd_req} and i_{sq_req}, as shown in Figure 7. The d-axis current component i_{sd_req} generates the rotor flux and the q-axis current component of the current i_{sq_req} generates the torque for the motor to run. Change the polarity of the i_{sq_req} current to change the rotation direction. The Current FOC control structure can be used for the current controller tuning, provided that the RFO is tuned correctly.

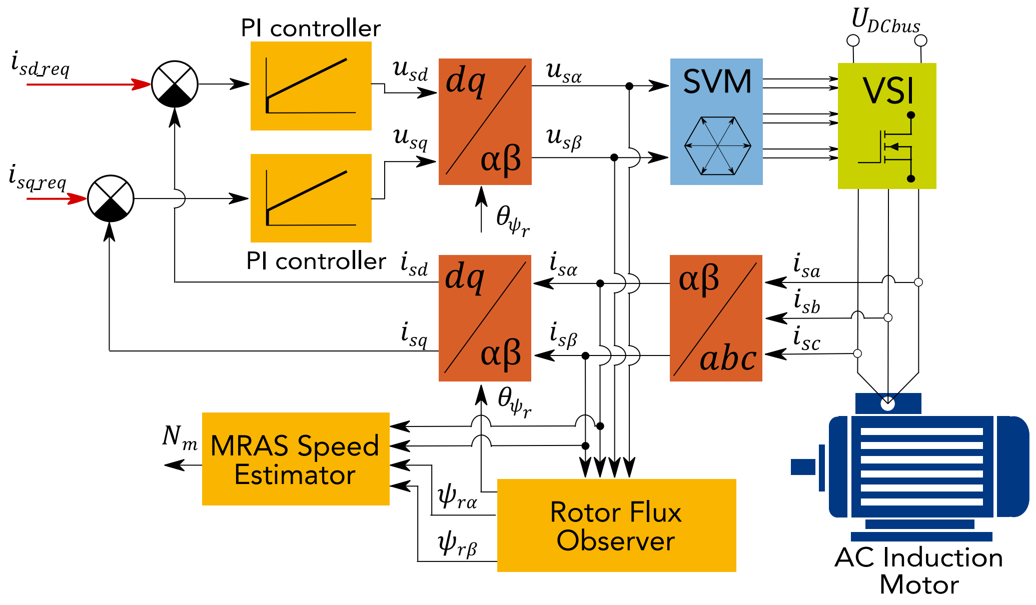


Figure 7. Current FOC (torque) control mode

The full ACIM sensorless FOC is activated by enabling the Speed FOC control structure. The block diagram is shown in Figure 8. Two outer control loops were added when compared to the Current FOC. The speed loop contains the PI controller, which controls the rotor speed and sets the q-axis current i_{sd_req}. The flux loop contains the Max Torque Per Ampere (MTPA) and Flux Weakening (FW) algorithms, which set the q-axis current i_{sd_req} to optimize the power efficiency and allow the motor to run at a speed that is higher than nominal. To run a motor at the required speed, simply enter the required value into the $Speed_{req}$ field. This control scheme is used for the speed PI controller and flux loop design (see Section 5.2.4, "Speed loop tuning" and Section 5.2.5, "Flux loop tuning"), which is the final stage of the ACIM sensorless application tuning.

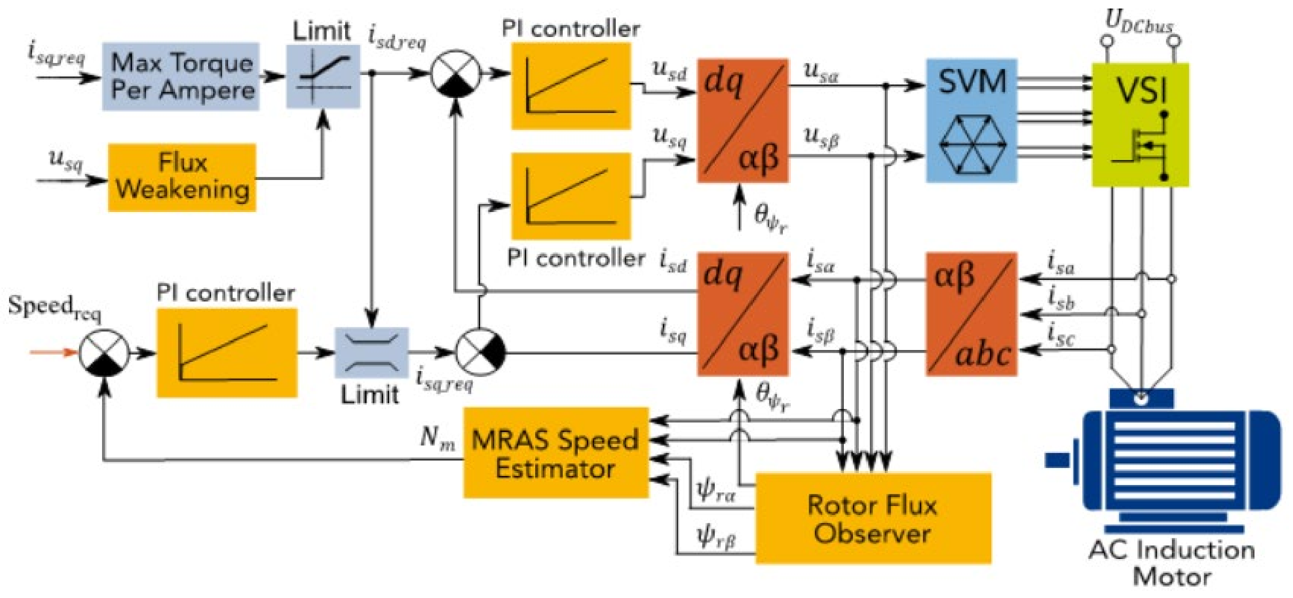


Figure 8. Speed FOC control mode

5.2. Application tuning using MCAT

The ACIM sensorless FOC algorithm tuning is described in this section. The flowchart of the complete process of connecting and running a new ACIM is shown in Figure 9. The control of the ACIM sensorless FOC application using MCAT is described in Section 5.1, "Application control using MCAT". The subsequent steps, including the tuning of the sensorless Rotor Flux Observer (RFO), current loops, speed loop, and flux loop, are described in the following sections. Only the expert MCAT tuning mode is described. When in the "Basic" mode, omit the grayed-out input fields and leave them at their pre-defined values. Most of the input field labels in the MCAT also show a short description of the item and the maximum range of input parameters when you hover over them with the mouse cursor.

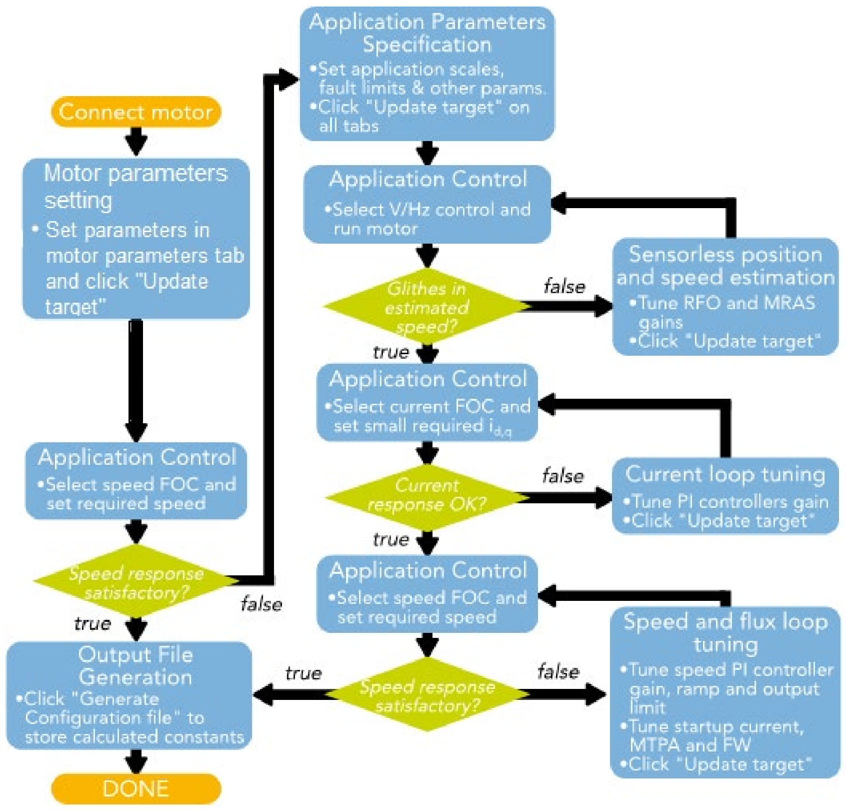

Connect motor
Motor parameters setting
• Set parameters in motor parameters tab and click "Update target"
Application Control
•Select speed FOC and set required speed
Speed response satisfactory?
false
true
Output File Generation
•Click "Generate Configuration file" to store calculated constants
DONE
Application Parameters Specification
•Set application scales, fault limits & other params.
•Click "Update target" on all tabs
Application Control
•Select V/Hz control and run motor
Glithes in estimated speed?
false
true
Sensorless position and speed estimation
•Tune RFO and MRAS gains
•Click "Update target"
Application Control
•Select current FOC and set small required $i_{d,q}$
Current response OK?
false
true
Current loop tuning
•Tune PI controllers gain
•Click "Update target"
Application Control
•Select speed FOC and set required speed
Speed response satisfactory?
true
false
Speed and flux loop tuning
•Tune speed PI controller gain, ramp and output limit
•Tune startup current, MTPA and FW
•Click "Update target"

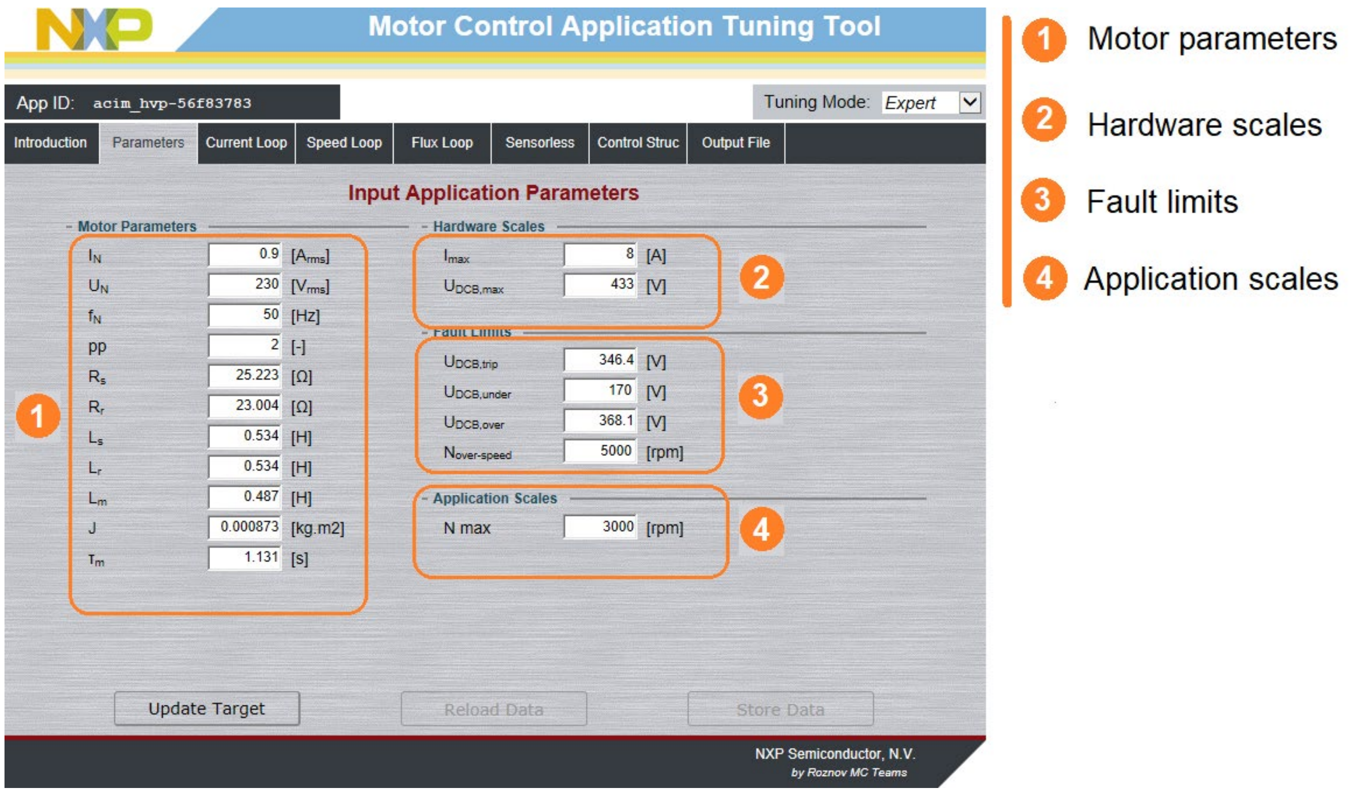
Figure 9. Running a new ACIM flowchart

5.2.1. Input Application Parameters tab

When the parameters of a connected ACIM are obtained or simply known before, navigate to the “Parameters” tab, as shown in Figure 10. On the left-hand side, modify the motor parameters (point 1) and the hardware board scales (point 2). Do not change the latter unless using a user-specific hardware. The right-hand side contains the “Fault Limits” area (point 3), which is accessible only in the “Expert” mode.

NOTE

Motor parameters can be measured by the motor identification algorithm using the Kinetis KV series platform with CM4 or CM7 cores and a floating-point unit. For more details, see application note AN5051, Section 5.1, “ACIM parameter identification”.



App ID: acim_hvp-56f83783

Tuning Mode: Expert

Input Application Parameters

Motor Parameters

I_N	0.9	$[A_{rms}]$
U_N	230	$[V_{rms}]$
f_N	50	[Hz]
pp	2	[-]
R_s	25.223	[Ω]
R_r	23.004	[Ω]
L_s	0.534	[H]
L_r	0.534	[H]
L_m	0.487	[H]
J	0.000873	[kg.m2]
T_m	1.131	[s]

Hardware Scales

I_{max}	8	[A]
$U_{DCB,max}$	433	[V]

Fault Limits

$U_{DCB,trip}$	346.4	[V]
$U_{DCB,under}$	170	[V]
$U_{DCB,over}$	368.1	[V]
$N_{over\text{-}speed}$	5000	[rpm]

Application Scales

N max	3000	[rpm]

Figure 10. MCAT Input Application Parameters tab

Table 2 shows the list of MCAT input parameters with their physical units, brief description, impacted algorithms, and accessibility status in the "Basic" mode:

Table 2. Parameters tab inputs

Input name	Units	Description	Use in constant calculation	Basic mode accessibility
I_N	A_{rms}	Nominal stator current	Speed and flux loop	yes
U_N	V_{rms}	Nominal stator current	Current and flux loop	yes
f_N	Hz	Nominal frequency	Speed and flux loop	yes
pp	—	Number of motor pole pairs	Speed control, RFO, and MRAS	yes
R_s	Ω	Stator resistance	Current loop and RFO	yes
R_r	Ω	Rotor resistance	Current loop and RFO	yes
L_s	H	Stator inductance	Current loop and RFO	yes
L_r	H	Rotor inductance	Current loop and RFO	yes
L_m	H	Magnetizing inductance	Current loop, flux loop, and RFO	yes
J	kgm^2	Moment of inertia	Speed loop	yes
T_m	S	Mechanical time constant	Speed loop	yes
I_{max}	A	Hardware current-sensing scale	Current sensing	yes
$U_{DCB,max}$	V	Hardware DC-bus voltage-sensing scale	Voltage sensing	yes
$U_{DCB,trip}$	V	Trigger value that switches an external DC-bus braking resistor on	Fault protection	no

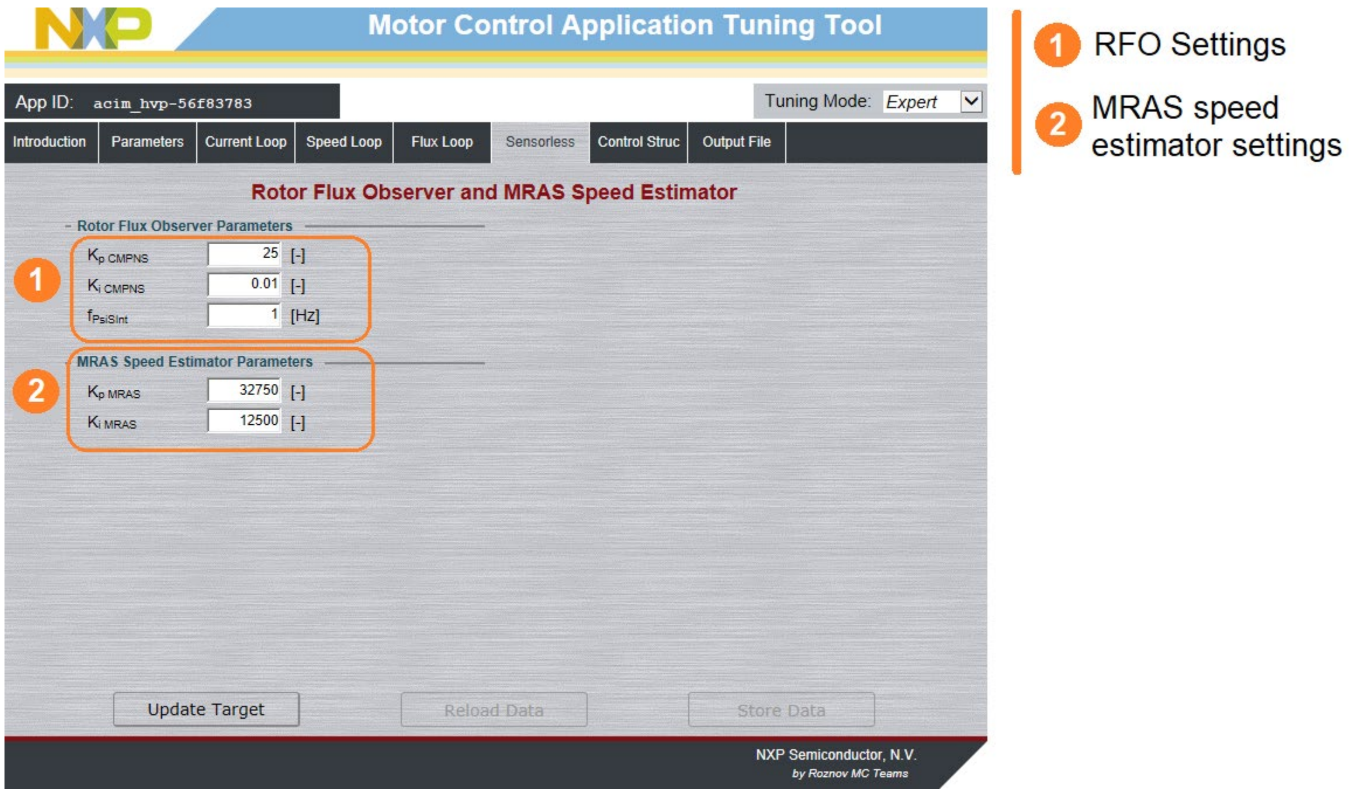
Table 2. Parameters tab inputs

Input name	Units	Description	Use in constant calculation	Basic mode accessibility
$U_{DCB,under}$	V	Voltage value that generates the DC-bus under-voltage fault	Fault protection	no
$U_{DCB,over}$	V	Voltage value that generates the DC-bus over-voltage fault	Fault protection	no
$N_{over\text{-}speed}$	rpm	Over-speed threshold	Fault protection	no
N_{max}	rpm	Application speed scale	Speed loop	yes

5.2.2. Sensorless rotor flux position and speed estimation

The rotor flux position and mechanical speed feedback signals are obtained using the sensorless RFO and Model Reference Adaptive System (MRAS) speed-estimation algorithm. For information about their principles, see *Sensorless ACIM Field-Oriented Control* (document DRM150). Tune both algorithms using the “Sensorless” sub-module MCAT tab, as shown in Figure 11. All the “Sensorless” sub-module tab inputs are listed in Table 3.

Most of the RFO parameters are calculated automatically by the MCAT and they do not need any tuning. The only parameters left to tune are the proportional gain $K_{p,CMPNS}$ and the integral gain $K_{i,CMPNS}$ of the RFO compensation PI controller. These parameters are usually set manually, because the settings do not vary greatly for different motors, and you can keep them at the default settings. A similar situation applies to the MRAS speed estimator and its proportional and integral gains of the internal PI controller ($K_{p,MRAS}$ and $K_{i,MRAS}$). To tune the parameters of these algorithms, run the motor in the scalar control mode, while referring to the “Speed” scope located in the “Scalar/Voltage Control” sub-block in the FreeMASTER project tree. Here you can see the estimated filtered rotor speed. The estimated and scalar rotor speeds are not going to exactly match the properly-tuned RFO and MRAS because of the speed slip.



Rotor Flux Observer and MRAS Speed Estimator

Rotor Flux Observer Parameters

$K_{p\ CMPNS}$ 25 [-]
$K_{i\ CMPNS}$ 0.01 [-]
$f_{PsiSInt}$ 1 [Hz]

MRAS Speed Estimator Parameters

$K_{p\ MRAS}$ 32750 [-]
$K_{i\ MRAS}$ 12500 [-]

Figure 11. MCAT sensorless position and speed estimation tab

Table 3. MCAT sensorless position and speed estimation tab inputs

Parameter name	Units	Description	Use in constant calculation	Basic mode accessibility
K_{PCMPNS}	—	Compensation PI controller proportional gain	RFO	no
K_{ICMPNS}	—	Compensation PI controller integral gain	RFO	no
$f_{PsiSInt}$	Hz	Stator flux integrator filter frequency	RFO	no
K_{PMRAS}	—	Compensation PI controller proportional gain	MRAS speed estimation	no
K_{IMRAS}	—	Compensation PI controller integral gain	MRAS speed estimation	no

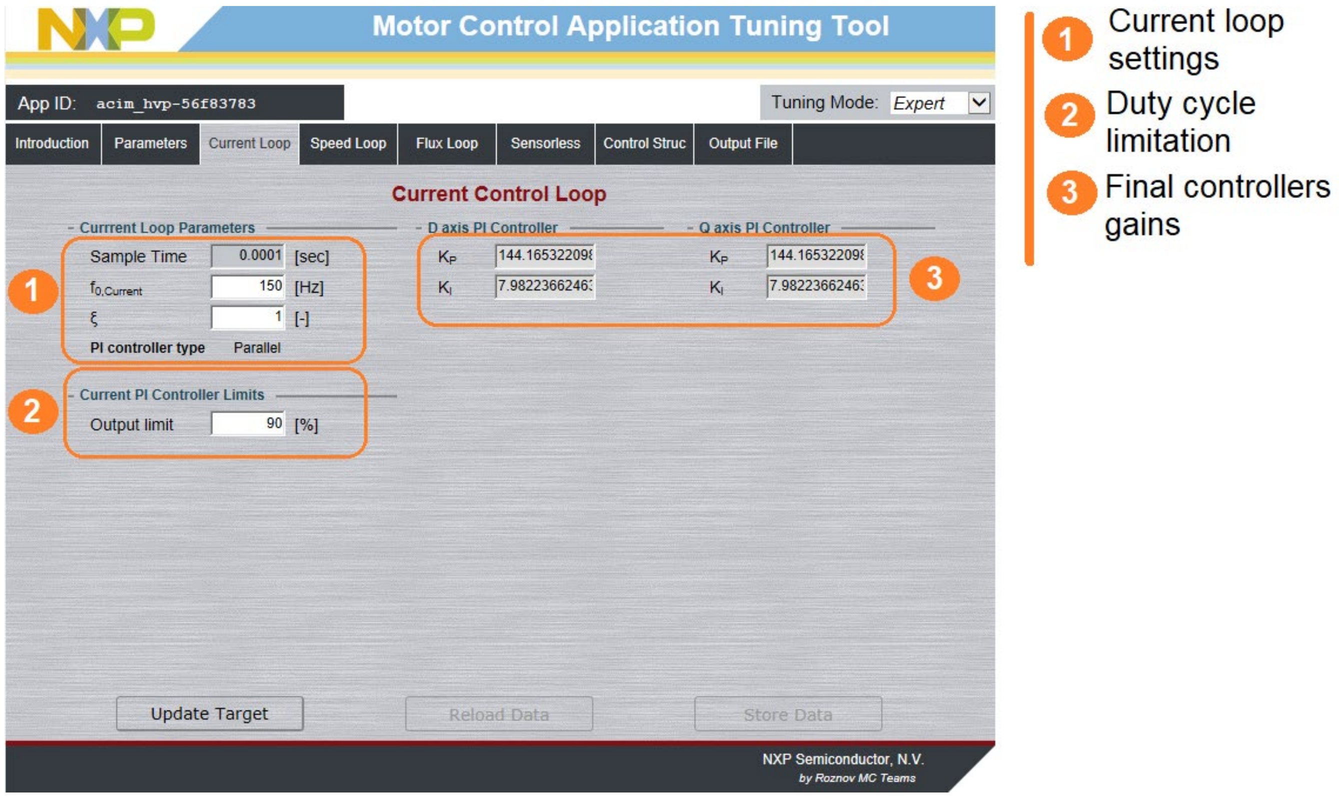
Part of the RFO algorithm requires an internal calculation of the stator flux, which involves pure integration that has problems with the integrator drift. These problems are solved by approximating the pure integrator with the low-pass filter. See *Sensorless ACIM Field-Oriented Control* (document DRM150) for more details. The low-pass filter cut-off frequency is set in the $f_{PsiSInt}$ input field and recommended to be set in the range from 1 Hz to 3 Hz. Higher values can lead to a high number of flux- and speed-estimation errors.

5.2.3. Current loop tuning

The “Current Loop” tab is designed for the current control loop tuning. The current control loop is the most inner loop in the cascade control structure of a vector-control algorithm. One of the FOC characteristics is a separate control of the rotor flux-producing (*d*-axis) and torque-producing (*q*-axis) components of the current. Therefore, the ACIM control structure has two current loops and each of them contains a PI controller. The “Current Loop” tab is shown in Figure 12. The individual fields are described in Table 4. Set all of the inputs on the left-hand side of the tab (points 1 and 2). The PI controller resulting gains are located on the right-hand side (point 3). The sampling time field is filled in automatically when the MCU platform is successfully detected by the MCAT and cannot be changed.

Table 4. MCAT current control loop tab inputs

Parameter name	Units	Description	Use in constant calculation	Basic mode accessibility
F0	Hz	Current control loop bandwidth	Current loop	no
z	—	Damping ratio of the current control loop	Current loop	no
Output limit	%	Current loop output limit in percentage of the DC-bus voltage	Current loop	no



App ID: acim_hvp-56f83783
Tuning Mode: Expert

Currrent Loop Parameters
Sample Time 0.0001 [sec]
$f_{0,Current}$ 150 [Hz]
ξ 1 [-]
PI controller type Parallel

Current PI Controller Limits
Output limit 90 [%]

D axis PI Controller
K_P 144.16532209
K_I 7.9822366246

Q axis PI Controller
K_P 144.16532209
K_I 7.9822366246

Figure 12. MCAT Current Control Loop tab

The simplified block diagrams of both the *d*-axis and *q*-axis current control loops are shown in Figure 13. The non-linear coupling parts of the stator voltage are ignored and treated as unmeasured errors entering the controlled system. The parasitic time constants (such as the inverter time constant) are ignored as well.

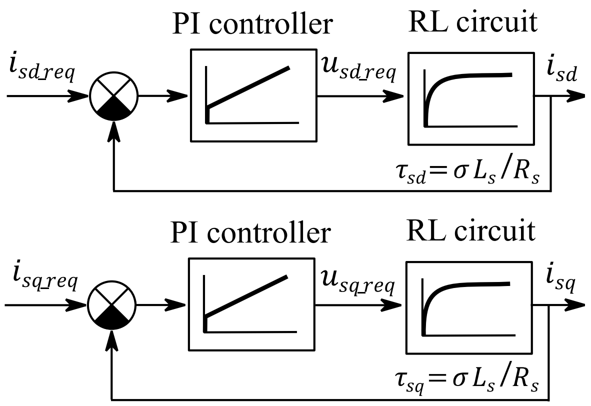


Figure 13. d-axis and q-axis current loop block diagram

Ignoring the non-linear coupling portion of the stator voltage (which is simply dealt with by the integral part of the current PI controllers), the transfer functions of the stator currents i_{sd} and i_{sq} are:

$$F_{isd}(s) = \frac{I_{sd}(s)}{U_{sd}(s)} = \frac{1/R_s}{\tau_{sd}s+1} = \frac{1}{\sigma L_s s + R_s} \quad \text{Eq. 1}$$

$$F_{isq}(s) = \frac{I_{sq}(s)}{U_{sq}(s)} = \frac{1/R_s}{\tau_{sq}s+1} = \frac{1}{\sigma L_s s + R_s} \quad \text{Eq. 2}$$

where s is the Laplace operator, τ_{sd} and τ_{sq} are the stator *d*-axis and *q*-axis electric time constants, and

$$\sigma = 1 - \frac{L_m^2}{L_s L_r} \quad [\text{-}] \quad \text{Eq. 3}$$

is the leakage coefficient.

The transfer function of the PI controller in a parallel form is:

$$F_{PI}(s) = K_p + \frac{K_i}{s} = \frac{K_p s + K_i}{s} \quad \text{Eq. 4}$$

where K_p is the proportional gain and K_i is the integral gain. The closed-loop *d*-axis and *q*-axis current transfer functions are:

$$F_{wisd}(s) = \frac{I_{sd}(s)}{I_{sd\_req}(s)} = \frac{F_{PI}(s)F_{isd}(s)}{1+F_{PI}(s)F_{isd}(s)} = \frac{\frac{K_{pd}}{K_{id}}s+1}{\frac{\sigma L_s}{K_{id}}s^2 + \frac{R_s+K_{pd}}{K_{id}}s+1} \quad \text{Eq. 5}$$

$$F_{wisq}(s) = \frac{I_{sq}(s)}{I_{sq\_req}(s)} = \frac{F_{PI}(s)F_{isq}(s)}{1 + F_{PI}(s)F_{isq}(s)} = \frac{\frac{K_{pq}}{K_{iq}}s + 1}{\frac{\sigma L_s}{K_{iq}}s^2 + \frac{R_s + K_{pq}}{K_{iq}}s + 1} \qquad \text{Eq. 6}$$

By comparing these transfer functions to the transfer function of a second-order system with the unity gain

$$F_{2nd}(s) = \frac{1}{\frac{s^2}{4\pi^2 f_0^2} + \frac{s\zeta}{\pi f_0} + 1}, \qquad \text{Eq. 7}$$

where f_0 is the system natural frequency (or bandwidth) and ζ is the system damping ratio, the following is obtained:

$$K_{pd} = K_{pq} = 4\pi f_0 \zeta \sigma L_s - R_s \qquad \text{Eq. 8}$$

$$K_{id} = K_{iq} = 4\pi^2 f_0^2 \sigma L_s \qquad \text{Eq. 9}$$

The proportional and integral gains of a discrete version of the current PI controller can be obtained using the bilinear transformation method:

$$K_{pz} = K_p \qquad \text{Eq. 10}$$

$$K_{iz} = T_s K_i \qquad \text{Eq. 11}$$

NOTE

The correct value of the integral gain (according to the bilinear transformation) must be half the value stated in Eq. 11. The division by two is not shown because it is conducted internally by the PI controller algorithm in the RTCESL (see more at www.nxp.com/rtcesl).

The effect of the damping ratio ζ on the step response of a second-order system $F_{2nd}(s)$ is shown in Figure 14. The MCAT allows setting the damping ratio ζ in the range from 0.5 to 2.0. It is not recommended to divert from the value of 1 too much. Choose the natural frequency in the range from tens to hundreds of Hz, but at least one order higher than the speed loop bandwidth. If the bandwidth value is too high, it leads to problems with the sampling frequency, voltage limitation, and stability.

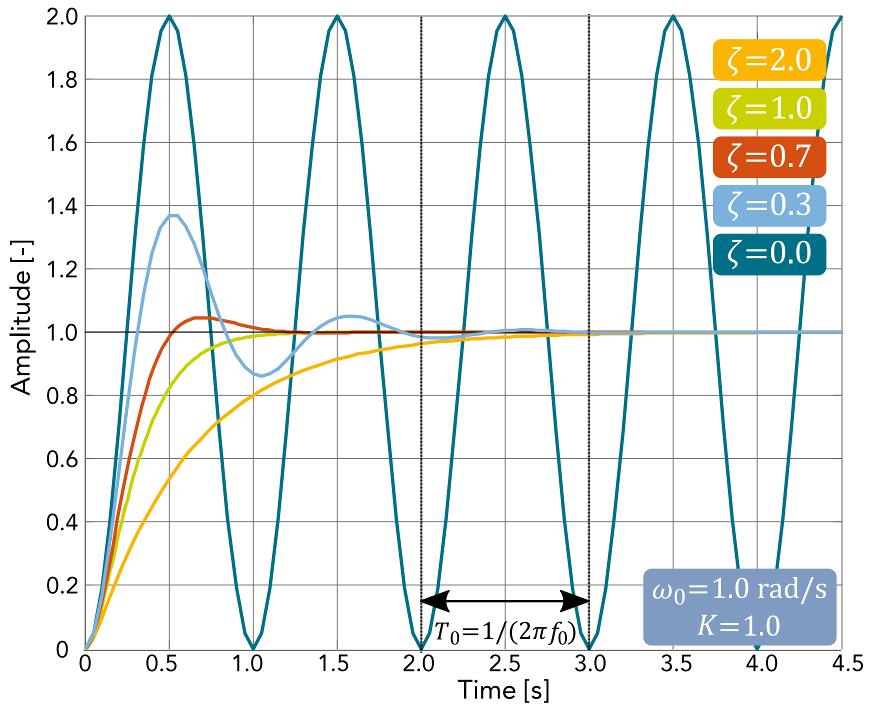


Figure 14. Second-order system steps response for various damping ratios

When comparing the transfer functions in Eq. 5, Eq. 6, and Eq. 7, the numerator (0 of the system) with the time constant K_p/K_i of the current closed-loop transfer function is ignored, because it has minor impact on the resulting system stability.

To check the current response, use the FreeMASTER recorder called Current Control, which is triggered during motor startup. The examples of the *d*-axis current response for different setups of the current loop bandwidth are shown in Figure 15. The ideal current response must not be too slow (as in case C), but it must neither contain a high overshoot. A very high current loop bandwidth can lead to instability. If you are not satisfied with the automatically-calculated current loop PI controller parameters, tune them manually. To do so, perform these steps:

1. Go to the “Current Loop” tab and select the “Expert” tuning mode.
2. Set the desired current loop bandwidth f_0 and click the “Update Target” button. It is recommended to start with a lower value and then keep increasing it until a desired response is achieved.
3. Select the “Current Loop” recorder. The message at the bottom of the recorder must read “Running, waiting for trigger…”.
4. Go to the “Control struc” tab, select the “Current FOC” control mode, and set small required values of the *d*-axis and *q*-axis currents.
5. Run the application and wait for the data to load.
6. Check the downloaded response in the recorder and repeat the procedure from step two (if necessary).

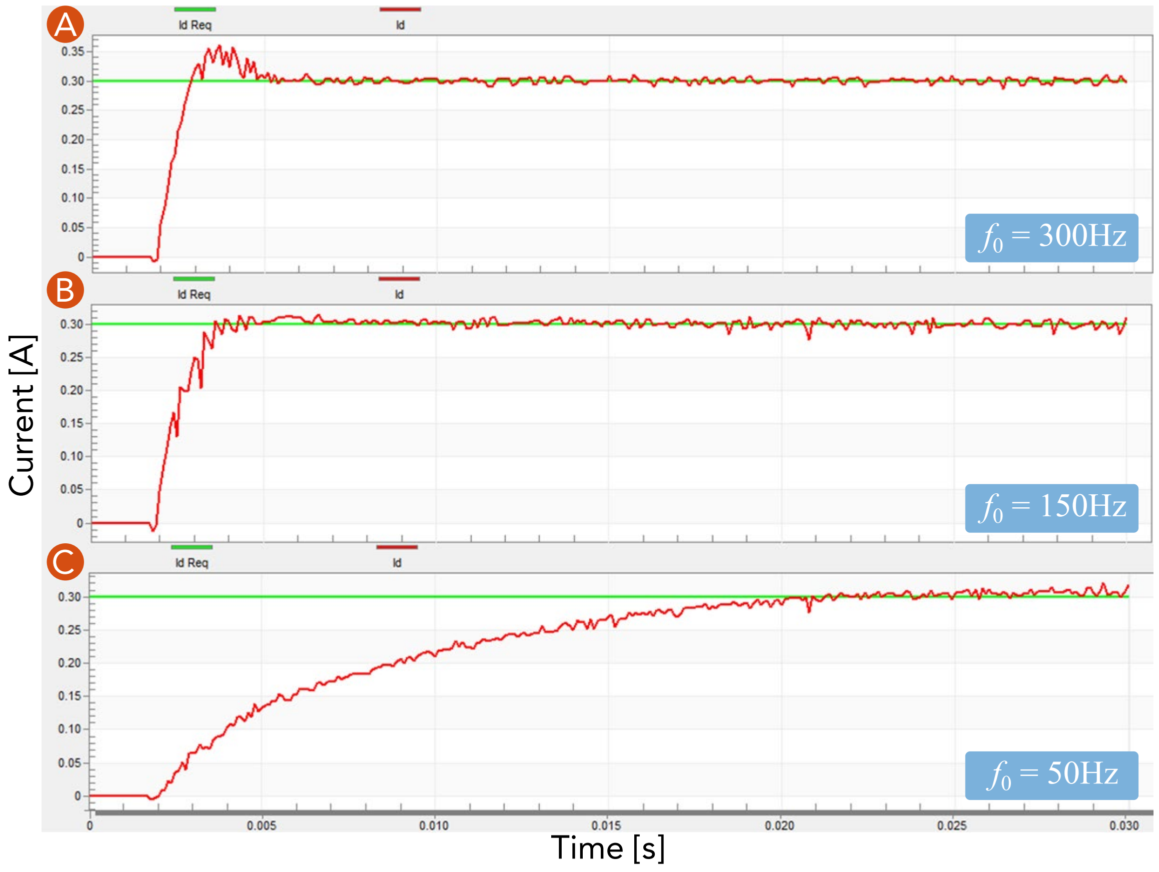
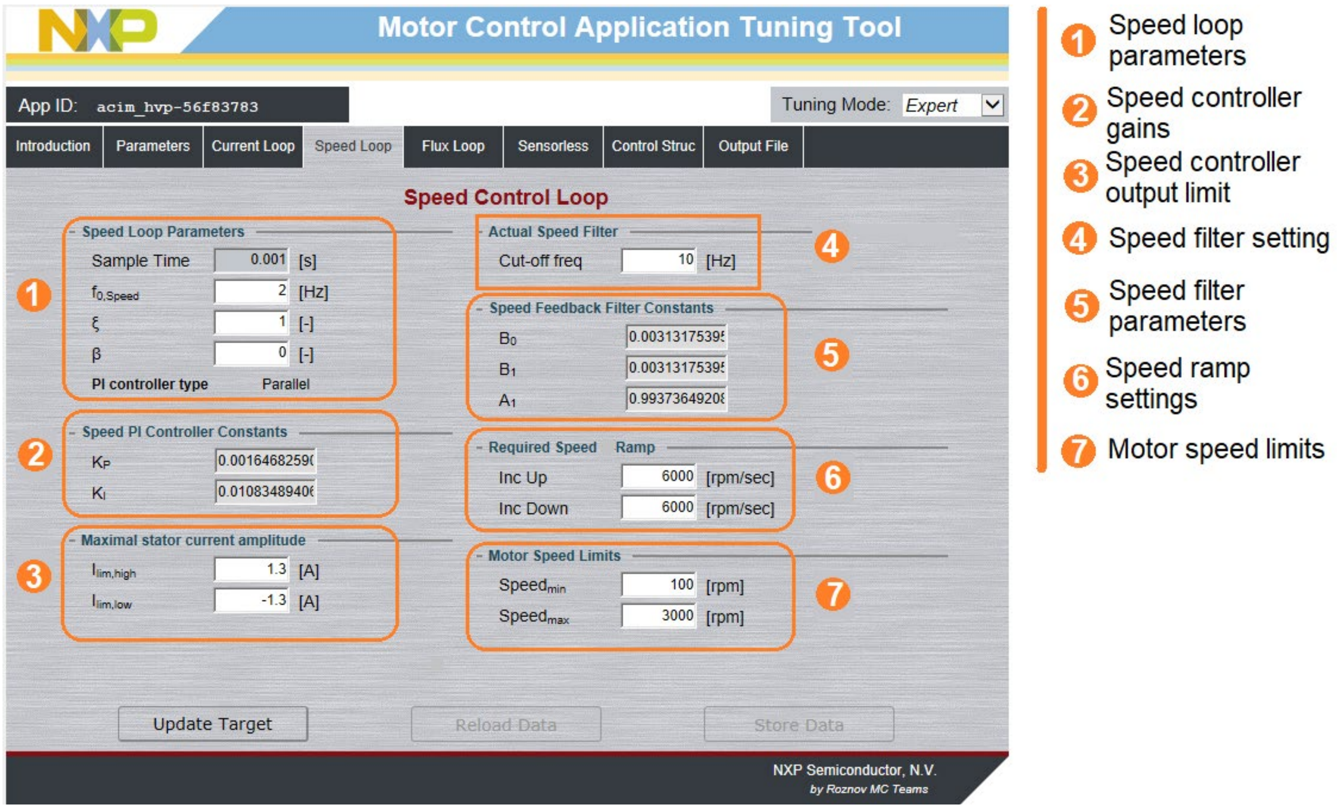


Figure 15. Response of the d-axis current to different settings of current loop bandwidth

5.2.4. Speed loop tuning

The “Speed Loop” tab is designed to tune the speed-control loop. The speed-control loop is an outer loop in the cascade-control structure of a vector-controlled ACIM. The speed loop consists of the PI controller, estimated speed filter, and ramp function, which limits the maximum, minimum, acceleration, and jerk of the required speed. A screenshot of the “Speed Loop” tuning page is shown in Figure 16 and the individual fields are described in Table 5.



App ID: acim_hvp-56f83783

Tuning Mode: Expert

Speed Control Loop

Speed Loop Parameters

Sample Time	0.001	[s]
$f_{0,Speed}$	2	[Hz]
ξ	1	[-]
β	0	[-]
PI controller type	Parallel	

Speed PI Controller Constants

K_P	0.00164682590
K_I	0.0108348940

Maximal stator current amplitude

$I_{lim,high}$	1.3	[A]
$I_{lim,low}$	-1.3	[A]

Actual Speed Filter

Cut-off freq	10	[Hz]

Speed Feedback Filter Constants

B_0	0.00313175395
B_1	0.00313175395
A_1	0.99373649208

Required Speed Ramp

Inc Up	6000	[rpm/sec]
Inc Down	6000	[rpm/sec]

Motor Speed Limits

$Speed_{min}$	100	[rpm]
$Speed_{max}$	3000	[rpm]

1. Speed loop parameters
2. Speed controller gains
3. Speed controller output limit
4. Speed filter setting
5. Speed filter parameters
6. Speed ramp settings
7. Motor speed limits

Figure 16. MCAT Speed Control Loop tab

Table 5. MCAT Speed Control Loop tab parameters

Parameter name	Units	Description	Use in constant calculation	Basic mode accessibility
Sample time	S	Speed loop sampling time period	Speed loop	no
F0	Hz	Speed control loop bandwidth	Speed loop	yes
ζ	—	Damping ratio of the speed control loop	Speed loop	no
β	—	Overshoot damping coefficient	Speed loop	no
$I_{lim,high}$	A	Speed loop output upper limit	Speed loop	no
$I_{lim,low}$	A	Speed loop output lower limit	Speed loop	no
Cut-off freq	Hz	Speed filter cut-off frequency	Speed loop	no
Acceleration up	rpm/s	Acceleration of the required speed	Speed loop	yes
Acceleration down	rpm/s^2	Acceleration of the required speed	Speed loop	yes
$Speed_{max}$	rpm	Maximum required speed	Speed loop	no
$Speed_{min}$	rpm	Minimum required speed	Speed loop	no

A simplified block diagram of the speed-control loop is shown in Figure 17. The simplification lies in ignoring the current loop dynamics, because it is presumed to be much faster than the dynamics of a mechanical system.

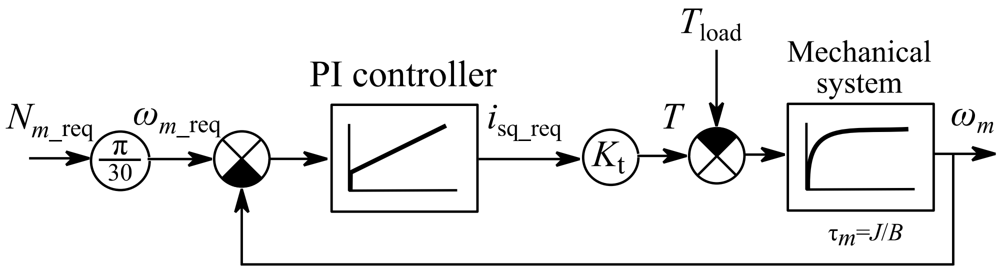


Figure 17. Speed loop block diagram

Calculate the torque constant K_t of an ACIM as follows:

$$K_t = \frac{3}{4} P_p \frac{L_m^2}{L_r} \quad \text{Eq. 12}$$

Note that the previous equation ignores the *d*-axis current. This value may change during the motor operation, which affects the behavior of the transient speed response. The torque constant is therefore adapted in run-time (according to the required *d*-axis current required value) and the speed controller is calculated for $i_{sd} = 1$ A.

When ignoring the load torque, the transfer function of a complete driven mechanical system is:

$$F_\omega(s) = \frac{\Omega_m(s)}{I_{sq\_req}(s)} = \frac{1/B}{\tau_m s + 1} = \frac{1}{Js + B} \quad \text{Eq. 13}$$

where τ_m is the mechanical time constant, J is the moment of inertia, and B is the mechanical viscous friction. Considering the PI controller to be in a parallel form according to Eq. 4, the closed speed control loop is:

$$F_{w\omega}(s) = \frac{\Omega_m(s)}{\Omega_{m\_req}(s)} = \frac{F_{PI}(s)F_\omega(s)}{1 + F_{PI}(s)F_\omega(s)} = \frac{\frac{K_{p\omega}}{K_{i\omega}} s + 1}{\frac{J}{K_t K_{i\omega}} s^2 + \frac{B + K_t K_{p\omega}}{K_t K_{i\omega}} s + 1} \quad \text{Eq. 14}$$

where $K_{p\omega}$ is the speed PI controller proportional gain and $K_{i\omega}$ is the integral gain. By comparing this transfer function with the transfer function of a second-order system in Eq. 7, it is obtained:

$$K_{p\omega} = \frac{4\zeta\pi f_0 J - B}{K_t} \quad \text{Eq. 15}$$

$$K_{i\omega} = \frac{4\pi^2 f_0^2 J}{K_t} \quad \text{Eq. 16}$$

The selection of damping ratio ζ and speed loop bandwidth f_0 follows similar rules as in the current loop. Choose a bandwidth at least one order smaller (in case of the current loop). Calculate the proportional and integral gains of a discrete version of the speed PI controller using the bilinear transformation method, as shown in Eq. 10 and Eq. 11.

To check the speed response, open the FreeMASTER scope named “Speed”, located under the “Speed Control” sub-block. If you are not satisfied with the speed response resulting from the automatically calculated parameters, tune the controller manually. To do so, perform these steps:

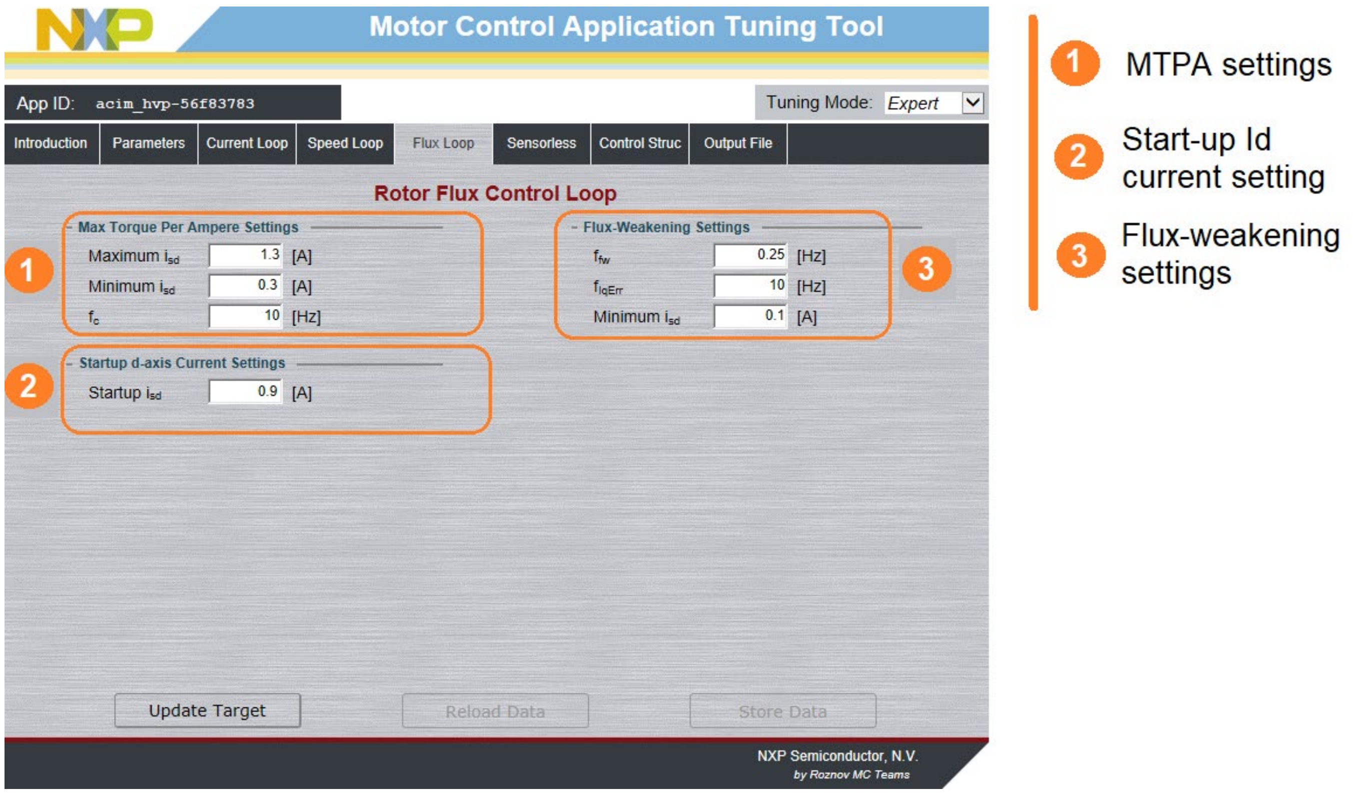
1. Go to the “Speed Loop” tab and select the “Expert” tuning mode.
2. Set the desired speed loop bandwidth f_0 and click the “Update Target” button. It is recommended to start with a lower value (in the range of Hz, depending on the mechanical time constant) and then increasing it until a desired response is achieved.
3. Select the “Speed” scope in the “Speed Control” sub-section.
4. Set the required speed and observe the response.
5. Check the downloaded response in the recorder and repeat from step 2 (if necessary).

5.2.5. Flux loop tuning

The “Flux Loop” tab is designed to tune the Max Torque Per Ampere (MTPA) and Flux Weakening (FW) algorithms, which forms the second outer loop in the cascade-control structure of the ACIM vector control. Both algorithms are more closely described in *Sensorless ACIM Field-Oriented Control* (document DRM150). The screenshot of the “Flux Loop” tuning page is shown in Figure 18 and the individual fields are described in Table 6.

Table 6. MCAT flux control loop tab parameters

Parameter name	Units	Description	Use in constant calculation	Basic mode accessibility
Maximum i_{sd}	A	Maximum d-axis current	Speed loop	no
Minimum i_{sd}	A	Minimum d-axis current	Speed loop	no
f_C	Hz	Required d-axis current filter	Speed loop	no
Startup i_{sd}	A	Startup d-axis current	Speed loop	no
f_{fw}	Hz	FW controller bandwidth	Speed loop	no
f_{IqErr}	Hz	Speed loop output lower limit	Speed loop	no
Minimum i_{sd}	A	Minimum d-axis current	Speed loop	no



Rotor Flux Control Loop

Max Torque Per Ampere Settings

Maximum i_{sd}	1.3	[A]
Minimum i_{sd}	0.3	[A]
f_c	10	[Hz]

Flux-Weakening Settings

f_{fw}	0.25	[Hz]
f_{IqErr}	10	[Hz]
Minimum i_{sd}	0.1	[A]

Startup d-axis Current Settings

Startup i_{sd}	0.9	[A]

Figure 18. MCAT flux loop tuning tab

The only parameters that are required to be set for the MTPA are the *d*-axis current limits $i_{sd\_req,max}$ and $i_{sd\_req,min}$ and the filter bandwidth. The upper limit must be set to a value that corresponds to the nominal amplitude of the rotor flux, which means:

$$i_{sd\_req,max} = 0.707 I_{phN} \quad [\mathrm{A}]$$ **Eq. 17**

Setting the lower *d*-axis current limit low allows for better power optimization (depends on the load). However, setting it too low may affect the RFO performance and lead to a control failure. It is recommended to set $i_{sd\_req,min}$ to at least 25 % of the upper limit $i_{sd\_req,min}$ (or more).

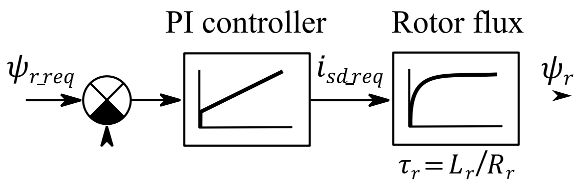


Figure 19. Rotor flux control loop block diagram

The rotor flux control loop to tune the flux-weakening PI controller is shown in Figure 19. The transfer function of the controlled rotor flux system is:

$$F_{\psi}(s) = \frac{\Psi_r(s)}{I_{sd\_req}(s)} = \frac{1/L_m}{\tau_r s + 1}$$ **Eq. 18**

Considering the PI controller to be in a parallel form according to Eq. 4, the open control loop is:

$$\boldsymbol{F_{0\psi}(s) = \frac{K_{p\Psi}s + K_{i\Psi}}{s}\frac{1/L_m}{\tau_r s + 1}} \qquad \textit{Eq. 19}$$

Placing the controller 0 to the system pole means:

$$\boldsymbol{K_{p\psi} = K\frac{\tau_r}{L_m}} \qquad \textit{Eq. 20}$$

$$\boldsymbol{K_{i\psi} = K\frac{1}{L_m}} \qquad \textit{Eq. 21}$$

The open loop transfer is reduced to $F_{0\psi}(s) = K/s$ $F_{0\Psi} = K/s$, where K is the general constant. Setting $K = 2\pi f_0$ leads to this closed loop transfer function:

$$\boldsymbol{F_{w\psi}(s) = \frac{\Psi_r(s)}{\Psi_{r\_req}(s)} = \frac{1}{\frac{1}{2\pi f_0}s + 1}} \qquad \textit{Eq. 22}$$

where f_0 is the flux-weakening controller bandwidth. The final discrete controller gains are therefore calculated as follows:

$$\boldsymbol{K_{p\psi z} = 2\pi f_0 \frac{\tau_r}{L_m}} \qquad \textit{Eq. 23}$$

$$\boldsymbol{K_{i\psi z} = T_s 2\pi f_0 \frac{1}{L_m}} \qquad \textit{Eq. 24}$$

5.2.6. MCAT output file generation

When you successfully tune the application and want to store all the calculated parameters to an embedded application, navigate to the “Output File” tab. View the list of all definitions generated by MCAT there. Clicking the “Generate Configuration File” button overwrites the older version of the *m1_acim_appconfig.h* file, which contains all FOC algorithm definitions. To generate the file into a correct location, connect the target MCU via FreeMASTER. Otherwise, when in the offline mode, the file is generated next to the **.pmp* file.

6. Conclusion

This application note describes the implementation of the reference application for ACIM on 32-bit DSC MC56F82748. The hardware-dependent part of the software, which includes peripheral initialization and application timing, is described in Section 3, “MCU peripheral settings”. The initialization and API of the Motor Control Peripheral Drivers (MCDRV), which allows for a simple and unified access to PWM and ADC on all supported devices, is in Section 4, “Motor-Control Peripheral Drivers”. The last part of the document describes application tuning and control using the FreeMASTER-based MCAT tool. All the steps necessary for running the ACIM-like parameter identification, current loop, speed loop, and flux loop tuning are described as well.

7. Acronyms and abbreviations

Table 7. Acronyms and abbreviations

Term	Meaning
AC	Alternating Current
ACIM	AC Induction Machine
ADC	Analog-to-Digital Converter
AN	Application Note
CPU	Central Processing Unit
CMP	Comparator
DC	Direct Current
DRM	Design Reference Manual
FOC	Field-Oriented Control
FW	Flux-Weakening
RTCESL	Real-Time Embedded Software Library
FTM	FlexTimer Module
GPIO	General-Purpose Input/Output
HVP	High-Voltage development Platform
I/O	Input/Output interface
MCAT	Motor Control Application Tuning tool
MCDRV	Motor Control Peripheral Drivers
MCU	Microcontroller Unit
MRAS	Model Reference Adaptive System
MTPA	Maximum Torque Per Ampere
PDB	Programmable Delay Block
PI	Proportional Integral controller
PWM	Pulse-Width Modulation
RFO	Rotor Flux Observer
UART	Universal Asynchronous Receiver/Transmitter
VSI	Voltage Source Inverter

8. References

These references are available on www.nxp.com:

1. *Sensorless ACIM Field Oriented Control* (document DRM150).
2. *MC56F83xxx Reference Manual* (document MC56F83XXXRM).
3. *NXP High-Voltage Motor Control Platform User's Guide* (document HVPMC3PHUG).
4. *HVP-MC56F82748 User’s Guide* (document HVPMC56F82748UG).
5. *Motor Control Application Tuning (MCAT) Tool for Three-Phase PMSM* (document AN4642).

9. Revision history

Table summarizes the changes done to this document since the initial release.

Table 8. Revision history

Revision number	Date	Substantive changes
0	03/2019	Initial release.
1	03/2020	Updates for DSC 56F83783.

Document Number: AN12379
Rev. 1
03/2020