AN5143
Application note

How to migrate motor control application software from SDK v4.3 to SDK v5.x

Introduction

The STM32 motor control software development kit (MC SDK) is part of the STMicroelectronics motor-control ecosystem. It is referenced as X-CUBE-MCSDK or X-CUBE-MCSDK-FUL according to the software license agreement applied. It includes the:

- ST MC FOC firmware library for permanent-magnet synchronous motor (PMSM) field-oriented control (FOC)
- ST MC Workbench software tool, a graphical user interface for the configuration of MC SDK firmware library parameters

This application note helps when migrating motor-control application software from the SDK v4.3 to the SDK v5.x framework. It covers firmware aspects as well as the use of the MC software tool.

Note: SDK v5.x must be used for new projects because it provides users with new APIs.

Contents

List of tables

List of figures

1 General information

The MC SDK is used for the development of motor-control applications running on STM32 32-bit microcontrollers based on the Arm®[a] Cortex®-M processor.

Table 1 presents the definition of acronyms that are relevant for a better understanding of this document.

Table 1. List of acronyms

Acronym	Description
API	Application programming interface
GUI	Graphical user interface
HAL	Hardware abstraction layer
ICS	Isolated current sensor
IDE	Integrated development environment
FOC	Field-oriented control
FW	Firmware
HFI	High frequency injection
LL	Low-level
MC	Motor control
MC WB	Motor control Workbench (STMicroelectronics software tool)
MP	Motor Profiler (STMicroelectronics software tool)
MTPA	Maximum torque per ampere
PFC	Power factor correction
PMSM	Permanent-magnet synchronous motor
SDK	Software development kit
SW	Software
SPL	Standard peripheral libraries

[a] Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

2 Related documents

Arm® documents

The following documents are available from the infocenter.arm.com web page:

1. Cortex®-M0 Technical Reference Manual
2. Cortex®-M3 Technical Reference Manual
3. Cortex®-M4 Technical Reference Manual

STMicroelectronics documents

The following documents are available from the *www.st.com* web page:

4. STM32F0 Series product data sheets
5. STM32F1 Series product data sheets
6. STM32F2 Series product data sheets
7. STM32F3 Series product data sheets
8. STM32F4 Series product data sheets
9. *X-NUCLEO expansion boards motor control - Selection guide* on-line presentation

3 SDK v5.x versus SDK v4.3 comparison summary

Table 2 provides a comparison overview of SDK v5.x against SDK v4.3 for the main migration topics.

Table 2. SDK v5.x versus SDK v4.3 comparison summary

Migration topic	SDK v4.3	SDK v5.x
Software installation	– Motor Profiler – MC Workbench [4.3] – ST-LINK/V2	– Motor Profiler – MC Workbench [5.0.0 or above] – STM32CubeMX [4.24.0 or above] – ST-LINK/V2
Supported microcontroller	Limited to the STM32F0, STM32F1, STM32F2, STM32F3, and STM32F4 Series.	– STM32F0, STM32F3, and STM32F4 Series are part of release 5.0. – STM32F1 Series is part of release 5.1. – Others Series such as STM32F7, STM32L4, and STM32H7 are planned in later releases.
Features	– Field-oriented control – Dual motor – Flux weakening – Feed forward – MTPA – PFC – Speed sensor (Hall / Encoder / HFI / Sensorless) – Current sensor (1 shunt / 3 shunts / ICS)	– Field-oriented control(1) – Dual motor(1) – Flux weakening(1) – Feed forward(1) – MTPA(1) – Speed sensor (Hall / Encoder / Sensorless)(1) – Current sensor (1 shunt / 3 shunts / ICS)(1) – PFC(2) – HFI(3)
Handling of ST board	Directly from the MC WB.	Directly from the MC WB.
Handling of custom control board	Directly from the MC WB but only with a limited number of MCUs.	Possible through STMicroelectronics reference board and use of STM32CubeMX (same MCUs as in SDK v4.3).
Architecture	Three main parts: – MCApplication – MCLibrary – UI_Library	Three main parts, re-arranged from SDK v4.3: – Motor cockpit – Motor control library – User interface library
Workspace	Two workspaces are needed: – Library – User application code	Only one workspace is needed, which manages both the middleware and the user application code.
API	Refer to *Chapter 5: Development workflow on page 11*.	SDK v4.3 APIs, used as reference, are: - Simplified; - Dedicated per motor.
Coding style	Object-oriented C code.	Cube architecture C code.
Drivers used	SPL	HAL - LL

Table 2. SDK v5.x versus SDK v4.3 comparison summary (continued)

Migration topic	SDK v4.3	SDK v5.x
Peripheral initialization	– Done inside FW main file through `#define` – All MCU code present in the code	– Automatically managed by STM32CubeMX – Only the needed code is generated
Readability	The main code is difficult to understand and modify because of: - The large file size - The many `#define` parts to be understood and handled by the user	– Automatically managed by STM32CubeMX – Only the needed code is generated
Debug	Not straightforward: - Data are hiding - Virtual functions are used	Easy, as no data are hiding and direct function are called.
MIPS	-	Single-motor CPU workload gain: – Up to 13% with STM32F072RB – Up to 3.5% with STM32F303RE – Up to 8.1% with STM32F446RE Dual-motor CPU workload gain: – Up to 20.8% with STM32F415ZG
Memory size	-	Single-motor MC library gain: – Up to 29.2% with STM32F072RB – Up to 21.9% with STM32F303RE – Up to 17% with STM32F446RE Dual-motor MC library gain: – Up to 14.9% with STM32F415ZG
MC Profiler	-	Same as SDK v4.3. The update of supported boards is planned.(3)
MC Workbench	– Header file generation only – Manual project generation: header files generated are copied into a user project workspace – The user must ensure project consistency	– Complete C project generation (for IDE) – Project generation is done automatically (using STM32CubeMX) including initialization code and toolchain project files – IAR Embedded Workbench® for Arm® (IAR Systems® AB), or µVision® IDE for Arm® (Keil® MDK) are supported(4)
MC monitor	-	Same as SDK v4.3.

1. Available from release v5.0.
2. Available from release v5.1.
3. Available in a later release.
4. The Atollic® TrueSTUDIO® framework is supported from the SDK v5.1 version on.

4 Supported STMicroelectronics boards

Table 3 lists the STMicroelectronics control and inverter boards supported in both SDK v4.3 and SDK v5.x.

Table 3. Control and inverter boards supported by both SDK v4.3 and SDK v5.x

MCU	Series	MC WB reference	MC WB type	STM32CubeMX board reference
STM32F030R8Tx	STM32F0	NUCLEO-F030R8[1]	Control board	NUCLEO-F030R8
STM32F072RB		NUCLEO-F072RB[1]	Control board	NUCLEO-F072RB
STM32F072VBTx		STM32072B-EVAL[1]	Control board	STM32072B-EVAL
STSPIN32F0		STEVAL-SPIN3201[2]	Inverter board	-
STM32F103ZGTx	STM32F1	STM3210E-EVAL[2]	Control board	STM3210E-EVAL
STM32F103RC		STEVAL-IHM034V2[2]	Inverter board	-
STM32F302R8Tx	STM32F3	NUCLEO-F302R8[1]	Control board	NUCLEO-F302R8
STM32F303RETx		NUCLEO-F303RE[1]	Control board	NUCLEO-F303RE
STM32F303VETx		STM32303E-EVAL[1]	Control board	STM32303E-EVAL
STM32F446RETx	STM32F4	NUCLEO-F446RE[1]	Control board	NUCLEO-F446RE
STM32F407IGHx		STM3240G-EVAL[1]	Control board	STM3240G-EVAL
STM32F417IGHx		STM3241G-EVAL[1]	Control board	STM3241G-EVAL
STM32F446ZETx		STM32446E-EVAL[1]	Control board	STM32446E-EVAL
STM32F415ZGT8		STEVAL-IHM039V1[1]	Inverter board	-

1. Supported from SDK v5.0.
2. Supported from SDK v5.1.

Table 4 lists the additional STMicroelectronics control and inverter boards supported only with SDK v5.1.

Table 4. Additional control and inverter boards supported only with SDK v5.1

MCU	Series	MC WB reference	MC WB type	STM32CubeMX board reference
STSPIN32F0A	STM32F0	STEVAL-SPIN3202	Inverter board	-
STM32F103RBT6	STM32F1	NUCLEO-F103RB	Control board	NUCLEO-F103RB
STM32F303CBT7	STM32F3	STEVAL-ESC001V1	Inverter board	-
STM32F303RETx		X-NUCLEO-IHM16M1, NUCLEO-F303RE	Inverter board	-

Table 5 lists the STMicroelectronics power boards supported in both SDK v4.3 and SDK v5.0.

Table 5. Power boards supported by both SDK v4.3 and SDK v5.0

MC WB reference	MC WB type
STEVAL-IHM023V3	Power board
STEVAL-IHM025V1	Power board
STEVAL-IHM028V2	Power board
STEVAL-IHM045V1	Power board
STEVAL-IPM05F	Power board
STEVAL-IPM07F	Power board
STEVAL-IPM10B	Power board
STEVAL-IPM10F	Power board
STEVAL-IPM15B	Power board
X-NUCLEO-IHM07M1	Power board
X-NUCLEO-IHM08M1	Power board
X-NUCLEO-IHM11M1	Power board

Table 6 lists the additional STMicroelectronics power boards supported only with SDK v5.1.

Table 6. Additional power boards supported only with SDK v5.1

MC WB reference	MC WB type
STEVAL-IPM08B	Power board
STEVAL-IPMNG3Q	Power board
STEVAL-IPMNG5Q	Power board
STEVAL-IPMNG8Q	Power board
STEVAL-IPMNM1N	Power board
STEVAL-IPMNM2N	Power board

5 Development workflow

The following IDEs are supported (refer to the release note for the versions tested):

- IAR Embedded Workbench® for Arm® (IAR Systems® AB)
- MDK tools for Arm® (Keil® MDK)
- TrueSTUDIO® for STM32 (Atollic®)

In MC SDK v5.x, the development workflow is similar to SDK v4.3 with the following exceptions, which provide improved efficiency and customer experience:

- The user does not need to copy MC Workbench output files to his project workspaces
- STM32CubeMX can be used to develop user software applications

When an SDK v4.3 project is opened with the SDK v5.x MC Workbench tool, a migration window is displayed, which informs about the conversion of output file format from SDK v4.3 to SDK v5.x.

The development workflow for SDK v5.x is presented in *Figure 1 on page 12*. The following steps are maintained from the development workflow for SDK v4.3:

- The Motor Profiler can be used to identify the main PMSM characteristics, which are further transferred to the MC Workbench.
- MC Workbench is the main entry point to start a new project.

In addition, it offers the following evolution features:

- STM32CubeMX is interfaced with, and background-called from the MC Workbench to generate the selected IDE project work frame.
- The user can customize the generated project with STM32CubeMX, his IDE, or both.
- The user can tune his application with the MC Workbench monitor feature as shown in *Figure 2 on page 12*.

Figure 1. Development workflow

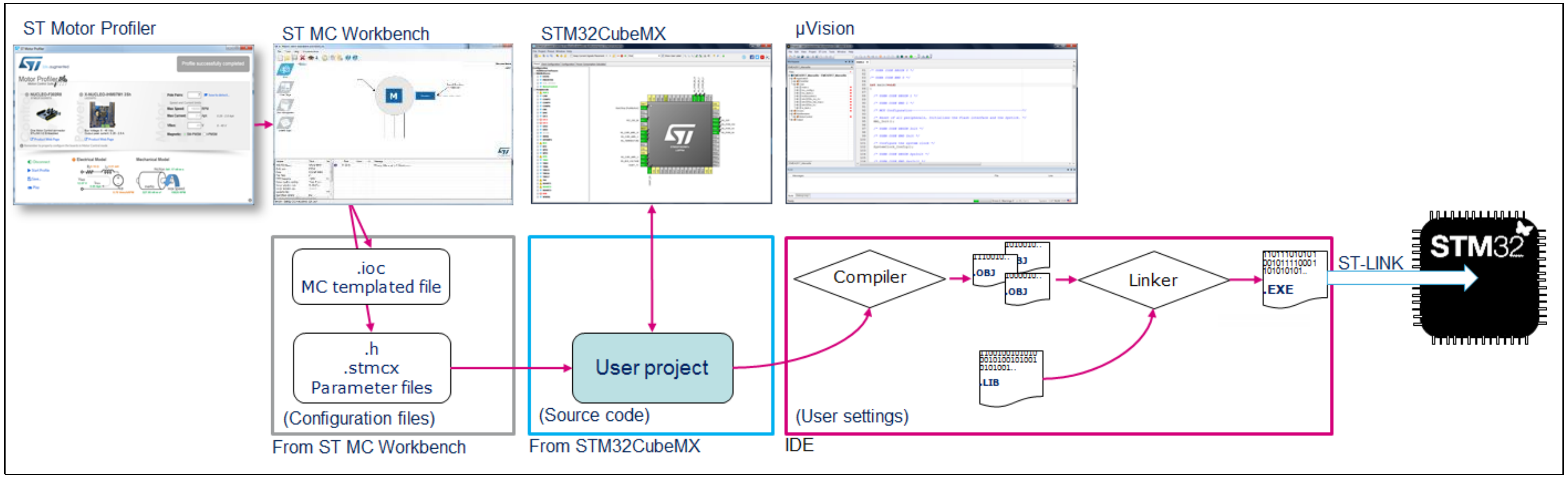
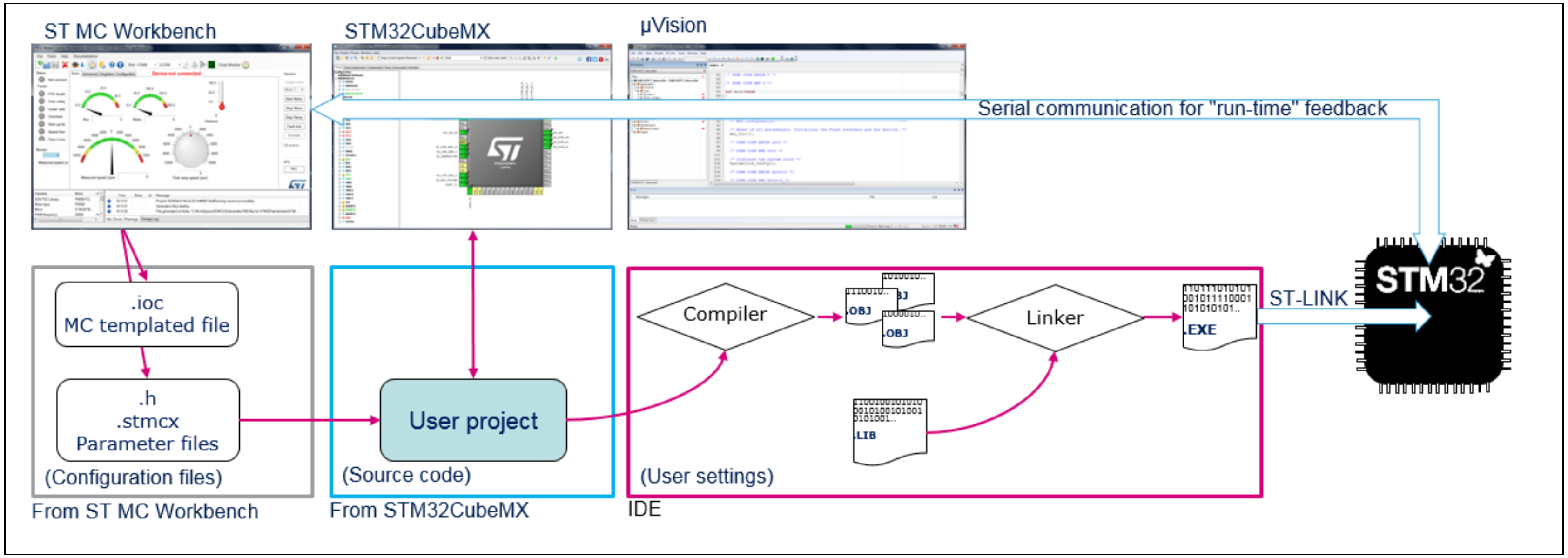


Figure 2. Tuning workflow



6 Simplification

6.1 Conversion of objects to pointers

All motor control objects are converted into data structure pointers to simplify their use, reference, readability, and potential debug. This also reduces the number of header files to be managed. *Table 7*. illustrates such a conversion for the Inrush Current Limiter feature.

Table 7. Objects to pointers conversion example

SDK v4.3	SDK v5.x
InrushCurrentLimiterClass.h	***InrushCurrentLimiter.h***

typedef enum
{
ICL_IDLE, ICL_ACTIVATION, ICL_ACTIVE,
ICL_DEACTIVATION, ICL_INACTIVE
} ICLState_t;
 	
typedef struct CICL_t *CICL;
 	
typedef const struct
{
  uint16_t hICLFrequencyHz;
  uint16_t hDurationms;
 	
} InrushCurrentLimiterParams_t,
*pInrushCurrentLimiterParams_t;
 	
CICL
ICL_NewObject(pInrushCurrentLimiterParams
_t pInrushCurrentLimiterParams);
 	
void ICL_Init(CICL this, CVBS oVBS, CDOUT
oDOUT);
	typedef enum
	{
	  ICL_IDLE,
	  ICL_ACTIVATION,
	  ICL_ACTIVE,
	  ICL_DEACTIVATION,
	  ICL_INACTIVE
	} ICL_State_t;
	
	typedef struct
	{
	  BusVoltageSensor_Handle_t *pVBS;
	  DOUT_handle_t *pDOUT;
	
	  ICL_State_t ICLstate;
	  uint16_t hICLTicksCounter;
	  uint16_t hICLTotalTicks;
	  uint16_t hICLFrequencyHz;
	  uint16_t hICLDurationms;
	} ICL_Handle_t;
	
	void ICL_Init(ICL_Handle_t *pHandle,
	BusVoltageSensor_Handle_t *pVBS,
	DOUT_handle_t *pDOUT);

Table 7. Objects to pointers conversion example (continued)

SDK v4.3	SDK v5.x
InrushCurrentLimiterPrivate.h	

typedef struct
{
  CVBS oVBS;
  CDOUT oDOUT;
 	
  ICLState_t ICLState;
  uint16_t hRemainingTicks;
  uint16_t hTotalTicks;
} Vars_t,*pVars_t;
 	
typedef InrushCurrentLimiterParams_t Params_t, *pParams_t;
 	
typedef struct
{
  Vars_t Vars_str;
  pParams_t pParams_str;
} _CICL_t, *_CICL;
	-

6.2 Translation of motor-control APIs

The tables in this section present the correspondence of APIs from SDK v4.3 to SDK v5.x:

Table 8. Translation from the *MCInterfaceClass.h* to the *mc_api.h* file

SDK v4.3	SDK v5.x

`void MCI_ExecSpeedRamp(CMCI this, int16_t hFinalSpeed, uint16_t hDurationms);`	`void MC_ProgramSpeedRampMotor1( int16_t hFinalSpeed, uint16_t hDurationms );`
	`void MC_ProgramSpeedRampMotor2( int16_t hFinalSpeed, uint16_t hDurationms );`

`void MCI_ExecTorqueRamp(CMCI this, int16_t hFinalTorque, uint16_t hDurationms);`	`void MC_ProgramTorqueRampMotor1( int16_t hFinalTorque, uint16_t hDurationms );`
	`void MC_ProgramTorqueRampMotor2( int16_t hFinalTorque, uint16_t hDurationms );`

`void MCI_SetCurrentReferences(CMCI this, Curr_Components Iqdref);`	`void MC_SetCurrentReferenceMotor1 ( Curr_Components Iqdref );`
	`void MC_SetCurrentReferenceMotor2 ( Curr_Components Iqdref );`

Table 8. Translation from the *MCInterfaceClass.h* to the *mc_api.h* file (continued)

SDK v4.3	SDK v5.x

`bool MCI_StartMotor(CMCI this);`	`bool MC_StartMotor1(void);`
	`bool MC_StartMotor2(void);`

`bool MCI_StopMotor(CMCI this);`	`bool MC_StopMotor1(void);`
	`bool MC_StopMotor2(void);`

`bool MCI_FaultAcknowledged(CMCI this);`	`bool MC_AcknowledgeFaultMotor1( void );`
	`bool MC_AcknowledgeFaultMotor2( void );`

`bool MCI_EncoderAlign(CMCI this);`	*Removed*

`CommandState_t MCI_IsCommandAcknowledged(CMCI this);`	`MCI_CommandState_t MC_GetCommandStateMotor1( void);`
	`MCI_CommandState_t MC_GetCommandStateMotor2( void);`

`State_t  MCI_GetSTMState(CMCI this);`	`State_t  MC_GetSTMStateMotor1(void);`
	`State_t  MC_GetSTMStateMotor2(void);`

`int16_t MCI_GetOccurredFaults(CMCI this);`	`uint16_t MC_GetOccurredFaultsMotor1(void);`
	`uint16_t MC_GetOccurredFaultsMotor2(void);`

`uint16_t MCI_GetCurrentFaults(CMCI this);`	`uint16_t MC_GetCurrentFaultsMotor1(void);`
	`uint16_t MC_GetCurrentFaultsMotor2(void);`

`int16_t MCI_GetMecSpeedRef01Hz(CMCI this);`	`int16_t MC_GetMecSpeedReferenceMotor1(void);`
	`int16_t MC_GetMecSpeedReferenceMotor2(void);`

`int16_t MCI_GetAvrgMecSpeed01Hz(CMCI this);`	`int16_t MC_GetMecSpeedAverageMotor1(void);`
	`int16_t MC_GetMecSpeedAverageMotor2(void);`

`int16_t MCI_GetTorque(CMCI this);`	`Not needed as never Implemented in SDK v4.3`

`STC_Modality_t MCI_GetControlMode(CMCI this);`	`STC_Modality_t MC_GetControlModeMotor1(void);`
	`STC_Modality_t MC_GetControlModeMotor2(void);`

`int16_t MCI_GetImposedMotorDirection(CMCI this);`	`int16_t MC_GetImposedDirectionMotor1(void);`
	`int16_t MC_GetImposedDirectionMotor2(void);`

`int16_t MCI_GetLastRampFinalSpeed(CMCI this);`	`int16_t MC_GetLastRampFinalSpeedMotor1(void);`
	`int16_t MC_GetLastRampFinalSpeedMotor2(void);`

`bool MCI_RampCompleted(CMCI this);`	`bool MC_HasRampCompletedMotor1(void);`
	`bool MC_HasRampCompletedMotor2(void);`

Table 8. Translation from the *MCInterfaceClass.h* to the *mc_api.h* file (continued)

SDK v4.3	SDK v5.x

`bool MCI_StopSpeedRamp(CMCI this);`	`bool MC_StopSpeedRampMotor1(void);`
	`bool MC_StopSpeedRampMotor2(void);`

`bool MCI_GetSpdSensorReliability(CMCI this);`	`bool MC_GetSpeedSensorReliabilityMotor1(void);`
	`bool MC_GetSpeedSensorReliabilityMotor2(void);`

`Curr_Components MCI_GetIab(CMCI this);`	`Curr_Components MC_GetIabMotor1(void);`
	`Curr_Components MC_GetIabMotor2(void);`

`Curr_Components MCI_GetIalphabeta(CMCI this);`	`Curr_Components MC_GetIalphabetaMotor1(void);`
	`Curr_Components MC_GetIalphabetaMotor2(void);`

`Curr_Components MCI_GetIqd(CMCI this);`	`Curr_Components MC_GetIqdMotor1(void);`
	`Curr_Components MC_GetIqdMotor2(void);`

`Curr_Components MCI_GetIqdHF(CMCI this);`	`Curr_Components MC_GetIqdHFMotor1(void);`
	`Curr_Components MC_GetIqdHFMotor2(void)`

`Curr_Components MCI_GetIqdref(CMCI this);`	`Curr_Components MC_GetIqdrefMotor1(void);`
	`Curr_Components MC_GetIqdrefMotor2(void);`

`Volt_Components MCI_GetVqd(CMCI this);`	`Volt_Components MC_GetVqdMotor1(void);`
	`Volt_Components MC_GetVqdMotor2(void);`

`Volt_Components MCI_GetValphabeta(CMCI this);`	`Volt_Components MC_GetValphabetaMotor1(void);`
	`Volt_Components MC_GetValphabetaMotor2(void);`

`int16_t MCI_GetElAngledpp(CMCI this);`	`int16_t MC_GetElAngledppMotor1(void);`
	`int16_t MC_GetElAngledppMotor2(void);`

`int16_t MCI_GetTeref(CMCI this);`	`int16_t MC_GetTerefMotor1(void);`
	`int16_t MC_GetTerefMotor2(void);`

`int16_t MCI_GetPhaseCurrentAmplitude(CMCI this);`	`int16_t MC_GetPhaseCurrentAmplitudeMotor1(void);`
	`int16_t MC_GetPhaseCurrentAmplitudeMotor2(void);`

`int16_t MCI_GetPhaseVoltageAmplitude(CMCI this);`	`int16_t MC_GetPhaseVoltageAmplitudeMotor1(void);`
	`int16_t MC_GetPhaseVoltageAmplitudeMotor2(void);`

`void MCI_SetIdref(CMCI this, int16_t hNewIdref);`	`void MC_SetIdrefMotor1( int16_t hNewIdref );`
	`void MC_SetIdrefMotor2( int16_t hNewIdref );`

`void MCI_Clear_Iqdref(CMCI this);`	`void MC_Clear_IqdrefMotor1(void);`
	`void MC_Clear_IqdrefMotor2(void);`

Table 9. Translation from the *MC.h* to the *mc_api.h* file

SDK v4.3	SDK v5.x
`void MC_RequestRegularConv(uint8_t bChannel, uint8_t bSamplTime);`	`void MC_ProgramRegularConversion(uint8_t bChannel, uint8_t bSamplTime);`
`uint16_t MC_GetRegularConv(void);`	`uint16_t MC_GetRegularConversionValue(void);`
`UDRC_State_t MC_RegularConvState(void);`	`UDRC_State_t MC_GetRegularConversionState(void);`

Table 10. Translation from the *MCTuningClass.h* to the *flux_weakening_ctrl.h* file

SDK v4.3	SDK v5.x
[*Flux Weakening]* ***FluxWeakeningCtrl class exported methods***	
`void FW_SetVref(CFW this, uint16_t hNewVref);`	`void FW_SetVref(FW_Handle_t *pHandle, uint16_t hNewVref);`
`uint16_t FW_GetVref(CFW this);`	`uint16_t FW_GetVref(FW_Handle_t *pHandle);`
`int16_t FW_GetAvVAmplitude(CFW this);`	`int16_t FW_GetAvVAmplitude(FW_Handle_t *pHandle);`
`uint16_t FW_GetAvVPercentage(CFW this);`	`uint16_t FW_GetAvVPercentage(FW_Handle_t *pHandle);`

Table 11. Translation from the *MCTuningClass.h* to the *feed_forward_ctrl.h* file

SDK v4.3	SDK v5.x
[Feed Forward] ***FeedForwardCtrl class exported methods***	
`void FF_SetFFConstants(CFF this, FF_TuningStruct_t sNewConstants);`	`void FF_SetFFConstants(FF_Handle_t *pHandle, FF_TuningStruct_t sNewConstants);`
`FF_TuningStruct_t FF_GetFFConstants(CFF this);`	`FF_TuningStruct_t FF_GetFFConstants(FF_Handle_t *pHandle);`
`Volt_Components FF_GetVqdff(CFF this);`	`Volt_Components FF_GetVqdff(FF_Handle_t *pHandle);`
`Volt_Components FF_GetVqdAvPIout(CFF this);`	`Volt_Components FF_GetVqdAvPIout(FF_Handle_t *pHandle);`

Table 12. Translation from the *MCTuningClass.h* to the *open_loop.h* file

SDK v4.3	SDK v5.x
[Open Loop] ***OLCtrl class exported methods***	
`void OL_UpdateVoltage(COL this, int16_t hNewVoltage);`	`void OL_UpdateVoltage(OpenLoop_Handle_t *pHandle, int16_t hNewVoltage);`

Table 13. Translation from the *MCTuningClass.h* to the *pid_regulator.h* file

SDK v4.3	SDK v5.x
[Proportional Integral] ***PI regulator class exported methods***	
`void PI_SetKP(CPI this, int16_t hKpGain);`	`void PID_SetKP(PID_Handle_t* pHandle, int16_t hKpGain);`
`void PI_SetKI(CPI this, int16_t hKiGain);`	`void PID_SetKI(PID_Handle_t* pHandle, int16_t hKiGain);`
`int16_t PI_GetKP(CPI this);`	`int16_t PID_GetKP(PID_Handle_t* pHandle);`
`uint16_t PI_GetKPDivisor(CPI this);`	`uint16_t PID_GetKPDivisor(PID_Handle_t* pHandle);`
`int16_t PI_GetKI(CPI this);`	`int16_t PID_GetKI(PID_Handle_t* pHandle);`
`uint16_t PI_GetKIDivisor(CPI this);`	`uint16_t PID_GetKIDivisor(PID_Handle_t* pHandle);`
`int16_t PI_GetDefaultKP(CPI this);`	`int16_t PID_GetDefaultKP(PID_Handle_t* pHandle);`
`int16_t PI_GetDefaultKI(CPI this);`	`int16_t PID_GetDefaultKI(PID_Handle_t* pHandle);`
`void PI_SetIntegralTerm(CPI this, int32_t wIntegralTermValue);`	`void PID_SetIntegralTerm(PID_Handle_t* pHandle, int32_t wIntegralTermValue);`
[Proportional Integral Derived] ***PID class exported methods***	
`void PID_SetPrevError(CPID_PI this, int32_t wPrevProcessVarError);`	`void PID_SetPrevError(PID_Handle_t* pHandle, int32_t wPrevProcessVarError);`
`void PID_SetKD(CPID_PI this, int16_t hKdGain);`	`void PID_SetKD(PID_Handle_t* pHandle, int16_t hKdGain);`
`int16_t PID_GetKD(CPID_PI this);`	`int16_t PID_GetKD(PID_Handle_t* pHandle);`

Table 14. Translation from the *MCTuningClass.h* to the *pwm_curr_fdbk.h* file

SDK v4.3	SDK v5.x
[Pulse Width Modulation Control] ***PWMnCurrFdbk class exported methods***	
`uint16_t PWMC_ExecRegularConv(CPWMC this, uint8_t bChannel);`	`uint16_t PWMC_ExecRegularConv(PWMC_Handle_t *pHandle, uint8_t bChannel);`
`void PWMC_ADC_SetSamplingTime(CPWMC this, ADConv_t ADConv_struct);`	`void PWMC_ADC_SetSamplingTime(PWMC_Handle_t *pHandle, ADConv_t ADConv_struct);`

Table 15. Translation from the *MCTuningClass.h* to the *revup_ctrl.h* file

SDK v4.3	SDK v5.x
[Rev Up Control] ***RevupCtrl class exported methods***	
`void RUC_SetPhaseDurationms(CRUC this, uint8_t bPhase, uint16_t hDurationms);`	`void RUC_SetPhaseDurationms(RevUpCtrl_Handle_t *pHandle, uint8_t bPhase, uint16_t hDurationms);`
`void RUC_SetPhaseFinalMecSpeed01Hz(CRUC this, uint8_t bPhase, int16_t hFinalMecSpeed01Hz);`	`void RUC_SetPhaseFinalMecSpeed01Hz(RevUpCtrl_Handle_t *pHandle, uint8_t bPhase,`
`void RUC_SetPhaseFinalTorque(CRUC this, uint8_t bPhase, int16_t hFinalTorque);`	`void RUC_SetPhaseFinalTorque(RevUpCtrl_Handle_t *pHandle, uint8_t bPhase, int16_t hFinalTorque);`
`uint16_t RUC_GetPhaseDurationms(CRUC this, uint8_t bPhase);`	`uint16_t RUC_GetPhaseDurationms(RevUpCtrl_Handle_t *pHandle, uint8_t bPhase);`
`int16_t RUC_GetPhaseFinalMecSpeed01Hz(CRUC this, uint8_t bPhase);`	`int16_t RUC_GetPhaseFinalMecSpeed01Hz(RevUpCtrl_Handle_t *pHandle, uint8_t bPhase);`
`int16_t RUC_GetPhaseFinalTorque(CRUC this, uint8_t bPhase);`	`int16_t RUC_GetPhaseFinalTorque(RevUpCtrl_Handle_t *pHandle, uint8_t bPhase);`
`uint8_t RUC_GetNumberOfPhases(CRUC this);`	`uint8_t RUC_GetNumberOfPhases(RevUpCtrl_Handle_t *pHandle);`

Table 16. Translation from the *MCTuningClass.h* to the *ntc_temperature_sensor.h* file

SDK v4.3	SDK v5.x
[Temperature Sensor] ***Temperature sensor class exported methods***	
`int16_t TSNS_GetAvTemp_C(CTSNS this);`	`int16_t NTC_GetAvTemp_C(NTC_Handle_t *pHandle);`
`uint16_t TSNS_CheckTemp(CTSNS this);`	`uint16_t NTC_CheckTemp(NTC_Handle_t *pHandle);`

Table 17. Translation from the *MCTuningClass.h* to the *digital_output.h* file

SDK v4.3	SDK v5.x
[Digital Output] ***DigitalOutput class exported methods***	
`DOutputState_t DOUT_GetOutputState(CDOUT this);`	`DOutputState_t DOUT_GetOutputState(DOUT_handle_t *pHandle);`

Table 18. Translation from the *MCTuningClass.h* to the *motor_power_measurement.h* file

SDK v4.3	SDK v5.x
[***Motor Power Measurement]*** ***MotorPowerMeasurement class exported methods***	
`int16_t MPM_GetElMotorPowerW(CMPM this);`	`int16_t MPM_GetElMotorPowerW(MotorPowMeas_Handle_t *pHandle);`
`int16_t MPM_GetAvrgElMotorPowerW(CMPM this);`	`int16_t MPM_GetAvrgElMotorPowerW(MotorPowMeas_Handle_t *pHandle);`

Table 19. Translation from the *MCTuningClass.h* to the *speed_pos_fdbk.h* file

SDK v4.3	SDK v5.x
[Speed and Position] ***SpeednPosFdbk class exported methods***	
`int16_t SPD_GetElAngle(CSPD this);`	`int16_t SPD_GetElAngle(SpeednPosFdbk_Handle_t *pHandle);`
`int16_t SPD_GetMecAngle(CSPD this);`	`int16_t SPD_GetMecAngle(SpeednPosFdbk_Handle_t *pHandle);`
`int16_t SPD_GetAvrgMecSpeed01Hz(CSPD this);`	`int16_t SPD_GetAvrgMecSpeed01Hz(SpeednPosFdbk_Handle_t *pHandle);`

Table 19. Translation from the *MCTuningClass.h* to the *speed_pos_fdbk.h* file (continued)

SDK v4.3	SDK v5.x
`int16_t SPD_GetElSpeedDpp(CSPD this);`	`int16_t SPD_GetElSpeedDpp(SpeednPosFdbk_Handle_t *pHandle);`
`bool SPD_Check(CSPD this);`	`bool SPD_Check(SpeednPosFdbk_Handle_t *pHandle);`
`int16_t SPD_GetS16Speed(CSPD this);`	`bool SPD_IsMecSpeedReliable(SpeednPosFdbk_Handle_t *pHandle, int16_t *pMecSpeed01Hz);`
-	`int16_t SPD_GetS16Speed(SpeednPosFdbk_Handle_t *pHandle);`
`uint8_t SPD_GetElToMecRatio(CSPD this);`	`uint8_t SPD_GetElToMecRatio(SpeednPosFdbk_Handle_t *pHandle);`
`void SPD_SetElToMecRatio(CSPD this, uint8_t bPP);`	`void SPD_SetElToMecRatio(SpeednPosFdbk_Handle_t *pHandle, uint8_t bPP);`

Table 20. Translation from the *MCTuningClass.h* to the *virtual_speed_sensor.h* file

SDK v4.3	SDK v5.x
[Virtual Sensor Speed] ***VSS class exported methods***	
`void  VSPD_SetMecAcceleration(CSPD this, int16_t  hFinalMecSpeed01Hz, uint16_t hDurationms);`	`void VSS_SetMecAcceleration(VirtualSpeedSensor_Handle_t *pHandle, int16_t hFinalMecSpeed01Hz, uint16_t hDurationms);`
`int16_t  VSPD_GetLastRampFinalSpeed(CSPD this);`	`int16_t VSS_GetLastRampFinalSpeed(VirtualSpeedSensor_Handle_t *pHandle);`

Table 21. Translation from the *MCTuningClass.h* to the *sto_speed_pos_fdbk.h* file

SDK v4.3	SDK v5.x
[State Observer] ***STO class exported methods***	
`Volt_Components STO_GetEstimatedBemf(CSTO_SPD this);`	`Volt_Components STO_GetEstimatedBemf(STO_Handle_t *pHandle);`
`Curr_Components STO_GetEstimatedCurrent(CSTO_SPD this);`	`Curr_Components STO_GetEstimatedCurrent(STO_Handle_t *pHandle);`
`void STO_GetObserverGains(CSTO_SPD this, int16_t *pC2, int16_t *pC4);`	`void STO_GetObserverGains(STO_Handle_t *pHandle, int16_t *pC2, int16_t *pC4);`

Table 21. Translation from the *MCTuningClass.h* to the *sto_speed_pos_fdbk.h* file (continued)

SDK v4.3	SDK v5.x
void STO_SetObserverGains(CSTO_SPD this, int16_t hC1, int16_t hC2);	void STO_SetObserverGains(STO_Handle_t *pHandle, int16_t hC1, int16_t hC2);
void STO_GetPLLGains(CSTO_SPD this, int16_t *pPgain, int16_t *pIgain);	void STO_GetPLLGains(STO_Handle_t *pHandle, int16_t *pPgain, int16_t *pIgain);
void STO_SetPLLGains(CSTO_SPD this, int16_t hPgain, int16_t hIgain);	void STO_SetPLLGains(STO_Handle_t *pHandle, int16_t hPgain, int16_t hIgain);
void STO_ResetPLL(CSTO_SPD this);	void STO_ResetPLL(STO_Handle_t *pHandle);
int32_t STO_GetEstimatedBemfLevel(CSTO_SPD this);	int32_t STO_GetEstimatedBemfLevel(STO_Handle_t *pHandle);
int32_t STO_GetObservedBemfLevel(CSTO_SPD this);	int32_t STO_GetObservedBemfLevel(STO_Handle_t *pHandle);
void STO_BemfConsistencyCheckSwitch(CSTO_SPD this, bool bSel);	void STO_BemfConsistencyCheckSwitch(STO_Handle_t *pHandle, bool bSel);
bool STO_IsBemfConsistent(CSTO_SPD this);	bool STO_IsBemfConsistent(STO_Handle_t *pHandle);

Table 22. Translation from the *MCTuningClass.h* to the *sto_cordic_speed_pos_fdbk.h* file

SDK v4.3	SDK v5.x
[***State Observer using CORDIC algorithm]*** ***STO_CORDIC class exported methods***	
Volt_Components STO_CR_GetEstimatedBemf(CSTO_CR_SPD this);	Volt_Components STO_CR_GetEstimatedBemf(STO_CR_Handle_t* pHandle);
Curr_Components STO_CR_GetEstimatedCurrent(CSTO_CR_SPD this);	Curr_Components STO_CR_GetEstimatedCurrent(STO_CR_Handle_t* pHandle);
void STO_CR_GetObserverGains(CSTO_CR_SPD this, int16_t *pC2, int16_t *pC4);	void STO_CR_GetObserverGains(STO_CR_Handle_t* pHandle, int16_t *pC2, int16_t *pC4);
void STO_CR_SetObserverGains(CSTO_CR_SPD this, int16_t hC1, int16_t hC2);	void STO_CR_SetObserverGains(STO_CR_Handle_t* pHandle, int16_t hC1, int16_t hC2);
int32_t STO_CR_GetEstimatedBemfLevel(CSTO_CR_SPD this);	int32_t STO_CR_GetEstimatedBemfLevel(STO_CR_Handle_t* pHandle);
int32_t STO_CR_GetObservedBemfLevel(CSTO_CR_SPD this);	int32_t STO_CR_GetObservedBemfLevel(STO_CR_Handle_t* pHandle);

Table 22. Translation from the *MCTuningClass.h* to the *sto_cordic_speed_pos_fdbk.h* file (continued)

SDK v4.3	SDK v5.x
void STO_CR_BemfConsistencyCheckSwitch(CSTO_CR_SPD this, bool bSel);	void STO_CR_BemfConsistencyCheckSwitch(STO_CR_Handle_t* pHandle, bool bSel);
bool STO_CR_IsBemfConsistent(CSTO_CR_SPD this);	bool STO_CR_IsBemfConsistent(STO_CR_Handle_t* pHandle);

Table 23. Translation from the *MCTuningClass.h* to the *speed_torq_ctrl.h* file

SDK v4.3	SDK v5.x
[Speed and Torque Control] ***SpeednTorqCtrl class exported methods***	
int16_t STC_GetMecSpeedRef01Hz(CSTC this);	int16_t STC_GetMecSpeedRef01Hz(SpeednTorqCtrl_Handle_t *pHandle);
int16_t STC_GetTorqueRef(CSTC this);	int16_t STC_GetTorqueRef(SpeednTorqCtrl_Handle_t *pHandle);
STC_Modality_t STC_GetControlMode(CSTC this);	STC_Modality_t STC_GetControlMode(SpeednTorqCtrl_Handle_t *pHandle);
uint16_t STC_GetMaxAppPositiveMecSpeed01Hz(CSTC this);	uint16_t STC_GetMaxAppPositiveMecSpeed01Hz(SpeednTorqCtrl_Handle_t *pHandle);
int16_t STC_GetMinAppNegativeMecSpeed01Hz(CSTC this);	int16_t STC_GetMinAppNegativeMecSpeed01Hz(SpeednTorqCtrl_Handle_t *pHandle);
Curr_Components STC_GetDefaultIqdref(CSTC this);	Curr_Components STC_GetDefaultIqdref(SpeednTorqCtrl_Handle_t *pHandle);
void STC_SetNominalCurrent(CSTC this, uint16_t hNominalCurrent);	void STC_SetNominalCurrent(SpeednTorqCtrl_Handle_t *pHandle, uint16_t hNominalCurrent);

Table 24. Translation from the *MCTuningClass.h* to the *state_machine.h* file

SDK v4.3	SDK v5.x
[State Machine] ***StateMachine class exported methods***	
State_t STM_GetState(CSTM this);	State_t STM_GetState(STM_Handle_t *pHandle);
uint32_t STM_GetFaultState(CSTM this);	uint32_t STM_GetFaultState(STM_Handle_t *pHandle);

Table 25. Translation from the *MCTuningClass.h* to the *bus_voltage_sensor.h* file

SDK v4.3	SDK v5.x
[Bus Voltage Sensor] ***BusVoltageSensor class exported methods***	
`uint16_t VBS_GetAvBusVoltage_V(CVBS this);`	`uint16_t VBS_GetAvBusVoltage_V(BusVoltageSensor_Handle_t *pHandle);`
`uint16_t VBS_CheckVbus(CVBS this);`	`uint16_t VBS_CheckVbus(BusVoltageSensor_Handle_t *pHandle);`

Table 26 lists APIs in the *MCTuningClass.h* file that are not exposed to users in SDK v5.0. These APIs only refer to the Motor Profiler tool, where a specific binary firmware is delivered to support its use.

Table 26. *MCTuningClass.h* APIs not exposed in SDK v5.x

SDK v4.3	SDK v5.x
[Self-Commissioning] ***Selfcommissioning class exported methods***	
`uint8_t SCC_GetState(CSCC this);`	Not applicable.
`uint8_t SCC_GetSteps(CSCC this);`	
`uint32_t SCC_GetRs(CSCC this);`	
`uint32_t SCC_GetLs(CSCC this);`	
`uint32_t SCC_GetKe(CSCC this);`	
`uint32_t SCC_GetVbus(CSCC this);`	
`uint32_t SCC_GetEstNominalSpeed(CSCC this);`	
`void SCC_ForceProfile(CSCC this);`	
`void SCC_StopProfile(CSCC this);`	
`void SCC_SetPolesPairs(CSCC this, uint8_t bPP);`	
`void SCC_SetNominalCurrent(CSCC this, float fCurrent);`	
`float SCC_GetNominalCurrent(CSCC this);`	
`void SCC_SetLdLqRatio(CSCC this, float fLdLqRatio);`	
`float SCC_GetLdLqRatio(CSCC this);`	
`void SCC_SetNominalSpeed(CSCC this, int32_t wNominalSpeed);`	
`int32_t SCC_GetNominalSpeed(CSCC this);`	
`int32_t SCC_GetEstMaxOLSpeed(CSCC this);`	
`int32_t SCC_GetEstMaxAcceleration(CSCC this);`	
`int16_t SCC_GetStartupCurrentS16(CSCC this);`	
`float SCC_GetStartupCurrentAmp(CSCC this);`	
`void SCC_SetCurrentBandwidth(CSCC this, float fCurrentBW);`	
`float SCC_GetCurrentBandwidth(CSCC this);`	
`uint16_t SCC_GetPWMFrequencyHz(CSCC this);`	
`uint8_t SCC_GetFOCRepRate(CSCC this);`	

Table 26. *MCTuningClass.h* APIs not exposed in SDK v5.x (continued)

SDK v4.3	SDK v5.x
[One Touch Tuning] ***One touch tuning class exported methods***	
void OTT_ForceTuning(COTT this);	Not applicable.
uint32_t OTT_GetNominalSpeed(COTT this);	
uint8_t OTT_GetSteps(COTT this);	
uint8_t OTT_GetState(COTT this);	
bool OTT_IsSpeedPITuned(COTT this);	
float OTT_fGetNominalSpeedRPM(COTT this);	
void OTT_SetPolesPairs(COTT this, uint8_t bPP);	
void OTT_SetNominalCurrent(COTT this, uint16_t hNominalCurrent);	
void OTT_SetSpeedRegulatorBandwidth(COTT this, float fBW);	
float OTT_GetSpeedRegulatorBandwidth(COTT this);	
float OTT_GetJ(COTT this);	
float OTT_GetF(COTT this);	
bool OTT_IsMotorAlreadyProfiled(COTT this);	

Table 27 lists APIs in the *MCTuningClass.h* file that are not exposed to users in SDK v5.0. These APIs are no present in SDK v5.0. In SDK v4.3, they are used to provide object visibility (also known as class export).

Table 27. *MCTuningClass.h* APIs not existing in SDK v5.x

SDK v4.3	SDK v5.x
[Motor Control Tuning] ***MCTuning class exported methods***	
`CFW MCT_GetFluxWeakeningCtrl(CMCT this);`	Not applicable.
`CFF MCT_GetFeedForwardCtrl(CMCT this);`	
`CHFI_FP MCT_GetHFICtrl(CMCT this);`	
`CPI MCT_GetSpeedLoopPID(CMCT this);`	
`CPI MCT_GetIqLoopPID(CMCT this);`	
`CPI MCT_GetIdLoopPID(CMCT this);`	
`CPI MCT_GetFluxWeakeningLoopPID(CMCT this);`	
`CPWMC MCT_GetPWMnCurrFdbk(CMCT this);`	
`CRUC MCT_GetRevupCtrl(CMCT this);`	
`CSPD MCT_GetSpeednPosSensorMain(CMCT this);`	
`CSPD MCT_GetSpeednPosSensorAuxiliary(CMCT this);`	
`CSPD MCT_GetSpeednPosSensorVirtual(CMCT this);`	
`CSTC MCT_GetSpeednTorqueController(CMCT this);`	
`CSTM MCT_GetStateMachine(CMCT this);`	
`CTSNS MCT_GetTemperatureSensor(CMCT this);`	
`CVBS MCT_GetBusVoltageSensor(CMCT this);`	
`CDOUT MCT_GetBrakeResistor(CMCT this);`	
`CDOUT MCT_GetNTCRelay(CMCT this);`	
`CMPM MCT_GetMotorPowerMeasurement(CMCT this);`	
`CSCC MCT_GetSelfCommissioning(CMCT this);`	
`COTT MCT_GetOneTouchTuning(CMCT this);`	

Table 28. Translation from the *MCTuningClass.h* to the *hifreqinj_fpu_ctrl.h* file

SDK v4.3	SDK v5.x[1]
[High Frequency Injection] ***HFICtrl class exported methods***	
`int16_t HFI_FP_GetRotorAngleLock(CHFI_FP this);`	`int16_t HFI_FP_GetRotorAngleLock(HFI_FP_Ctrl_Handle_t *pHandle);`
`int16_t HFI_FP_GetSaturationDifference(CHFI_FP this);`	`int16_t HFI_FP_GetSaturationDifference(HFI_FP_Ctrl_Handle_t *pHandle);`
`int16_t HFI_FP_GetCurrent(CHFI_FP this);`	`int16_t HFI_FP_GetCurrent(HFI_FP_Ctrl_Handle_t *pHandle);`
`CPI HFI_FP_GetPITrack(CHFI_FP this);`	`PID_Handle_t* HFI_FP_GetPITrack(HFI_FP_Ctrl_Handle_t *pHandle);`
`void HFI_FP_SetMinSaturationDifference(CHFI_FP this, int16_t hMinSaturationDifference);`	`void HFI_FP_SetMinSaturationDifference(HFI_FP_Ctrl_Handle_t *pHandle, int16_t MinSaturationDifference);`

1. This feature is not implemented in SDK v5.0 but in later versions.

Table 29. Translation from the *MCTasks.h* to the *mc_tasks.h* file

SDK v4.3	SDK v5.x
`void MCboot(CMCI oMCIList[],CMCT oMCTList[]);`	`void MCboot( MCI_Handle_t* pMCIList[], MCT_Handle_t* pMCTList[] );`
`void MC_Scheduler(void);`	`void MC_Scheduler(void);`
`void TSK_SafetyTask(void);`	`void TSK_SafetyTask(void);`
`uint8_t TSK_HighFrequencyTask(void);`	`uint8_t TSK_HighFrequencyTask(void);`
`void TSK_DualDriveFIFOUpdate(void *oDrive);`	`void TSK_DualDriveFIFOUpdate(void *oDrive);`
`void TSK_HardwareFaultTask(void);`	`void TSK_HardwareFaultTask(void);`

Table 30. Translation from the *MC.h* to the *mc_extended_api.h* file

SDK v4.3	SDK v5.x
`CMCI GetMCI(uint8_t bMotor);`	`MCI_Handle_t * GetMCI(uint8_t bMotor);`
`CMCT GetMCT(uint8_t bMotor);`	`MCT_Handle_t* GetMCT(uint8_t bMotor);`
`void MC_SetDAC(DAC_Channel_t bChannel, MC_Protocol_REG_t bVariable);`	`void MC_SetDAC(DAC_Channel_t bChannel, MC_Protocol_REG_t bVariable);`
`void MC_SetUserDAC(DAC_UserChannel_t bUserChNumber, int16_t hValue);`	`void MC_SetUserDAC(DAC_UserChannel_t bUserChNumber, int16_t hValue);`

7 Cubification

Cubification is the migration of a motor-control application from object-oriented C source code to standard C source code so that it complies with the use of STM32CubeMX.

7.1 Conversion of library use from SPL to HAL or LL

The SPL2LL-Converter tool from STMicroelectronics converts SPL to LL. It is available for download on *www.st.com* to support users during this conversion stage. A presentation and an application note are also available to assist users using the tool.

7.2 Use of STM32CubeMX tools

STM32CubeMX is the STM32Cube initialization code generator from STMicroelectronics. It aims at supporting users developing their applications.

STM32CubeMX can be downloaded from *www.st.com.* The *STM32CubeMX for STM32 configuration and initialization C code generation* user manual (UM1718) is also available to guide users through their use of the tool.

8 Revision history

Table 31. Document revision history

Date	Revision	Changes
5-Mar-2018	1	Initial release.
23-Apr-2018	2	Updated *Table 2: SDK v5.0 versus SDK v4.3 comparison summary*.
10-Jul-2018	3	Extended document scope to SDK v5.x: – Updated document title – Updated *Supported microcontrollers*, *Features*, *MIPS*, and *Memory size* in *Table 2* – Updated *Table 3* – Added *Table 4*, *Table 5*, and *Table 6* – Updated *Table 8* with dual-motor APIs

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2018 STMicroelectronics – All rights reserved