MotionChip™ II Configuration Setup

TECHNOSOFT

User Manual

Preliminary

TECHNOSOFT

MotionChip™ II
Configuration Setup

P091.055.MCII.STP.UM.0806

Technosoft S.A.
Buchaux 38
CH-2022 BEVAIX
Switzerland
Tel.: +41 (0) 32 732 5500
Fax: +41 (0) 32 732 5504
contact@technosoftmotion.com
www.technosoftmotion.com/

Read This First

Whilst Technosoft believes that the information and guidance given in this manual is correct, all parties must rely upon their own skill and judgment when making use of it. Technosoft does not assume any liability to anyone for any loss or damage caused by any error or omission in the work, whether such error or omission is the result of negligence or any other cause. Any and all such liability is disclaimed.

About This Manual

This book is a technical reference manual for the **MotionChip™ II** registers, parameters and variables. Describes the MotionChip II operation and how to setup its registers and parameters starting from user application data needed to program the MotionChip II in the Technosoft Motion Language (in short **TML**) environment.

Scope of This Manual

The TML programming of drives based on **MotionChip II** involves 2 steps:

Step 1 - Parameters setup
Step 2 - Motion programming

This manual describes the first step – Parameters setup.

The goal of second step is the programming of the motion in TML environment in accordance with the user application data. This step is described in the user manual ***MotionChip II TML Programming***.

Both steps can be performed using **IPM Motion Studio** – a development platform offering easy-to-use graphical programming for devices based on MotionChip II. The output from IPM Motion Studio is a TML program, which can be downloaded into the non-volatile memory of the drive and can be started automatically after power on.

Depending on your application configuration, you have the following options for splitting the tasks between your host and your drive based on MotionChip II:

1. ***Host control is absent***. The complete motion application is programmed in the drive using TML
2. ***Host control is done via I/O handshake***. The host commands are set via digital or analogue signals. The drive answers also using digital signals
3. ***Minimal host control via a communication channel***. The host control is reduced at calling motion functions implemented in the drive non-volatile memory and requesting

status information. The motion functions from the drive memory can be developed separately using IPM Motion Studio

4. ***Extended host control via a communication channel***. The host sends all the TML commands needed to program the motion, but does not perform the drive setup. This is done via a TML program executed automatically after power-on. The TML program can be developed using IPM Motion Studio.
5. ***Full host control via a communication channel.*** In this case the host performs both the drive setup and the motion programming. There is no TML program stored in the drive.

You need this manual only if you plan to use option 5.

Notational Conventions

This document uses the following conventions:

- TML – Technosoft Motion Language
- PCR.6-0 it's referring to bits 6 to 0 of the PCR register
- Program examples are shown with a special font. Here is an example:

```
ENIO#36;   //Configure dual function pin as I/O line 36
user_1 = IN#36;   //Read I/O line 36 data into variable user_1
```

Related Documentation from Technosoft

MotionChip™ II TML Programming (part no. P091.055.MCII.TML.UM.xxxx) describes in detail TML basic concepts, motion programming, functional description of TML instructions for high level or low level motion programming, communication channels and protocols. Also give a detailed description of each TML instruction including syntax, binary code and examples.

MotionChip™ II Data sheet (part no. P091.055.MCII-QFP100.DSH.xxxx) presents the MotionChip II features and specifications, and how to interface it with typical external devices.

IPM Motion Studio User Manual (part no. P091.088.E075.UM.xxxx) describes how to use the IPM Motion Studio – the complete development platform for MotionChip II including: motion system setup & tuning wizard, motion sequence programming wizard, testing and debugging tools like: data logging, watch, control panels, on-line viewers of TML registers, parameters and variables, etc.

If you Need Assistance …

If you want to …	Contact Technosoft at …
Visit Technosoft online	World Wide Web: http://www.technosoftmotion.com/
Receive general information or assistance	World Wide Web: http://www.technosoftmotion.com/ Email: contact@technosoftmotion.com
Ask questions about product operation or report suspected problems	Fax: (41) 32 732 55 04 Email: hotline@technosoftmotion.com
Make suggestions about or report errors in documentation	Mail: Technosoft SA Buchaux 38 CH -2022 Bevaix, NE Switzerland

Trademarks

MotionChip is a trademark of Technosoft SA.

This page is empty

Contents

Figures

Tables

This page is empty

1. MotionChip II Overview

1.1. MotionChip II concept

Embedding the latest digital signal processor structures Technosoft **MotionChip II** is a high performance ready to run motion controller. Motion Chip II does not require any DSP code development, thus being ideal for rapid and cost effective design of fully digital, intelligent drives for various motor types.

The MotionChip II can:

- Operate stand-alone or in master/slave, multiple axis configuration
- Control three motor types: brushless DC, brushless AC and DC brush
- Implement various command structures: open loop, torque, speed, position
- Work with different motion and protection sensors (position, speed, current, voltage, temperature)
- Use different communication channels such as RS232, RS485 or CAN-bus
- Execute advanced motion language commands and motion sequences

Intended to cover a major part of basic and complex motion applications, the MotionChip II has the special advantage to be a highly flexible structure at the level of:

- Motion structure configuration (selection of motor technology, control type, sensors type)
- Motion implementation with high-level motion language commands

Relative to existing solutions, the MotionChip II offers many advantages:

- Usable for different motor technologies
- Implementation of multiple motion control configurations, including vector control for AC drives
- Implementation of complete digital control loops, including current/torque control
- Powerful motion language including motion modes, decision blocks, function calls, event-driven motion updates, interrupts
- Stand-alone or slave operation
- Minimal requirements for setup configuration and use
- Easy to embed in user's hardware structures
- Software-less device (no programming effort required)
- High-level development tools for application build, test and debug

1.2. Specifications

Due the high degree of hardware and advance software integration, the MotionChip II is the ideal solution for motion control structures implementation.

Here is a summary of MotionChip II features:

- High-performance ready to run DSP Motion Controller, 40MHz, 40MIPS;
- Single-Chip solution for control of brushless DC, brushless AC (PMSM) and DC brush motors;

- Operation Modes: Stand-alone – executes motion sequences from local memory; Slave – multi-axis cases;
- Communication channels: serial RS-232, serial RS-485 and CAN-bus;
- Flexible structure allowing: open-loop, torque, speed or position close-loop control;
- Typical sampling rates: 10kHz torque loop, 1kHz speed/position loop;
- Feedback signals: 1 – 3 Currents; Hall sensors; Position read from: incremental encoder (on-chip interface), potentiometer (analogue); Speed estimated from position. Two temperature sensors; DC-bus voltage (V_{DC});
- Programmable with high-level Technosoft Motion Language (TML);
- No DSP Code Development Required;
- Motion language including motion modes, decision blocks, functions, arithmetic & logic unit;
- Accurate profile generator with automatic round-off correction: Position range: 32-bits; Speed/acceleration range: 16-bit integer part, 16-bit fractional part;
- 18 programmable event triggers when motion modes can be changed on-the-fly;
- 12 programmable TML interrupts;
- Main Inputs: Power Drive Fault (interrupt input); Enable/Disable (interrupt input); 2 Limit switches (interrupt inputs); Encoder Index (capture input); 2^{nd} Encoder Index (capture input); Start mode: Stand Alone/Slave (wait commands from a master)
- Main Outputs: 4-6 PWM commands; Brake transistor command; Interrupt to host; Ready
- Up to 17 general-purpose I/O
- Advanced PWM command methods: V_{DC} variation compensation; Dead-time compensation; 3^{rd} harmonic injection; Wobbling for EMI reduction
- Brake transistor control
- Integrated protections: over current, over voltage, under voltage, over temperature, I^2t, control error
- Development tools available: IPM Motion Studio allows you to configure, parameterize and program the motion for your application in a user friendly graphical interface, it also includes a compiler, command interpreter, graphical display for traced variables, watch functions

1.3. Programming the MotionChip II.

The MotionChip II is programmable using the Technosoft Motion Language (TML). TML is a high-level language allowing you to:

- Setup a drive built with MotionChip II for a given application
- Program and execute motion sequences

The setup part consists in assigning the right values for the TML registers and parameters. Through this process you can:

- Describe your application configuration (as motor and sensors type)
- Select specific operation settings (as motor start mode, PWM mode, sampling rates, etc.)
- Setup the controllers' parameters (current, speed, position), etc.

The next part is for motion programming. Here the TML allows you to:

- Set various motion modes (profiles, contouring, electronic gearing or camming, etc.)
- Change the motion modes and/or the motion parameters on-the-fly
- Execute homing sequences
- Control the program flow through:
    - Conditional jumps and calls of TML functions
    - TML interrupts generated on pre-defined or programmable conditions (protections triggered, detection of transitions on limit switch or capture inputs, etc.)
    - Waits for programmed events to occur
- Handle digital I/O and analogue input signals
- Execute arithmetic and logic operations
- Perform data transfers between axes
- Control motion of an axis from another one via motion commands sent between axes
- Send commands to a group of axes (multicast). This includes the possibility to start simultaneously motion sequences on all the axes from the group

Due to a powerful instruction set, the motion programming in TML is quick and easy even for complex motion applications. The result is a high-level motor-independent program which once conceived may be used in other applications too.

1.4. TML Environment

The TML environment includes three basic components:

1. "TML processor"
2. Trajectory generator
3. Motor control kernel

The software-implemented "TML processor" represents the core of the TML environment. It decodes and executes the TML commands. Like any processor, it includes specific elements as program counter, stack, ALU, interrupt management and registers.

The trajectory generator computes the position, speed, torque or voltage reference at each sampling step, depending on the selected motion mode.

The motor-control kernel implements the control loops including: the acquisition of the feedback sensors, the controllers, the PWM commands, the protections, etc.

When the "motion processor" executes a motion command, it translates them into actions upon the trajectory generator and/or the motor control kernel.

1.5. Program Execution

The TML programs are executed sequentially, one instruction after the other. A 16-bit instruction pointer (IP) controls the program flow. As the binary code of a TML instruction may have up to 5 words, during its execution the IP is increased accordingly. When the execution of a TML instruction ends, the IP always points to the next TML instruction, or more exactly to the first word of its binary code.

The sequential execution may be interrupted by one of the following causes:

- A TML command received through a communication channel (on-line commands);
- A branch to the interrupt service routine (ISR) when a TML interrupt occurs;
- The need to send the master position to the slave axes when the current axis is set as master for electronic gearing or camming
- A `GOTO` or `CALL` instruction;
- A return from a TML function – `RET` or from a TML interrupt – `RETI;`
- During the execution of the instructions: `WAIT!` (wait event), `SEG` (new contour segment) and data transfers between axes of type `local_variable = [x]remote_variable`, which all keep the IP unchanged (i.e. loop on the same instruction) until a specific condition is achieved
- After execution of the `END` instruction.

The on-line commands have the highest priority and act like interrupts: when an on-line command is received through any communication channel, it starts to be executed immediately after the current TML instruction is completed.

If an on-line command is received during a wait loop, e.g. when `WAIT!` or `SEG` commands are processed, the wait loop is temporary suspended, to permit the execution of the on-line command.

The TML works with 3 types of commands, presented in Table 1.1.

Table 1.1 *Type of TML commands*

TML Command Type	Execution	
	From a TML program	**Send via communication**
Immediate	√	√
Sequential	√	-
On-line	-	√

The immediate commands may be send via a communication channel, or can reside a TML program. These commands don't require any wait loops to complete. Their execution is straightforward and can't be interrupted by other TML commands.

The sequential commands require a wait loop to complete i.e. will not permit IP to advance until the wait condition becomes true. In this category enter commands like:

```
WAIT!;          // Wait a programmed event to occur
```

`SEG` *`Time, Increment;`* // Set a contour segment with parameters *Time* and *Increment* to be executed when the previous one ends

`local_variable = [x]remote_variable;` // Get value of *remote_variable* from axis x and put it in *local_variable*

The sequential commands can reside only in a TML program saved in the local memory.

Remark: *If a sequential command is sent via a communication channel, it is immediately executed as if the wait loop condition is always true.*

The on-line commands may be sent only via a communication channel. These commands can't be included in a TML program. The on-line commands do not have an associated mnemonic and syntax rules as they are do not need to be recognized by the TML compiler.

1.6. TML Program Structure

The main section of a TML program starts with the instruction BEGIN and ends with the instruction `END`. It is divided into two parts:

- Setup part
- Motion programming part

The setup part starts after `BEGIN` and lasts until the `ENDINIT` instruction, meaning "END of INITitialization". This part of the TML program consists mainly of assignment instructions, which shall set the TML registers and the TML parameters in accordance with your application data. When the `ENDINIT` command is executed, key features of the TML environment are initialized according with the setup data. After the `ENDINIT` execution, the basic configuration involving the motor and sensors types or the sampling rates, cannot be changed unless a reset is performed.

The motion programming part starts after the `ENDINIT` instruction until the `END` instruction. All the TML programs (the main section) should end with the TML instruction `END`. When `END` instruction is encountered, the sequential execution of a TML program is stopped.

Apart from the main section, a TML program also includes the TML interrupt vectors table, the interrupt service routines (ISRs) for the TML interrupts and the TML functions. A typical structure for a TML program is presented in Figure 1.1

```
      BEGIN;              // TML program start
       ...
                          // Setup part of the main section
       ...
      ENDINIT;            // end of initialization
       ...
                          // Motion programming part of the main section
       ...
      END;                // end of the main section

InterruptTable:           // start of the interrupt vectors table
   @Int0_Axis_disable_ISR;
   @Int1_PDPINT_ISR;
   @Int2_Software_Protection_ISR;
   @Int3_Control_Error_ISR;
   @Int4_Communication_Error_ISR;
   @Int5_Wrap_Around_ISR;
   @Int6_Limit_Switch_Positive_ISR;
   @Int7_Limit_Switch_Negative_ISR;
   @Int8_Capture_ISR;
   @Int9_Motion_Complete_ISR;
   @Int10_Update_Contour_Segment_ISR;
   @Int11_Event_Reach_ISR;
Int0_Axis_disable_ISR:        // Int0_Axis_disable_ISR body
      ...
      RETI;                   // RETurn from TML ISR
      ...
Int11_Event_Reach_ISR:        // Int11_Event_Reach_ISR body
      ...
      RETI;                   // RETurn from TML ISR
Function1:                    // Start of the first TML function named Function1
      ...
      RET;                    // RETurn from TML function named Function 1
      ...
FunctionX:                    // Start of the last TML function named FunctionX
      ...
      RET;                    // RETurn from the last TML function named Function X
```

Figure 1.1 *Typical structure of a TML Program*

1.7. Development tools for MotionChip II

A complete range of development tools is available for the MotionChip II user. It starts with a basic evaluation kit, the MotionChip™ II Starter Kit (**MCIISK**), accompanied by power module PM-50. On the software side, the powerful **IPM Motion Studio** platform allows you to configure, parameterize and program the motion for your application.

The MotionChip™ II Starter Kit boards contains the MotionChip II, external E^2ROM and RAM memories, an RS-232 interface, and extension connectors, to interface the MotionChip II with external power modules and/or external I/O interfaces. The MCIISK kit can be combined with specific power amplifier modules and motors, thus becoming a complete motion structure for the evaluation of motion applications based on the MotionChip II. See the "**MotionChip™ II Starter Kit User Manual**" for more details about MCIISK kit contents and features.

With the MCIISK board you receive a template for IPM Motion Studio, which is a high level graphical Windows environment for MotionChip II applications development. It allows you to configure and parameterize a motion system (including tuning and auto tuning of controllers) and to define motion sequences using high level integrated tools which automatically generate Technosoft Motion Language source code (TML instructions). Embedded code development tools allow you to further edit or directly compile, link and generate executable code to be downloaded to the MotionChip II. Finally, advanced graphics tools — like data logger, control panel and view/watch of TML parameters, registers and memory — can be used to analyze the behavior of the motion system. See the "**IPM Motion Studio User Manual**" for more details about IPM Motion Studio contents and features.

This page is empty

2. Motor control configurations

MotionChip II is designed to control brushless motors and brush DC motors.

Brushless motors are vector controlled and can be driven with sinusoidal commutation (PMSM) or with trapezoidal commutation (BLDC).

Each type of motor can be controlled in position, speed and current/torque closed loop or can be commanded in open loop. In the following pages for each type of motor it will be presented supported control modes.

2.1. Brushless motor with sinusoidal commutation (PMSM)

The brushless motor with sinusoidal commutation can be controlled in test (open loop), current/torque, speed or position control mode. In any of these configurations, the three phases of the motor are supplied from the power converter with PWM voltages having a sinusoidal fundamental component. In all closed loops MotionChip II implements a vector control scheme; a d-q reference frame is used for the control implementation. In such cases, a position sensor is needed.

Position mode. The system controls the motor position.

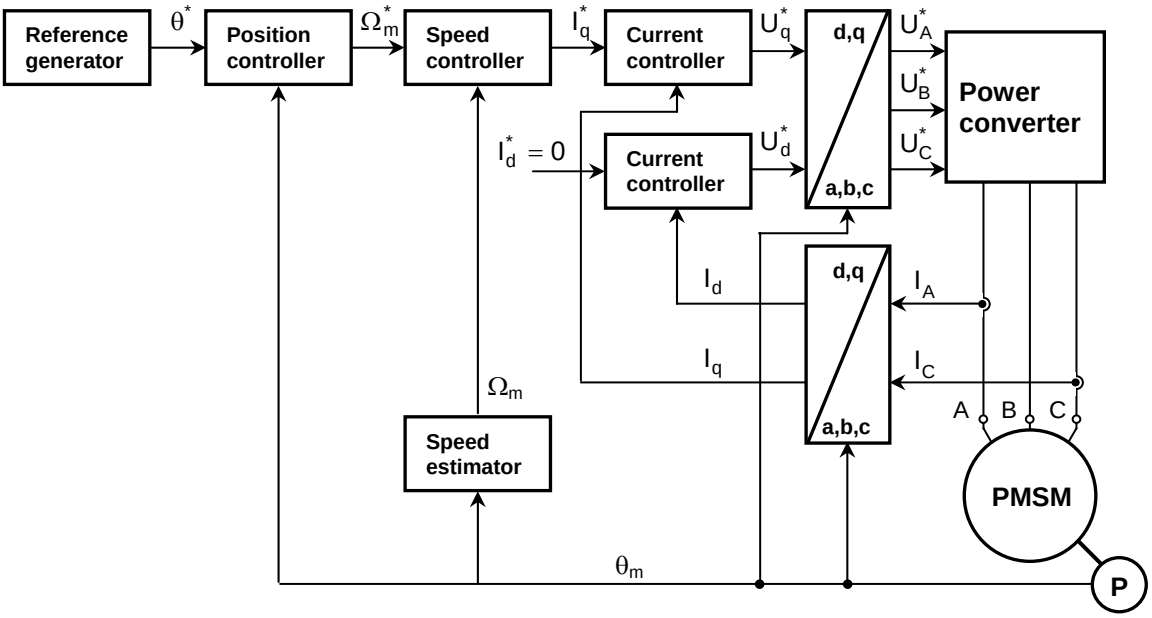


Figure 2.1. *Scheme of PMSM operating in position mode*

The control of PMSM is based on direct and inverse coordinate transformations. In the direct transformation the phase motor currents (two of them are measured, the third is computed from the first two) are transformed in $\mathbf{i_{ds}}$ and $\mathbf{i_{qs}}$ defined in stator stationary frame. In the next step $\mathbf{i_{ds}}$ and $\mathbf{i_{qs}}$, based on position information, are transformed in $\mathbf{i_d}$ and $\mathbf{i_q}$ currents defined in the rotating frame attached to the rotor. The inverse coordinate transformations are used to obtain the phase voltages references used by PWM converter.

System position is imposed by the reference generator. Implemented controllers are PI type. Required sensors are: current sensors and position sensors; the speed is estimated from position.

Position mode without speed loop. The system controls the motor position, but the speed loop is disabled. The scheme contains a slow loop represented by the position loop and a fast loop represented by current loop. Required sensors are: current sensors and position sensors. The reference generator imposes the system position.

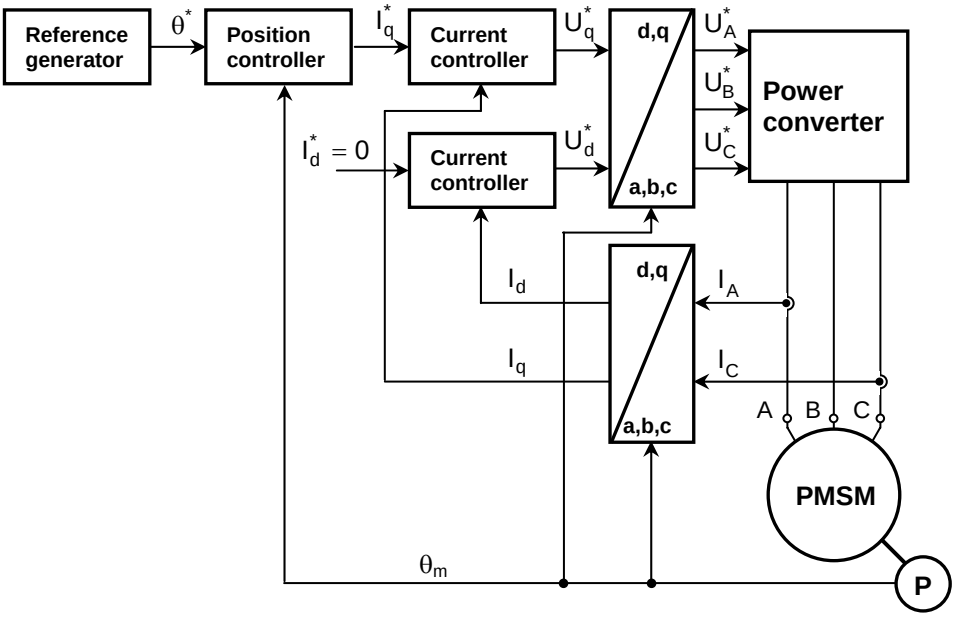


Figure 2.2. *Scheme of PMSM motor operating inposition mode without speed loop*

Speed mode. The system controls the motor speed. The scheme contains a slow loop represented by the speed loop and a fast loop represented by the current loop. Required sensors for these scheme are: current sensors and position sensor; the speed is estimated from position information. The reference generator imposes the system speed.

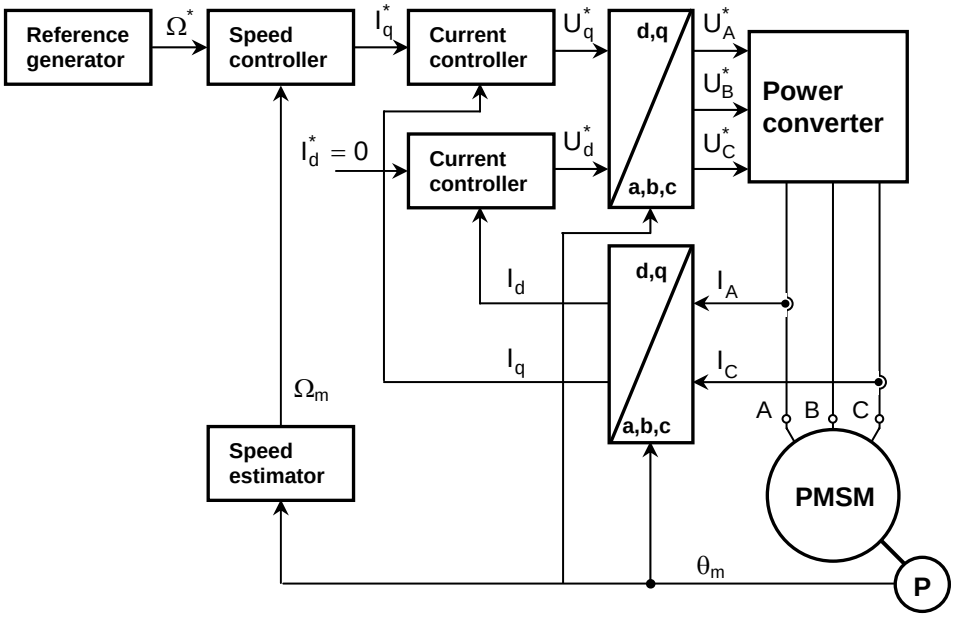


Figure 2.3. *Scheme of PMSM operating in speed mode*

Torque/current mode. The system controls the motor torque. The scheme contains only a fast loop represented by the current loop. Required sensors for these scheme are: current sensors, and position sensor. The reference generator imposes the system torque/current.

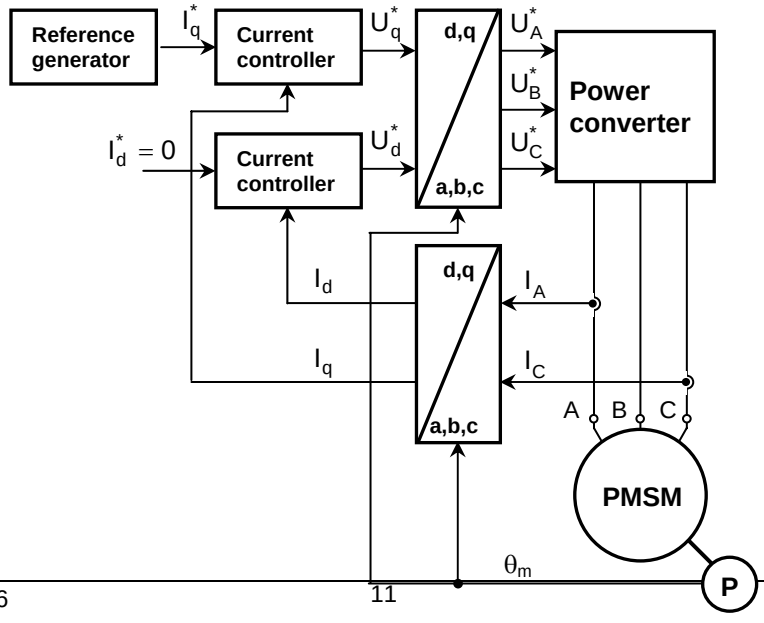


Figure 2.4. *Scheme of PMSM operating in torque/current control*

Test mode. In the test mode, the motor is driven in open loop**.** For these scheme there ar no required sensors. The reference in test mode is a variable frequency, applied to U/f block and to voltage commutation block.

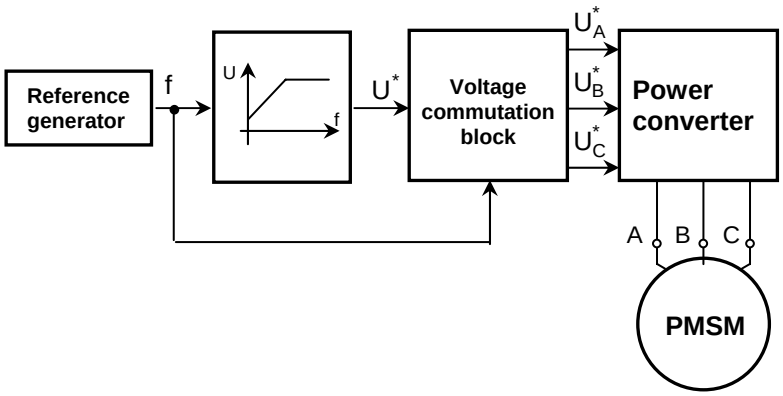


Figure 2.5. *Scheme of PMSM operating in test mode*

2.2. Brushless motor with trapezoidal commutation (BLDC)

The BLDC motor can be driven in test, current/torque, speed, position control mode. In any of these configurations, only two phases of the motor are supplied from the power converter at one moment, while the third phase is unconnected. The connection sequence is decided based on the information supplied by Hall sensors spaced at 120^{o} (electrical) one of the other, and changed accordingly during the motion.

The motion is controlled by varying the average value of the voltage applied to the motor, through specific PWM command of the power converter switches.

There are several possible combinations of Hall sensor positions, depending on their relative position to the motor phase windings. You need to specify the case of your application in order to implement a correct commutation scheme for the motor.

Position mode. The position of the motor is controlled by the system. The scheme contains three loops, a position loop and as inner loops, a speed and a current control loop. Sensors needed in this control mode are position sensor, currents sensors and Hall sensors; the speed is estimated from position information. The reference generator imposes the motor position.

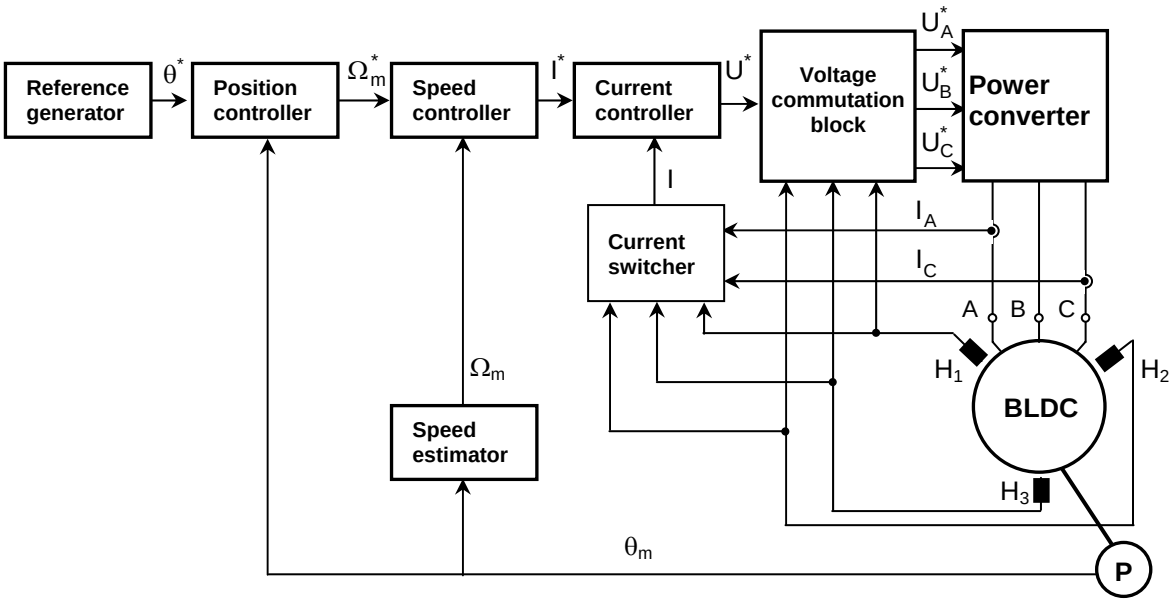
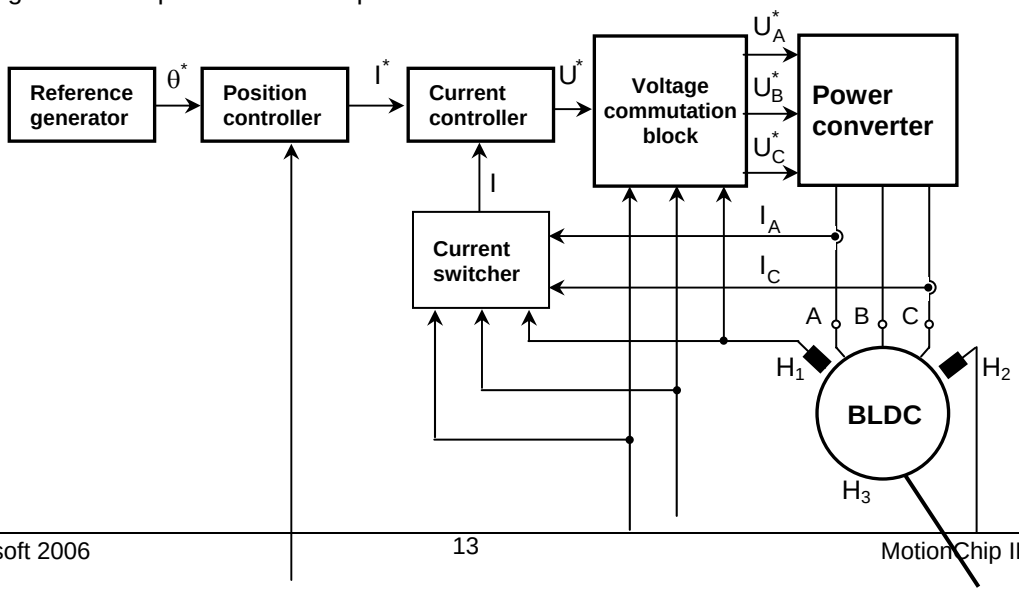


Figure 2.6. *Scheme of BLDC motor operating in position mode*

Position mode without speed loop. The system controls the position of the motor. The scheme contains only two loops, a position loop and, as inner loop, a current control loop. Sensors needed in this control mode are position sensor, currents sensors and Hall sensors. The reference generator imposes the motor position.



.

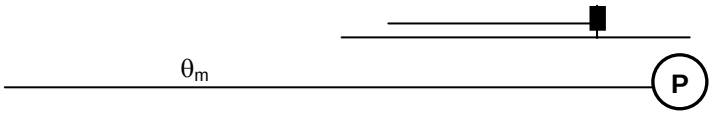


Figure 2.7. *Scheme of BLDC motor operating in position mode without speed loop*

Speed mode. The system controls the speed of the motor. The scheme contains only two loops, a position loop and, as inner loop, a current control loop. Sensors needed in this control mode are currents sensors and Hall sensors; the speed is estimated from position information. The reference generator imposes the motor speed.

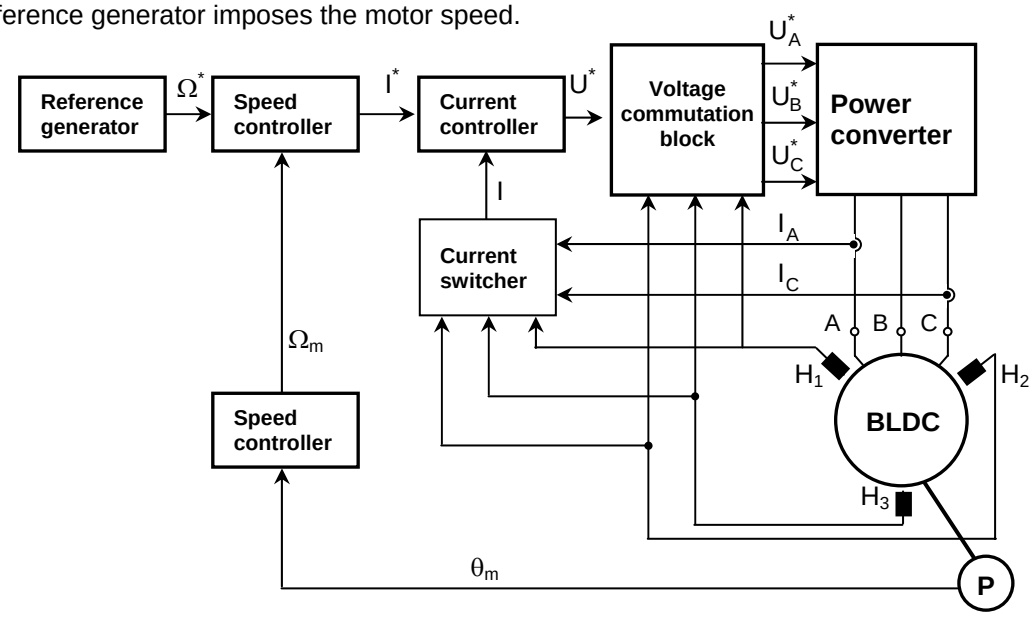


Figure 2.8. *Scheme of BLDC motor operating in speed mode*

Torque/current mode. The system controls the torque/current of the motor. The scheme contains only the current control loop. Sensors needed in this control mode are current sensors and Hall sensors. The reference generator imposes the motor torque.

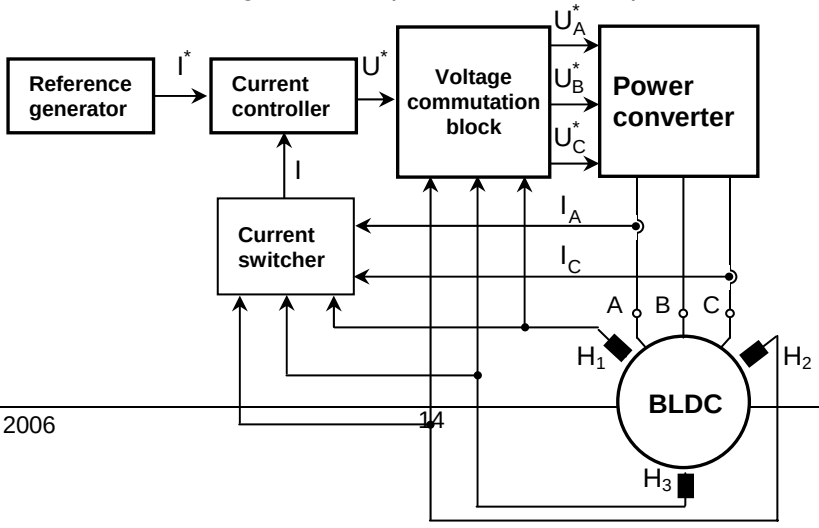


Figure 2.9. *Scheme of BLDC operating in torque/current mode*

Test mode. No control loop is closed, and the motor is controlled in open loop. A DC voltage reference is used to drive the motor. The voltage commutation block implements (by software) the computation of the phase voltages references, U_A^*, U_B^* and U_C^*, applied to the inverter. Practically, the 6 full compare PWM outputs of the DSP controller are directly driven by the program, based on these reference voltages. Sensors needed for test mode are Hall sensors.

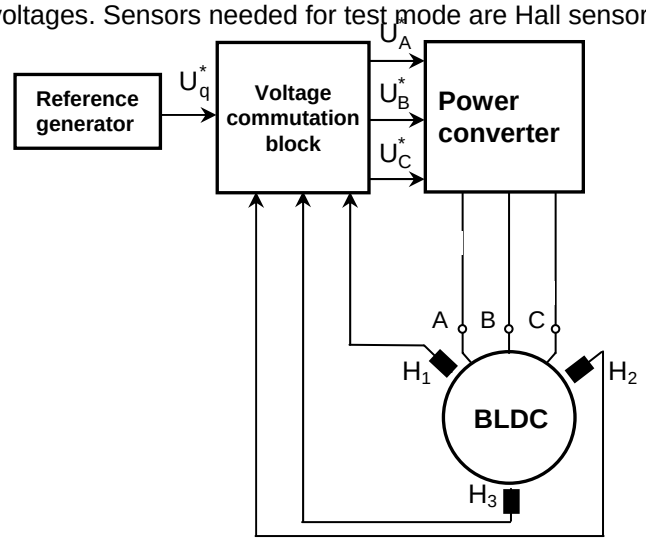


Figure 2.10. *Scheme of BLDC operating in test mode*

2.3. DC brushed motors

A brushed DC motor can operate in test, torque/current, speed or position control modes. Correspondingly, specific sensors are needed in the system configuration. Also depending on the operating mode, the output of the reference generator module will represent the reference for the controlled variable.

Position mode. The position of the motor is the controlled variable. The scheme contains three loops, a position loop and as inner loops, a speed and a current control loop. Sensors needed in this control mode are: position sensor and currents sensors; the speed is estimated from position information. The reference generator imposes the motor position.

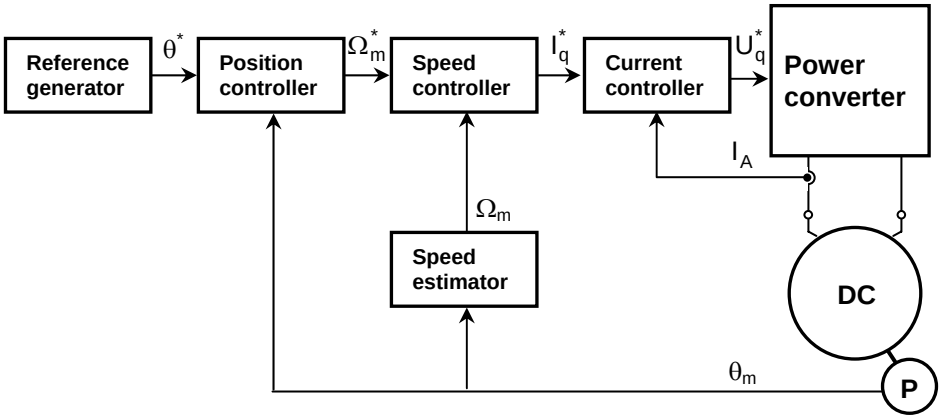


Figure 2.11. *Scheme of DC brushed motor in position mode*

Position mode without speed loop. The position of the motor is the controlled variable. The scheme contains three loops, a position loop and as inner loops, a current control loop. Sensors needed in this control mode are position sensor and currents sensors. The reference generator imposes the motor position.

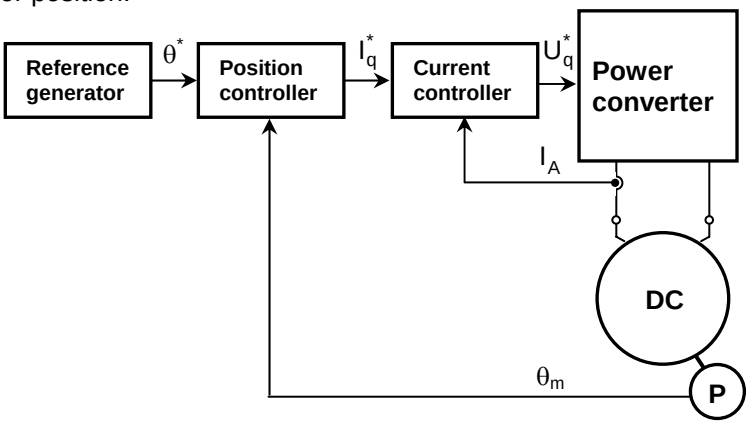


Figure 2.12. *Scheme of DC brushed motor operating in position mode without speed loop*

Speed mode. The system controls the motor speed. The scheme contains only two loops, a speed loop and, as inner loop, a current control loop. Sensors needed in this control mode are

currents sensors; the speed is estimated from position information. The reference generator imposes the motor speed.

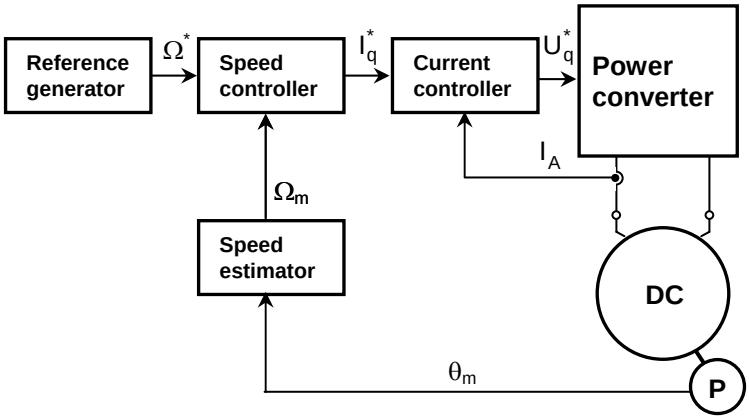


***Figure 2.13**. Scheme of DC motor operating in speed mode*

Torque/current mode. The system controls the torque/current of the motor. The scheme contains only a current control loop. For this type of control scheme, you need only currents sensors. The reference generator imposes the motor torque.

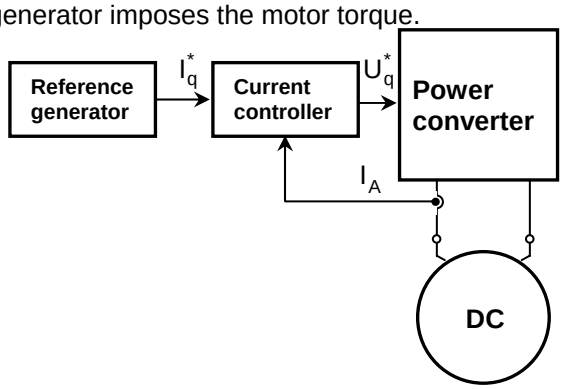


***Figure 2.14.** Scheme of DC brushed motor in torque/current mode*

Test mode. No control loop is closed, and the motor is moved in open loop. A DC voltage is used to impose motor reference. In this mode four DSP PWM outputs command power converter's transistors.

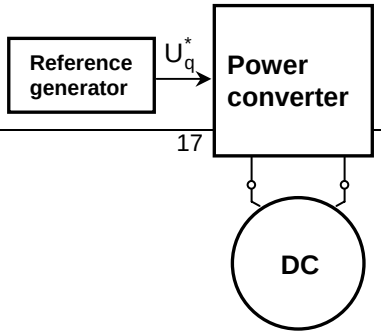


Figure 2.15. *Scheme of DC brushed motor operating in test mode*

This page is empty

3. Application setup

MotionChip II contains three functional units: the communication unit, the motion language processor unit, and the motor control unit:

- the communication unit, through which the MotionChip II can get and send motion language commands using serial (RS-232 or RS-485) or CAN-bus;
- the “motion language processor” unit decodes motion language commands received from communication unit or fetches them from a local memory;
- the motor control unit that executes the motion commands.

This manual covers aspects related with motor control unit. All the other components of the MotionChip II, which are not directly related with the motor control operations, are covered in TML Programming user manual.

Motion control applications built with MotionChip II have the following components:

- Real Time Control Kernel (RTC)
- Motor
- Power converter
- Sensors
- Controllers
- Protections

To setup the application components, the user must select appropriate values for TML register and translate hardware specifications in values understandable for the digital signal processor. The next sections present components implementation peculiarities.

3.1. Real time kernel

The operations executed by the MotionChip II to control the motor are performed in a real-time environment, i.e. all the motion control operations are activated at regular time intervals. There are two real-time operating levels for the MotionChip II:

- One related with the reference generator, speed and/or position control loops. These modules are activated at regular period time intervals called "**Slow sampling time period**".
- One related with the current / torque control and update of PWM signals commanding the power converter. These modules are activated at regular period time intervals called "**Fast sampling time period**".

The MotionChip II, through specific TML instructions, will initialize and activate the real-time kernel for motion control implementation. Once activated (after the execution of the "AXISON" TML command), the slow and fast control loops are executed, at fixed time intervals, corresponding to programmed sampling time periods

All the other functions of the MotionChip II, which are not directly related with the motor control operations, are implemented outside these real-time operating levels. This basically relates with the motion language processor unit.

I/O operations, including communication functions, are implemented in separate modules, activated by specific hardware or software-based processor interrupts. As concerns the communication unit, the drivers for the different communication interfaces are activated using an interrupt structure, basically for data send and receive commands. The processing of communication data is performed in the motion language processor unit.

In order to setup the RTC you must compute the following TML parameters:

INPUTS: **–** desired current loop sampling time ***Ts_C*** [s]

– desired speed loop sampling time ***Ts_S*** [s]

– desired PWM frequency for power stage transistors command ***PWM_freq*** [Hz]

– MotionChip II clock ***DSP_clock*** [Hz]

OUTPUTS: **–** TML parameters: **CLPER, SLPER** and **PWMPER**

RELATIONS: **CLPER=***PWM_freq*Ts_C*; if(CLPER < 2) CLPER = 2**;**
SLPER=*PWM_freq*Ts_S*; if(SLPER < 4) SLPER = 4;
PWMPER = *DSP_clock* / (*PWM_freq* * 2)

Remarks:

- *Since the previous TML parameters are integer values, you must choose such values for the three "engineering" parameters, so that the previous relations give exact integer results for the corresponding TML values. Only in these cases you'll get exactly the required values in the real operation of the processor.*
- *Since the PWM period parameter is also related to the operation of the power converter, note that changes of only one of these parameters, without considering the others, may lead to inconsistent results. As one can remark from the previous relations, the sampling periods must be in fact multiples of the PWM period.*
- *Minimum value for CLPER is 2 and for SLPER is 4*
- *The minimum recommended sampling period for the current loop is 100 microseconds.*

3.2. Motor

MotionChip II supports DC brushless, AC brushless (PMSM) and DC brush motors. Depending on motor type, the user sets the corresponding bits in the System Control Register (B14–B12 of the **SCR** register).

Table below shows for each type of motors the corresponding bits combination.

SCR register (B14 | B13 | B12 default: 000)

B14 B13 B12	Motor Type
0 0 0	BLDC
0 0 1	DC
0 1 0	PMSM
0 1 1	Reserved
1 0 0	Reserved

A BLDC motor is supplied only on two phases at one moment, the third phase is unconnected. Supplied phases sequence is decided based on the information received from Hall sensors, spaced at 120^{o} (electrical) one of the other, and changed accordingly during the motion. You need to specify the Hall sensors case for your application in order to implement a correct commutation scheme for the motor.

For permanent magnet synchronous motors (PMSM), the control system it is based on absolute position of the rotor. Depending on the position sensor type and your application requirements, you need to select the appropriate method to detect the initial absolute position of the motor. MotionChip II offers 3 methods to start the PMSM motor:

- **b/a current command; incremental sensors**. A controlled DC current is imposed to the motor, first on phase B and then on phase A. When the motor has aligned with phase A, the position measurement is initialized. The method can be used for motors having an incremental position sensor.
- **b/a voltage command; incremental sensors.** A controlled DC voltage is imposed to the motor, first on phase B and then on phase A. When the motor has aligned with phase A, the position measurement is initialized. The method can be used for motors having an incremental position sensor.
- **for PMSM with Hall and encoder sensors**, drive the motor in trapezoidal mode (speed control) until the first transition of a Hall sensor. At that point, the position measurement is initialized and the motor is automatically switched to sinusoidal mode.

The start method is selected by setting bits **B4–B2** in the Operating System Register (**OSR**). Table below show start method implemented in MotionChip II, for each method is presented bits combination.

OSR register (B4 | B3 | B2 default: 000)

B4-B2	PMSM start method
000	a / b, current command; incremental sensors
001	a / b, voltage command; incremental sensors
010	Hall / encoder (speed loop)
011	Reserved
100	Reserved

Depending on the start method for PMSM motors the user need to specify several parameters.

PMSM start with b/a current command and incremental sensors

INPUTS: – nominal motor current ***In*** [A]
– power stage maximal current ***ImaxPS*** [A]
– current scaling factor ***Kif*** [bits/A]
– the value of the current to be used during the initialization of the motor position, expressed in percentages of the nominal current ***StartCurrent*** [%]
– the value of the time interval for which the current is applied on phase A of the motor in order to align on A phase ***time_A*** [s]
– the value of the time interval for which the current is applied on phase A of the motor in order align on A phase ***time_B*** [s]

OUTPUTS: – initial current, i.e. the current value during the motor start ***initial_current*** [A]
– TML parameters: **REFTST, RINCTST, T1ONA, T1ONB**

RELATIONS: ***initial_current*** = minimum(*In*, *ImaxPS*);
REFTST = *initial_current* * *Kif* * *StartCurrent* / 100;
RINCTST = *initial_current* * *Kif* * *StartCurrent* / 100;
T1ONA = *time_A* * 65536;
T1ONB = *time_B* * 65536;

PMSM start with b/a voltage command and incremental sensors

INPUTS: – nominal motor current ***In*** [A]
– motor phase resistance ***R*** [ohm]
– measured DC-link /supply voltage ***Vdc*** [V]
– power stage dead band time ***DeadBand*** [s]
– desired PWM frequency for power stage transistors command ***PWM_freq*** [Hz]
– command voltage scaling factor ***Kuf*** [bits/V]
– the value of the current used to be used during the initialization of the motor position, expressed in percentages of the nominal current ***StartCurrent*** [%]
– the value of the time interval for which the voltage is applied on phase A of the motor in order to align on A phase ***time_A*** [s]
– the value of the time interval for which the voltage is applied on phase A of the motor in order to align on B phase ***time_B*** [s]

OUTPUTS: – initial voltage, i.e. the voltage value applied during motor start ***initial_voltage*** [V]
– TML parameters: **REFTST, RINCTST, T1ONA, T1ONB**

RELATIONS: **initial_voltage** = sqrt(2)*_In_*_R_ + 4/3*_DeadBand_*_PWM_freq_*_Vdc_);
REFTST = initial_voltage * *Kuf* * *StartCurrent*/ 100;
RINCTST = initial_voltage * *Kuf* * *StartCurrent*/ 100;
T1ONA = *time_A* * 65535;
T1ONB = *time_B* * 65535;

Start PMSM with Hall and encoder sensors

For PMSM motor start with Hall sensors and encoder there are no parameters needed to setup.

PMSM field position computation

Some parameters are related with the computation of the electric angle information during the motion (needed for coordinate transformation computations). Both motor and position sensor data-related parameters are needed to compute these parameters.

INPUTS: – position sensor resolution ***PS_resolution*** [bits]
– motor pole pairs ***pp*** [-]

OUTPUTS: – TML parameters*:* **MECRESL**, **ENC2THL, ELRES**[1]

RELATIONS: **MECRESL** = *PS_resolution*;
ENC2THL = *pp* * 2^32 / *PS_resolution*;
ELRES = *PS_resolution* / *pp*;

3.3. Power converter

PWM modes

The PWM strategy used to drive the DC and AC motors is the symmetrical PWM. The system control loops are synchronized with the PWM waveforms. Specific interrupts are generated by the PWM at the middle point of the pulse. Also motor currents measurement is performed in synchronization with the PWM, issue important if the measurement schematic uses shunts mounted in the bottom of each inverter leg.

The PWM frequency is programmable within a quite large range. The accuracy of the PWM frequency is related to the DSP clock frequency (40 MHz), thus the PWM period is modifiable with a 25 ns increment resolution. The PWM resolution depends on the programmed frequency. Table 3.1 presents the relation between them.

[1] It is used only for start method with HALL and encoder.

Table 3.1. *PWM resolution versus PWM frequency*

PWM frequency [kHz]	PWM resolution
25	1/800
20 (default)	1/1000
16	1/1250
10	1/2000
5	1/4000

At power-on or after any MotionChip II reset, all PWM outputs remain in the high-Z state until TML command `AXISON` is first executed. PWM outputs go immediately to the high-impedance state in one of the following cases:

- when $\overline{\text{PDPINT}}$ input goes low (high to low transition)
- when ENABLE input goes low (high to low transition) and as long as ENABLE remains low
- when TML command `AXISOFF` is executed;

Bits **B7**–**B6** of the **OSR** register must be set according to the type of PWM technique used in the system.

OSR register (B7 | B6 default: 00)

B7-6	PWM type
00	Standard symmetric
01	Dead time and U_{DC} compensation
10	Dead time and U_{DC} compensation,3rd harmonic injection
11	Wobbling with U_{DC} compensation,3rd harmonic injection

INPUTS: – power stage maximal current ***ImaxPS*** [A]
– motor maximal current ***ImaxM*** [A]
– current measurement method; can be on inverter lower legs or on motor supply lines ***CurrentMeasurementType*** [-]
– DC-link/supply voltage scaling factor ***Kuf_m*** [bits/V]
– desired PWM frequency for power stage transistors command ***PWM_freq*** [Hz]
– reference DC-link/supply voltage used for compensation ***VdcCompensation*** [V]

OUTPUTS: – system maximal current ***ImaxSys*** [A]
– TML parameters: **PWMPER**, **VDCN**, **SATPWM**

RELATIONS: ***ImaxSys*** = minimum(*ImaxM*, *ImaxPS*)
PWMPER = *DSP_clock* / (*PWM_freq* * 2)
VDCN = *VdcCompensation* * *Kuf_m*
If (*CurrentMeasurementType* == inverter legs) // current sensors on inverter legs
    SATPWM = 65535 * 2e-6 * *PWM_freq*
else // current sensors on motor phases
    SATPWM = 0;

Dead time compensation

In order to prevent the short-circuit between the upper and lower transistors of each inverter leg during transistors commutation, a dead time must be introduced from the switching OFF of one transistor and the switching ON of the complementary one. The MotionChip II provides a programmable dead-band for each PWM output pair (1-2, 3-4, 5-6) from 0 to 12μs. The default value is 0μs. Hence you must set the TML parameter **DBT** with a nonzero value or use external dead-band logic in order to drive the power stage of a three-phase motor or that of a DC brush motor i.e. the configurations where each PWM output drives one switching device. Table 3.2. presents the correspondence between DBT parameter values and dead-band values.

Table 3.2. *Dead-band values*

Dead-band [μs]	DBT value
0.25	2784
0.5	2788
0.75	4064
1	2792
1.3	3556
1.5	4072
1.75	2540
2.0	2796
2.2	3052
2.6	3564
2.8	3820
3.0	4076
3.2	2288
4.0	2800
4.4	3056
4.8	3312
5.2	3568
6	4080
8	2808
12	4088

INPUTS: – desired dead band time ***DeadBand*** [s]

OUTPUTS: – TML parameter: **DBT**

RELATIONS: – see **Table 3.2**

Third harmonic injection

In order to be able to obtain an increase of the output voltage supplied by the power converter feeding three-phase motors, a third harmonic component can be injected in the output reference voltage. This can lead to an equivalent increase of the output voltage of up to 15%. No specific TML parameters are associated to this feature; only the specific PWM mode using the third harmonic injection must be set in the configuration.

U_{DC} compensation

If the DC link voltage varies during the operation of the drive, an alteration of the system performances can be produced. Depending on the dynamic of the control loops, usually such variations can be compensated by the closed current control loop. If these voltage variations are important, and the control can't compensate them, you can also activate the U_{DC} compensation feature. In this case, the PWM unit will correct the reference voltage applied to the motor, by a factor depending on the measured voltage in the DC link of the power converter. Obviously, a DC-link voltage measurement sensor must be installed in the system, and the corresponding configuration bit must be set in the OSR register.

Bit **B5** of the **OSR** register must be set according to this option.

OSR register (B5 default: 0)

B5	U_{DC} compensation
0	No
1	Yes

INPUTS: – DC-link/supply voltage scaling factor ***Kuf_m*** [bits/V]
– requested voltage level of DC-link/supply, used for compensation ***VdcCompensation*** [V]

OUTPUTS: – TML parameters: **VDCN**

RELATIONS: **VDCN** = *VdcCompensation * Kuf_m*

Ud,q saturation mode

During the computation of command voltages for vector control structures, specific methods may be used to treat the saturation of the direct and quadrature command voltage components.

- **Independent uq, ud**: The direct and quadrature voltage command components are saturated independently one to each other, at the limit voltage that can be generated (in TML, at value 32767 - SATPWM). Note that if saturation of the voltage occurs on any of the components, the voltage vector generated by the system is not oriented in the same direction as the reference voltage.
- **uq = f(ud)**: The quadrature voltage command component is saturated in relation to the direct voltage command value. This approach keeps the same orientation of the applied voltage as the reference one, and proportionally reduces both d and q voltage reference components, in order to obtain a maximum resultant voltage.

Bit **B8** of the **OSR** register must be set according to saturation method desired.

OSR register (B8 default: 0)

B8	Ud,q saturation
0	Independent uq, ud
1	uq = f(ud)

INPUTS: **–** current measurement method. Current sensors can be on inverter lower legs or on motor supply phases. ***CurrentMeasurementType*** [-]
– desired PWM frequency for power stage transistors command ***PWM_freq*** [Hz]

OUTPUTS: – TML parameters: **SATPWM**

RELATIONS: If (*CurrentMeasurementType* == inverter legs) // current sensors on inverter legs
SATPWM = 65535 * 2e-6 * *PWM_freq*;
else // current sensors on motor phases
SATPWM = 0;

Brake

The MotionChip II can monitor the DC-bus voltage level if a voltage feedback is provided on the analogue input VDC. In some cases, during motor brakes, the DC-bus voltage may grow to dangerous values due to the current injected by the motor into the DC-link. One solution to reduce this over-voltage is to connect temporary a resistor across the DC-link through a switching device (see Figure 3.1).

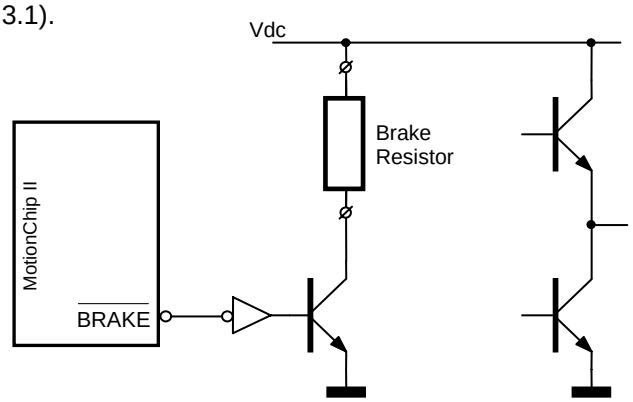


Figure 3.1. *Brake pin usage*

The MotionChip II has an active-low PWM output called $\overline{\text{BRAKE}}$ dedicated to command the switching device from a brake circuit. When this function is enabled, the $\overline{\text{BRAKE}}$ output is automatically activated with a 50% duty cycle, if the DC-bus voltage bypasses a programmed voltage trigger value. The $\overline{\text{BRAKE}}$ output remains active until the DC-bus voltage drops under this trigger value. The activation of the BRAKE command can be signaled by a TML interrupt if the over-voltage protection limit is set at the same value as the BRAKE trigger level.

The user can activate or deactivate the brake command. Bit **B9** of the **OSR** register must be set according to this option.

OSR register (B9 default: 0)

B9	Brake command
0	Inactive
1	Active

INPUTS: – DC-link/supply voltage scaling factor ***Kuf_m*** [bits/V]
– requested trigger voltage level ***Vbrake*** [V]

OUTPUTS: – TML parameters: **BRAKELIM**

RELATIONS: **BRAKELIM** = *Vbrake * Kuf_m*;

3.4. Sensors

3.4.1. Current sensors

In case of three-phase motors, the MotionChip II uses for torque control the motor current in phase A, read on input I_A and the motor current in phase C read on input I_C. In case of DC brush motors, the MotionChip II reads the motor current on I_A input. The current sensors may be placed on the motor lines or on the lower legs of the inverter.

By default, the values measured on I_A and I_C inputs are interpreted with an offset of half A/D input range. Hence, values higher than mid-point are interpreted as positive currents, while values lower than mid-point are interpreted as negative values. Due to current measurement scheme imperfections, sometimes the A/D mid-points do not correspond exactly to a zero current. For these situations, the MotionChip II can detect automatically the true offset, for each of the A/D inputs. This offset will be considered for current feedback interpretation. Note that these offsets will be stored in the TML parameters **AD0OFF** and **AD1OFF**. You can also set these parameters manually, if their values are known for your measurement scheme. Figure 3.2 shows how to measure the currents for DC brush or three-phase motors using shunts on the lower legs of the inverter. Current feedback signal polarity relative to motor current is emphasized.

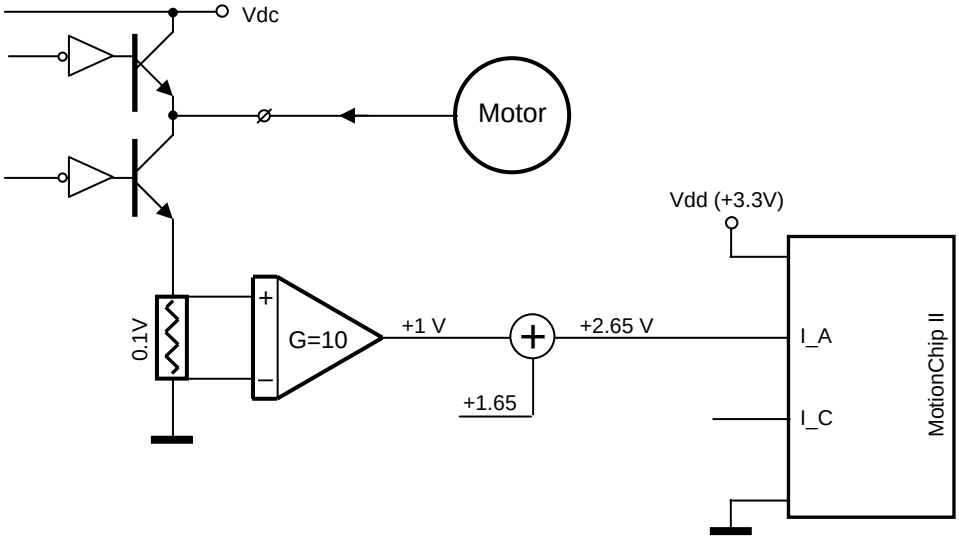


Figure 3.2. *Current measurement from lower-legs of a 2 or 3-phase power converter*

Bits **B1–B0** of the **OSR** register must be set according to desired offset detection method.

OSR register (B1 | B0 default: 00)

B1-0	Current Offset Detection
00	No detect
01	Detect without PWM
10	Detect with PWM

INPUTS: – current gain factor ***CGa*** [V/A]
– current measurement shift ***SFTCRT*** [bits]

OUTPUTS: – current scaling factor ***Kif*** [bits/A]
– TML parameters: **SFTCRT**

RELATIONS: ***Kif*** = *CGa* * 32767 / (1.65 * 2^SFTCRT);
SFTCRT = *SFTCRT*;

3.4.2. Speed sensors

The speed is estimated as the variation of the position over one sampling period of the speed / position control loop. This method can be implemented for any motion structure that has a position sensor defined.

Be aware that this method has a resolution depending on the number of equivalent lines per rotation, of the position sensor, and the value of the slow sampling period of the MotionChip II control module. Figure 3.3 presents the principle of the method.

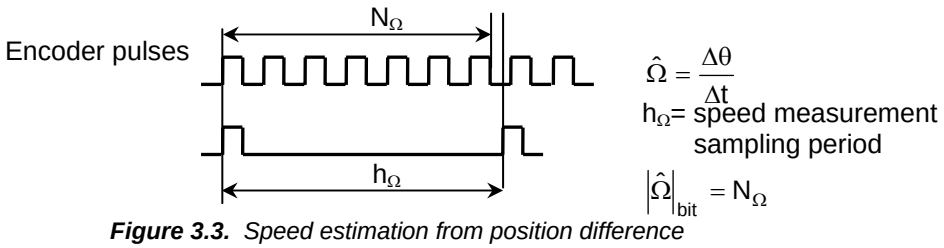


Figure 3.3. *Speed estimation from position difference*

Speed estimate accuracy resolution: $[\text{accuracy}] = \log_2(N_\Omega)_{\Omega max}$.

Example: $\Omega_{max} = 3000\,\text{rpm}$

$h_\Omega = 1\,\text{ms}$

$N_{encoder} = 1000$ lines/rot

$N_{V\,max} = (4xN_{encoder})xh_\Omega x\Omega_{max} x1/60 \Rightarrow [\text{accuracy}] < 8$ bits

Note also that the minimum speed that can be obtained using this method is the equivalent of one pulse per sampling period, giving an estimation of value 1 (bit).
If the speed is computed as a position difference, the speed scaling factor relation depends on the mechanical resolution of the position sensor (***PS_resolution***) and on the speed/position control loop sampling period (***Ts_S***).
To specify speed sensor's presence in the motion application set the corresponding bits (**B5-B3**) of the **SCR** register.

SCR register (B5 | B4 | B3 default: 000)

B5-3	**Speed sensor**
000	Position difference
001	Reserved
010	Reserved
011	Reserved
100	Reserved
110	Reserved
101	Reserved
111	None

INPUTS: – position sensor resolution, equivalent lines per rotation ***PS_resolution*** [bits]
– requested speed loop sampling period ***Ts_S*** [s]

OUTPUTS: – speed scaling factor ***Kvf*** [bits/(rad/s)]

RELATIONS: ***Kvf*** = *PS_resolution* * *Ts_S* / (2* PI)

3.4.3. Position sensors

Quadrature encoder pulse

The MotionChip II has an integrated interface for quadrature encoder pulse (QEP). This makes the interfacing with these sensors straightforward. Figure 3.4 presents the connection of a quadrature encoder to the MotionChip II. Both quadrature signals A and B must be connected to the MotionChip II for proper operation of the measurement interface. Note that for an encoder with N_{lines} per rotation, a PS_resolution = $4xN_{lines}$ number of counts is obtained from the QEP interface. The index signal Z can be optionally connected to the ENC1Z capture input of the MotionChip II, and used for zero position-offset detection.

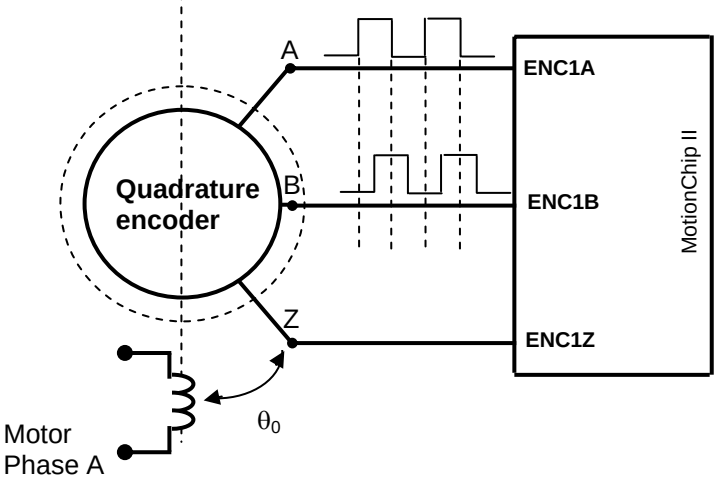


Figure 3.4. *Interfacing a quadrature encoder with the MotionChip II*

Note: A and B signals must be correlated with the succession of A, B and C phases of three-phase motors. Specific tests must be performed in order to validate this correlation. The measured position information in the MotionChip II must increase when rotating the motor in the sense from A phase to B phase. (This can be obtained for example with a test voltage having a positive phase increment TINCTST). If the position decreases as the motor rotates with a positive phase increment, interchange A and B signals connection to the MotionChip II **OR** any two of motor phases connection to the power inverter. **IF THE CORRECT CORRELATION BETWEEN THE ENCODER SIGNALS AND THE MOTOR PHASES SUCCESSION IS NOT ASSURED, THE MOTOR CANNOT BE CONTROLLED IN ANY CONFIGURATION USING THE MEASURED POSITION INFORMATION!**

INPUTS: – number of encoder lines per one mechanical revolution ***No_encoder_lines*** [bits]
– motor pole pairs ***pp*** [-]

OUTPUTS: – position scaling factor ***Kpf*** [bits]
– position sensor resolution - equivalent lines per rotation ***PS_resolution*** [bits]
–TML parameters: **MECRESL**, **ENC2THL, ELRES**

RELATIONS: ***PS_resolution*** = 4 * *No_encoder_lines*;
Kpf = *PS_resolution* / (2 *PI);
MECRESL = *PS_resolution*;
ENC2THL = *pp* * 2^32 / *PS_resolution*;
ELRES = *PS_resolution* / *pp*;

Linear Hall

The MotionChip II can use as position feedback the signals of 3 linear Hall sensors, read on inputs LH1, LH2 and LH3/MTEMP. The Hall signals must have the same amplitude and be phased at 120 electrical degrees.

The MotionChip II interprets the linear Hall signals using offsets and gains determined experimentally. Due to this approach, the MotionChip II is independent of the measurement scheme. Also, the imperfections of the measurement scheme and/or of the Hall sensors are eliminated; sometimes the signals' amplitudes aren't equal. The gains and the offsets are determined using a specific test. The test detects, for each of the A/D inputs the true offset and the gain factor, besides that it estimates the zero position offset versus motor phase A. At the end of the test, the offsets and the gains are saved in the EEPROM, in the memory area **x**FB0 to **x**FB8 (**x** depends on the EEPROM size, x=3 for 64k, x=5 for 128k and x=7 for 256k). Hence, the test execution is required only once at run-time the values of the offset and gains are read from the EEPROM and copied in the TML parameters **KSLIP**, **SFTCWEAK**, **AD3OFF**, **AD6OFF**, **SRECTCOMP** and **SIN2REC**. If several applications use the same type of linear Hall sensors, the parameters from the EEPROM can be included in the software file, thus reducing the programming time by eliminating the test execution.

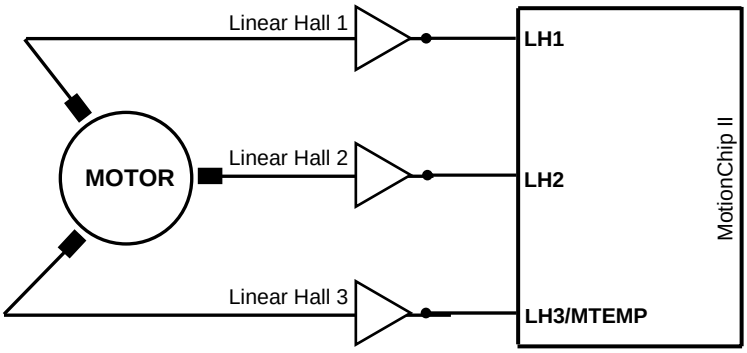


Figure 3.5. *Linear Hall sensors connections to MotionChip II*

Remarks: *In configurations with position feedback from linear Hall sensors, the motor temperature monitoring is not available, since the AD input for monitoring the motor temperature is used to read the third linear Hall signal.*

INPUTS: – number of equivalent encoder lines per one mechanical revolution ***No_encoder_lines*** [bits] 2^X * pp, recommended value X = 9
– motor pole pairs ***pp*** [-]

OUTPUTS: – position scaling factor ***Kpf*** [bits]

– position sensor resolution - equivalent lines per rotation ***PS_resolution*** [bits]
–TML parameters: **MECRESL, DBTTH**

RELATIONS: ***PS_resolution*** = 4 * *No_encoder_lines*;
Kpf = *PS_resolution* / (2 *PI);
MECRESL = *PS_resolution*;
DBTTH = *16- log_2(PS_resolution/pp)*

To specify the position sensor's presence in the motion application set the corresponding bits (**B2-B0**) of the **SCR** register.

SCR register (B2 | B1 | B0 default: 000)

B2-0	Position sensor
000	Encoder
100	Linear Hall
001	Reserved
010	Reserved
011	Reserved
110	Reserved
101	Reserved
111	None

3.4.4. Hall sensors

In order to drive correctly the BLDC motor you must know the correlation between Hall sensors distribution and motor phases. There are 12 possible combinations in which the sensors can be distributed. According to the combination actually implemented there are two TML parameters needed to be set, **HALLCASE** and **HALLDIR.**

The algorithm used to find the Hall sensor distribution case for a certain motor is the following:

- The motor phases are supplied with three DC voltages: ***u_a = U***, ***u_c = u_b = - U/2*** and the Hall sensor output signals (*H1, H2, H3*) are read. In fact, in this step the motor rotor is positioned in front of phase *A*.
- The motor phases are supplied with three voltages: ***u_b = U, u_c = u_a = - U /2*** and the Hall sensor output signals (*H1, H2, H3*) are read. In fact, in this step the motor rotor is positioned in front of phase *B*.

Using the Table 3.3, the corresponding values of parameters HALLCASE and HALLDIR can be selected, based on the read combinations of signals H1, H2, H3.

Table 3.3. *HALLCASE and HALLDIR detection for BLDC motor*

HALLCASE	HALLDIR	Rotor aligned with phase B (H3, H2, H1)		Rotor aligned with phase A (H3, H2, H1)	
2	0x35	(0, 1, 0)	2	(1, 0, 0)	4
4	0x63	(0, 0, 1)	1	(1, 0, 0)	4
6	0x35	(1, 0, 0)	4	(0, 1, 0)	2

8	0x56	(1, 0, 0)	4	(0, 0, 1)	1
10	0x63	(0, 0, 1)	1	(0, 1, 0)	2
12	0x56	(0, 1, 0)	2	(0, 0, 1)	2
3	0x42	(1, 0, 1)	5	(0, 1, 1)	3
5	0x14	(1, 1, 0)	6	(0, 1, 1)	3
7	0x42	(0, 1, 1)	3	(1, 0, 1)	5
9	0x21	(0, 1, 1)	3	(1, 1, 0)	6
11	0x14	(1, 1, 0)	6	(1, 0, 1)	5
13	0x21	(1, 0, 1)	5	(1, 1, 0)	6

3.4.5. Temperature sensors

Up to two temperature sensors (from motor and/or power converter temperature sensors) can be connected to the MotionChip II analog input pins MTEMP1 and DTEMP. These analogue inputs may range in the 0 to +3.3 V and can be monitored by software during the motion. The MTEMP and DTEMP inputs can have only positive values. Hence, the values read are interpreted without any offset.

Remark: *In configurations that use linear Hall sensors, the analogue input for motor temperature sensor is used to read the third linear Hall sensor. Hence, in such configurations the motor temperature monitoring is not available.*

The user specifies if a temperature sensor is used and its value is read through the corresponding A/D channel by setting bits **B8** and **B7** of the **SCR** register.

SCR register (B8 default: 0)

B8	**Temperature 2**
0	No
1	Yes

SCR register (B7 default: 0)

B7	**Temperature 1**
0	No
1	Yes

INPUTS: – drive temperature sensor gain factor TempSensorGain [V/°C]
– drive temperature sensor output at 0 °C TempSensorOutputAtoC [V]
– drive over temperature trigger level Temp_2_Max [°C]
– motor temperature sensor type MotorTempSensorType [-]

OUTPUTS: – temperature sensor offset **TempOffset** [bits]
– TML parameters: **T1MAXPROT**, **T2MAXPROT**, **MTSTYPE**

REALTIONS: **if (MotorTempSensorType == NTC)**

MTSTYPE = 0 //sensor type NTC
else
MTSTYPE = 1 //sensor type PTC
T1MAXPROT = 32767; //for NTC/PTC sensors type
TempOffset = 65472 /3.3* TempSensorOutputAtoC;
T2MAXPROT = *Temp_2_Max* * TempSensorGain* 65472 /3.3+TempOffset;

3.4.6. DC Voltage sensor measurement

The MotionChip II has a dedicated input, VDC**,** for DC-bus (DC-link) voltage feedback. When this feedback is provided, the following features of the MotionChip II can be enabled:

- Compensation of the DC-bus voltage variation;
- Brake control;
- Over-voltage and under-voltage protections.

The DC-link voltage variable is considered as unsigned measurement, i.e. measured 0 V correspond to 0 value of the variable, while measured 3.3V correspond to the maximum measurable DC-link voltage, ***VdcMaxMeasurable***. A scaling factor (***Kuf_m***) is used to scale the measured value, expressed in V, to a value expressed in bits.

INPUTS: – maximal measurable DC-link supply voltage ***VdcMaxMeasurable*** [V]

OUTPUTS: – DC-link /supply voltage scaling factor ***Kuf_m*** [bits/V]

RELATIONS: ***Kuf_m*** = 65472 / *VdcMaxMeasurable*;

3.5. Controllers

The MotionChip II can operate, depending on the configuration of the motion system, with up to three controllers (current, speed and position). For all cases, the reference generator block of the motion system imposes the reference for the outer control loop, giving also the type of the configuration (position, speed or torque control). The inner controllers can be present or not, depending on availability of needed sensors and on system configuration. Figure 3.6 shows a complete control scheme implemented in MotionChip II.

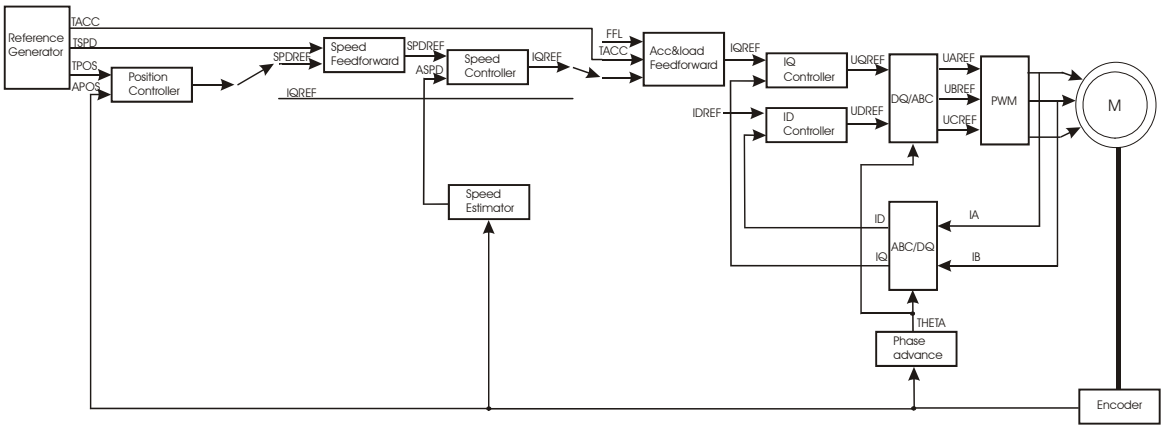


Figure 3.6. Complete control scheme

In PMSM applications the fast control loop requires two identical 16-bit digital PI controllers to adjust currents on D and Q axes. For DC motor applications (Brushless DC, Brush DC) it is used only the IQ controller.

The speed control loop contains a 16-bit digital PI controller. For position control a 16-bit digital PID controller is used. By canceling the derivative component it can be used as a 16-bit PI controller.

In order to improve dynamic system behavior MotionChip II implements also two feedforward blocks, a speed feedforward block and an acceleration & load feedforward block.

The information related to the control loops used in a motion configuration is contained in the Motion Command Register (bits B10-B8). This information is automatically updated during the motion, by the MotionChip II TML processing unit, at the execution of update commands (as UPD instruction).

In MotionChip II the controllers' coefficients are represented using two numbers. One is a Q15 fractional number containing the value of the coefficient scaled in the [-1, +1] range. The other is a shift factor, having a value that represents the power of 2, which multiplies the scaled part in order to give the initial value of the coefficient.

For example, if K_real = 2.4, then K_scaled = 2.4 / 4 = 0.6, and K_shift = 2.

Next paragraphs present general considerations regarding the model of the current, speed and position controllers used in MotionChip II.

3.5.1. Current controllers

The current through the circuit is controlled using a discrete PI controller having the following transfer function:

$$R(z) = Kp + \frac{Ki}{1 - z^{-1}} \cdot Ts\_C$$

where: Kp is the proportional factor, Ki is the integral factor, *Ts_C* is the current loop sampling period.

Coefficients Kp and Ki are usually obtained by tuning the closed-loop system (see Figure 3.7) Usually this is done based on the desired dynamic performances imposed for the closed loop structure.

Remark: *in the following paragraphs the current controller's coefficients are referred with the following notations Ki_C_real and Kp_C_real. The Ki_C_real coefficient includes the sampling period Ts_S.*

$$Kp\_C\_real = Kp \quad Ki\_C\_real = Ki \times Ts\_C$$

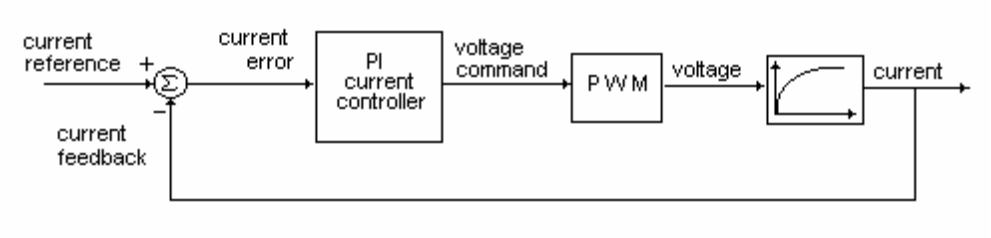


Figure 3.7. *Structure of current control loop*

MotionChip II implementation of Q-axis current controller is presented in Figure 3.8. Figure 3.9 presents D-axis current controller implementation.

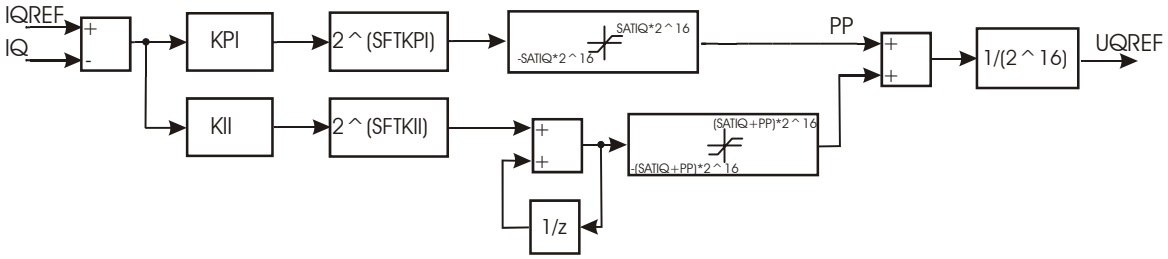


Figure 3.8. *Q-axis current controller implementation*

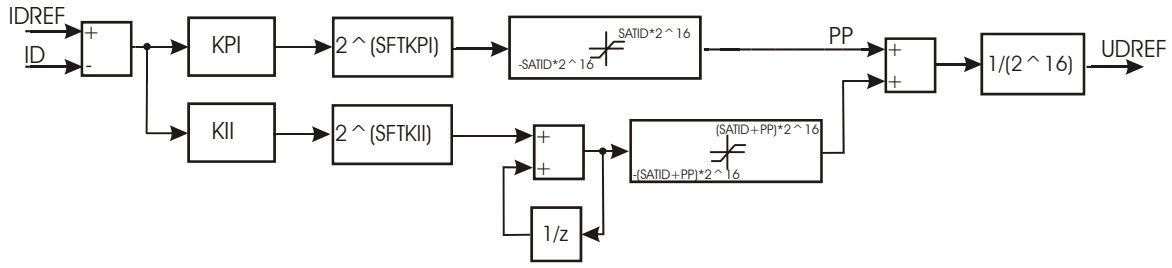


Figure 3.9. *D-axis current controller implementation*

Controller inputs are the reference current (IQREF/IDREF), given by the acceleration & load feedforward block and the feedback current (IQ/ID) given by the current sensors. The outputs of current controllers are numerical values of the reference voltages (UQREF/UDREF).

Current controller parameters:

- numerical coefficients representing the proportional part (KPI, SFTKPI);
- numerical coefficients representing the integral part (KII, SFTKII) ;
- saturation coefficient (SATIQ/SATID);

Relationships are therefore required between the numerically expressed parameters (KPI, SFTKPI etc.) and the coefficients resulting from the controller tuning (Kp and Ki). These relationships must take into account the input scaling factors (current scaling factor) and the output ones (voltage scale factor).

Here are the computational relations of the link factors. The computational relations for current control parameters use *Kp_crt* and *Ki_crt* as inputs, given by:

$$\begin{cases} Kp\_crt = Kp\_C\_real \times \dfrac{Kuf}{Kif} \\ Ki\_crt = Ki\_C\_real \times \dfrac{Kuf}{Kif} \end{cases}$$

where *Kuf* – command voltage scaling factor (see par. 3.7 for details).

Kif – current scaling factor (see par. 3.7 for details).

Current controller setup

INPUTS:
– current scaling factor ***Kif*** [bits/A]
– command voltage scaling factor ***Kuf*** [bits/V]
– PWM frequency ***PWM_freq*** [Hz]
– current controller integral factor ***Ki_C_real***
– current controller proportional factor ***Kp_C_real***
– current measurement method, can be on inverter lower legs or on motor supply lines ***CurrentMeasurementType*** [-]

OUTPUTS:
– scaled current controller integral factor ***Ki_C_scl***
– scaled current controller proportional factor ***Kp_C_scl***
– TML parameters: **KPI**, **SFTKPI**, **KII**, **SFTKII**, **SATPWM**, **SATID**, **SATIQ**

RELATIONS: ***Ki_C_scl*** = *Ki_C_real * Kuf / Kif*;
Kp_C_scl = *Kp_C_real * Kuf / Kif*;
SFTKPI = smallest power of 2 for which *Kp_C_scl* < 2^SFTKPI
SFTKII = smallest power of 2 for which *Ki_C_scl* < 2^SFTKII
KPI = 32767 * *Kp_C_scl* / 2^SFTKPI;
KII = 32767 * *Ki_C_scl* / 2^SFTKII;
if (*CurrentMeasurementType* == inverter legs) // current sensors on inverter legs
SATPWM = 65535 * 2e-6 * *PWM_freq*
else // current sensors on motor phases
SATPWM = 0;
SATID = SATPWM;
SATIQ = SATPWM;

3.5.2. Speed controller

In applications using a cascade control structure with a closed speed loop generating output reference for the current/torque controller, the mechanical system can be controlled using a PI controller whose transfer function is:

$$R(z) = Kp + \frac{Ki}{1 - z^{-1}} \cdot Ts\_S$$

where: Kp represents the proportional factor, Ki is the integral factor, *Ts_S* is the speed loop sampling time.

Coefficients Kp and Ki are usually obtained by tuning the closed-loop system (see Figure 3.10). Usually this is done based on the desired dynamic performances imposed for the closed loop structure.

Remark: *in the following paragraphs the controller's coefficient are referred with the following notations Kp_S_real and Ki_S_real. The Ki_S_real coefficient includes the sampling period Ts_S.*

$$Kp\_S\_real = Kp \quad Ki\_S\_real = Ki \times Ts\_S$$

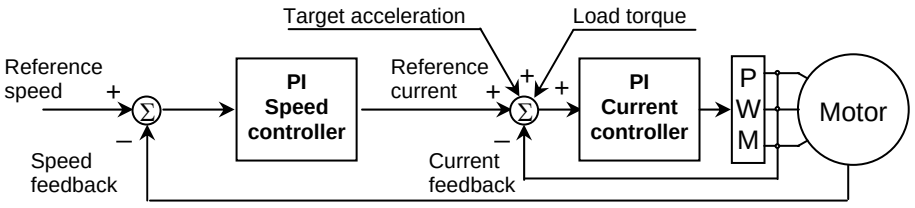


Figure 3.10. *Structure of speed and current control loops*

MotionChip II implementation of the speed controller is presented in Figure 3.11.

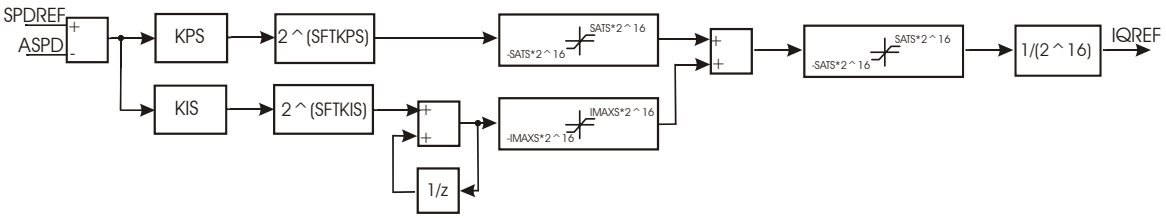


Figure 3.11. *MotionChip II speed controller implementation scheme*

As shown in Figure 3.11, the inputs in the speed controller are the reference speed (SPDREF), given by the speed feedforward block and the feedback speed (ASPD), given by the speed sensor. The controller output is the numerical value of the reference current (IQREF).

Speed controller parameters:

- numerical coefficients representing the proportional part (KPS, SFTKPS);
- numerical coefficients representing the integral part (KIS, SFTKIS);
- saturation coefficient for integral part (IMAXS);
- saturation coefficient (SATS).

Relationships are therefore required between the numerically expressed parameters (KPS, SFTKPS etc.) and the coefficients resulting from the controller tuning (Kp, Ki**,** etc). These relationships must take into account the input scaling factors (speed scaling factor) and the output ones (current scale factor).

Here are the computational relations of the link factors. The computational relations for current control parameters use *Kp_spd* and *Ki_spd* as inputs, given by:

$$\begin{cases} Kp\_spd = Kp\_S\_real \cdot \dfrac{Kvf}{Kif} \\ Ki\_spd = Ki\_S\_real \cdot \dfrac{Kvf}{Kif} \end{cases}$$

where Kvf is the speed scaling factor, and *Kif* is the current scaling factor (for details see par. 3.7).

Speed controller setup

INPUTS: – current scaling factor ***Kif*** [bits/A]
– speed scaling factor **Kvf** [bits/rad/s]
– system maximal current ***ImaxSys*** [A]
– real speed controller integral factor ***Ki_S_real*** [-]
– real speed controller proportional factor ***Kp_S_real*** [-]
IPSatS (integral part saturation) [%]

OUTPUTS: – scaled speed controller integral factor ***Ki_S_scl*** [-]
– scaled speed controller proportional factor ***Kp_S_scl*** [-]
– maximal output current ***I_Max*** [A]
–TML Parameters: **KPS**, **SFTKPS**, **KIS**, **SFTKIS**, **SATS**, **IMAXS**,

RELATIONS ***Ki_S_scl*** = *Ki_S_real* * *Kif* / Kvf;
Kp_S_scl = *Kp_S_real* * *Kif* / Kvf;
I_Max = *ImaxSys*;
SFTKPS = smallest positive power of 2 for which *Kp_S_scl* < 2^SFTKPS
SFTKIS = smallest positive power of 2 for which *Ki_S_scl* < 2^SFTKIS
KPS = 32767 * *Kp_C_scl* / 2^SFTKPS;
KIS =32767 * *Ki_C_scl* / 2^SFTKIS;
SATS = 32767 - *I_Max* * *Kif*;
IMAXS = (1 - IPSatS / 100) * 32767;

3.5.3. Position controller

In applications using a cascade control structure with a closed position loop generating output reference for the speed controller, the mechanical system can be controlled using a PI controller. For configuration without a speed controller as inner loop, a PID configuration is required. The general form of the position controller transfer function is:

$$R(z) = Kp + \frac{Ki}{1 - z^{-1}} \cdot Ts\_S + \frac{K_d}{Ts\_S} \cdot (1 - z^{-1}) \cdot (1 + z^{-1} \cdot Kdf)$$

where : Kp represents the proportional factor, Ki is the integral factor, Kd is the derivative factor, Kdf a filtering term of the derivative factor, *Ts_S* is the position loop sampling time (equal to the speed sampling time for the MotionChip II).

Coefficients Kp, Ki and Kd are usually obtained by tuning the closed-loop system (see Figure 3.12). Usually this is done based on the desired dynamic performances imposed for the closed loop structure.

Remark: *in the following paragraphs the controller's coefficient are referred with the following notations Kp_P_real, Ki_P_real and Kd_P_real. The Ki_P_real and Kd_P_real coefficients include the sampling period Ts_S.*

$$Kp\_P\_real = Kp\ ;\quad Ki\_P\_real = Ki \times Ts\_S\ ;\quad Kd\_P\_real = Kd \times Ts\_S$$

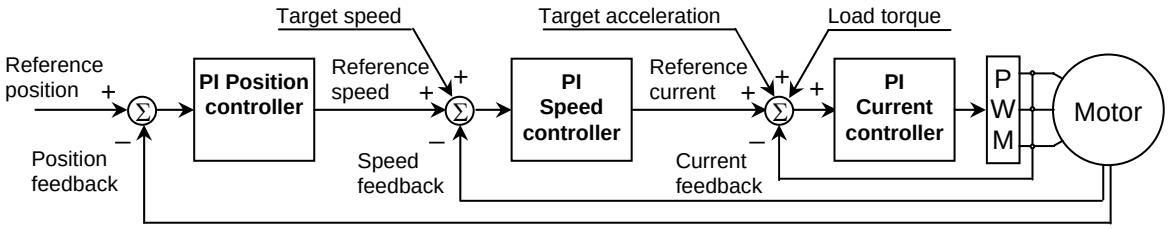


Figure 3.12. *Structure of position control with feedforward*

In applications using a position control structure without an inner speed control loop, the control uses a discrete PID controller generating as output a reference current (Figure 3.13).

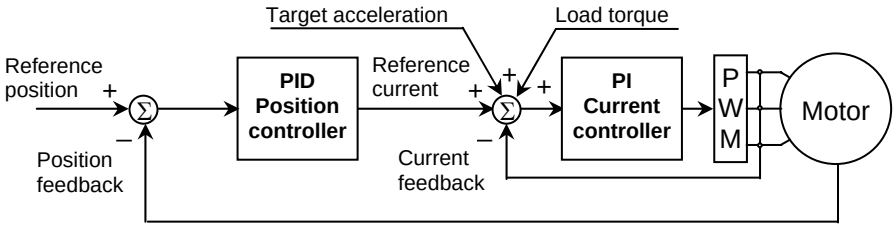


Figure 3.13. *Structure of position control loop without inner speed control loop*

MotionChip II implementation of the position controller is presented in Figure 3.14.

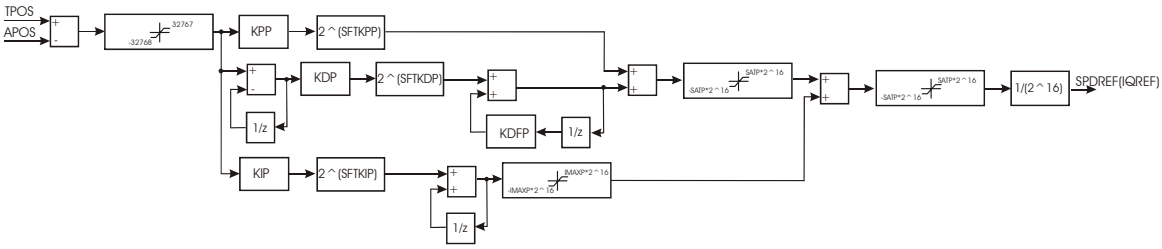


Figure 3.14. *MotionChip II position controller implementation scheme*

As shown in Figure 3.14, the position controller inputs are the target position (TPOS), given by the reference generator and the feedback position (APOS), given by the position sensor. The output of the controller is the numerical value of the reference speed (SPDREF), or current reference (IQREF), when the speed loop is disabled.

Position controller parameters:

- numerical coefficients representing the proportional part (KPP, SFTKPP);
- numerical coefficients representing the integral part (KIP, SFTKIP);
- saturation coefficient for integral part (IMAXP);
- numerical coefficients representing the derivative part (KDP, SFTKDP);
- derivative filter coefficient (KDFP);
- saturation coefficient (SATP).

Relationships are therefore required between the numerically expressed parameters (KPP, SFTKPP, etc.) and the coefficients resulting from the controller tuning (Kp, Ki**,** etc). These relationships must take into account the input scaling factors (position scaling factor) and the output ones (speed scale factor).

Here are the computational relations of the link factors. The computational relations for position control parameters use *Kp_pos*, *Ki_pos* and *Kd_pos* as inputs, given by:

$$\begin{cases} Kp\_pos = Kp\_P\_real \cdot \dfrac{Kpf}{Kvf} \\ Ki\_pos = Ki\_P\_real \cdot \dfrac{Kpf}{Kvf} \\ Kd\_pos = Kd\_P\_real \cdot \dfrac{Kpf}{Kvf} \end{cases}$$

where *Kpf* is the position scaling factor, and Kvf is the speed scaling factor (see par. 3.7 for details). Similar relations are used if the output of the position controller is the current command (no speed controller as inner loop replacing the speed scaling factor with the current scaling factor *Kif*).

Position controller setup with inner speed controller loop (speed output command)

INPUTS:
– DC-link/supply voltage ***Vdc*** [V]
– motor torque constant ***K*** [Nm/A]
– speed scaling factor ***Kvf*** [bits/rad/s]
– position scaling factor ***Kpf*** [bits/rad]
– real position controller integral factor ***Ki_P_real***
– real position controller proportional factor ***Kp_P_real***
– requested integral part saturation **IPSatP** [%]

OUTPUTS:
– scaled position controller integral factor ***Ki_P_scl***
– scaled position controller proportional factor ***Kp_P_scl***
– maximal output speed ***N_Max*** [rad/s]
– TML parameters: **KIP**, **SFTKIP**, **KPP**, **SFTKPP**, **SATP**, **IMAXP**

RELATIONS: ***Ki_P_scl*** =*Ki_P_real* * Kvf / *Kpf*;
Kp_P_scl =*Kp_P_real* * Kvf / *Kpf*;
N_Max =*Vdc/K*;

SFTKPP = smallest positive power of 2 for which *Kp_P_scl* < 2^SFTKPP
SFTKIP = smallest positive power of 2 for which *Ki_P_scl* < 2^SFTKIP
KIP = 32767 * *Ki_P_scl* / 2^SFTKIP;
KPP = 32767 * *Kp_P_scl* / 2^SFTKPP;
SATP = 32767 - *N_Max* * Kvf;
IMAXP = (1 - IPSatP / 100) * 32767;

Position controller setup without inner speed controller loop (current output command)

INPUTS:
– system maximum current ***ImaxSys*** [A]
– current sensor scaling factor ***Kif*** [bits/A]
– position sensor scaling factor ***Kpf*** [bits/rad]
– real position controller integral factor ***Ki_P_real*** [-]
– real position controller proportional factor ***Kp_P_real*** [-]
– real position controller derivative factor ***Kd_P_real*** [-]
– requested integral part saturation **IPSatP** [%]
– derivative filter factor **filter_D** [-]

OUTPUTS: – scaled position controller integral factor ***Ki_P_scl***
– scaled position controller proportional factor ***Kp_P_scl***
– scaled position controller derivative factor ***Kd_P_Scl***
– maximal output current ***I_Max*** [A]
– TML parameters: **KIP**, **SFTKIP**, **KPP**, **SFTKPP**, **KDP**, **SFTKDP**, **SATP**, **IMAXP, KDFP**

RELATIONS: ***Ki_P_scl*** = *Ki_P_real * Kif / Kpf*;
Kp_P_scl = *Kp_P_real * Kif / Kpf*;
Kd_P_scl = *Kd_P_real * Kif / Kpf*;
I_Max = *ImaxSys*;

SFTKPP = smallest positive power of 2 for which *Kp_P_scl* < 2^SFTKPP
SFTKIP = smallest positive power of 2 for which *Ki_P_scl* < 2^SFTKIP
SFTKDP = smallest positive power of 2 for which *Kd_P_scl* < 2^SFTKDP
KIP = 32767 * *Ki_P_scl* / 2^SFTKIP;
KPP = 32767 * *Kp_P_scl* / 2^SFTKPP;
KDP = 32767 * *Kd_P_scl* / 2^SFTKDP;
KDFP = filter_D * 32767;
SATP = 32767 - *I_Max * Kif*;
IMAXP = (int)((1 - IPSatP / 100) * 32767);

3.5.4. Advanced features

3.5.4.1. Speed feedforward

As seen in Figure 3.15 the inputs of the speed feedforward block are the target speed (TSPD) given by the reference generator and the reference speed (SPDREF) given by the position controller. The block output represents the numerical value of the reference speed (SPDREF) applied to the speed controller.

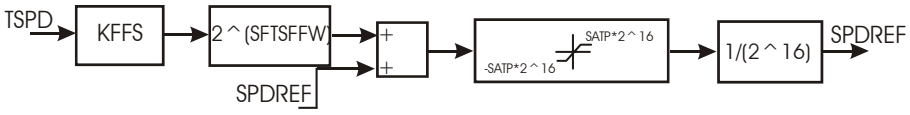


Figure 3.15. *Speed feedforward block implementation*

Speed feedforward block parameters:
- target speed coefficients (KFFS, SFTSFFW)
- speed feedforward block saturation coefficient (SATP).

Speed feedforward setup:

INPUTS: – scaled speed feedforward factor ***KFFS_scl*** [-]

OUTPUTS: – TML parameters: **KFFS**, **SFTSFFW**

RELATIONS: **KFFS** = 32767* *KFFS_scl*/(2^SFTSFFW);
KFFL = 32767
if(*KFFS_scl* <= 1)
SFTSFFW = 0
else
SFTSFFW = smallest positive power of 2 for which *Kp_P_scl* < 2^SFTKPP

3.5.4.2. Acceleration and load feedforward

Figure 3.16 presents the structure of the acceleration and load feedforward block. The inputs are the target acceleration (TACC) given by the reference generator, the feedforward load (FFL) from load sensor and the reference current (IQREF) given by the speed controller, or by the position controller when the speed loop is disabled. The block output is the numerical value of the q axis reference current (IQREF) applied to the current controller.

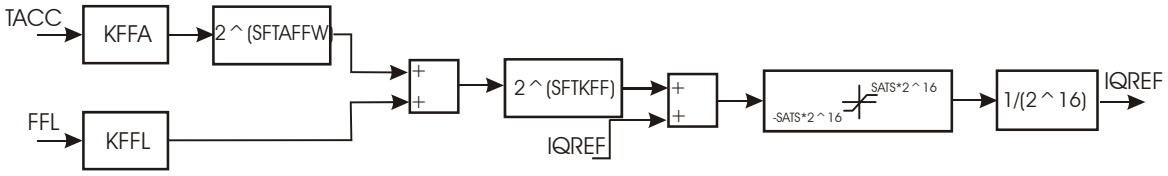


Figure 3.16. *Acceleration & load feedforward block implementation*

Acceleration and load feedforward block parameters:
- acceleration feedforward coefficients (KFFA, SFTSFFW);
- load feedforward coefficient (KFFL);
- feedforward shift coefficient (SFTKFF);
- feedforward block saturation coefficient (SATS).

Acceleration and load feedforward setup

INPUTS: – scaled acceleration feedforward factor **KFFA_scl** [-]

OUTPUTS: **– Global_shift**
– TML parameters*:* **KFFA**, **SFTAFFW**, **KFFL**, **SFTKFF**

RELATIONS: if(KFFA_scl <= 1)
Global_shift = 0
else
Global_shift = smallest positive power of 2 for which KFFA_scl < 2^Global_shift
KFFA = (integer)(32767* KFFA_scl/(2^Global_shift));
KFFL = 32767
if (Global_shift > 11)
SFTAFFW = 11
else
SFTAFFW = Global_shift

```
if (Global_shift > 11)
        SFTKFF = Global_shift – SFTAFFW
else
        SFTKFF = 0
```

3.5.4.3. Phase advance

MotionChip II uses Phase Advance block in field position computation for high speed application with PMSM, to eliminate measurement and computation delays. For this block there are no settings required from the user.

3.5.4.4. Position control suspend

The MotionChip II can suspend the execution of the control loops in case of a positioning application when the motor has arrived to the target position and is standing still. If the difference between the position reference and the actual position is lower than an accepted position error limit, for a given period of time, the execution of the control loops is suspended. The control is resumed when the position error exceeds again the accepted position error limit.

Position control suspend setup

INPUTS: – accepted position error limit **POSOKLIM** [bits]
– position control suspend trigger time **TONPOSOK** [slow loop samplings]

OUTPUTS: – TML parameters: **POSOKLIM**, **TONPOSOK**

3.6. Protections

Several protection functions can be activated during the motion. Once activated, a protection is monitored, and will generate a TML interrupt when the corresponding limit value is overridden. Depending on the selections done in the previous steps, some of the choices won't be possible.

Protection masks are set in Protections Control Register (**PCR**). The same register contains also information related to the status of the protections. Individual flags can be tested for decisions related to protection activation. A specific interrupt flag is set if an active protection occurs, in the **ISR** and **ICR** registers (see par. 4.1 for registers descriptions).

Bits **B5-B0** of the **PCR** register must be set according to this option.

PCR register (B5 - B0 default: 0)

B5	**Under voltage protection**
0	Not activated
1	Activated
B4	**Over voltage protection**
0	Not activated
1	Activated
B3	**Over temp2 protection**
0	Not activated
1	Activated
B2	**Over temp1 protection**
0	Not activated
1	Activated
B1	**I2t protection**
0	Not activated
1	Activated
B0	**Imax protection**
0	Not activated
1	Activated

3.6.1. Over current protection

The over current protection checks if the motor currents override a maximum accepted limit, for a given period of time. This check is done in the fast control loop of the real-time kernel of the MotionChip II. If the limit is overridden, the corresponding status bit in the PCR register is set. Also, if the "software protections" TML interrupt is enabled (unmasked), a TML interrupt is generated.

The maximal current protection is set if the absolute value of the measured torque component of the current overrides a limit specified by the user, for a given period of time. The protection is not activated if the override time is smaller than the imposed time limit.

Set (Imax Protection) IF (abs(i_{tq}) > i_{Max_Prot}) for more than MaxSamplingTime

INPUTS: – current sensor scaling factor ***Kif*** [bits/A]
– requested speed loop sampling period ***Ts_S*** [s]
– current trigger value for over current protection ***IMAX_PROT*** [A]
– Imax sampling time ***Time_IMAX_PROT*** [s]

OUTPUTS: – TML Parameters: **IMAXPROT**, **TIMAXPROT**

IMAXPROT = *IMAX_PROT * Kif*;
TIMAXPROT = *Time_IMAX_PROT* / *Ts_S*;

3.6.2. I2t protection

MotionChip II offers the possibility to protect your motor against long term over currents. In order to set this protection the user chooses a point on motor thermal protection curve (Figure 3.17). The point gives the parameter *I_I2t*, the maximum value for motor over current and *t_I2t*, the maximum time allowed to over current to flow through motor. Parameters *I_I2t* and *t_I2t* together with the nominal motor current I_n define the surface S_{I2t} (integral limit).

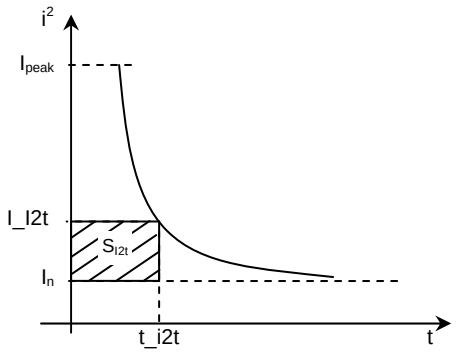


Figure 3.17. *Motor I2t thermal protection curve*

If the motor instantaneous current is greater than the nominal current I_n and the I2t protection is activated, the difference between the square of the instantaneous current and the square of the nominal current is integrated and compared with the S_{I2t} value (see Figure 3.18). When the integral equals the S_{I2t} surface the, corresponding status bit in the PCR register is set. Also, if the "Software protections" TML interrupt is enabled (unmasked), a TML interrupt is generated.

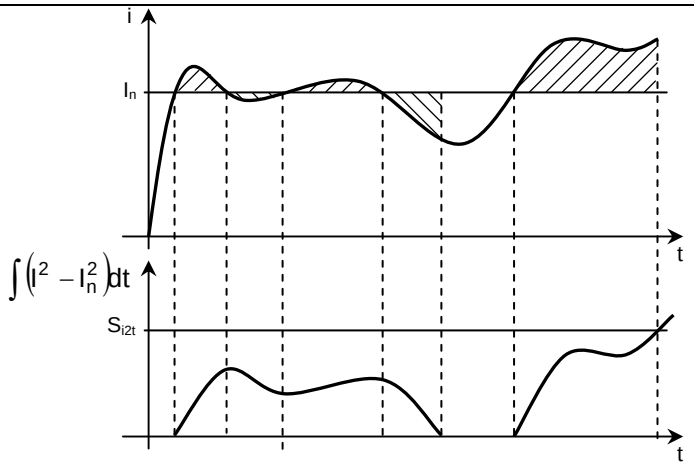


Figure 3.18. *I^2t protection implementation*

INPUTS: – current scaling factor ***Kif*** [bits/A]
– motor nominal current ***In*** [A]
– speed loop sampling time ***Ts_S*** [s]
– maximum value for motor over current ***I_I2t*** [A]
– time at over current ***t_I2t*** [s]

OUTPUTS: – TML Parameters: **II2TPROT, SFI2T, I2TINLIM**

RELATIONS: **II2TPROT** = *In * Kif*;
SFI2T= 2^26**Ts_S*/*t_I2t*
I2TINLIM = (*t_I2t*/*Ts_S*)*[(I_I2t**Kif*)^2-(*In***Kif*)^2]*SFI2T/32767/32767;

3.6.3. Over voltage protection

The maximal voltage protection checks if the DC-link voltage overrides a maximum accepted limit. This check is done in the fast control loop of the real-time kernel of the MotionChip II. If the limit is overridden, the corresponding status bit in the PCR register is set. Also, if the "Software protections" TML interrupt is enabled (unmasked), a TML interrupt is generated.

The maximal voltage protection is set if the value of the measured DC-link voltage overrides a limit specified by the user.

Set (U max protection) IF $U_{DC} > U_{DCMax}$

INPUTS: – voltage scaling factor ***Kuf*** [bits/V]
– protection maximum voltage ***UMAX_PROT*** [V]

OUTPUTS – TML Parameters **UMAXPROT**

RELATIONS: **UMAXPROT** = *Kuf * UMAX_PROT*;

3.6.4. Under voltage protection

The minimal voltage protection checks if the DC-link voltage is below a minimum accepted limit. This check is done in the fast control loop of the real-time kernel of the MotionChip II. If the limit is attained, the corresponding status bit in the PCR register is set. Also, if the "software protections" TML interrupt is unmasked, a TML interrupt is generated.

The minimal voltage protection is set if the value of the measured DC-link voltage is below a limit specified by the user.

Set (U min protection) IF $U_{DC} < U_{DCMin}$

INPUTS: – voltage scaling factor ***Kuf*** [bits/V]
– required protection minimum voltage ***UMIN_PROT*** [V] [*user*]

OUTPUTS: – TML Parameters: **UMAXPROT**

RELATIONS: **UMINPROT** = *Kuf * UMIN_PROT* ;

3.6.5. Over temperature protections

The maximal temperature protection checks if the analog values read on pins TEMP_1 or TEMP_2 override maximum accepted limits. This check is done in the fast control loop of the real-time kernel of the MotionChip II. If a limit is overridden, the corresponding status bit in the PCR register is set. Also, if the "Software protections" TML interrupt is enabled (unmasked), a TML interrupt is generated.

The maximal temperature protection is set if the values measured on TEMP_1 or TEMP_2 analog inputs of MotionChip II override user specified limits.

Set (Temp_1 max protection) IF $Temp1 > Temp1_{Prot}$

Set (Temp_2 max protection) IF $Temp2 > Temp2_{Prot}$

The "Temperature" inputs can measure any analog value, which must operate below a specified limit, and must signalize when this limit is overridden.

INPUTS: – drive temperature sensor gain factor TempSensorGain [V/°C]
– drive temperature sensor output at 0 °C TempSensorOutputAtoC [V]
– drive over temperature trigger level Temp_2_Max [°C]
– motor temperature sensor type MotorTempSensorType [-]

OUTPUTS: – temperature sensor offset **TempOffset** [bits]
– TML parameters: **T1MAXPROT**, **T2MAXPROT**, **MTSTYPE**

RELATIONS:
```
if (MotorTempSensorType == NTC)
        MTSTYPE = 0          //sensor type NTC
else
        MTSTYPE = 1          //sensor type PTC
T1MAXPROT = 32767;           //for NTC/PTC sensors type
```

TempOffset = 65472 /3.3* TempSensorOutputAtoC;
T2MAXPROT = *Temp_2_Max* * TempSensorGain* 65472 /3.3+TempOffset;

3.6.6. Control error protection

The MotionChip II can supervise the error between the reference and the actual value of the outer loop controlled variable (position, speed or current). If the difference between the two values overrides a maximum accepted limit, for a given period of time, a control error protection occurs. This check is done in the fast control loop of the real-time kernel of the MotionChip II. If the limit is overridden, the corresponding status bit in the MSR register is set. Also, if the "Control error" TML interrupt is enabled (unmasked), a TML interrupt is generated.

The control error protection is set if the absolute value of the measured controlled variable overrides a limit specified by the user, for a given period of time. The protection is not activated if the override time is smaller than the imposed time limit.

Set (Control error) IF (abs(error in outer loop) > max_accepted_error) for more than MaxSamplingTime

3.7. Scaling factors

The values you set in the TML parameters must be always expressed in internal units. The conversion from international standard units in MotionChip II internal units is made using scaling factors. The next section presents the scaling factors used by MotionChip II.

Position scaling factor

Scaling factor for quadrature encoder is:

$$Kpf = \frac{4 \cdot N\_encoder\_lines}{2\pi} \left[\frac{bits}{rad}\right]$$

where N_encoder_lines – number of encoder lines per revolution.

Speed scaling factor

Speed scaling factor when it's estimated from position information is:

$$Kvf = \frac{4 \cdot N\_encoder\_lines \cdot Ts\_S}{2\pi} \left[\frac{bits}{rad/s}\right]$$

where N_encoder_lines – number of encoder lines per revolution.

Ts_S – speed loop sampling time [s].

Acceleration scaling factor

Acceleration scaling factor when it's estimated from position information is:

$$Kaf = \frac{4 \cdot N\_encoder\_lines \cdot Ts\_S^2}{2\pi} \left[\frac{bits}{rad/s^2}\right]$$

where N_encoder_lines – number of encoder lines per revolution.

Ts_S – speed loop sampling time [s].

Current scaling factor

Scaling factor for current sensor is:

$$Kif = \frac{65472}{2 \cdot ImaxPS} \left[\frac{bits}{Amps}\right]$$

where: *ImaxPS* is power stage peak current i.e. the maximum measurable current [A].

This formula results from the simplification of the following one:

$$Kif = \frac{CGa\,[V/A]}{3.3\,[V]} \cdot \frac{65472}{2^{SFTCRT}} \cdot \left[\frac{bits}{Amps}\right]$$

where: *CGa* is the current gain factor and SFTCRT is the current shift factor.

The current gain factor represents the ratio between the voltage range of a MotionChip II A/D input (i.e. 3.3V) and the maximum measurable motor current range corresponding to the A/D input range. For example, if the maximum measurable current is in the range +/-5A, the current gain factor will be: *CGa* = 3.3 V / 10 A = 0.33 V/A.

The current shift factor SFTCRT, is set by default to 0 and gives some degree of flexibility to the user. It can be used in order to change the scaling factor of current measurement. Consequently, this will affect the coefficients of the current, speed and position controllers. If, for a given system structure, some of these coefficients result outside of the accepted range of values, you can try to increase SFTCRT and re-compute the controllers coefficients.

Supply/DC-link voltage scaling factor

Scaling factor for supply/DC-link voltage measurement is:

$$Kuf\_m = \frac{65472}{VdcMaxMeasurable} \left[\frac{bits}{volts}\right]$$

where: *VdcMaxMeasurable* – is the maximum measurable supply/DC-link voltage [V].

Voltage command scaling factor

The significance of the voltage commands as well as the scaling factors, depend on the motor technology and the control method used.

- For the PMSM motors control with standard PWM command the scaling factor for voltage command is computed with the following relation:

$$Kuf = \frac{2 \cdot 32767}{Vdc} \left[\frac{bits}{volts}\right]$$

where: Vdc – is the DC-link/supply voltage [V]

When Udc compensation is activated for PWM command, the scaling factor for voltage command is computed with the following relation:

$$Kuf = \frac{2 \cdot 32767}{Vdc \cdot \frac{CVDC}{PWM\_per}} \left[\frac{bits}{volts}\right]$$

where: CVDC – ratio between the nominal voltage value and the measured DC-link/supply voltage value. It is used to implement the compensation of DC link/ supply voltage variations;

PWM_per – PWM period;

Vdc – is the DC-link/supply voltage [V].

For PWM technique with 3rd harmonic injection the CVDC is equal with 1.1* *PWM_per*.

- For DC or BLDC motors the voltage command scaling factor is:

$$Kuf = \frac{32767}{Vdc} \left[\frac{bits}{volts}\right]$$

where: Vdc – is the DC-link/supply voltage [V].

Temperature scaling factor

The temperature variables are considered as unsigned measurements, i.e. measured 0 V correspond to 0 value of the variable, while measured 3.3 V correspond to the maximum measurable temperature. Scaling factor for temperature measurement is:

$$KTf = \frac{TempSensorGain}{3.3V} \cdot 65472 \left[\frac{bits}{degree}\right]$$

where *TempSensorGain* – represents temperature sensors gain, computed as the ratio between the voltage applied at DSP input pin and the measured temperature; its expressed in Volt/degree.

The measured temperature expressed in bits must be corrected by adding the temperature sensor offset (Temp_Off) expressed in bits. The temperature sensor offset is expressed as

$$Temp\_Off = TempSensorOutputAt0°C \cdot \frac{65472}{3.3V} [bits]$$

Time scaling factor

All variables time related are expressed in internal units using the following scaling factor:

$$Ktf = \frac{1}{Ts\_S}\left[\frac{1}{s}\right]$$

where *Ts_S* – speed loop sampling time in seconds

Current increment scaling factors

The correspondence with the international standard (SI) units for current increment is:

$$K\_C\_inc = \frac{65472 \times Ts\_S}{2ImaxPS}$$

where:

ImaxPS – is the power stage maximum current [A]
Ts_S – is the speed loop sampling period [s]

Voltage (command) increment scaling factors

Like in the case of the voltage command units, scaling factors from international standard (SI) units to the voltage increment units depends on the on the motor technology and the control method used.

For the brushed DC and brushless DC motors, the scaling factors is:

$$K\_V\_inc = \frac{32767 \times Ts\_S}{Vdc}$$

For the brushless AC motor, the scaling factor is:

$$K\_V\_inc = \frac{65534 \times Ts\_S}{1.1Vdc}$$

where:

Vdc – is the DC-link/supply voltage [V]
Ts_S – is the speed loop sampling period [s]

Electrical angle scaling factor

Electrical angle scaling factor is:

$$K\_th = \frac{32768}{\pi}$$

The electrical angle is the mechanical angle divided by the number of pole pairs. For example when a brushless motor with 2 pairs does half of revolution (i.e. 180 mechanical degrees) this corresponds to 360 electrical degrees

Electrical angle increment scaling factor

For electrical angle increment the scaling factor used is:

$$K\_th\_inc = \frac{32767 \times Ts\_C \times pp}{\pi}$$

where:

pp – is the number of pair poles
Ts_C – is the current loop sampling period [s]

Master Position scaling factor

When the master position is sent via a communication channel, the master position units depend on the type of position sensor present on the master axis.

When the master position is an encoder the scaling factor is:

$$Kpf\_master = \frac{4 \times No\_encoder\_lines}{2 \times \pi}$$

where:

No_encoder_lines – is the master's number of encoder lines per revolution

Master Speed scaling factor

The master speed is computed in internal units (IU) as master position units /slow loop sampling period i.e. the master position variation over one position/speed loop sampling period.

When the master position is an encoder the scaling factor is:

$$Kvf\_master = \frac{4 \times No\_encoder\_lines \times Ts\_S}{2 \times \pi}$$

where:

No_encoder_lines – is the master's number of encoder lines per revolution

Ts_S – is the slow loop sampling period[s]

4. TML data

4.1. TML Registers

Configuration registers (R/W):

- **OSR – Operating Settings Register.** Contains system operating settings, as current offset detection mode, PMSM motor start procedure etc.
- **SCR – System Configuration Register.** Contains information that defines the motor and sensors type used in the motion system structure.

Control registers (R/W):

- **CCR – Communication Control Register.** Contains settings for SPI communication channel.
- **ICR – Interrupt Control Register.** Enables/disables TML interrupts.
- **PCR.5-0 – Protections Control Register.** Activates/deactivates different protections in the system, as maximum current, I^2t, over and under voltage and over temperature.

Status registers (RO):

- **AAR – Axis Address Register.** Contains axis address (AXISID) and group address (GROUPID) of the motion axis.
- **CBR – CAN Baud rate Register.** Contains information on CAN controller related parameters.
- **CER – Communication Error Register.** Contains error bits for communication channels, serial (RS232/RS485) and CAN.
- **CSR – Communication Status Register.** Contains status information about the different communication channels, as CAN, SPI baud rate, serial baud rate, etc.
- **ISR – Interrupt Status Register.** Contains interrupt flags corresponding to the different events that can generate a TML interrupt.
- **MCR – Motion Command Register.** Used to examine the motion reference type, active control loops, positioning type, etc.
- **MSR – Motion Status Register.** Mainly used internally by the TML kernel, the register bits give indications about motion progress and specific motion events as software protections, control error, wrap-around, limit switches, captures, contour segments, events, axis status, etc.
- **PCR.13-8 – Protections Control Register.** Shows the status of different protections in the system, as maximum current, over and under voltage etc.

Some of these registers are read/write (R/W), while some are read-only (RO) registers. The configuration registers **MUST** be defined before the "`ENDINIT`" TML command is issued. The

command registers may be defined / changed also during the motion (if these changes do not affect the motion or functionality of the system, and are not in contradiction with other settings in the system). The status registers are read-only registers, and can only be read by the user.

Each TML register has also a specific data memory address. This address is needed to build binary TML commands, to be sent via a communication channel, towards the MotionChip II.

Mnemonic	Register Name	Type	Address	Par.
OSR	**O**perating **S**ettings **R**egister	cfg	0x0302	**4.1.1**
SCR	**S**ystem **C**onfiguration **R**egister	cfg	0x0300	**4.1.2**
CCR	**C**ommunication **C**ontrol **R**egister	cmd	0x030A	**4.1.3**
ICR	**I**nterrupt **C**ontrol **R**egister	cmd	0x0304	**4.1.4**
PCR	**P**rotections **C**ontrol **R**egister	cmd/stat	0x0303	**4.1.5**
AAR	**A**xis **A**ddress **R**egister	stat	0x030C	**4.1.6**
CBR	**C**AN **B**aud rate **R**egister	stat	0x030D	**4.1.7**
CER	**C**ommunication **E**rror **R**egister	stat	0x0301	**4.1.8**
CSR	**C**ommunication **S**tatus **R**egister	stat	0x030B	**4.1.9**
ISR	**I**nterrupt **S**tatus **R**egister	stat	0x0306	**4.1.10**
MCR	**M**otion **C**ommand **R**egister	stat	0x0309	**4.1.11**
MSR	**M**otion **S**tatus **R**egister	stat	0x0308	**4.1.12**

The following paragraphs present each of the TML registers function, setting and usage. All TML registers are 16-bit registers. Bits are numbered from Bit 0 (LSB) to Bit 15 (MSB). Default values after reset, for each bit, are given bellow the register description table.

4.1.1. OSR – Operating Settings Register (configuration, R/W)

Purpose: OSR is a 16-bit command register, containing specific information related to the operation of the motion system. Current offset detection scheme, PMSM motors start procedure, Udc compensation, Ud,q saturation method, PWM type, etc. are settings defined in this register.

TML Address: 0x0302

Contents. OSR information is structured as follows:

15	14-12	11	10	9	8
ELGMD	**Reserved**	**LOGGER**	**Reserved**	**BKCMD**	**UDQSAT**
0	000	0	0	0	0

7-6	5	4-2	1-0
PWM	**UDCCOMP**	**PMSMST**	**CRTOFF**
00	0	000	00

Bit 15 ELGMD. Electronic gearing master mode
- 0 = Send actual position to slave axes
- 1 = Send target position to slave axes

Bit 14-12 Reserved

Bit 11 LOGGER. PMSM start logging
- 0 = No data logging possible during PMSM motor start
- 1 = Data logging possible during PMSM motor start

Bit 10 Reserved

Bit 9 BKCMD. Brake command
- 0 = Brake command inactive
- 1 = Brake command active

Bit 8 UDQSAT. Ud,q command saturation method
- 0 = Use independently saturated commands on d and q axes
- 1 = Compute Uq from Ud and saturation limit. Uq = f(Ud)

Bits 7-6 PWM. PWM command method
- 00 = Standard symmetric PWM
- 01 = Dead-time and U_{DC} compensation
- 10 = Dead-time, U_{DC} compensation, third harmonic injection
- 11 = Wobbling with U_{DC} compensation, third harmonic injection

Bit 5 UDCCOMP. U_{DC} compensation
- 0 = Inactive
- 1 = Active

Bits 4-2 PMSMST. PMSM motor start method

000 = a / b, current mode, incremental sensors
001 = a / b, voltage mode, incremental sensors
010 = Hall / Encoder speed loop
011 = Reserved
100 = Reserved
101 = Reserved
110 = Reserved
111 = Reserved

Bits 1-0 CRTOFF. Current offset detection

00 = No current offset detection
01 = Detection without PWM activated
10 = Detection with PWM activated
11 = Reserved

Setting: This register is initialized during the default configuration phase of the motion system. Its content will normally be set before starting the motion of the system. Use local or on-line TML commands in order to set this register.

TML instruction for OSR register setting:

OSR = value16;

These settings will be used during the motion performing, in the real-time control module.

Example: In order to configure the following settings:

- Electronic gearing mode **–** master sends actual position
- Data logger **–** no data logging possible during PMSM motor start
- Brake command – active
- Ud,q command saturation method – independent
- PWM type – dead-time, U_{DC} compensation, third harmonic injection
- U_{DC} compensation – inactive
- PMSM start method – a / b voltage, incremental sensor
- Current offset detection – with active PWM

Use the following assignment instruction:

OSR = 0x0286;

4.1.2. SCR – System Configuration Register (configuration, R/W)

Purpose: SCR is a 16-bit configuration register, containing information that defines the motor and sensors type used in the motion system structure.

TML Address: 0x0300

Contents. SCR information is structured as follows:

15	14-12	11-9	8	7	6	5-3	2-0
Reserved	**MOTOR**	**Reserved**	**TSNS2**	**TSNS1**	**Reserved**	**SSNS**	**PSNS**
0	000	000	0	0	0	000	000

Bit 15 Reserved

Bits 14-12 MOTOR. Motor type

000 = Brushless DC (BLDC)
001 = Brushed DC
010 = Brushless AC (PMSM)
100 = Reserved
101 = Reserved
011 = Reserved
110 = Reserved
111 = Reserved

Bits 11-9 Reserved

Bit 8 TSNS2. Drive/ power stage temperature sensor

0 = No temperature sensor 2
1 = Use temperature sensor 2

Bit 7 TSNS1. Motor temperature sensor

0 = No temperature sensor 1
1 = Use temperature sensor 1

Bit 6 Reserved

Bits 5-3 SSNS. Speed sensor

000 = Position difference
001 = Reserved
010 = Reserved
100 = Reserved
011 = Reserved
101 = Reserved
110 = Reserved
111 = None

Bits 0-2 PSNS. Position sensor

000 = Quadrature encoder on QEP
100 = Linear Hall
001 = Reserved
010 = Reserved
011 = Reserved
101 = Reserved
110 = Reserved
111 = None

Setting: This register **MUST** be initialized at motion system configuration phase, before the execution of the "ENDINIT" TML command. It **MUST NOT** to be changed latter, during the operation of the motion system.

TML instruction for SCR register setting:

SCR = value16;

These settings are used during the operation of the real time control module.

Example: In order to configure the following settings:

- Motor type – PMSM:
- Drive/ power stage temperature sensor – No
- Motor temperature sensor – Yes
- Speed sensor – Position difference
- Position sensor – Quadrature encoder on QEP

Use the following assignment instruction:

SCR = 0x2080;

4.1.3. CCR – Communication Control Register (command, R/W)

Purpose: CCR is a 16-bit command register, containing specific settings for the communication devices SPI.

TML Address: 0x030A

Contents. CCR information is structured as follows:

15-2	1	0
Reserved	Reserved	SPIMEM
0	1	1

Bits 15-2 Reserved

Bit 1 Reserved

Bit 0 SPIMEM. SPI E^2ROM memory

0 = Not installed

1 = Installed

Setting: This register is initialized during the default configuration phase of the motion system. Its content will normally be set before starting the motion of the system. Use local or on-line TML commands in order to set this register.

TML instruction for CCR register setting:

CCR = value16;

These settings will be used during the operation of the communication channels.

Example: In order to configure SPI memory as installed use the following assignment instruction:

CCR = 0x0001;

4.1.4. ICR – Interrupt Control Register (command, R/W)

Purpose: ICR is a 16-bit command register, containing the masks for TML interrupts. All the unmasked bits of this register will allow the generation of a TML interrupt at the occurrence of the associated specific situation.

TML Address: 0x0304

Contents. ICR information is structured as follows:

15	14-12	11	10	9	8
GIM	**Reserved**	**EVNIM**	**SEGIM**	**MOTIM**	**PCAPIM**
0	000	0	0	0	0

7	6	5	4	3	2	1	0
LSWNIM	**LSWPIM**	**WRPIM**	**CMERIM**	**CTRERIM**	**SWPRIM**	**PDPIM**	**DSLBIM**
0	0	0	0	0	0	0	0

Bit 15 GIM. Global Interrupt Mask for all TML interrupts
0 = Disable
1 = Enable

Bits 14-12 Reserved.

Bit 11 EVNIM. Interrupt mask for TML events
0 = Disable
1 = Enable

Bit 10 SEGIM. Interrupt mask for "Possible contour segment update" TML interrupt
0 = Disable
1 = Enable

Bit 9 MOTIM. Interrupt mask for "Motion complete" TML interrupt
0 = Disable
1 = Enable

Bit 8 PCAPIM. Interrupt mask for "Position capture" TML interrupt
0 = Disable
1 = Enable

Bit 7 LSWNIM. Interrupt mask for "Negative limit switch" TML interrupt

0 = Disable

1 = Enable

Bit 6 LSWPIM. Interrupt mask for "Positive limit switch" TML interrupt

0 = Disable

1 = Enable

Bit 5 WRPIM. Interrupt mask for "Position wrap-around" TML interrupt

0 = Disable

1 = Enable

Bit 4 CMERIM. Interrupt mask for "Communication error" TML interrupt

0 = Disable

1 = Enable

Bit 3 CTRERIM. Interrupt mask for "Control error" TML interrupt

0 = Disable

1 = Enable

Bit 2 SWPRIM. Interrupt mask for "Software protections" TML interrupt

0 = Disable

1 = Enable

Bit 1 PDPIM. Interrupt mask for "Power drive protection" TML interrupt

0 = Disable

1 = Enable

Bit 0 DSLBIM. Interrupt mask for "Axis disable" TML interrupt

0 = Disable

1 = Enable

Setting: This register is initialized during the default configuration phase of the motion system. Use local or on-line TML commands in order to activate / deactivate TML interrupts, at any time during motion system operation.

TML instruction for ICR register setting:

ICR = value16;

The **ICR** register must be set according to the TML interrupts that the user wants to activate during the motion. The TML processor will check its content, once any of the TML interrupts flag is set. Only interrupts that are enabled will generate a TML interrupt request. See par. 4.5 for details about TML interrupts.

Example: In order to enable the following TML interrupts:

- Axis disable
- Power drive interrupt
- Communication error

All the remaining TML interrupts are disabled. TML interrupts are globally enabled.

Use the following assignment instruction:

ICR = 0x0013;

4.1.5. PCR – Protections Control Register (command / status, R/W)

Purpose: PCR is a 16-bit command and status register, containing both masks and status information for TML protections that can be generated during the motion.

TML Address: 0x0303

Contents. PCR information is structured as follows:

15-14	13	12	11	10	9	8
Reserved	**UVPRS**	**OVPRS**	**OT2PRS**	**OT1PRS**	**I2TPRS**	**IMXPRS**
00	0	0	0	0	0	0

7-6	5	4	3	2	1	0
Reserved	**UVPRM**	**OVPRM**	**OT2PRM**	**OT1PRM**	**I2TPRM**	**IMXPRM**
00	0	0	0	0	0	0

Bit 15-14 Reserved.

Bit 13 UVPRS. Status of "Under voltage protection"
0 = Off
1 = On

Bit 12 OVPRS. Status of "Over voltage protection"
0 = Off
1 = On

Bit 11 OT2PRS. Status of "Over temperature 2 protection"
0 = Off
1 = On

Bit 10 OT1PRS. Status of "Over temperature 1 protection"
0 = Off
1 = On

Bit 9 I2TPRS. Status of "I^2t motor thermal protection"
0 = Off
1 = On

Bit 8 IMXPRS. Status of "Maximum motor current protection"
0 = Off
1 = On

Bits 7-6 Reserved.

Bit 5 UVPRM. Mask for "Under voltage protection"
0 = Protection not activated
1 = Protection activated

Bit 4 OVPRM. Mask for "Over voltage protection"
0 = Protection not activated
1 = Protection activated

Bit 3 OT2PRM. Mask for "Over temperature 2 protection"
0 = Protection not activated
1 = Protection activated

Bit 2 OT1PRM. Mask for "Over temperature 1 protection"
0 = Protection not activated
1 = Protection activated

Bit 1 I2TPRM. Mask for "I^2t motor thermal protection"
0 = Protection not activated
1 = Protection activated

Bit 0 IMXPRM. Mask for "Maximum motor current protection"
0 = Protection not activated
1 = Protection activated

Setting: This register is initialized during the default configuration phase of the motion system. Use local or on-line TML commands in order to activate / deactivate external interrupt generation at specific TML interrupts, at any time during motion system operation.

TML instruction for PCR register setting:

PCR = value16;

Bits 5-0 must be set according to par. 3.6. Protections. These settings will be used during the real time motion control in order to check the activated protections status.

Bits 13-8 can be read to get information about protections status. If any of these protections is activated, the corresponding "Software protections" interrupt flag from the ISR register will be set. Correspondingly, if that TML interrupt is enabled, a TML interrupt request is generated.

Example: In order to configure the following protections:

- Over voltage
- Under voltage
- Maximum motor current

Use the following assignment instruction:

PCR = 0x0031;

In order to examine protections status, read the PCR register and examine the corresponding bits from the upper byte of the register. Note that a TML interrupt flag is set in the corresponding bit of the Interrupt Status Register (ISR), if an activated protection occurs.

For example, if PCR content is read as **0x0131**, it means that the protection for maximum motor current was activated (bit 8 of PCR is set), due to motor current exceeding maximum accepted value.

4.1.6. AAR – Axis Addresses Register (status, RO)

Purpose: AAR is a 16-bit status register, containing information that defines the individual and group addresses of the motion axis.

TML Address: 0x030C

Contents. AAR information is structured as follows:

15	14	13	12	11	10	9	8
GR7	**GR6**	**GR5**	**GR4**	**GR3**	**GR2**	**GR1**	**GR0**
0	0	0	0	0	0	0	1

7-0
AxisID

Read from external hardware or set to 0xFF by software

Bits 15-8 GRn. Group n selection

0 = The motion axis does not belong to group n

1 = The motion axis belongs to group n

Bits 7-0 AXISID. Axis address

Xx = Individual identification address for the motion axis

Setting: This register is initialized during the default configuration phase of the motion system. Depending on the states of MotionChip II analogue inputs lines ADCIN10 to ADCIN14, the following initialization is performed:

- if all analogue inputs pins ADCIN10 to ADCIN14 are high a software initialization is performed and the AxisID = 255
- if at least one of the ADCIN10 to ADCIN14 are low a hardware initialization is performed, the AXISID value depends on their combination (see MotionChip™ II TML programming User Manual)

Apart from the Axis ID, each drive has also a **group ID**. The group ID represents a way to identify a group of drives, for a multicast transmission. Each drive can be programmed to be member of one or several of the 8 possible groups. By default all the drives are set as members of group 1.

Both axis and group fields can be dynamically changed at any time during motion system operation, by using local or on-line specific TML commands.

TML instruction for axis identifier setting (indirectly reflected in the AAR AXISID field):

AXISID value16;

or

AXISID var16;

TML instruction for axis group setting (indirectly reflected in the AAR AXISID field):

GROUPID value16;

or

GROUPID var16;

The AAR register will be set in order to indicate the individual and group identifiers. See MotionChip II TML Programming User Manual for details about multi-axis operation of the motion system.

These settings will be used during the operation of the communication channels.

Example: In order to configure the following settings:

- Group selection: **[10011010]** axis belongs to groups 2, 4, 5, 8
- Individual identification address: **[00010010]** 012h

Use the following TML instructions:

GROUPID 0x9A;

AXISID 0x12;

4.1.7. CBR – CAN Baud rate Register (status, R/W)

Purpose: CBR is a 16-bit status register, containing information to setup up the communication baud rate parameters for CAN controller.

TML Address: 0x030D

Contents. CBR information is structured as follows:

15-8	7-0
CANBTR1	**CANBTR0**
32	73

Bits 15-8 CANBTR1. CAN bus Timing Register 1 (BTR1)
xx = CAN controller bus timing register 1

Bits 7-0 CANBTR0. CAN bus Timing Register 0 (BTR0)
xx = CAN controller bus timing register 0

Remark. The default settings of BTR0 and BTR1 registers correspond to a 500 kBits/s CAN communication speed.

Setting: This register is initialized during the default configuration phase of the motion system. Specific TML commands must be used in order to change the CAN communication baud rate during motion system operation.

TML instruction for CBR register setting:

CANBR value16;

These settings will be used in order to setup the CAN controller, and to communicate via this channel. For more details see MotionChip II TML Programming User Manual.

Example: In order to configure CAN communication channel at 1 MBits/s use the following assignment instruction

CANBR 0x1273;

4.1.8. CER – Communication Error Register (status, RO)

Purpose: CER is a 16-bit status register, containing status information about communication errors on CAN, SPI and SCI communication channels.

TML Address: 0x0301

Contents. CER information is structured as follows:

15-8	7
Reserved	**SPITTO**
0	0

6	5	4	3	2	1	0
CANBER	**CANTER**	**CANRER**	**Reserved**	**SCIRTO**	**SCITTO**	**SCIRER**
0	0	0	0	0	0	0

Bits 15-8 Reserved.

Bit 7 SPITTO. SPI timeout on write operation
- 0 = No SPI timeout on write operation
- 1 = SPI timeout on write operation

Bit 6 CANBER. CAN bus off error
- 0 = No CAN bus off error
- 1 = CAN bus off error

Bit 5 CANTER. CAN Tx overrun error
- 0 = No CAN transmission overrun error
- 1 = CAN transmission overrun error

Bit 4 CANRER. CAN Rx overrun error
- 0 = No CAN reception overrun error
- 1 = CAN reception overrun error

Bit 3 Reserved.

Bit 2 SCIRTO. SCI Rx timeout error
- 0 = No SCI reception timeout error
- 1 = SCI reception timeout error

Bit 1 SCITTO. SCI Tx timeout error

0 = No SCI transmission timeout error

1 = SCI transmission timeout error

Bit 0 SCIRER. SCI Rx error

0 = No SCI reception error

1 = SCI reception error

Setting: This register is initialized during the default configuration phase of the motion system. During system operation, the communication section of the system will indirectly set the register content.

No TML instructions may be used in order to set this register!

Use local or on-line TML commands in order to read CSR register contents and get the communication error status information, at any time during motion system operation.

Example. For example, if CER content is read as **0x0002**, it means that the system had a transmission error timeout on the SCI communication channel.

4.1.9. CSR – Communication Status Register (status, RO)

Purpose: CSR is a 16-bit status register, containing status information about the communication channels of the system, SPI and SCI.

TML Address: 0x030B

Contents. CSR information is structured as follows:

15	14	13-11	10	9-8	7-1	0
ELGEAR	**AXIDSTP**	**SCIBAUD**	**Reserved**	**SPIBAUD**	**Reserved**	**SCITYPE**
0	0	100	0	10	0	0

Bit 15 ELGEAR. Electronic gearing master flag
- 0 = No data to send, while in master gearing mode
- 1 = Data to send, while in master gearing mode

Bit 14 AXIDSTP. Axis ID setup flag
- 0 = The initial axis ID was imposed by software
- 1 = The initial axis ID was imposed by hardware

Bits 13-11 SCIBD. SCI serial communication channel baud rate
- 000 = 9600
- 001 = 19200
- 010 = 38400
- 011 = 56600
- 100 = 115200
- 101 = Reserved
- 110 = Reserved
- 111 = Reserved

Bit 10 Reserved

Bits 9-8 SPIBD. SPI baud rate
- 00 = 1 MHz
- 01 = 2 MHz
- 10 = 5 MHz
- 11 = Reserved

Bits 7-1 Reserved

Bit 0 SCITYPE. Serial communication driver type

0 RS-232

1 RS-485

Setting: This register is initialized during the default configuration phase of the motion system. During system operation, the register contents will be indirectly set, by using specific local or on-line TML commands that define communication parameters.

Use the following TML instructions to setup specific bits of CSR:

SCIBR value16;

SPIBR value16;

CANOC value16;

DO NOT USE assignment TML instructions in order to set the communication parameters specified in this register!

Use local or on-line TML commands in order to read CSR register contents and get the communication settings information, at any time during motion system operation.

Example. For example, if CSR content is read as **0x2100**, it means that the system uses a RS-232 driver on the SCI, 115200 baud rate on the SCI and 2 MHz SPI communication baud rate.

4.1.10. ISR – Interrupt Status Register (status, RO)

Purpose: ISR is a 16-bit status register, containing the interrupt flags for TML interrupts. Only unmasked TML interrupts (see Interrupt Control Register - ICR) will generate a TML interrupt request.

TML Address: 0x0306

Contents. ISR information is structured as follows:

15-12	11	10	9	8
Reserved	**EVNIF**	**SEGIF**	**MOTIF**	**PCAPIF**
0	0	0	0	0

7	6	5	4	3	2	1	0
LSWNIF	**LSWPIF**	**WRPIF**	**CMERIF**	**CTRERIF**	**SWPRIF**	**PDPIF**	**DSLBIF**
0	0	0	0	0	0	0	0

Bits 15-12 Reserved.

Bit 11 EVNIF. Interrupt mask for TML events
- 0 = Not triggered
- 1 = Triggered

Bit 10 SEGIF. Interrupt flag for "Possible contour segment update" TML interrupt
- 0 = Not triggered
- 1 = Triggered

Bit 9 MOTIF. Interrupt flag for "Motion complete" TML interrupt
- 0 = Not triggered
- 1 = Triggered

Bit 8 PCAPIF. Interrupt flag for "Position capture" TML interrupt
- 0 = Not triggered
- 1 = Triggered

Bit 7 LSWNIF. Interrupt flag for "Negative limit switch" TML interrupt
- 0 = Not triggered
- 1 = Triggered

Bit 6 LSWPIF. Interrupt flag for "Positive limit switch" TML interrupt

0 = Not triggered

1 = Triggered

Bit 5 WRPIF. Interrupt flag for "Position wrap-around" TML interrupt

0 = Not triggered

1 = Triggered

Bit 4 CMERIF. Interrupt flag for "Communication error" TML interrupt

0 = Not triggered

1 = Triggered

Bit 3 CTRERIF. Interrupt flag for "Control error" TML interrupt

0 = Not triggered

1 = Triggered

Bit 2 SWPRIF. Interrupt flag for "Software protections" TML interrupt

0 = Not triggered

1 = Triggered

Bit 1 PDPIF. Interrupt flag for "Power drive protection" TML interrupt

0 = Not triggered

1 = Triggered

Bit 0 DSLBIF. Interrupt flag for "Axis disable" TML interrupt

0 = Not triggered

1 = Triggered

Setting: This register is initialized during the default configuration phase of the motion system. During system operation, different internal sections of the system will indirectly set the register content, at specific context events occurrence. The ISR register is internally set during the operation of the motion system. The real time control module internally and dynamically updates its information. The user **MAY NOT CHANGE** its contents. The user can use this register in TML program sequences. The interrupt flags can be tested by polling, or used to generate TML interrupts, corresponding to the associated events. Use local or on-line TML commands in order to read ISR register contents and get TML interrupts flags status, at any time during system operation.

No TML instructions may be used in order to set this register!

Example. For example, if ISR content is read as **0x0140**, it means that the system got "position capture" and "positive limit switch" TML interrupts. If activated in the ICR register, these interrupts will start the execution of the associated, user-defined, TML interrupt service routines.

4.1.11. MCR – Motion Command Register (status, RO)

Purpose: MCR is a 16-bit status register, containing specific information related to the operation of the motion system. Reference type, active control loops, electronic gearing, positioning type, motion mode information, etc. are settings defined in this register.

TML Address: 0x0309

Contents. MCR information is structured as follows:

15	14	13	12	11	10	9	8
MMODE	**MODECHG**	**POSTYPE**	**RELPOSEV**	**ELGEAR**	**POSLP**	**SPDLP**	**CRTLP**
1	0	0	0	0	0	0	0

7-6	5	4-0
EXTREF	**REFLOC**	**REFTYPE**
00	0	10000

Bit 15 MMODE. Motion mode

0 = Same motion mode
1 = New motion mode

Bit 14 MODECHG. When motion mode changed

0 = Update the reference
1 = Keep the reference

Bit 13 POSTYPE. Positioning type

0 = Relative
1 = Absolute

Bit 12 RELPOSEV. Relative position events

0 = Do not set new origin at relative position events
1 = Set new origin at relative position events

Bit 11 ELGEAR. Electric gearing

0 = Reset the axis as a master axis, when in gearing mode
1 = Set the axis as a master axis, when in gearing mode

Bit 10 POSLP. Position loop status

0 = Off
1 = On

Bit 9 SPDLP. Speed loop status

0 = Off
1 = On

Bit 8 CRTLP. Current loop status

0 = Off
1 = On

Bits 6-7 EXTREF. External reference type

00 = On-line reference
01 = Analogue reference
10 = Reserved
11 = Reserved

Bit 5 REFLOC. Reference generator location

0 = Use the reference generator in the slow control loop (speed / position loop)
1 = Use the reference generator in the fast control loop (current loop)

Bits 4-0 REFTYPE. Reference type

00000 = External reference
00001 = Trapezoidal reference
00010 = Contouring position / speed

00011 = Contouring torque / voltage
00100 = Pulse & direction
00101 = Electronic gearing – slave
00110 = Electronic camming – slave
01000 = Test
10001 = Stop trapezoidal (Stop3)
10000 = Stop 0 /1 /2

Setting: This register is initialized during the motion mode configuration phase. Its content **MUST NOT** be set directly by the user (through an assignment instruction, for example). Specific local or on-line TML commands that set the motion mode will specifically set the contents of this register.

These settings will be used in the real-time control module of the application (motor control loops and reference generator module). See par. 3.1 for details about the real-time kernel, and TML Instructions User Manual for details about the reference generator module.

Example: TML instructions for indirect MCR register setting:

```
MODE PP3;
MODE SP1;
MODE PC2;
MODE TEF;
SGM;
```

etc.

4.1.12. MSR – Motion Status Register (status, RO)

Purpose: MSR is a 16-bit status register, containing information about motion system status. Some of its bits, when set, will determine specific actions in the real-time control kernel.

TML Address: 0x0308

Contents. MSR information is structured as follows:

15	14	13	12	11	10	9	8
UPDATE	**EVNRS**	**AXISST**	**DEBUG**	**EVNS**	**CNTSGS**	**MOTS**	**PCAPS**
0	0	0	0	0	0	1	0

7	6	5	4	3	2	1	0
LSWNS	**LSWPS**	**WRPS**	**Reserved**	**CTRERS**	**SWPRS**	**Reserved**	**DSLBS**
0	0	0	0	0	0	0	0

Bit 15 UPDATE
- 0 = No update
- 1 = Update

Bit 14 EVNRS. Event status
- 0 = Reset after update
- 1 = Set for update

Bit 13 AXISST. Axis status
- 0 = Axis off
- 1 = Axis on

Bit 12 DEBUG
- 0 = Debug off
- 1 = Debug on

Bit 11 EVNS. Events
- 0 = No program event, or last event
- 1 = Last program event reached

Bit 10 CNTSGS. Contour segment
- 0 = Don't update
- 1 = Update

Bit 9 MOTS. Motion status

0 = In motion
1 = Motion complete

Bit 8 PCAPS. Position capture

0 = Not triggered
1 = Triggered

Bit 7 LSWNS. Negative limit switch

0 = Not triggered
1 = Triggered

Bit 6 LSWPS. Positive limit switch

0 = Not triggered
1 = Triggered

Bit 5 WRPS. Position wrap around

0 = Not triggered
1 = Triggered

Bit 4 Reserved.

Bit 3 CTRERS. Control error

0 = Not triggered
1 = Triggered

Bit 2 SWPRS. Software protections

0 = Not triggered
1 = Triggered

Bit 1 Reserved.

Bit 0 DSLBS. Motion initialization status

0 = Not initialized
1 = Initialized

Setting: This register is initialized during the default configuration phase of the motion system. The MSR register is internally set during the operation of the motion system. Its information is internally used and dynamically updated by the reference generator and the real time control

module. The user **MAY NOT CHANGE** its contents. Also, the internal use of this register does not recommend its use/testing in TML program sequences. Most of its bits have corresponding interrupt flags in the ISR register, which can be tested by polling, or used to generate TML interrupts, corresponding to the associated events.

No TML instructions may be used in order to set this register!

Use local or on-line TML commands in order to read MSR register contents and get the motion status information, at any time during system operation.

Example. For example, if MSR content is read as **0x0A01**, it means that the system is in a motion complete, motion initialized, and last program event reached.

4.2. TML Parameters

TML programs use some specific data information structured through the TML parameters. Having reserved TML mnemonics, the TML parameters are used in order to setup and/or examine the values of specific parameters of the different motion system components.

Some of the TML parameters share the same memory address. They are used in application configurations that exclude each other, and thus are not needed at the same time.

Some of the TML parameters must be initialized before activating the motion system real time structure (before the `ENDINIT` TML command), or after activating the motion control functions (after the execution of the `AXISON` TML command). The others need to be initialized only before being used in the TML environment, and can be changed at any time if needed.

This paragraph will present, in detail, all the TML parameters. The information is grouped as follows:

- Address – the address in data memory (dm) where the parameter is.
- Numerical type – the parameter can be integer 16 bits, unsigned integer 16 bits, long 32 bits, unsigned long 32 bits or fixed point 32 bits: (16 bits signed integer part).(16 bits fractional part).
- Default value – after RESET the parameters are set to a default value. This will be given in decimal and hexadecimal form in MotionChip II internal units (IU).
- Description – a short description of the parameters will be presented.
- Scaling factor – where is the case the scaling factor will be given for you to understand how the value in IU has been obtained from the physical value (SI).
- Setting – present how the parameter must be used correctly.
- Example – where is the case a short example is given for an easier understanding of the meaning and the range of the parameter's values.
- Remarks – if is necessary some remarks will be also given.

Parameters descriptions may contain specific notation; their significance is presented in Table 4.1.

Table 4.1 *Significance of notations used in parameters descriptions*

Name	Description
CGa	Current sensor gain factor
DeadBand	Inverter dead band time
OffAt0oC	Drive temperature sensor output at 0°C
TGa	Drive temperature sensor gain
DSP_clock	DSP clock frequency
ERR_position	Maximum accepted error for position loop
ERR_speed	Maximum accepted error for speed loop
I_Max	Maximal output current
In	Nominal motor current
No_encoder_lines	Encoder lines
pp	Motor pole pairs
PWM_freq	PWM frequency
spd_max	Controller maximum speed output
start_ref_inc_ratio	Increment per sampling, for current reference
Temp_1_PROT	Protection temperature 1
Temp_2_PROT	Protection temperature 2
time_ERR_pos	Time interval for accepted error of position
time_ERR_spd	Time interval for accepted error of speed
time_IMAX_PROT	Imax sampling time
Ts_C	Current loop sampling time
Ts_S	Speed loop sampling time
u_d_max	Maximum voltage outputs
u_q_max	Maximum voltage outputs
UMAX_PROT	Protection maximum voltage
UMIN_PROT	Protection minimum voltage
V_DC	DC voltage measured on AD4

The following table presents the TML parameters, grouped by functionality.

1. Real-time kernel

Parameter Name	Data Type	Significance	Address	Par.
CLPER	**uint**	**Current loop sampling time**	**0x0250**	**4.2.1.1**
SLPER	**uint**	**Speed loop sampling time**	**0x0251**	**4.2.1.2**

2. Motors

Parameter Name	Data Type	Significance	Address	Par.
T1ONA	uint	Time on A phase (PMSM motor)	0x0284	4.2.2.1
T1ONB	uint	Time on B phase (PMSM motor)	0x0285	4.2.2.2

3. Sensors

Parameter Name	Data Type	Significance	Address	Par.
ALPHA	long	Threshold value for detection of linear Hall signal missing	0x0295	4.2.3.1
AD0OFF	uint	Offset for AD0 channel	0x0244	4.2.3.2
AD1OFF	uint	Offset for AD1 channel	0x0245	4.2.3.2
AD2OFF	uint	Offset for AD2 channel	0x0246	4.2.3.2
AD3OFF	uint	Offset for 2nd linear Hall	0x0247	4.2.3.3
AD4OFF	uint	Offset for AD4 channel	0x0248	4.2.3.2
AD5OFF	uint	Offset for AD5 channel	0x0249	4.2.3.2
AD6OFF	uint	Offset for 1st linear Hall	0x024A	4.2.3.4
AD7OFF	uint	Offset for AD7 channel	0x024B	4.2.3.2
CADIN	int	Analog reference scaling factor	0x025C	4.2.3.5
ENC2THL	long	Scaling constant for electrical angle computation	0x024C	4.2.3.6
FILTER1	int	Coefficient for the first order filter for analogue reference	0x029D	4.2.3.7
HALLCASE	int	Hall sensors configuration parameter	0x0259	4.2.3.8
KSLIP	int	Gain of the 3rd linear Hall		
MASTERRES	long	Master position sensor resolution	0x081A	4.2.3.9
MECRESL	long	Position sensor increments / mechanical revolution	0x024E	4.2.3.10
MTSTYPE	uint	Motor temperature sensor type	0x028C	4.2.3.11
PHASEADV	int	Phase advance factor	0x0257	4.2.3.12
SFTADIN	int	Shift for analog reference scaling factor	0x025D	4.2.3.5
SFTCRT	int	Current measurement shift factor	0x0290	4.2.3.13
SFTCWEAK	int	Offset of the 3rd linear Hall	0x0258	4.2.3.14
SIN2REC	int	Gain of the 2nd linear Hall	0x0287	
SRECTCOMP	Int	Gain of the 1st linear Hall	0x028F	
VDCN	uint	DC bus voltage nominal value (V_dc compensation)	0x025A	4.2.3.15

4. Power Converters

Parameter Name	Data Type	Significance	Address	Par.

BRAKELIM	**uint**	**Brake activation level**	**0x028A**	**4.2.4.1**
DBT	**uint**	**Power stage dead-time**	**0x0253**	**4.2.4.2**
PWMPER	**uint**	**PWM period**	**0x0252**	**4.2.4.3**
SATPWM	**int**	**Saturation limit for voltage command**	**0x0254**	**4.2.4.4**

5. Controllers

Parameter Name	Data Type	Significance	Address	Par.
IMAXP	**int**	**Saturation limit for integral part for position controller**	**0x0266**	**4.2.5.6**
IMAXS	**int**	**Saturation limit for integral part for speed controller**	**0x026C**	**4.2.5.7**
KDFP	**int**	**Derivative term filter coefficient for position controller**	**0x0264**	**4.2.5.1**
KDP	**int**	**Derivative term coefficient for position controller**	**0x0262**	**4.2.5.1**
KFFA	**int**	**Acceleration feedforward factor**	**0x026E**	**4.2.5.2**
KFFL	**int**	**Load torque feedforward factor**	**0x026F**	**4.2.5.3**
KFFS	**int**	**Speed feedforward factor**	**0x026D**	**4.2.5.4**
KII	**int**	**Integrative term coefficient for currents controllers**	**0x0273**	**4.2.5.5**
KIP	**int**	**Integrative term coefficient for position controller**	**0x0260**	**4.2.5.6**
KIS	**int**	**Integrative term coefficient for speed controller**	**0x0269**	**4.2.5.7**
KPI	**int**	**Proportional term coefficient for currents controllers**	**0x0271**	**4.2.5.8**
KPP	**int**	**Proportional term coefficient for position controller**	**0x025E**	**4.2.5.9**
KPS	**int**	**Proportional term coefficient for speed controller**	**0x0267**	**4.2.5.10**
POSOKLIM	**uint**	**Accepted position error limit to suspend control**	**0x036A**	**4.2.5.11**
SATID	**int**	**Saturation limit for ud command**	**0x0275**	**4.2.5.12**
SATIQ	**Int**	**Saturation limit for uq command**	**0x0276**	**4.2.5.13**
SATP	**int**	**Saturation limit for speed command**	**0x0265**	**4.2.5.14**
SATS	**int**	**Saturation limit for current command**	**0x026B**	**4.2.5.15**
SFTAFFW	**int**	**Shift feedforward acceleration**	**0x0291**	**4.2.5.2**
SFTKDP	**int**	**Derivative term shift for position controller**	**0x0263**	**4.2.5.1**
SFTKFF	**int**	**Feedforward shift**	**0x0270**	**4.2.5.16**
SFTKII	**int**	**Integrative term shift for currents controllers**	**0x0274**	**4.2.5.5**
SFTKIP	**int**	**Integrative term shift for position controller**	**0x0261**	**4.2.5.6**
SFTKIS	**int**	**Integrative term shift for speed controller**	**0x026A**	**4.2.5.7**
SFTKPI	**int**	**Proportional term shift for currents controllers**	**0x0272**	**4.2.5.8**
SFTKPP	**int**	**Proportional term shift for position controller**	**0x025F**	**4.2.5.9**
SFTKPS	**int**	**Proportional term shift for speed controller**	**0x0268**	**4.2.5.10**
SFTSFFW	**int**	**Shift feedforward speed**	**0x0292**	**4.2.5.4**

TONPOSOK	**uint**	**Position trigger time to suspend control**	**0x036B**	**4.2.5.17**

6. Reference generator

Parameter Name	Data Type	Significance	Address	Par.
CACC	**fixed**	**Acceleration command for position/speed profile modes**	**0x02A2**	**4.2.6.1**
CAMOFF	**long**	**CAM offset**	**0x03AD**	**4.2.6.2**
CAMSTART	**int**	**RAM address where is CAM table copied**	**0x03AC**	**4.2.6.3**
CPOS	**long**	**Position command for position profile mode**	**0x029E**	**4.2.6.4**
CSPD	**fixed**	**Speed command for position/speed profile modes**	**0x02A0**	**4.2.6.5**
EFLEVEL	**Int**	**Activate/deactivate synchronization with master**	**0x02C7**	**4.2.6.6**
EREF	**long/ fixed**	**External reference for all external modes**	**0x02A8**	**4.2.6.7**
GEAR	**fixed**	**Gearing factor for slave axis**	**0x02AC**	**4.2.6.8**
GEARMASTER	**int**	**Denominator of gear ratio**	**0x0255**	**4.2.6.8**
GEARSLAVE	**int**	**Nominator of gear ratio**	**0x0256**	**4.2.6.8**
MPOS0	**long**	**Initial master reference**	**0x02E5**	**4.2.6.9**
POS0	**long**	**Origin of relative position for event tests**	**0x02B8**	**4.2.6.10**
REF0	**long/ fixed**	**Initial reference in torque/voltage test mode**	**0x02A8**	**4.2.6.11**
REFTST	**int**	**Test reference saturation value**	**0x0281**	**4.2.6.12**
RINCTST	**int**	**Test reference increment value**	**0x0280**	**4.2.6.13**
SLAVEID	**int**	**ID of slave axes referenced from a master axis**	**0x0311**	**4.2.6.14**
THTST	**int**	**Test reference electric angle**	**0x0282**	**4.2.6.15**
TIME0	**long**	**Origin of relative time for event tests**	**0x02BE**	**4.2.6.16**
TINCTST	**int**	**Test reference electric angle increment**	**0x0283**	**4.2.6.17**

7. Protections

Parameter Name	Data Type	Significance	Address	Par.
ERRMAX	**uint**	**Maximum accepted control error in the outer loop**	**0x02C5**	**4.2.7.1**
I2TINTLIM	**ulong**	**I^2t protection integrator limit**	**0x0815**	**4.2.7.2**
II2TPROT	**int**	**I^2t protection current limit**	**0x0814**	**4.2.7.3**
IMAXPROT	**int**	**Maximum current – protection limit**	**0x0297**	**4.2.7.5**
NI2T	**long**	**I^2t computation interval**	**0x0255**	**4.2.7.6**
SFI2T	**int**	**I^2t protection scaling factor**	**0x0819**	**4.2.7.4**
T1MAXPROT	**uint**	**Temp1. maximum – protection value**	**0x0298**	**4.2.7.7**
T2MAXPROT	**uint**	**Temp2. maximum – protection value**	**0x0299**	**4.2.7.8**
TERRMAX	**uint**	**Control error protection trigger time**	**0x02C6**	**4.2.7.9**
TIMAXPROT	**uint**	**Maximum current protection trigger time**	**0x02C4**	**4.2.7.10**
UMAXPROT	**uint**	**DC bus voltage maximum – protection value**	**0x029A**	**4.2.7.11**
UMINPROT	**uint**	**DC bus voltage minimum – protection value**	**0x029B**	**4.2.7.12**

Parameter Name	Data Type	Significance	Address	Par.
INTTABLE	int	Start address of TML interrupts vector table	0x0307	4.2.8.1

4.2.1. Real-time kernel related TML parameters

4.2.1.1. CLPER. Current loop sampling time

ADDRESS: 0x0250

NUMERICAL TYPE: unsigned integer

DEFAULT VALUE: 2 (0x0002)

DESCRIPTION: The CLPER parameter is used for the setting of the fast control loop (current) sampling period.

SETTING: CLPER depends on the PWM frequency and on the frequency of the fast control loop (current/field control loop). See par. 3.1 for details about the structure of the real-time motion kernel operation. CLPER is expressed as a multiple of PWM periods, and is computed using the next relation:

$$\mathbf{CLPER} = \boldsymbol{PWM\_freq} \cdot \boldsymbol{Ts\_C}$$

EXAMPLE:

If: PWM frequency = 20000 [Hz], *Ts_C* = 0.00001 [s]

Then: CLPER = 20000 · 0.00001 = 2

REMARKS: When choosing a fast loop frequency, remember that the minimum value of parameter CLPER is 2.

4.2.1.2. SLPER. Speed loop sampling time

ADDRESS: 0x0251

NUMERICAL TYPE: unsigned integer

DEFAULT VALUE: 20 (0x0014)

DESCRIPTION: The SLPER parameter is used for the setting of the slow control loop (speed/position) sampling period.

SETTING: SLPER depends on the PWM frequency and on the frequency of the slow control loop (speed / position control loop). See par. 3.1 for details about the structure of the real-time motion kernel operation. SLPER is expressed as a multiple of PWM periods, and is computed using the following relation:

$$\mathbf{SLPER} = \boldsymbol{PWM\_freq} \cdot \boldsymbol{Ts\_S}$$

EXAMPLE:

If: *PWM_freq* = 20000 [Hz], *Ts_S* = 0.0001 [s]

Then: SLPER = 20000 · 0.0001 = 20

4.2.2. Motors related TML parameters

4.2.2.1. T1ONA. Time for wait on A phase (PMSM motor)

ADDRESS: 0x0284

NUMERICAL TYPE: unsigned integer

DEFAULT VALUE: - 1 (0xFFFF)

DESCRIPTION: The T1ONA parameter controls the phase A supply time, at the start of a PMSM motor, using initial positioning through supply of phases A and B of the motor, with a controlled DC voltage or current.

SETTING: T1ONA is related to the time needed to the motor to move when supplying phase A with a controlled DC voltage or current. See par. 3.2 for details and related parameters associated to the specific starting procedures of the PMSM motor.

In order to setup this parameter, a possible method is to use its maximal value (65535) - corresponding to a delay of approximate one second, and to reduce it as long as the motor start is performed correctly.

When the starting procedure is finished the value of T10NA is halved from the set value.

4.2.2.2. T1ONB. Time for wait on B phase (PMSM motor)

ADDRESS: 0x0285

NUMERICAL TYPE: unsigned integer

DEFAULT VALUE: -1 (0xFFFF)

DESCRIPTION: The T1ONB parameter controls the phase B supply time, at the start of a PMSM motor, using initial positioning through supply of phases A and B of the motor, with a controlled DC voltage or current.

SETTING: T1ONB is related to the time needed to the motor to move when supplying phase B with a controlled DC voltage or current. See par. 3.2 for details and related parameters associated to the specific starting procedures of the PMSM motor.

In order to setup this parameter, a possible method is to use its maximal value (65535) - corresponding to a delay of approximate 1 second, and to reduce it as long as the motor start is performed correctly.

When the starting procedure is finished the value of T10NB is halved from the set value.

4.2.3. Sensors related TML parameters

4.2.3.1. ALPHA(L). Threshold value for detection of linear Hall signal missing

ADDRESS: 0x0295

NUMERICAL TYPE: uint

DEFAULT VALUE: 0 (0x0000)

DESCRIPTION: The ALPHA(L) parameter sets the threshold value for the linear Hall signal missing detection. If the linear Hall signals results in a value below then ALPHA(L) the **Position Wraparound** interrupt is triggered and the PWM outputs are disabled.

SETTING: The parameter ALPHA(L) is set with the threshold value read from the EEPROM memory. The threshold value is automatically determined and saved in the EEPROM by the linear Hall gains and offsets detection test.

4.2.3.2. AD0OFF, AD1OFF, AD2OFF, AD4OFF, AD5OFF, AD7OFF. Offset for AD0, AD1, AD2, AD4, AD5, AD7, channels

ADDRESS: AD0OFF: 0x0244; AD1OFF: 0x0245; AD2OFF: 0x0246; AD4OFF: 0x0248; AD5OFF: 0x0249; AD7OFF: 0x024B;

NUMERICAL TYPE: unsigned integer

DEFAULT VALUE: 32736 (0x7FE0)

DESCRIPTION: ADxOFF parameters (x=0 to 7) represent the offset used to compute measure variables ADx (x=0 to 7), from the conversion data given by the analog-to-digital converters. See par. 3.4.1 for details on A/D conversion scheme implementation and parameters.

SETTING: As the ADx measure result falls within the range [0, 0xFFC0] (10bits, left aligned), the ADxOFF parameter enables the “polarization” of this range as unipolar or bipolar values.

ADxOFF can be used as follows:

Case 1: to define a positive unipolar range for ADx measure in the range [0, X_{max}]: set AD2OFF = 0;

Case 2: to define a positive bipolar range for ADx measure in the range [-X_{max}, X_{max}]: set AD2OFF =0x7FE0

Case 3: to define a negative unipolar range for ADx measure in the range [-X_{max}, 0]: set AD2OFF = 0xFFC0;

4.2.3.3. AD3OFF, SIN2REC. Offset and gain for 2nd linear Hall

ADDRESS: AD3OFF: 0x0247; SIN2REC: 0x0290

NUMERICAL TYPE: AD3OFF: unsigned integer; SIN2REC: integer

DEFAULT VALUE: AD3OFF: 32736 (0x7FE0); SIN2REC: 500 (0x01F4)

DESCRIPTION: AD3OFF and SIN2REC parameters represent the offset and the gain factor used by MotionChip II to interpret the values read from the second linear Hall. See par. 3.4.1 for details on A/D conversion scheme implementation and parameters.

SETTING: The AD3OFF and SIN2REC are set with offset and gain values read from the EEPROM. The offset and gain values are automatically determined and saved in the EEPROM by the linear Hall gains and offsets detection test.

4.2.3.4. AD6OFF, SRETCOMP. Offset and gain for 1st linear Hall

ADDRESS: AD6OFF: 0x0247; SRETCOMP: 0x028F

NUMERICAL TYPE: AD6OFF: unsigned integer; SRETCOMP: integer

DEFAULT VALUE: AD6OFF: 32736 (0x7FE0); SRETCOMP: 33 (0x0021)

DESCRIPTION: AD6OFF and SRETCOMP parameters represent the offset and the gain factor used by MotionChip II to interpret the values read from the first linear Hall. See par. 3.4.1 for details on A/D conversion scheme implementation and parameters.

SETTING: The AD6OFF and SRETCOMP are set with offset and gain values read from the EEPROM. The offset and gain values are automatically determined and saved in the EEPROM by the linear Hall gains and offsets detection test.

4.2.3.5. CADIN, SFTADIN. Analogue reference scaling factor and shift

ADDRESS: CADIN: 0x025C; SFTADIN: 0x025D

NUMERICAL TYPE: integer

DEFAULT VALUE: CADIN =32767(0x7FFF) SFTADIN = 0 (0x0000)

DESCRIPTION: CADIN parameter is used to scale the analog reference measured from the conversion data given by the analog-to-digital converter.

SETTING: CADIN and SFTADIN depend on the peak value of the desired range and on the interpretation of the ADC input. The CADIN and SFTADIN parameters enable the “polarization” of ADC input range as unipolar or bipolar values.

These parameters can be used as follows:

Case 1: to define a positive unipolar range for analog reference measure in the range $[0,X_{max}]$;

Case 2: to define a positive bipolar range for analog reference measure in the range $[X_{max}, X_{max}]$;

Case 3: to define a negative unipolar range for analog reference measure in the range $[-X_{max}, 0]$;

CADIN and SFTADIN parameters are computed as follows:

Case 1: CADIN_fixed = X_{max} * 32767 / 32736 (computed as a fixed)

Case 2: CADIN_fixed = X_{max} * 32767 / 16368 (computed as a fixed)

Case 3: CADIN_fixed = X_{max} * 32767 / 32736 (computed as a fixed)

If **CADIN_fixed** < **32767**

SFTADIN = 0 and CADIN = CADIN_fixed.

If **CADIN_fixed** > **32767**

SFTADIN = the least integer power of 2 for which **CADIN_fixed <2^SFTADIN * 32767**

CADIN = CADIN_fixed / 2^SFTADIN (computed as an integer)

EXAMPLE:

Case 1: [0, X_{max}] = [0, 100000] => CADIN_fixed = 100000 * 32767 / 32736 = 100094.69

SFTADIN = 2 : 100094.69 < 2^2 * 32767 = 131068

CADIN = 100094.69 / 4 = 25024

4.2.3.6. ENC2THL. Scaling constant for electrical angle computation

ADDRESS: 0x024C

NUMERICAL TYPE: long

DEFAULT VALUE: 0 (0x00000000)

DESCRIPTION: ENC2THL parameter is used to scale the measured position information, in order to obtain the electrical angle (theta).

SETTING: ENC2THL depends on the number of encoder lines and pole pairs. ENC2THL is set as follows:

ENC2THL = 2^32 * *pp* / (4*No_encoder_lines*)

EXAMPLE:

If: *No_encoder_lines* = 500; *pp* = 2

Then: ENC2THL = 2^32 * 2 /(4* 500) = 4980792

4.2.3.7. FILTER1. Coefficient for the first order filter for analogue reference

ADDRESS: 0x029D

NUMERICAL TYPE: integer

DEFAULT VALUE: 32767 (0x7FFF)

DESCRIPTION: FILTER1 parameter represents the 1st order filter coefficient used to filter the reference signal, when measured from an A/D converter input. See MotionChip II TML Programming User Manual details about analogue references.

SETTING: FILTER1 is used to implement a first order filter on the measured A/D reference value. For the maximum value (0x7FFF), the measured value is not filtered. Complete attenuation is obtained for value 0.

The filtering relation for an input x, output of the filter y, is:

$$y_n = y_{n-1} + (x_n - y_{n-1}) * \text{FILTER1}$$

Depending on the measurement environment (noise level), adjust the value of FILTER1, starting from its maximal value, until accurate measurements are obtained.

4.2.3.8. HALLCASE. Hall sensors configuration parameter

ADDRESS: 0x0259

NUMERICAL TYPE: integer

DEFAULT VALUE: 32767 (0x7FFF)

DESCRIPTION: HALLCASE parameter is used to set the Hall sensor configuration mode.

SETTING: HALLCASE depends on the Hall sensor configuration, as related to the motor phases. See par. 3.4.4 for details about Hall sensors and configuration selection method.

4.2.3.9. MASTERRES. Master position sensor resolution

ADDRESS: 0x081A

NUMERICAL TYPE: long

DEFAULT VALUE: 0 (0x00000000)

DESCRIPTION: MASTERRES parameter indicates the master position sensor mechanical resolution for one motor rotation. The slaves need the master resolution to compute correctly the master position and speed (i.e. position increment).

SETTING: MASTERRES depends on the position sensor type.

For sensors connected to the QEP interface (quadrature encoders), MASTERRES is set as follows:

MASTERRES = 4 * *No_encoder_lines*

For other position sensors (not connected to the QEP interface), MASTERRES is set as follows:

MASTRERRES = No_sensor_bits/ rotation

EXAMPLE:

If: For an encoder having *No_encoder_lines* = 1000

Then: MASTERRES = 4 * 1000 = 4000

Remark: *The master resolution is a 32-bit long integer value. If master position is not cyclic (i.e. the resolution is equal with the whole 32-bit range of position), set master resolution to 0x80000001. When this value is used, no modulo operation is performed on the position counted from the 2nd encoder inputs.*

4.2.3.10. MECRESL. Position sensor increments per mechanical revolution

ADDRESS: 0x024E

NUMERICAL TYPE: long

DEFAULT VALUE: 7995494 (0x007A0066)

DESCRIPTION: MECRESL parameter is used to indicate the position sensor mechanical resolution for one motor rotation.

SETTING: MECRESL depends on the position sensor type.

For sensors connected to the QEP interface (quadrature encoders), MECRESL is set as follows:

MECRESL = 4 * *No_encoder_lines*

For other position sensors (not connected to the QEP interface), MECRESL is set as follows:

MECRESL = (*No_sensor_bits*/ rotation)

EXAMPLE:

If: For an encoder having *No_encoder_lines*= 500

Then: MECRESL = 4 * 500 = 2000

4.2.3.11. MTSTYPE. Motor Temperature Sensor Type

ADDRESS: 0x028C

NUMERICAL TYPE: integer

DEFAULT VALUE: 0 (0x0000)

DESCRIPTION: MTSTYPE is used to indicate the type of the motor temperature sensor, PTC or NTC.

SETTING: For sensors with positive temperature coefficient (PTC) MTSTYPE = 1 and for sensors with negative temperature coefficient (NTC) MTSTYPE = 0.

4.2.3.12. PHASEADV. Phase advance factor

ADDRESS: 0x0257

NUMERICAL TYPE: integer

DEFAULT VALUE: 0x1FFF

DESCRIPTION: PHASEADV it is used in field position computation for high speed PMSM applications, to eliminate measurement and computation delays.

SETTING: PHASEADV must be set before AXISON instruction.

4.2.3.13. SFTCRT. Current measurement shift factor

ADDRESS: 0x0290

NUMERICAL TYPE: integer

DEFAULT VALUE: 0 (0x0000)

DESCRIPTION: SFTCRT parameter is used to change the current scaling factor, and thus to increase the overall representation accuracy for speed controller coefficients.

SETTING: SFTCRT parameter can be set between 0 and 6. Shifting of current values does not generate truncation errors, due to the limited, 10 bits accuracy, of the measured motor currents. On the other side, the maximum shift of a controller coefficient is 12. Thus, for values of controller coefficients bigger then 4096, one will be forced to use a truncated value for their representation.

But the speed controller coefficients depend also on the current scaling factor. Then, if the current scaling factor can be modified through the use of a supplementary scaling factor (SFTCRT parameter), which do not alter the computational accuracy, an increased range can be used for speed controller coefficients, preserving the same overall mathematical accuracy.

For details about controllers' models and parameters, see par. 3.5.

EXAMPLE:

Suppose the proportional speed controller coefficient value after tuning is: *Kp_spd* = 5000

Implementing this coefficient implies that SFTKPS = 13 : 2^13 > 5000 (13 is the least integer power of 2 for which KPS < 2^SFTKPS). In this case, a limitation of the *Kp_spd* coefficient to 2^12 = 4096 is imposed, leading to a deviated controller.

The problem can be solved using the SFTCRT parameter through which the measured current value is reduced (divided by 2^SFTCRT). This leads to a reduction of the current scaling factor and, implicitly, to a reduction of the speed controller coefficients (which are proportional with the current scaling factor).

4.2.3.14. SFTCWEAK, KSLIP. Offset and gain for 3rd linear Hall

ADDRESS: SFTCWEAK: 0x0290; KSLIP: 0x0279

NUMERICAL TYPE: SFTCWEAK: integer; KSLIP: integer;

DEFAULT VALUE: SFTCWEAK: 0 (0x0000); KSLIP: 15 (0x000F)

DESCRIPTION: SFTCWEAK and KSLIP parameters represent the offset and the gain factor used by MotionChip II to interpret the values read from the third linear Hall. See par. 3.4.1 for details on A/D conversion scheme implementation and parameters.

SETTING: The SFTCWEAK and KSLIP are set with offset and gain values read from the EEPROM. The offset and gain values are automatically determined and saved in the EEPROM by the linear Hall gains and offsets detection test.

4.2.3.15. VDCN. DC bus voltage nominal value (Udc compensation)

ADDRESS: 0x025A

NUMERICAL TYPE: unsigned integer

DEFAULT VALUE: 46000 (0XB3B)

DESCRIPTION: VDCN parameter is used to implement the compensation of DC-link/supply voltage (Vdc) variation by changing the PWM command. Its value represents the nominal Vdc value. See also par. 4.3.3.1 for related variable CVDC description.

SCALING FACTOR: *Kuf_m*. For more details about scaling factors see par. 3.7.

SETTING: VDCN depends on the voltage scaling factor and on the rated DC bus voltage values:

VDCN = *Vdc * Kuf_m*

EXAMPLE:

If: *Kuf_m* = 65472[bits]/426[V]; Vdc = 325 [V]

Then: VDCN = 325 * 153.69 = 49949

4.2.4. Power converter related TML parameters

4.2.4.1. BRAKELIM. Brake activation level

ADDRESS: 0x028A

NUMERICAL TYPE: unsigned integer

DEFAULT VALUE: -1 (0xFFFF)

DESCRIPTION: BRAKELIM parameter defines the DC voltage value for which the brake becomes active, when the braking is activated.

SCALING FACTOR: *Kuf_m*. For more details about scaling factors see par. 3.7.

SETTING: BRAKELIM must be set if the breaking mode will be activated in the OSR register. Its value is compared with the DC voltage value and the braking is activated once the DC voltage is bigger than the BRAKELIM value. If the DC voltage becomes smaller than BRAKELIM, the braking is deactivated. See par. 3.3 about brake command details.

4.2.4.2. DBT. Power stage dead-time

ADDRESS: 0x0253

NUMERICAL TYPE: unsigned integer

DEFAULT VALUE: 20 (0x014)

DESCRIPTION: DBT parameter is used to set the dead band time for the PWM output commands. See par. 3.3 for details regarding the power converters PWM command implementation aspects.

SETTING: DBT depends on the dead band time of the power converter devices. Possible values for DBT parameter are presented in Table 3.2.

4.2.4.3. PWMPER. PWM period

ADDRESS: 0x0252

NUMERICAL TYPE: unsigned integer

DEFAULT VALUE: 500

DESCRIPTION: PWMPER parameter is used to set the value of the PWM period. See par. 3.3 for details regarding the power converters PWM command implementation aspects.

SETTING: PWMPER depends on the values of the PWM frequency and of the DSP clock. It is computed as follows:

PWMPER = *DSP_clock* / (2PWM_freq*)**

EXAMPLE:

If: *PWM_freq* = 20000 [Hz]; *DSP_clock* = 40 MHz [ns]

Then: PWMPER = 40e6/ (2*20000) = 1000

4.2.4.4. SATPWM. Saturation limit for voltage command

ADDRESS: 0x0254

NUMERICAL TYPE: integer

DEFAULT VALUE: 2620 (0x0A3C)

DESCRIPTION: SATPWM parameter is used to limit the maximum value of the reference voltage command. The reference voltage saturation is required for accurate current measurement.

SETTING: SATPWM depends on the PWM frequency and on the saturation range. It is computed as follows:

SATPWM = 65535 * (saturation range) * (*PWM_freq*) * 2

EXAMPLE:

If: *PWM_freq* = 20000 [Hz]; Saturation range = 1 [μs] (recommended)

Then: SATPWM = 65535 * 1e-6 * 20000 * 2 = 2621

4.2.5. Controllers related TML parameters

4.2.5.1. KDP, SFTKDP, KDFP. Derivative term coefficients for position controller

KDP: derivative factor for position controller

SFTKDP: derivative factor shift for position controller

KDFP: filter derivative factor for position controller

ADDRESS: KDP: 0x0262; SFTKDP: 0x0263; KDFP: 0x0264

NUMERICAL TYPE: integer

DEFAULT VALUE: KDP = 0 (0x0000); SFTKDP = 0 (0x0000); KDFP = 0 (0x0000)

DESCRIPTION: KDP, SFTKDP, KDFP parameters are used to implement the derivative component of the PID position controller.

SETTING: KDP, SFTKDP, KDFP depend on the derivative component of the PID position controller coefficients, obtained from the tuning, on the controller maximum speed output and on the speed scaling factor. See par. 3.5.3 for details about position controller model and parameters. KDP, SFTKDP, KDFP are computed as follows:

SFTKDP = the least integer power of 2 for which **KD_P_scl < 2^SFTKDP**

KDP = Kd_P_scl * 32767 / 2^SFTKDP

KDFP = *filter_D32767**

EXAMPLE:

If: *Kd_P_scl* = 1.3e-2

Then: SFTKDP = 0 : 2^0 = 1 > 1.3e-2

KDP = 1.3e-2 * 32767 / 2^0 = 426

REMARKS: The maximum shift value is 12, which implies that the maximum value of controller coefficient KDP is 4096.

4.2.5.2. KFFA, SFTAFFW. Acceleration feedforward factor and shift for current controller

KFFA: acceleration feedforward factor for speed controller

SFTAFFW: acceleration feedforward shift factor for speed controller

ADDRESS: KFFA: 0x026E; SFTAFFW: 0x0291

NUMERICAL TYPE: integer

DEFAULT VALUE: KFFA = 0 (0x0000); SFTAFFW = 0 (0x0000)

DESCRIPTION: KFFA and SFTAFFW parameters are used to implement the acceleration feedforward term in the acceleration & load feedforward block. They are used in the multiplication with the target acceleration issued by the reference generator module. The obtained term is added at the output of the speed controller, as an acceleration feedforward term.

SETTING: KFFA represents the multiplication coefficient. The SFTAFFW parameter represents the shift used to extract the acceleration feedforward component from the target acceleration. See par. 3.5.4.2 for details about acceleration & load feedforward block model and parameters.

These parameters depend on the system model and parameters. Theoretically, they can be computed from these parameters (mainly system inertia). Practically, try-and-error methods can be used to “tune” optimal values for these parameters. Use as start values the default ones (0).

4.2.5.3. KFFL. Load torque feedforward factor for current controller

ADDRESS: 0x026F

NUMERICAL TYPE: integer

DEFAULT VALUE: 0 (0x0000)

DESCRIPTION: KFFL parameter is used to implement the load torque feedforward term in the acceleration & load feedforward block. It is used in the multiplication with the measured load torque value. The obtained term is added at the output of the speed controller, as a load torque feedforward term.

SETTING: KFFL represents the multiplication coefficient of the load torque value. See par. 3.5.4.2 for details about speed controller model and parameters.

This parameter depends on the system model and parameters. Theoretically, it can be computed from these parameters. Practically, try-and-error methods can be used to “tune” optimal values for this parameter. Use as start value the default one (0).

4.2.5.4. KFFS, SFTSFFW. Speed feedforward factor and shift for speed controller

KFFS: speed feedforward factor for speed controller

SFTSFFW: speed feedforward shift factor for speed controller

ADDRESS: KFFS: 0x026D; SFTSFFW: 0x0292

NUMERICAL TYPE: integer

DEFAULT VALUE: KFFS = 0 (0x0000); SFTSFFW = 0 (0x0000)

DESCRIPTION: KFFS and SFTSFFW parameters are used to implement the speed feedforward term added at the input of the PI speed controller. They are used in the multiplication with the target speed issued by the reference generator module. The obtained term is added at the output of the speed controller, as a speed feedforward term.

SETTING: KFFS represents the multiplication coefficient. The SFTSFFW parameter represents the shift used to extract the speed feedforward component from the target speed. See par. 3.5.4.1 for details about speed feedforward block model and parameters.

These parameters depend on the system model and parameters. Theoretically, they can be computed from these parameters (mainly system viscous friction). Practically, try-and-error methods can be used to "tune" optimal values for these parameters. Use as start values the default ones (0).

4.2.5.5. KII, SFTKII. Integrative term coefficients for current controllers

KII: integrative factor for current controller

SFTKII: integrative factor shift for current controller

ADDRESS: KII: 0x0273; SFTKII: 0x0274

NUMERICAL TYPE: integer

DEFAULT VALUE: KII = 0 (0x0000); SFTKII =0 (0x0000)

DESCRIPTION: KII and SFTKII parameters are used to implement the integral component of the PI current controllers.

SETTING: KII and SFTKII depend on the integral component of the PI current controller coefficients, obtained from the tuning and on the controller maximum voltage outputs (*u_d_max*, *u_q_max*). See par. 3.5.1 for details about current controller's model and parameters. KII and SFTKII are computed as follows:

SFTKII = the least integer power of 2 for which **KII < 2^SFTKII**

KII = Ki_C_scl * 32767 / 2^SFTKII

EXAMPLE:

If: *Ki_C_scl* = 0.35

Then: SFTKII = 0 : 2^0 = 1 > 0.35

KII = 0.35 * 32767 / 1 = 11468

REMARK: The maximum shift value is 12, which implies that the maximum value of controller coefficient KII is 4096.

4.2.5.6. KIP, SFTKIP, IMAXP. Integrative term coefficients for position controller

KIP: integrative factor for position controller

SFTKIP: integrative factor shift for position controller

IMAXP: integrative part saturation limit for position controller

ADDRESS: KIP: 0x0260; SFTKIP: 0x0261; IMAXP: 0x0266

NUMERICAL TYPE: integer

DEFAULT VALUE: KIP = 0 (0x0000); SFTKIP = 0 (0x0000); IMAXP =0 (0x0000)

DESCRIPTION: KIP, SFTKIP, IMAXP parameters are used to implement the integrative component of the PID position controller.

SETTING: KIP, SFTKIP, IMAXP depend on the integrative component of the PID position controller coefficients, obtained from the tuning, on the controller maximum speed output (*spd_max*) and on the speed scaling factor. See par. 3.5.3 for details about position controller model and parameters. KDP, SFTKDP, KDFP are computed as follows:

SFTKIP = the least integer power of 2 for which **KIP < 2^SFTKIP**

KIP = Ki_P_scl * 32767 / 2^SFTKIP

IMAXP = user defined. Defines the saturation value of the integrative part.

EXAMPLE:

If: *Ki_P_scl* = 9.6e-4

Then: SFTKIP = 0 : 2^0 = 1 > 9.6e-4

KIP = 9.6e-4 * 32767 / 2^0 = 31

REMARK: The maximum shift value is 12, which implies that the maximum value of controller coefficient KIP is 4096.

4.2.5.7. KIS, SFTKIS, IMAXS. Integrative term coefficients for speed controller

KIS: integrative factor for speed controller

SFTKIS: integrative factor shift for speed controller

IMAXS: integrative part saturation limit

ADDRESS: KIS: 0x0269; SFTKIS: 0x026A; IMAXS: 0x026C

NUMERICAL TYPE: integer

DEFAULT VALUE: KIS = 0 (0x0000); SFTKIS = 0 (0x0000); IMAXS = 0 (0x0000)

DESCRIPTION: KIS, SFTKIS, IMAXS parameters are used to implement the integrative component of the PI speed controller.

SETTING: KIS, SFTKIS, IMAXS depend on the integrative component of the PI speed controller coefficients, obtained from the tuning, on the controller maximum current output (*I_max*) and on the current scaling factor. See par. 3.5.2 for details about speed controller model and parameters. KIS, SFTKIS, IMAXS are computed as follows:

SFTKIS = the least integer power of 2 for which **KIS < 2^SFTKIS**

KIS = Ki_S_scl * 32767 / 2^SFTKIS

IMAXS = user defined. Defines the saturation value of the integrative part.

EXAMPLE:

If: *Ki_S_scl* = 7.81

Then: SFTKIS = 3 : 2^3 = 8 > 7.81

KIS = 7.81 * 32767/ 8 = 31988

REMARKS: The maximum shift value is 12, which implies that the maximum value of controller coefficient KIS is 4096.

4.2.5.8. KPI, SFTKPI. Proportional term coefficients for current controllers

KPI: proportional factor for current controllers

SFTKPI: proportional factor shift for current controllers

ADDRESS: KPI: 0x0271; SFTKPI: 0x0272

NUMERICAL TYPE: integer

DEFAULT VALUE: KPI = 0 (0x0000); SFTKPI = 0 (0x0000)

DESCRIPTION: KPI and SFTKPI parameters are used to implement the proportional component of the PI current controllers.

SETTING: KPI and SFTKPI depend on the proportional component of the PI current controller coefficients, obtained from the tuning and on the controller maximum voltage outputs (*u_d_max*, *u_q_max*). See par. 3.5 for details about current controller's model and parameters. KPI and SFTKPI are computed as follows:

SFTKPI = the least integer power of 2 for which **KPI < 2^SFTKPI**

KPI = Kp_C_scl * 32767 / 2^SFTKPI

EXAMPLE:

If: *Kp_C_scl* = 1.2

Then: SFTKPI = 1 : 2^1 = > 1.2

KPI = 1.2 * 32767 / 2 = 19660

REMARKS: The maximum shift value is 12, which implies that the maximum value of controller coefficient KPI is 4096.

4.2.5.9. KPP, SFTKPP. Proportional term coefficients for position controller

KPP: proportional factor for position controller

SFTKPP: proportional factor shift for position controller

ADDRESS: KPP: 0x025E; SFTKPP: 0x025F

NUMERICAL TYPE: integer

DEFAULT VALUE: KPP = 0 (0x0000); SFTKPP = 0 (0x0000)

DESCRIPTION: KPP and SFTKPP parameters are used to implement the proportional component of the PID position controllers.

SETTING: KPP and SFTKPP depend on the proportional component of the PID position controller coefficients, obtained from the tuning and on the controller maximum speed output (*spd_max*). See par. 3.5 for details about position controller's model and parameters. KPP and SFTKPP are computed as follows:

SFTKPP = the least integer power of 2 for which **KPP < 2^SFTKPP**

KPP = Kp_P_scl * 32767 / 2^SFTKPP

EXAMPLE:

If: *Kp_P_scl* = 6.82e-2

Then: SFTKPP = 0 : 2^0 = 1 > 6.82e-2

KPP = 6.82e-2 * 32767 / 2^0 = 2234

REMARKS: The maximum shift value is 12, which implies that the maximum value of controller coefficient KPP is 4096.

4.2.5.10. KPS, SFTKPS. Proportional term coefficients for speed controller

KPS: proportional factor for speed controllers

SFTKPS: proportional factor shift for speed controllers

ADDRESS: KPS: 0x0267; SFTKPS: 0x0268

NUMERICAL TYPE: integer

DEFAULT VALUE: KPS = 0 (0x0000); SFTKPS = 0 (0x0000)

DESCRIPTION: KPS and SFTKPS parameters are used to implement the proportional component of the PI speed controllers.

SETTING: KPS and SFTKPS depend on the proportional component of the PI speed controller coefficients, obtained from the tuning and on the controller maximum current output (*I_max*). See par. 3.5 for details about speed controller's model and parameters. KPS and SFTKPS are computed as follows:

SFTKPS = the least integer power of 2 for which **KPS < 2^SFTKPS**

KPS = Kp_S_scl * 32767 / 2^SFTKPS

EXAMPLE:

If: *Kp_S_scl* = 125

Then: SFTKPS = 7 : 2^7 =128 > 125

KPS = 125 * 32767 / 128 = 31999

REMARKS: The maximum shift value is 12, which implies that the maximum value of controller coefficient KPS is 4096.

4.2.5.11. POSOKLIM. Accepted position error limit to suspend control

ADDRESS: 0x036A

NUMERICAL TYPE: unsigned integer

DEFAULT VALUE: 0 (0x0000)

DESCRIPTION: POSOKLIM parameter is used to suspend (to lock) the execution of the control loops if the position error is lower than this value for a time greater than the one specified with the TONPOSOK parameter. This special feature of the position control loop can be useful in such applications where the vibrations of the motor are undesired from the moment when the load has arrived to its target position.

SCALING FACTOR: *Kpf*. For more details about this feature see par. 3.5.4.4.

SETTING: POSOKLIM can be set based on the application requirements and the position sensor resolution. If the position sensor resolution is good enough and the application accepts a certain position error, POSOKLIM could be set to a value slightly lower than the accepted position error.

4.2.5.12. SATID. Saturation limit for ud command

ADDRESS: 0x0275

NUMERICAL TYPE: integer

DEFAULT VALUE: 0 (0x0000)

DESCRIPTION: SATID parameter is used at the implementation of PI current controllers. It represents the D axis current controller output limit.

SETTING: SATID depends on the current controller coefficients resulting from the tuning of (*Kp_crt*, *Ki_crt*) and on the maximum controller voltage outputs (*u_d_max*, *u_q_max*). See par. 3.5.1 for details about current controller's model and parameters. SATID is computed as follows:

SATID = 65535 * 2e-6 * *PWM_freq*

EXAMPLE:

If: *PWM_freq* = 20000 [Hz]

Then: SATID = 65535 * 2e-6 * 20000 = 2621

4.2.5.13. SATIQ. Saturation limit for uq command

ADDRESS: 0x0276

NUMERICAL TYPE: integer

DEFAULT VALUE: 0

DESCRIPTION: SATIQ parameter is used at the implementation of PI current controllers. It represents the Q axis current controller output limit.

SETTING: SATIQ depends on the current controller coefficients resulting from the tuning of (*Kp_crt*, *Ki_crt*) and on the maximum controller voltage outputs (*u_d_max*, *u_q_max*). See par. 3.5.1 for details about current controller's model and parameters. SATIQ is computed as follows:

SATIQ = 65535 * 2e-6 * *PWM_freq*

EXAMPLE:

If: *PWM_freq* = 10000 [Hz]

Then: SATIQ = 65535 * 2e-6 * 10000 = 1311

4.2.5.14. SATP. Saturation limit for speed command

ADDRESS: 0x0265

NUMERICAL TYPE: integer

DEFAULT VALUE: 0 (0x0000)

DESCRIPTION: SATP parameter is used at the implementation of PID position controller. It represents the position controller output limit.

SETTING: SATP depends on the PID position controller coefficients resulting from the tuning of (*Kp_pos*, *Ki_pos*, *Kd_pos*); on the controller maximum speed output (*spd_max*) and on the speed-scaling factor. See par. 3.5.3 for details about position controller's model and parameters. SATP is computed as follows:

SATP = 32767 – (*N_max*) * Kvf

EXAMPLE:

If: *N_max* = 314 [rad/s]

Then: *Kvf*= 0.318 [bit/rad/s]

SATP = 32767 – 314 * 0.318 = 32667

4.2.5.15. SATS. Saturation limit for current command

ADDRESS: 0x026B

NUMERICAL TYPE: integer

DEFAULT VALUE: 0 (0x0000)

DESCRIPTION: SATS parameter is used at the implementation of PI speed controller. It represents the speed controller output limit.

SETTING: SATS depends on the PI speed controller coefficients resulting from the tuning (*Kp_spd*, *Ki_spd*); on the maximum output current of the PI speed controller (*I_max*); on the current scaling factors; on the speed feedforward scaling factors. See par. 3.5.3 for details about speed controllers model and parameters. SATS is computed as follows:

SATS = 32767 – *I_max * Kif*

EXAMPLE:

If: *I_max* = 2 [A]

Then: *Kif* = 6553 [bit/A]

SATS = 32767 – 2 * 6553 = 19661

4.2.5.16. SFTKFF. Global shift for acceleration & load feedforward block

ADDRESS: 0x0270

NUMERICAL TYPE: integer

DEFAULT VALUE: 0 (0x0000)

DESCRIPTION: SFTKFF parameter is used to implement the feedforward term for PI current controller. It represents the global shift for the feedforward term that is implemented in the acceleration and load feedforward block.

SETTING: SFTKFF depends on the PI speed controller coefficients resulting from the tuning of (*Kp_spd*, *Ki_spd*); on the maximum output current of the PI speed controller (*I_max*); on the current scaling factors; on the speed, acceleration & load feedforward scaling factors. See par. 3.5.4.2 for details about speed controller and acceleration & load feedforward block models and parameters. SFTKFF is computed as follows:

SFTKFF = the least integer power of 2 for which **KFFA < 2^SFTAFFW + KFFL < 2^SFTKFF**

REMARKS: The maximum shift value is 12, which implies maximum values of 4096 for feedforward block coefficients.

4.2.5.17. TONPOSOK. Position trigger time to suspend control

ADDRESS: 0x036B

NUMERICAL TYPE: unsigned integer

DEFAULT VALUE: -1 (0xFFFF)

DESCRIPTION: TONPOSOK parameter is used to specify the time during which the position error must be lower than POSOKLIM to initiate the position control suspend feature.

SETTING: TONPOSOK can be set based on the time given to the position to remain in the accepted position error limit. The value must be expressed in number of samplings of the position loop (slow loop).

4.2.6. Reference generator related TML parameters

4.2.6.1. CACC. Acceleration command for position/speed profile modes

ADDRESS: 0x02A2

NUMERICAL TYPE: fixed - fixed point 32 bits: (16 bits signed integer part).(16 bits fractional part)

DEFAULT VALUE: 0.5 (0x00008000)

DESCRIPTION: CACC parameter represents the absolute value of the acceleration or deceleration for a positioning or speed profile movement.

SCALING FACTOR: *Kaf*. For more details about scaling factors see par. 3.7.

SETTING: CACC must be set before an immediate or event-based update command is issued. At the update command execution, the value of CACC is transferred to the reference generator module and is used to compute the position or speed reference. See MotionChip II TML Programming User Manual for details about profiles positioning.

EXAMPLE:

In order to set the motion acceleration at value 10.5, use the TML assignment command

CACC = 10.5;

Or send the corresponding binary code, where the value of CACC is set to 0x000A8000 (0x000A ↔ 10, 0x8000 ↔ 0.5).

4.2.6.2. CAMOFF. CAM offset

ADDRESS: 0x03AD

NUMERICAL TYPE: long

DEFAULT VALUE: 0 (0x0000)

DESCRIPTION: CAMOFF parameter it's used in electronic camming modes allowing shifting the cam profile versus the master position. It represents the offset between the master position and slave axis positions in electronic camming.

SETTING: Must be set at the same time with the cam table. The following relation exists between: the master position (MREF), the cam offset (CAMOFF), the cam table X input (MPOS0) and the master resolution (MASTERRES):

MPOS0 = (MREF – CAMOFF) % MASTERRES

4.2.6.3. CAMSTART. RAM address where the cam table is copied

ADDRESS: 0x03AC

NUMERICAL TYPE: integer

DEFAULT VALUE: 0 (0x0000)

DESCRIPTION: CAMSTART parameter points to SRAM program memory start address for a cam table. When several cam tables are used, switching between them resumes to set CAMSTART to the right address i.e. the beginning of next the cam table to use. CAMSTART is automatically set by the `INITCAM` command, which copies the cam table from the EEPROM to the SRAM memory

SETTING: CAMSTART parameter is initialized after the execution of `INITCAM` instruction with the RAM address where the cam table was copied.

4.2.6.4. CPOS. Position command for position profile mode

ADDRESS: 0x029E

NUMERICAL TYPE: long (32 bits signed integer)

DEFAULT VALUE: 0 (0x00000000)

DESCRIPTION: CPOS parameter represents the value of the command position for a positioning profile movement. CPOS is considered relative, after a `CPR` command or absolute after a `CPA` command. After reset, CPOS is relative. By default, in a relative positioning, CPOS value is added to the actual position APOS of the motor at the moment when update command is executed. If `TUM1` command is issued after the positioning profile mode is set, CPOS value is added to the target position TPOS at the moment when update command is executed.

SCALING FACTOR: *Kpf*. For more details about scaling factors see par.3.7.

SETTING: CPOS must be set before an immediate or event-based update command is issued. At the update command execution, the value of CPOS is transferred to the reference generator module and is used to compute the position reference. See MotionChip II TML Programming User Manual for details about profiles positioning.

EXAMPLE:

In order to set the motion position at value 130, for positioning profile mode, use the TML assignment command

CPOS = 130;

Or send the corresponding binary code, where the value of CPOS is set to 0x00000082 (0x82 ↔ 132).

4.2.6.5. CSPD. Speed command for position/speed profile modes

ADDRESS: 0x02A0

NUMERICAL TYPE: fixed - fixed point 32 bits: (16 bits signed integer part).(16 bits fractional part)

DEFAULT VALUE: 0 (0x00000000)

DESCRIPTION: CSPD parameter represents the value of the slew speed for a positioning profile movement or the value of the jog speed for a speed profile movement. In a speed profile the CSPD sign indicates the direction of the movement. In a positioning profile, the CSPD sign is disregarded.

SCALING FACTOR: Kvf. For more details see par. 3.2 Scaling factors.

SETTING: CSPD must be set before an immediate or event-based update command is issued. At the update command execution, the value of CSPD is transferred to the reference generator

module and is used to compute the position or speed reference (depending on the motion mode settings). See TML Instructions Set User Manual for details about profiles positioning.

EXAMPLE:

In order to set the motion speed at value 600.5, for positioning or speed profile mode, use the TML assignment command

CSPD = 600.5;

Or send the corresponding binary code, where the value of CSPD is set to 0x02588000 (0x0258 ↔ 600, 0x8000 ↔ 0.5).

4.2.6.6. EFLEVEL. Activate/deactivate synchronization with master

ADDRESS: 0x02C7

NUMERICAL TYPE: integer

DEFAULT VALUE: 32767 (0x7FFF)

DESCRIPTION: EFLEVEL parameter activates or deactivates the synchronization mechanism between the master and slave in electronic gearing or camming. The master send it's position via communication.

SETTING: The user must set EFLEVEL before starting the operation of the axis in the slave-gearing mode.

EXAMPLE: In order to activate the synchronization with the master use the following TML assignment command:

EFLEVEL=0;

Remark: *The synchronization must be enabled only after the master starts sending its position and must be disabled before or immediately after the master stops sending its position. Do not leave a slave with the synchronization enabled while the master is disabled. During this period the motor control performance is slightly degraded*

4.2.6.7. EREF. External reference for all external modes

ADDRESS: 0x02A8

NUMERICAL TYPE: long (32 bits signed integer) OR fixed – fixed-point 32 bits: (16 bits signed integer part).(16 bits fractional part)

ACCESS: R/W

DESCRIPTION: EREF parameter stores the value of the reference, for all external reference modes.

SETTING: EREF must be set with the reference value, for all external "***on-line***" reference modes. See par. 4.6.6 for details about external references and their parameters and setting. Depending

on the reference type, its value will be considered as a long integer (for position references), or as a fixed-point 32-bit fractional value (for speed, torque or voltage references).

EXAMPLE:

In order to set the motion speed at value 600.5, for on-line external speed control mode, send the corresponding binary code of the TML assignment command

EREF = 600.5;

(i.e. set EREF to 0x02588000 (0x0258 ↔ 600, 0x8000 ↔ 0.5).

4.2.6.8. GEAR. GEARMASTER. GEARSLAVE. Gearing factor for slave axis

ADDRESS: GEAR: 0x02AC, GEARMASTER: 0x0255, GEARSLAVE: 0x03AD

NUMERICAL TYPE:

GEAR: fixed - fixed point 32 bits: (16 bits signed integer part).(16 bits fractional part)

GEARMASTER: unsigned integer

GEARSLAVE: integer

DEFAULT VALUE:

GEAR: 1 (0x00000001)

GEARMASTER: -31072 (0x86A0)

GEARSLAVE: 1 (0x0001)

DESCRIPTION: GEAR, GEARMASTER and GEARSLAVE parameters are used to set the gearing factor used by an axis when operating in slave gearing mode. GEAR is multiplied with the master position increment (MREF – MPOS0), to give the target position increment of the slave. GEARSLAVE and GEARMASTER represent the numerator and denominator of the Slave / Master ratio. GEARSLAVE is a signed integer, while GEARMASTER is an unsigned integer. GEARSLAVE sign indicates the direction of movement: positive – same as the master, negative – reversed to the master. GEAR is a fixed value containing the result of the gear ratio i.e. the result of the division GEARSLAVE / GEARMASTER. In order to eliminate any cumulative errors the electronic gearing slave mode includes an automatic compensation of the round off errors when the gear ratio has an irrational value like: Slave = 1, Master = 3, giving a ratio of 1/3 = 0.33333 which can't be represented exactly

SETTING: The user must set GEAR before starting the operation of the axis in the slave-gearing mode. This parameter can be set directly using an assignment instruction, or is computed based on GEARMASTER and GEARSLAVE values. See MotionChip II TML Programming User Manual for details about electronic gearing.

EXAMPLE:

In order to set the gearing factor at value 3.5, for slave gearing mode, use the TML assignment command

GEAR = 3.5;

Or send the corresponding binary code, where the value of GEAR is set to 0x00038000 (0x0003 ↔ 3, 0x8000 ↔ 0.5).

4.2.6.9. MPOS0. Initial master reference

ADDRESS: 0x02E5

NUMERICAL TYPE: long (32 bits signed integer)

DEFAULT VALUE: 0 (0x00000000)

SCALING FACTOR: *Kpf_master*. For more details about scaling factors see par. 3.7.

DESCRIPTION: MPOS0 parameter is used by an axis when operating in slave gearing mode to compute its new target position with the formula:

New TPOS = old TPOS + (MREF – MPOS0) x GEAR

where, MREF is the master position, sent by the master and MPOS0 the previous master position.

SETTING: Before activating the slave mode MPOS0 acts like a parameter that has to be initialized by the master with its target or actual position. After the slave mode is set, MPOS0 becomes a variable, which is updated by the slave at each position / speed sampling period with the previous value of MREF.

EXAMPLE:

In order to set MPOS0 on the slave axis 1, use on the master axis the TML assignment command

[1] MPOS0 = APOS;

This sets MPOS0 of slave axis 1 with the value of APOS of the master axis

4.2.6.10. POS0. Origin of relative position for event tests

ADDRESS: 0x02B8

NUMERICAL TYPE: long (32 bits signed integer)

DEFAULT VALUE: 0 (0x00000000)

DESCRIPTION: POS0 parameter is used to mark the absolute position value, from which the relative position will be measured when a relative position event is set. By default, POS0 is updated each time a new update command is executed. As effect, POS0 is set to the value of the actual position APOS of the motor. Using `TUM1` command it is also possible to set POS0 to the value of the target position TPOS. After a `RAOU` (reset automatic origin update) command, POS0 remains unchanged when a new update is executed. This operation mode can be used to monitor a relative position event while the motion modes or parameters are changing. With command `SAOU` (set automatic origin update) the default operation mode is restored. POS0 is used as parameter only if TML relative positioning events are programmed with `RAOU` mode set.

SETTING: POS0 must be set before setting a TML relative event, and only under RAOU command. See MotionChip II TML Programming User Manual for details about motion events. In the other cases, POS0 is a variable automatically updated.

EXAMPLE:

In order to set POS0 at value 260, use the TML assignment command

POS0 = 260;

Or send the corresponding binary code, where the value of POS0 is set to 0x00000104 (0x0104 ↔ 260).

4.2.6.11. REF0. Initial reference in torque/voltage mode

ADDRESS: 0x02A8

NUMERICAL TYPE: long / fixed - fixed point 32 bits: (16 bits signed integer part).(16 bits fractional part)

DEFAULT VALUE: 0 (0x00000000)

DESCRIPTION: REF0 parameter is used to set the initial value of the reference, when a contouring torque or voltage mode is activated.

SETTING: REF0 must be set before starting a contouring torque or voltage mode. It will represent the initial value used by the reference generator, from which the contouring values are computed. See TML Instruction Set User Manual for details about contouring mode.

EXAMPLE:

In order to set the origin for the voltage at value 15.25, use the TML assignment command

REF0 = 15.25;

Or send the corresponding binary code, where the value of REF0 is set to 0x000F4000 (0x000F ↔ 15, 0x4000 ↔ 0.25).

4.2.6.12. REFTST. Test reference saturation value

ADDRESS: 0x0281

NUMERICAL TYPE: integer

DEFAULT VALUE: 0 (0x0000)

DESCRIPTION: REFTST parameter is used to implement the starting modes in Brushless Motor applications (AC control mode), or to implement the test-operating mode; it represents the saturation value of the generated reference.

SCALING FACTOR: *Kuf_m* for the voltage reference or *Kif* for the current reference. For more details about scaling factors see par. 3.7.

SETTING: REFTST depends on the reference type (current, voltage), on the input value and on the scaling factor associated to each type of input. See par. 3.2 for details about starting procedures for PMSM motors. REFTST is computed as follows:

REFTST = (saturation value) * (scaling factor)

4.2.6.13. RINCTST. Test reference increment value

ADDRESS: 0x0280

NUMERICAL TYPE: integer

DEFAULT VALUE: 0 (0x0000)

DESCRIPTION: RINCTST parameter is used to implement the starting modes in Brushless Motor applications (AC control mode), or to implement the test-operating mode; it represents the increment value for the reference generated.

SCALING FACTOR: *K_V_inc* for the voltage reference or *K_C_inc* for the current reference. For more details about scaling factors see par. 3.7

SETTING: RINCTST depends on the reference type (current, voltage), on the input value and on the scaling factor associated to each type of input. See par. 3.2 for details about starting procedures for PMSM motors. RINCTST is computed as follows:

RINCTST = (saturation value) * (scaling factor)

4.2.6.14. SLAVEID. ID of slave axis receiving reference from the master axis

ADDRESS: 0x0311

NUMERICAL TYPE: integer

DEFAULT VALUE: 0 (0x0000)

DESCRIPTION: SLAVEID parameter is used to define the ID of the axes that must get the reference from a master axis. This will be needed in gearing master mode operation.

SETTING: SLAVEID must be set before starting a master operation mode. The SLAVEID parameter must contain the ID for the slave axis (if one slave is defined in the system), or the ID of the group of axes that will be slaves of the master one. See TML Instructions Set User Manual for details about multiple axis operation. The slave axis or group ID is set in SLAVEID using the standard 16-bit format of the ID field where ID value is set on bits 11 to 4 and bit 12 represents the axis/group selection.

EXAMPLE:

In order to send commands to groups 1, 4, 7 use the TML assignment command

SLAVEID = 0x1490; // I/G = 1,4,7; ID = 01001001;

Or send the corresponding binary code, where the value of SLAVEID is set to 0x1490.

4.2.6.15. THTST. Test reference electric angle

ADDRESS: 0x0282

NUMERICAL TYPE: integer

DEFAULT VALUE: 32768 (0x8000)

DESCRIPTION: THTST parameter is used to implement the test mode for PMSM motors, or to implement the test-operating mode; it represents the electric angle value imposed to the motor.

SCALING FACTOR: K_th. For more details about scaling factors see par. 3.7

SETTING: THTST depends on the reference type (current, voltage). For details about PMSM motors see par. 3.2. The user sets THTST as required by the test to be performed.

4.2.6.16. TIME0. Origin of relative time for event tests

ADDRESS: 0x02BE

NUMERICAL TYPE: long (32 bits signed integer)

DEFAULT VALUE: 0 (0x00000000)

DESCRIPTION: TIME0 parameter is used to mark the absolute time value, from which a relative time interval will be measured in RTIME variable. The computing formula is:

RTIME = ATIME – TIME0

When a relative time event is activated, TIME0 is automatically updated with the value of ATIME - the absolute timer.

SCALING FACTOR: *Ktf*. For more details about scaling factors see par. 3.7.

SETTING: TIME0 can be used to memorize ATIME value at a certain moment. RTIME will then provide the time elapsed from this moment.

EXAMPLE:

In order to set memorize ATIME use the TML assignment command

TIME0 = ATIME;

4.2.6.17. TINCTST. Test reference electric angle increment

ADDRESS: 0x0283

NUMERICAL TYPE: integer

DEFAULT VALUE: 1 (0x0001)

DESCRIPTION: TINCTST parameter is used to implement the starting modes for PMSM, or to implement the test-operating mode; it represents the increment value for the electric angle.

SCALING FACTOR: *K_V_inc* for voltage command or *K_C_inc* for current command. For more details about scaling factors see par. 3.7

SETTING: TINCTST depends on the reference type (current, voltage), on the input value and on the scaling factor associated to each type of input. See par. 3.2 for details about starting procedures for PMSM motors. TINCTST is computed as follows:

TINCTST = (current/ voltage increment) * (scaling factor)

4.2.7. Protections related TML parameters

4.2.7.1. ERRMAX. Maximum accepted control error in the most outer loop

ADDRESS: 0x02C5

NUMERICAL TYPE: unsigned integer

DEFAULT VALUE: 32767 (0x7FFF)

DESCRIPTION: ERRMAX parameter represents the maximum accepted control error of the most outer loop of the controlled system. If the error between the reference and the feedback exceeds this value for a time that is greater than the value specified by the TERRMAX parameter, the corresponding control error flag (CTRERIF) is set in the Interrupt Status Register – ISR.

SCALING FACTOR: *Kpf* for position control or *Kvf* for speed control. For more details about scaling factors see par. 3.7.

SETTING: ERRMAX depends on the maximum accepted error for the variable that is controlled in the outer loop and on its scaling factor. See par. 3.6.6 for details about control error protection. It is expressed as follows:

ERRMAX = (max. accepted error of the controlled variable) * (controlled variable scaling factor)

EXAMPLE:

If: the outer loop is a speed control loop and

(max. accepted speed error) = *ERR_spd* = 10 rad/s;

(speed scalling factor) = *Kvf* = 105 bits/rad/s

Then: ERRMAX = 10 * 105 = 1050;

4.2.7.2. I2TINTLIM. I^2t Protection Integrator limit

ADDRESS: 0x0815

NUMERICAL TYPE: ulong

DEFAULT VALUE: 0 (0x00000000)

DESCRIPTION: I2TINTLIM is the maximum value of the I^2t motor thermal protection; exceeding this value triggers the corresponding bit in PCR register.

SETTING: I2TINTLIM is computed based on over current level, time at over current (*I_I2t* respectively *t_I2t*) defined by the user and nominal current (see par. 3.6).

4.2.7.3. II2TPROT. I^2t Protection Current Limit,

ADDRESS: 0x0814

NUMERICAL TYPE: integer

DEFAULT VALUE: 0 (0x0000)

DESCRIPTION: II2TPROT parameter is the reference for computing the I^2t protection.

SETTING: II2TPROT is computed based on motor nominal current.

II2TPROT= *In * Kif*

EXAMPLE:

If: *In* = 1.2 A

Then: *Kif* = 1985 [bit/A]

II2TPROT = 1.2 * 1985 = 2382

4.2.7.4. SFI2T I^2t protection scaling factor

ADDRESS: 0x0819

NUMERICAL TYPE: integer

DEFAULT VALUE: 0 (0x0000)

DESCRIPTION: SFI2T is used to scale the integral computed for I^2t motor thermal protection.

SETTING: SFI2T is computed based on time at over current *t_I2t* defined by the user (see par. 3.6.2).

4.2.7.5. IMAXPROT. Maximum current protection limit

ADDRESS: 0x0297

NUMERICAL TYPE: integer

DEFAULT VALUE: 32767 (0x7FFF)
DESCRIPTION: IMAXPROT parameter represents the current values which, when exceeded for a time greater than the interval specified by TIMAXPROT, will activate the maximum current protection.

SCALING FACTOR: *Kif*. For more details about scaling factors see par. 3.7.

SETTING: IMAXPROT depends on the motor and power converter maximum current. See par. 3.6.1 for details related to the protections. IMAXPROT depends on maximum protection current and current scaling factor. It is expressed as follows:

IMAXPROT = *IMAX_PROT* Kif*

EXAMPLE:

If: *IMAX_PROT* = 10 A; *Kif* = 125 bits/A

Then: IMAXPROT = 10 * 125 = 1250;

4.2.7.6. NI2T. I2t computation interval (I2t protection)

ADDRESS: 0x0255

NUMERICAL TYPE: long

DEFAULT VALUE: 100000 (0x000186A0)

DESCRIPTION: NI2T parameter represents the time after which one I2t protection iteration is executed.

SCALING FACTOR: *Ktf*. For more details about scaling factors see par. 3.7.

SETTING: NI2T depends on thermal model parameter values of the motor (mainly on the thermal time constant of the motor). See par. 3.6.2 for details related to the thermal protection model and parameters. NI2T depends on slow loop frequency analysis. NI2T will be set using the following relation:

NI2T = (analysis time value) · *Ktf*

EXAMPLE:

If: Analysis time value = 60 [s]; *Ktf* = 1000 [bits/s]

Then: NI2T = 60 · 1000= 60000

4.2.7.7. T1MAXPROT. Motor Temperature maximum protection value

ADDRESS: 0x0298

NUMERICAL TYPE: unsigned integer

DEFAULT VALUE: -1 (0xFFFF)

DESCRIPTION: T1MAXPROT parameter represents the temperature value which, when exceeded, will activate the maximum motor temperature protection.

SETTING: T1MAXPROT depends on the motor sensor type. See par. 3.6 for details related to the protections. MotionChip II will set automatically T1MAXPROT with 32767 if your motor temperature sensor is PTC or NTC.

4.2.7.8. T2MAXPROT. Drive/ power stage Temperature maximum protection value

ADDRESS: 0x0299

NUMERICAL TYPE: unsigned integer

DEFAULT VALUE: -1 (0xFFFF)

DESCRIPTION: T2MAXPROT parameter represents the temperature values which, when exceeded, will activate the maximum drive/ power stage temperature protection.

SCALING FACTOR: *KTf*. For more details about scaling factors see par. 3.7.

SETTING: T2MAXPROT depends on the power converter maximum temperature. See par. 3.6.5 for details related to over temperature protection. The user based on system parameters will set T2MAXPROT value. T2MAXPROT depends on the protection temperature and on the temperature scale factor. It is expressed as follows:

T2MAXPROT = *Temp_2_PROT * KTf+OffAt0oC*65472/3.3V*

EXAMPLE:

If: *Temp_2_PROT* = 150 deg; *KTf* = 200 bits/deg, OffAt0oC = 0,5 V = 9920

Then: T2MAXPROT = 150 * 200+9920 = 39920;

4.2.7.9. TERRMAX. Control error protection trigger time

ADDRESS: 0x02C6

NUMERICAL TYPE: unsigned integer

DEFAULT VALUE: -1 (0xFFFF)

DESCRIPTION: TERRMAX parameter represents the trigger time for the control error protection. If the error between the reference and the feedback exceeds ERRMAX for a time that is greater than this parameter, the corresponding control error flag is set in the Interrupt Status Register – ISR.

SCALING FACTOR: *Ktf*. For more details about scaling factors see par. 3.7.

SETTING: TERRMAX depends on the time interval during which the error between the reference and the feedback can exceed the ERRMAX limit and the sampling time of the outer loop. See par. 3.6 for details related to the protections. It is expressed as follows:

TERRMAX = (time interval for accepted error of the controlled variable) * *Ktf*

EXAMPLE:

If: the outer loop is a speed control loop and

(time interval for accepted error) = time_ERR_spd = 2 s;

Ktf = 1000 bits/s

Then: TERRMAX = 2 · 1000 = 2000;

4.2.7.10. TIMAXPROT. Maximum current protection trigger time

ADDRESS: 0x02C4

NUMERICAL TYPE: unsigned integer

DEFAULT VALUE: -1(0xFFFF)

DESCRIPTION: TIMAXPROT parameter represents the trigger time for the maximum current protection. If the error between the reference current IQREF and the feedback current IQ exceeds IMAXPROT for a time that is greater than this parameter, the maximum current protection is activated.

SCALING FACTOR: *Ktf*. For more details about scaling factors see par. 3.7.

SETTING: TIMAXPROT depends on the time interval during which the error in the current controller can exceed the IMAXPROT limit and the sampling time of the current loop. See par. 3.6. Maximal current protection for details. It is expressed as follows:

TIMAXPROT = *time_IMAX_PROT* · *Ktf*

EXAMPLE:

If: *time_IMAX_PROT* = 0.1 sec; *Ktf* = 1000 bits/sec

Then: TERRMAX = 0.1 · 1000= 100;

4.2.7.11. UMAXPROT. Protection – maximal voltage value

ADDRESS: 0x029A

NUMERICAL TYPE: unsigned integer

DEFAULT VALUE: -1 (0xFFFF)

DESCRIPTION: UMAXPROT parameter represents the voltage values which, when exceeded, will activate the over voltage protection.

SCALING FACTOR: *Kuf_m*. For more details about scaling factors see par.3.7.

SETTING: UMAXPROT depends on the maximum protection voltage and on the voltage scale factor. See par. 3.6.3 for details related to the protection. UMAXPROT is expressed as follows:

UMAXPROT = *UMAX_PROT* * *Kuf_m*

EXAMPLE:

If: *UMAX_PROT* = 300 V; *Kuf*_m= 50 bits/V

Then: UMAXPROT = 300 * 50 = 15000;

4.2.7.12. UMINPROT. Protection – minimal voltage value

ADDRESS: 0x029B

NUMERICAL TYPE: unsigned integer

DEFAULT VALUE: 0 (0x0000)

DESCRIPTION: UMINPROT parameter represents the voltage values which, when felled behind, will activate the under voltage protection.

SCALING FACTOR: *Kuf*. For more details about scaling factors see par.3.7.

SETTING: UMINPROT depends on the maximum protection voltage and on the voltage scale factor. See par. 3.6 for details related to the protections. UMINPROT is expressed as follows:

UMINPROT = *UMIN_PROT * Kuf_m*

EXAMPLE:

If: *UMIN_PROT* = 100 V; *Kuf*_m= 50 bits/V

Then: UMINPROT = 100 * 50 = 5000;

4.2.8. Motion Language parameters

4.2.8.1. INTTABLE. Start address of TML interrupts vector table

ADDRESS: 0x0307

NUMERICAL TYPE: integer

DEFAULT VALUE: 0 (0x0000)

DESCRIPTION: INTTABLE parameter represents the starting address of the TML interrupt vector table.

SETTING: INTTABLE value will be set by the user to point to the starting address of the TML interrupt vector table, which contains pointers to the TML interrupt routines, associated with each of the TML interrupts. (See MotionChip II TML programming User Manual).

INTTABLE = address of interrupt vector table;

EXAMPLE:

```
INTTABLE = IntVect;  // set interrupt table
    …
    …

IntVect:                          // interrupt vector table
    @Int0_Disable;
    @Int1_PDPINT;
    @Int2_SoftProtection;
```

```
@Int3_ControlError;
@Int4_CommError;
@Int5_WrapAround;
@Int6_LimitSwitchP;
@Int7_LimitSwitchM;
@Int8_Capture;
@Int9_MotionComplete;
@Int10_UpdateContourSeg;
@Int11_EventReached;
```

4.3. TML variables

TML programs use some specific data information structured through the TML variables. Having reserved TML mnemonics, the TML variables are used in the control module of the program. Their values can be examined at any time during the execution of the program. Activating the specific logger module, real-time data tracing can also be implemented for any of these variables.

Only those TML variables will have a meaning, which are used for a given application, depending on the motion system configuration or command settings.

The TML variables are internally initialized before activating the motion system real time structure (before the `ENDINIT` TML command), or before activating the motion control functions (before the execution of the `AXISON` TML command).

Most TML variables are read-only (RO). Modifying their value during motion execution may cause improper operation of the motion system. In specific situations, some of the TML variables can also be written (are R/W variables). These cases are explicitly mentioned in the next paragraphs.

This paragraph will present, in detail, all the TML variables. The information is grouped as follows:

- Address – the address in data memory (dm) where the parameter is.
- Numerical type – the parameter can be integer 16 bits, unsigned integer 16 bits, long 32 bits, unsigned long 32 bits or fixed point 32 bits: (16 bits signed integer part).(16 bits fractional part)
- Access – the variable can be read only (RO) or read/write (RW).
- Description – a short description of the variable will be presented.
- Scaling factor – where is the case the scaling factor will be given for you to understand how the value in IU has been obtained from the physical value (SI).
- Setting – present how the parameter must be used correctly.
- Remarks – if is necessary some remarks will be also given

The following tables present the TML variables, grouped by functionality. The remaining part of this paragraph will present, in detail, all the TML variables.

1. Motors

Variable Name	Data Type	Significance	Address	Par.
APOS	long	Actual position	0x0228	4.3.1.1
ASPD	fixed	Actual speed	0x022C	4.3.1.2
IA	int	A phase current	0x0239	4.3.1.3
IB	int	B phase current	0x023A	4.3.1.3
IC	int	C phase current	0x023B	4.3.1.3
UAREF	int	A phase reference voltage	0x0236	4.3.1.4
UBREF	int	B phase reference voltage	0x0237	4.3.1.4
UCREF	int	C phase reference voltage	0x0238	4.3.1.4

2. Sensors

Variable Name	Data Type	Significance	Address	Par.
AD0	uint	1st AD channel (current i_A)	0x023C	4.3.2.1
AD1	uint	2nd AD channel (current i_C)	0x023D	4.3.2.2
AD2	uint	3rd AD channel	0x023E	4.3.2.3
AD3	uint	4th AD channel	0x023F	4.3.2.4
AD4	uint	5th AD channel (DC link voltage)	0x0240	4.3.2.5
AD5	uint	6th AD channel (reference)	0x0241	4.3.2.6
AD6	uint	7th AD channel	0x0242	4.3.2.7
AD7	uint	8th AD channel (Temperature 2)	0x0243	4.3.2.8
CAPPOS	long	Captured position	0x02BC	4.3.2.9
CAPPOS2	long	Master captured position	0x081E	4.3.2.10
ELPOS	int	Measured (not scaled) field position angle	0x0221	4.3.2.11
THETAINC	int	Increment (not scaled) of field position angle	0x020E	4.3.2.12

3. Power Converters

Variable Name	Data Type	Significance	Address	Par.
CVDC	int	Ratio of DC bus variation (V_DC Compensation)	0x025B	4.3.3.1

4. Controllers

Variable Name	Data Type	Significance	Address	Par.
COSTH	int	Cosine theta	0x0226	4.3.4.1
CRTERR	int	Q axis current error	0x0231	4.3.4.2
HALL	int	Hall sensors information	0x0227	4.3.4.3
ID	int	D axis current	0x0234	4.3.4.4
IDREF	int	D axis reference current	0x0233	4.3.4.5
IQ	int	Q axis current	0x0230	4.3.4.6

IQREF	int	Q axis reference current	0x022F	4.3.4.7
POSERR	int	Position error	0x022A	4.3.4.8
SINTH	int	Sinus theta	0x0225	4.3.4.1
SPDERR	int	Speed error	0x022E	4.3.4.9
SPDREF	int	Reference speed	0x022B	4.3.4.10
THETA	int	Field position angle	0x0224	4.3.4.11
UDREF	int	D axis reference voltage	0x0235	4.3.4.12
UQREF	int	Q axis reference voltage	0x0232	4.3.4.13

5. Reference generator

Variable Name	Data Type	Significance	Address	Par.
ATIME	long	Absolute system time	0x02C0	4.3.5.1
APOS2	long	Master actual position	0x081C	4.3.5.2
BETA	long	Master synchronization	0x0293	4.3.5.3
CDREF	fixed	Reference increment for a contour segment	0x02A4	4.3.5.4
CTIME	uint	No. of samplings for a contour segment	0x02A6	4.3.5.5
MREF	long	Reference send from master	0x02AA	4.3.5.6
MSPD	int	Master speed	0x0820	4.3.5.7
RPOS	long	Relative position for event tests	0x02BA	4.3.5.8
RTIME	long	Relative time for event tests	0x02C2	4.3.5.9
TACC	fixed	Target acceleration	0x02B6	4.3.5.10
TPOS	long	Target position	0x02B2	4.3.5.11
TREF	long/ fixed	Target reference	0x02AE	4.3.5.12
TSPD	fixed	Target speed	0x02B4	4.3.5.13

6. Motion Language

Variable Name	Data Type	Significance	Address	Par.
PROD	48 bits	Result of a TML multiply operation	0x030E	4.3.6.1

7. General-purpose Pre-defined Variables

Variable Name	Data Type	Significance	Address	Par.
VAR_I1	int	Predefined user variable	0x0366	4.3.7.1
VAR_I2	int	Predefined user variable	0x0367	4.3.7.1
VAR_LF	fixed/long	Predefined user variable	0x0368	4.3.7.2

4.3.1. Motor related TML variables

4.3.1.1. APOS. Motor actual position

ADDRESS: 0x0228

NUMERICAL TYPE: long

ACCESS: R/W

DESCRIPTION: APOS variable contains the value of the actual position of the motor, represented as a 32-bit integer variable. Depending on the position sensor, its value is directly set with the value read from the sensor, or the position increment between two sampling moments is added to the actual position value. See par. 3.4.3 for details about position sensors models and parameters.

SCALING FACTOR: *Kpf*. For more details about scaling factors see par.3.7.

SETTING: APOS is updated at each speed / position control loop sampling time interval (i.e. the slow sampling time, see SLPER parameter). If the APOS variable wraps around, a “wrap around” bit is set in the MSR and ISR registers.

APOS can be read at any moment using TML instructions or the data tracer capabilities of TML environment.

4.3.1.2. ASPD. Motor actual speed

ADDRESS: 0x022C

NUMERICAL TYPE: fixed – fixed point 32 bits: (16 bits signed integer part).(16 bits fractional part)

ACCESS: R/W

DESCRIPTION: ASPD variable contains the value of the actual speed of the motor, represented as a 32-bit fixed-point value (16 bits integer part, 16 bits fractional part). Depending on the speed sensor, its value is directly set with the value read from the sensor, or a speed estimate is computed, using position variation between two sampling moments, or using the time length of a position sensor pulse. See par. 3.4.2 for details about speed sensors models and parameters.

SCALING FACTOR: *Kvf*. For more details about scaling factors see par.3.7.

SETTING: ASPD is updated at each speed / position control loop sampling time interval (i.e. the slow sampling time, see SLPER parameter).

ASPD can be read at any moment using TML instructions or the data tracer capabilities of TML environment.

4.3.1.3. IA, IB, IC. Motor actual currents

ADDRESS: IA: 0x0239; IB: 0x023A; IC: 0x023B

NUMERICAL TYPE: integer

ACCESS: RO

DESCRIPTION: IA, IB, IC variables represent the values of motor currents. These values are normally represented as signed fractional numbers. Only analog to digital current measurement is considered. See par. 3.4.1 for details about current sensors models and parameters.

SCALING FACTOR: *Kif*. For more details about scaling factors see par.3.7.

SETTING: IA, IB, IC are updated at each current control loop sampling time interval (i.e. the fast sampling time, see CLPER parameter). Two currents, IA and IC, are computed based on the measured A/D channels connected to the current sensors, corrected with the values of offsets defined for these channels. Third current, IB, is obtained making the assumption that the currents' sum is zero.

Currents values are computed as follows (the difference between the channel offset and the measured value is right-shifted with a number of bits given by the parameter SFTCRT, see par.3.4 for details):

IA = (AD0OFF – AD0) >> SFTCRT;

IC = (AD1OFF – AD1) >> SFTCRT;

IB = -(IA+IC)

Where AD0OFF and AD1OFF are the offsets detected or set by the user on channels 0 and 1 (see their corresponding description in par. 4.2.3.2). AD0 and AD1 are the measured values from the A/D channels 0 and 1 (see their description below).

REMARKS:

1. For DC motors, current IA is used to implement motor control module.

4.3.1.4. UAREF, UBREF, UCREF. Motor phase voltage commands

ADDRESS: UAREF: 0x0236; UBREF: 0x0237; UCREF: 0x0238

ACCESS: RO

NUMERICAL TYPE: integer

DESCRIPTION: UAREF, UBREF, UCREF variables represent the values of the command motor voltages. These values are the output of the inner active control loop (if any is defined) or, for voltage operating mode, the output of the reference generator module. See par. 3.3.5 for details about controllers' models and parameters.

SCALING FACTOR: *Kuf*. For more details about scaling factors see par.3.7.

SETTING: UAREF, UBREF, UCREF are updated at each current control loop sampling time interval (i.e. the fast sampling time, see CLPER parameter). They are computed depending on the motion mode, specifically depending on the active control loops. Corresponding to their values, the PWM outputs are set and will command the power transistors.

REMARKS:

1. For three-phase motors (PMSM, BLDC), all these values are set and used to drive the three-phase power inverter.
2. For DC motors, only UAREF and UBREF variables are used, to implement a four-quadrant operation of the motor.

4.3.2. Sensor related TML variables

4.3.2.1. AD0. 1st AD channel (current i_A)

ADDRESS: 0x023C

NUMERICAL TYPE: unsigned integer

ACCESS: RO

DESCRIPTION: AD0 variable represents the value of the measured A/D channel corresponding to phase A motor current. It corresponds to channel ADCIN6. See par. 3.4.1 for details about the operation of A/D channels.

SETTING: AD0 is updated at each current control loop sampling time interval (i.e. the fast sampling time, see CLPER parameter). Its value is used to set the variable IA, used by the control module.

4.3.2.2. AD1. 2nd AD channel (current i_C)

ADDRESS: 0x023D

NUMERICAL TYPE: unsigned integer

ACCESS: RO

DESCRIPTION: AD1 variable represents the value of the measured A/D channel corresponding to phase C motor current. It corresponds to channel ADCIN5. See par. 3.4.1 for details about the operation of A/D channels.

SETTING: AD1 is updated at each current control loop sampling time interval (i.e. the fast sampling time, see CLPER parameter). Its value is used to set the variable IC, used by the control module.

4.3.2.3. AD2. 3rd AD channel

ADDRESS: 0x023E

NUMERICAL TYPE: unsigned integer

ACCESS: RO

DESCRIPTION: AD2 variable represents the value of the measured A/D channel corresponding to an external analogue signal. It corresponds to channel ADCIN2. **On MotionChip II 3rd AD channel is not used.**

SETTING: –

4.3.2.4. AD3. 4th AD channel

ADDRESS: 0x023F

NUMERICAL TYPE: unsigned integer

ACCESS: RO

DESCRIPTION: AD3 variable represents the value of the measured A/D channel. By default, it corresponds to channel ADCIN7. **On MotionChip II 4th AD channel is not used.**

SETTING: –

4.3.2.5. AD4. 5th AD channel (DC link voltage)

ADDRESS: 0x0240

NUMERICAL TYPE: unsigned integer

ACCESS: RO

DESCRIPTION: AD4 variable represents the value of the measured A/D channel corresponding to DC link voltage. It is used if the DC link voltage compensation is activated (indicated by setting corresponding bit *"U_{DC} compensation"* in OSR register). By default, it corresponds to channel ADCIN4.

SCALING FACTOR: *Kuf*_m. For more details about scaling factors see par.3.7.

SETTING: AD4 is updated at each current control loop sampling time interval (i.e. the fast sampling time, see CLPER parameter). The control module uses this value, if the DC link voltage compensation is activated (see par. 3.3 for details about DC-link/supply voltage compensation).

4.3.2.6. AD5. 6th AD channel (reference)

ADDRESS: 0x0241

NUMERICAL TYPE: unsigned integer

ACCESS: RO

DESCRIPTION: AD5 variable represents the value of the measured A/D channel corresponding to an analogue reference signal (indicated by using corresponding TML mode setup instructions). By default, it corresponds to channel ADCIN3.

SETTING: AD5 is updated at each current control loop sampling time interval (i.e. the fast sampling time, see CLPER parameter). The control module uses this value, if an analogue

reference must be used by the motion system (see par. 4.2.5 for details about analogue reference generators). The variable AD5 is used to set TREF (target reference), needed in the control module. The setting is performed using the following relation:

$$\textbf{TREF = filtered (AD5 – AD5OFF) * CADIN * } 2^{\textbf{SFTADIN}} \textbf{ / 65536}$$

If an analogue speed reference is applied on this A/D channel, the SPDREF variable is set, using the following relation:

$$\textbf{SPDREF = filtered (AD5 – AD5OFF) * CADIN * } 2^{\textbf{SFTADIN}} \textbf{ / 65536}$$

REMARKS:

1. A digital first-order low-pass filter is used to filter the input reference signal. This feature is useful to avoid noises that can determine mechanical oscillations of the system.

4.3.2.7. AD6. 7h AD channel

ADDRESS: 0x0242

NUMERICAL TYPE: unsigned integer

ACCESS: RO

DESCRIPTION: AD6 variable represents the value of the measured A/D channel. By default, it corresponds to channel ADCIN0. **On MotionChip II 4th AD channel is not used.**

SETTING: –

4.3.2.8. AD7. 8th AD channel (Drive temperature)

ADDRESS: 0x0243

NUMERICAL TYPE: unsigned integer

ACCESS: RO

DESCRIPTION: AD7 variable represents the value of the measured A/D channel corresponding to drive temperature sensor. By default, it corresponds to channel ADCIN1.

SCALING FACTOR: *KTf*. For more details about scaling factors see par.3.7.

SETTING: AD7 is updated at each current control loop sampling time interval (i.e. the fast sampling time, see CLPER parameter). The control module uses this value in the protections section, to check if parameter T2MAXPROT limit is not exceeded.

4.3.2.9. CAPPOS. Captured position

ADDRESS: 0x02BC

NUMERICAL TYPE: long

ACCESS: RO

DESCRIPTION: CAPPOS variable represents a value of the measured motor position sensor, captured at the occurrence of an external capture on the DSP. See TML Instructions Set User Manual for details about the use of captures in the TML environment.

SCALING FACTOR: *Kpf*. For more details about scaling factors see par.3.7.

SETTING: CAPPOS is set as the value of the position sensor, captured on one of the capture inputs Z1+ and Z1- of the drive (these capture inputs corresponds to DSP pins IOPA3/CAP1, respectively IOPA4/CAP2). It can be used to correct the position information, related to an external I/O input device (switch), installed in a specific position of the motion structure (homing functions, for example).

4.3.2.10. CAPPOS2. Master captured position

ADDRESS: 0x081E

NUMERICAL TYPE: long

ACCESS: RO

DESCRIPTION: CAPPOS2 variable represents the value of the master position read at second encoder inputs, captured at the occurrence of an external capture on the DSP. See TML Instructions Set User Manual for details about the use of captures in the TML environment.

SCALING FACTOR: *Kpf_master*. For more details about scaling factors see par.3.7.

SETTING: CAPPOS2 is set as the value of the second position sensor, captured on one of the capture pins Z2+ and Z2- of the drive (these capture inputs corresponds to DSP pins IOPE7/CAP4, respectively IOPF0/CAP5).

4.3.2.11. ELPOS. Measured (not scaled) field position angle

ADDRESS: 0x0221

NUMERICAL TYPE: integer

ACCESS: RO

DESCRIPTION: ELPOS variable represents the value of the electric angle position as measured from the position sensor.

SETTING: ELPOS is updated at each current control loop sampling time interval (i.e. the fast sampling time, see CLPER parameter). The control module uses this value to compute the scaled field position angle – THETA, by multiplying it with the ENC2THL parameter.

4.3.2.12. THETAINC. Electrical angle increment

ADDRESS: 0x020E

NUMERICAL TYPE: integer

ACCESS: RO

DESCRIPTION: THETAINC variable represents the increment of the electric angle as measured from the position sensor, before scaling.

SETTING: THETAINC is updated at each current control loop sampling time interval (i.e. the fast sampling time, see CLPER parameter). The control module uses this value to compute the scaled field position angle – THETA, by adding it to ELPOS variable and then multiplying the result with the ENC2THL parameter.

4.3.3. Power Converters related TML variables

4.3.3.1. CVDC. Ratio of DC bus variation (V_DC Compensation)

ADDRESS: 0x025B

NUMERICAL TYPE: integer

DEFAULT VALUE: 0

DESCRIPTION: CVDC parameter is used to implement the compensation of DC link voltage (V_{DC}) variation in the PWM command. Its value represents the ratio between the nominal and the measured V_{DC} values.

SETTING: CVDC is computed each control loop sampling if the U_{DC} Compensation feature is activated. In the computation of the duty cycles of the PWM signals, the values are multiplied with the CVDC variable to compensate for the variations of the DC bus voltage: if the DC bus voltage is higher than the nominal value described by the VDCN parameter, the duty cycles of the phase voltages are decreased. The CVDC parameter is computed with the following formula:

$$CVDC = PWM\_per \cdot \frac{VDCN}{V\_DC \cdot Kuf\_m}$$

4.3.4. Controllers related TML variables

4.3.4.1. COSTH, SINTH. Cosine, sine of electric angle

ADDRESS: COSTH: 0x0226; SINTH: 0x0225

NUMERICAL TYPE: integer

ACCESS: RO

DESCRIPTION: COSTH and SINTH variables represent the values of the sine, respectively cosine of the electric angle position.

SETTING: COSTH and SINTH are updated at each current control loop sampling time interval (i.e. the fast sampling time, see CLPER parameter). The control module uses these values for vector control of AC motors, for coordinate transformations.

4.3.4.2. CRTERR. Q axis current error

ADDRESS: 0x0231

NUMERICAL TYPE: integer

ACCESS: RO

DESCRIPTION: CRTERR variable represents the value of current error, on the Q axis (torque axis) of the motor, computed as the difference between the reference and the measured current in the Q axis of the motor.

SCALING FACTOR: *Kif*. For more details see par. 3.2 Scaling factors

SETTING: CRTERR is updated at each current control loop sampling time interval (i.e. the fast sampling time, see CLPER parameter). The control module uses these values for current control loop implementation. This value is related to the Q (torque) axis current component of the motor. (See par. 3.5.1 for details about current controllers).

4.3.4.3. HALL. Hall sensors information

ADDRESS: 0x0227

NUMERICAL TYPE: integer

ACCESS: RO

DESCRIPTION: HALL variable codifies the position information read from Hall sensors, in the case of BLDC motors.

SETTING: HALL is set corresponding to the position information read from the Hall sensors and Hall configuration. See par. 3.4.4 for details about Hall sensors use and parameters.

4.3.4.4. ID. D axis current

ADDRESS: 0x0234

NUMERICAL TYPE: integer

ACCESS: RO

DESCRIPTION: ID variable represents the value of the measured field current component. The control module uses this value for vector control of AC motors.

SCALING FACTOR: *Kif*. For more details about scaling factors see par.3.7.

SETTING: ID is updated at each current control loop sampling time interval (i.e. the fast sampling time, see CLPER parameter). It is obtained from the measured motor currents, as projected on the field axis of the motor (for vector control schemes).

REMARKS: *This variable is not used in the case of DC motors control.*

4.3.4.5. IDREF. D axis reference current

ADDRESS: 0x0233

NUMERICAL TYPE: integer

ACCESS: RO

DESCRIPTION: IDREF variable represents the value of the reference field current component. The control module uses this value for vector control of AC motors.

SCALING FACTOR: *Kif*. For more details about scaling factors see par.3.7.

SETTING: IDREF is updated at each current control loop sampling time interval (i.e. the fast sampling time, see CLPER parameter). It is obtained as the output of the field control block, and is usually constant up to the rated speed of the motor, and is modified at speeds higher than the rated value, in the field weakening operating region.

REMARKS: *This variable is not used in the case of DC motors control.*

4.3.4.6. IQ. Q axis current

ADDRESS: 0x0230

NUMERICAL TYPE: integer

ACCESS: RO

DESCRIPTION: IQ variable represents the value of the measured torque current component. The control module uses this value for vector control of AC motors.

SCALING FACTOR: *Kif*. For more details about scaling factors see par.3.7.

SETTING: IQ is updated at each current control loop sampling time interval (i.e. the fast sampling time, see CLPER parameter). It is obtained from the measured motor currents, as projected on the torque axis of the motor (for vector control schemes).

4.3.4.7. IQREF. Q axis reference current

ADDRESS: 0x022F

NUMERICAL TYPE: integer

ACCESS: RO

DESCRIPTION: IQREF variable represents the value of the reference torque current component. The control module uses this value for vector control of AC motors.

SCALING FACTOR: *Kif*. For more details about scaling factors see par.3.7.

SETTING: IQREF is updated at each current control loop sampling time interval (i.e. the fast sampling time, see CLPER parameter). It is obtained as the output of the outer loop connected to the current controller on the Q axis, i.e.: speed controller (if configured in the motion structure); position / user controller (if no speed controller was configured in the motion structure); reference generator (if no speed or position / user loops were configured).

4.3.4.8. POSERR. Position error

ADDRESS: 0x022A

NUMERICAL TYPE: integer

ACCESS: RO

DESCRIPTION: POSERR variable represents the value of motor position error, computed as the difference between the reference and the measured position of the motor.

SCALING FACTOR: *Kpf*. For more details about scaling factors see par.3.7.

SETTING: POSERR is updated at each speed / position control loop sampling time interval (i.e. the slow sampling time, see SLPER parameter). The control module uses these values for position control loop implementation. (See par. 3.5.3 for details about position controllers).

4.3.4.9. SPDERR. Speed error

ADDRESS: 0x022E

NUMERICAL TYPE: integer

ACCESS: RO

DESCRIPTION: SPDERR variable represents the value of motor speed error, computed as the difference between the reference and the measured speed of the motor.

SCALING FACTOR: *Kvf*. For more details about scaling factors see par.3.7.

SETTING: SPDERR is updated at each speed / position control loop sampling time interval (i.e. the slow sampling time, see SLPER parameter). The control module uses these values for speed control loop implementation. (See par. 3.5.2 for details about speed controllers).

4.3.4.10. SPDREF. Reference speed

ADDRESS: 0x022B

NUMERICAL TYPE: integer

ACCESS: RO

DESCRIPTION: SPDREF variable represents the value of the motor speed reference.

SCALING FACTOR: Kvf. For more details about scaling factors see par.3.7.

SETTING: SPDREF is updated at each speed control loop sampling time interval (i.e. the slow sampling time, see SLPER parameter). It is obtained as the output of the outer loop connected to the speed controller, i.e. position / user controller, or reference generator (if no position / user loops were configured).

4.3.4.11. THETA. Field position angle

ADDRESS: 0x0224

NUMERICAL TYPE: integer

ACCESS: RO

DESCRIPTION: THETA variable represents the value of the electric angle position.

SCALING FACTOR: $\text{Kth} = \frac{32767}{180}\left[\frac{\text{bits}}{\text{deg}}\right]$

SETTING: THETA is updated at each current control loop sampling time interval (i.e. the fast sampling time, see CLPER parameter). The control module uses these values for vector control of AC motors, for coordinate transformations.

4.3.4.12. UDREF. D axis reference voltage

ADDRESS: 0x0235

NUMERICAL TYPE: integer

ACCESS: RO

DESCRIPTION: UDREF variable represents the value of the reference field voltage component. The control module uses this value for vector control of AC motors.

SCALING FACTOR: *Kuf*. For more details about scaling factors see par.3.7.

SETTING: UDREF is updated at each current control loop sampling time interval (i.e. the fast sampling time, see CLPER parameter). It is obtained as the output of the inner loop connected to the D axis, i.e.: D-axis current controller (if configured in the motion structure); field control block (if no current controllers are configured in the motion structure).

REMARKS: This variable is not used in the case of DC motors control.

4.3.4.13. UQREF. Q axis reference voltage

ADDRESS: 0x0232

NUMERICAL TYPE: integer

ACCESS: RO

DESCRIPTION: UQREF variable represents the value of the reference field voltage component. The control module uses this value for vector control of AC motors.

SCALING FACTOR: *Kuf*. For more details about scaling factors see par.3.7.

SETTING: UQREF is updated at each current control loop sampling time interval (i.e. the fast sampling time, see CLPER parameter). It is obtained as the output of the inner loop connected to the Q axis, i.e.: Q-axis current controller (if configured in the motion structure); speed controller (if no current controllers are configured in the motion structure); position / user controller (if neither current, nor speed controller were configured in the motion structure); reference generator (if no current, speed or position / user loops were configured).

4.3.5. Reference related TML variables

4.3.5.1. ATIME. Absolute system time

ADDRESS: 0x02C0

NUMERICAL TYPE: long

ACCESS: RO

DESCRIPTION: ATIME variable represents the value of the motion system actual time.

SCALING FACTOR: *Ktf*. For more details about scaling factors see par.3.7.

SETTING: ATIME is updated at each speed / position control loop sampling time interval (i.e. the slow sampling time, see SLPER parameter). ATIME is incremented with 1 at each update. This variable is used to implement time-based TML events.

4.3.5.2. APOS2 Master actual position

ADDRESS: 0x081C

NUMERICAL TYPE: long

ACCESS: R/W

DESCRIPTION: APOS2 variable contains the value of the actual position of the master.

SETTING: APOS2 is computed based on master speed MSPD at each speed / position control loop sampling time interval.

APOS2 can be read at any moment using TML instructions or the data tracer capabilities of TML environment.

4.3.5.3. BETA. Master synchronization

ADDRESS: 0x0293

NUMERICAL TYPE: long

ACCESS: R/W

DESCRIPTION: BETA variable it is used by the slave axes to synchronize with the master, when the reference is send via a communication channel.

SETTING: BETA contains the reference send by the master. If the content of BETA it is not changed during one slow loop the slave inserts a variable delay in the slow loop (maximum one sampling). BETA is not used when the master send the reference from the second encoder.

4.3.5.4. CDREF. Reference increment for a contour segment

ADDRESS: 0x02A4

NUMERICAL TYPE: fixed – fixed point 32 bits: (16 bits signed integer part).(16 bits fractional part)

ACCESS: RO

DESCRIPTION: CDREF variable is used for contouring references and represents the reference increment to be used for the actual contouring segment.

SETTING: CDREF is updated with the last value of the reference increment, as defined with a SEG instruction.

4.3.5.5. CTIME. Number of samplings for a contour segment

ADDRESS: 0x02A6

NUMERICAL TYPE: unsigned integer

ACCESS: RO

DESCRIPTION: CTIME variable is used for contouring references and represents the number of samplings to be used for the actual contouring segment.

SCALING FACTOR: *Ktf*. For more details about scaling factors see par.3.7.

SETTING: CTIME is updated with the last value of the number of samplings, as defined with a SEG instruction.

4.3.5.6. MREF. Reference send from master

ADDRESS: 0x02AA

NUMERICAL TYPE: long

ACCESS: R/W

DESCRIPTION: MREF variable stores the value of the master position, in electronic gearing multi-axis configurations. The master updates MREF variable on all slave axes, once at each position / speed loop sampling period. The value written into MREF may represent:

- The master actual position e.g. master APOS value, if bit 15 is set to 1 in OSR register
- The master target position e.g. master TPOS value, if bit 15 is set to 0 in OSR register (default)

On slave axes, the target position is computed with the following formula:

New TPOS = old TPOS + (MREF – MPOS0) x GEAR

where, MPOS0 represents the previous MPOS sent by the master.

SETTING: MREF is automatically set and updated by the master. The slave axes use MREF to compute their new target position. The target position increment is computed by each slave axis as the master position increment x gear ratio given by the GEAR parameter. MREF variable is used only in slave mode (if MODE GSx, x = 0,1,2,3 instruction was executed).

4.3.5.7. MSPD. Master speed

ADDRESS: 0x0820

ACCESS: RO

NUMERICAL TYPE: integer

DESCRIPTION: MSPD variable stores the value of the master speed, in electronic gearing.

SCALING FACTOR: *Kvf_master*. For more details about scaling factors see par. 3.7.

SETTING: MSPD is computed by the slave axis, at each speed / position control loop sampling time interval. If the master sends position information via communication channels the speed is computed as follows:

MSPD=MREF-MPOS0

If the slave axis receives position reference from second encoder, the speed is computed as the difference between two consecutive captured values from timer 4.

4.3.5.8. RPOS. Relative position for event tests

ADDRESS: 0x02BA

NUMERICAL TYPE: long

ACCESS: R/W

DESCRIPTION: RPOS variable represents the difference between the actual position of the motor APOS and the origin of relative position measurement, stored in variable POS0.

SETTING: RPOS is updated at each speed / position control loop sampling time interval (i.e. the slow sampling time, see SLPER parameter). It is used in relative position events tests, in order to decide if an event must be set or not (in RPO or RPU instructions). Also, it can be used to test if the motor has reached a specific relative position as compared with a reference point.

4.3.5.9. RTIME. Relative time for event tests

ADDRESS: 0x02C2

NUMERICAL TYPE: long

ACCESS: RO

DESCRIPTION: RTIME variable represents the difference between the actual motion system absolute time ATIME and the origin of the relative time measurement, stored in variable TIME0.

SCALING FACTOR: *Ktf*. For more details about scaling factors see par.3.7.

SETTING: RTIME is updated at each speed / position control loop sampling time interval (i.e. the slow sampling time, see SLPER parameter). It can be used to test the reaching of a specific relative delay as compared with a reference time moment.

4.3.5.10. TACC. Target acceleration

ADDRESS: 0x02B6

NUMERICAL TYPE: fixed – fixed point 32 bits: (16 bits signed integer part).(16 bits fractional part)

ACCESS: RO

DESCRIPTION: TACC variable represents the value of the reference acceleration.

SCALING FACTOR: *Kaf*. For more details about scaling factors see par.3.7.

SETTING: TACC is updated at each speed / position control loop sampling time interval (i.e. the slow sampling time, see SLPER parameter). It is computed in the reference generator module, for position / speed profile or contouring modes. TACC is used by the acceleration feedforward term in the speed controller.

4.3.5.11. TPOS. Target position

ADDRESS: 0x02B2

NUMERICAL TYPE: long

ACCESS: RO

DESCRIPTION: TPOS variable represents the value of the reference position.

SCALING FACTOR: *Kpf*. For more details about scaling factors see par.3.7.

SETTING: TPOS is updated at each speed / position control loop sampling time interval (i.e. the slow sampling time, see SLPER parameter). It is computed in the reference generator module for all positioning modes and also during speed profile or contouring modes.

4.3.5.12. TREF. Target reference

ADDRESS: 0x02AE

NUMERICAL TYPE: long / fixed – fixed point 32 bits: (16 bits signed integer part).(16 bits fractional part)

ACCESS: RO

DESCRIPTION: TREF variable represents the value of the reference value, for all internal reference types, in any of the accepted motion modes (position, speed, torque or voltage mode).

SETTING: TREF is updated at each speed / position control loop sampling time interval (i.e. the slow sampling time, see SLPER parameter). It is computed in the reference generator module and is used in the control module as an input to the outer controller of the motion structure.

4.3.5.13. TSPD. Target speed

ADDRESS: 0x02B4

NUMERICAL TYPE: fixed – fixed point 32 bits: (16 bits signed integer part).(16 bits fractional part)

ACCESS: RO

DESCRIPTION: TSPD variable represents the value of the reference speed.

SCALING FACTOR: *Kvf*. For more details about scaling factors see par.3.7.

SETTING: TSPD is updated at each speed / position control loop sampling time interval (i.e. the slow sampling time, see SLPER parameter). It is computed in the reference generator module, for all speed modes and also during position profile or contouring modes.

4.3.6. Motion Language variables

4.3.6.1. PROD. Result of a TML multiply operation

ADDRESS: 0x030E

NUMERICAL TYPE: 48 bits

ACCESS: R/W

DESCRIPTION: PROD variable contains the result of the last TML multiply operation performed, including the left or right shift.

SETTING: PROD is updated after each TML multiply operation. Either the 32MSB of the PROD or the 32LSB may be copied into a 32-bit variable using a 32-bit assignment instruction. Use PROD(H) mnemonic to access the 32MSB of the result and PROD or PROD(L) for the 32LSB.

4.3.7. General-purpose pre-defined user variables

4.3.7.1. VAR_I1, VAR_I2. Integer predefined user variables

ADDRESS: VAR_I1: 0x0366; VAR_I2: 0x0367

NUMERICAL TYPE: integer

ACCESS: RW

DESCRIPTION: VAR_I1 and VAR_I2 variables represent predefined integer variables used by MotionChip II during system setup, i.e. interrupts service routines.

SETTING: You can set and retrieve the two variables anywhere in a TML instruction accepting an integer variable as an operand, as long they do not interfere with a predefined functionality.

EXAMPLE:

```
VAR_I1 = ASPD;
```

VAR_I2 = 5;

4.3.7.2. VAR_LF. Long / Fixed predefined user variable

ADDRESS: VAR_LF: 0x0368

NUMERICAL TYPE: fixed/long

ACCESS: RW

DESCRIPTION: VAR_LF variable represents a predefined variable used by MotionChip II during system setup, as a fixed or long variable.

SETTING: You can set and retrieve the VAR_LF variable anywhere in a TML instruction accepting a fixed or long variable as an operand, as long they do not interfere with a predefined functionality.

EXAMPLE:

VAR_LF = APOS;

VAR_LF = 10.33 ;

TECHNOSOFT