T E C H N O S O F T

Technosoft is a Third Party of Texas Instruments
supporting the TMS320C24xx and TMS320F281x
DSP controllers of the C2000 family.

To help you get your project started rapidly, Technosoft offers the

DMCode - MS(BL) MATLAB Library

A collection of motion control blocks usable to program Technosoft Motion Control Kits based on the TMS320F2812 DSP controller

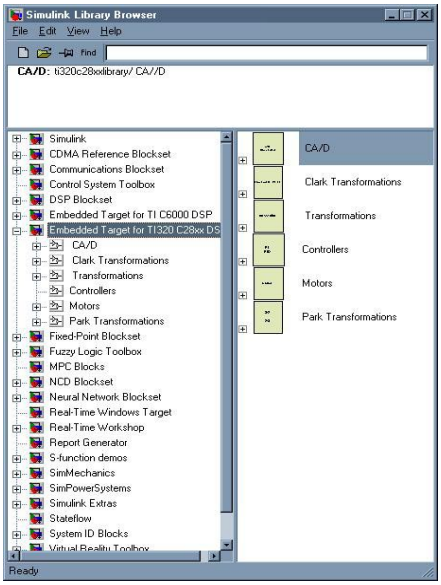
You operate with a **complete** development environment, both at hardware and software levels. The solution offers many significant advantages, based on its typical features:

- Automatic C code generation: eliminates the need to handwrite C and Assembly code
- Visual modeling and simulation: selection of control structure, optimization of control parameters for specific application aspects
- Analysis on the DSP system: validation of the solution on the real control environment
- "Plug-and-play" approach: you get a ready-to-run platform. The first straightforward step is to set up the H/W and S/W; then you can already test that all parts operate properly (simulation, code generation, download and execution on DSP structure)

Extended Target for TI320 C28xx DSP Library

MATLAB-Simulink motion control library used to simulate different control models for synchronous and asynchronous motors.

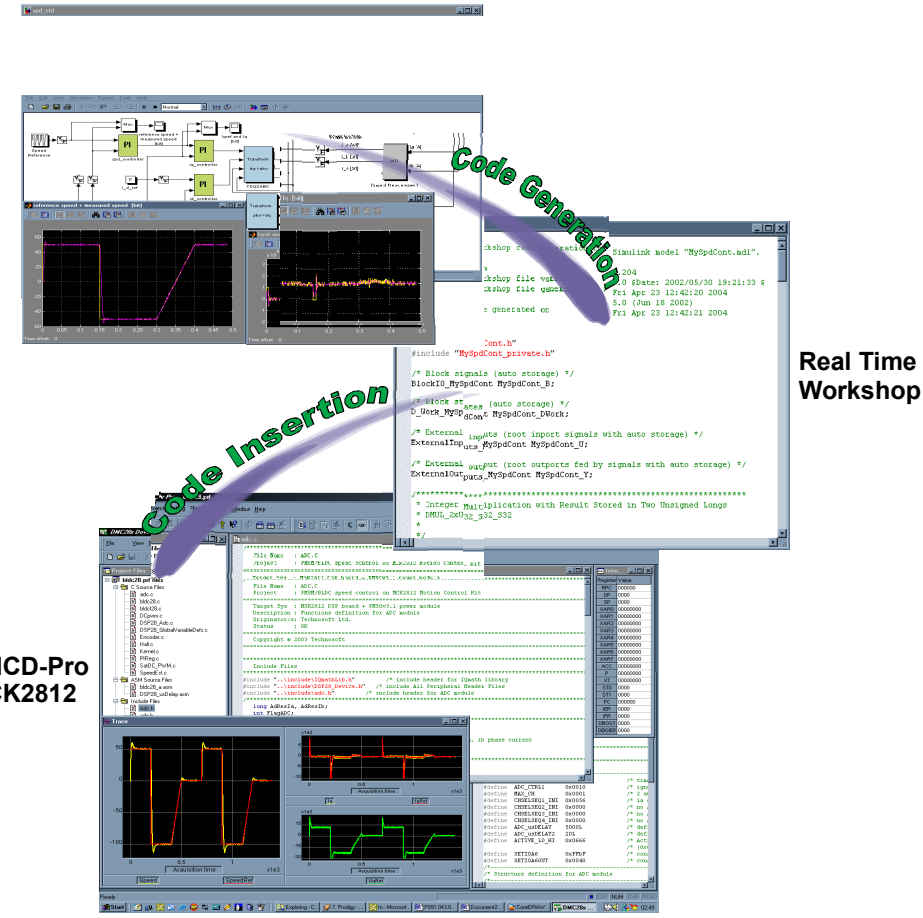
- TI320C28xxLibrary: Simulink library model
  - Blocks for coordinate transformations
  - Blocks for PI and PID controllers
  - Blocks for A/D converters
  - Blocks for electrical motors



- TI320C28xxtarget : target file used to generate C code for real time implementation

Once the system has been simulated, and you are satisfied with its expected behavior, you can **generate** C/C++ code for the control blocks of the system, in order to implement and test it on the 2812 DSP controller.

Matlab-Simulink



DMCD-Pro (Digital Motion Control Developer Pro)

Digital Motion Control Developer for the integrated DSP software development with TMS320F28xx

- Incorporated Debugger Watch Windows
- Memory and I/O registers view / modify
- Integrated source code editor with powerful programming options
- Project Management System
- Tracing Module
- Plug-Ins
- Reference Generator Module
- Application Sources (Optional)

Fully integrated DSP software development environment

- Windows environment with DSP-specific functions gets you started quickly

Incorporated Debugger

- Observe / edit global variables during the debugging process
- Breakpoints, single stepping, stopping and continuing the current program
- View / edit of both data and program memory contents of the DSP target board
- Disassembly window with disassembled instructions with symbolic information for an effective debugging
- View / edit of I/O and internal registers of the DSP processor

Integrated source code editor with powerful programming options

- Each file has its own window, and you can edit several views of the same file
- Advanced search and replace mechanism
- Syntax coloring for C and ASM (TI's assembly syntax is also supported)
- Bookmarks management

Project Management System

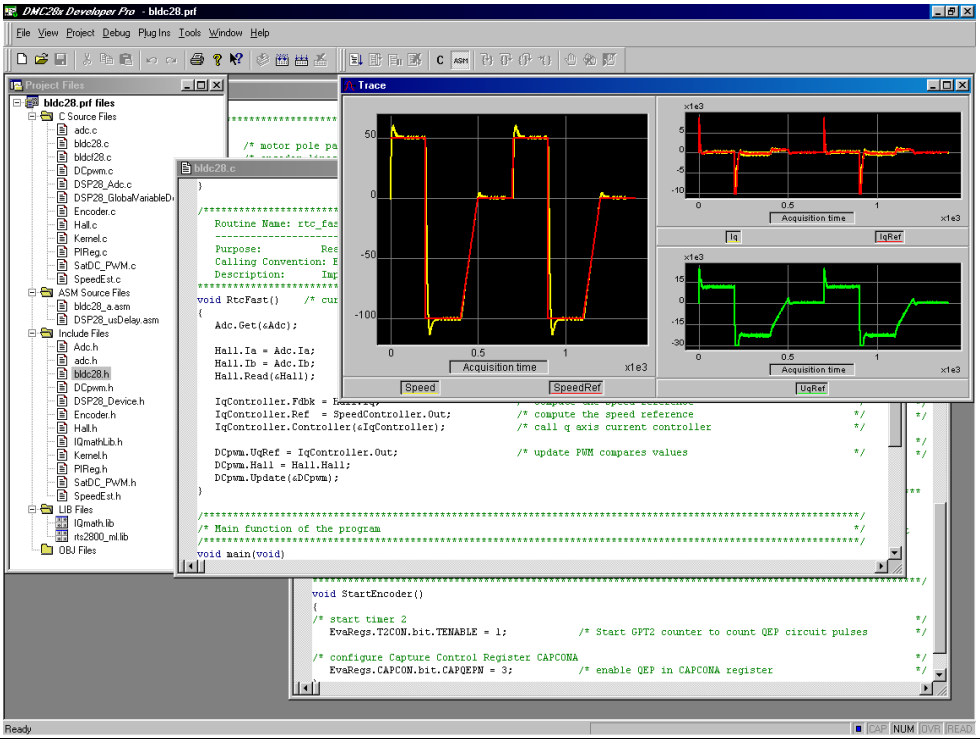
- The system provides an effective way of quickly visualizing, accessing, and manipulating all the project files and their dependencies.
- The result is a concise, highly organized project management system that promotes a very efficient development process.

Tracing module

- The system provides an advanced graphical tool for the analysis and evaluation of motion control applications.
- The program variables may be stored during the real-time execution of the motion, and then uploaded and visualized in the graphical environment.

Plug-ins

- This module allows you to use external module functions in your DSP applications. Basically, you may select one or more external modules from a list containing all available external modules.
- If the reference generator plug-in is included in your application, you may define the motion reference at a high level in DMC Developer, download it, and execute it automatically on the DSP board.



Brushless DC motion demo application

Real time control method of driving the three-phase brushless motor

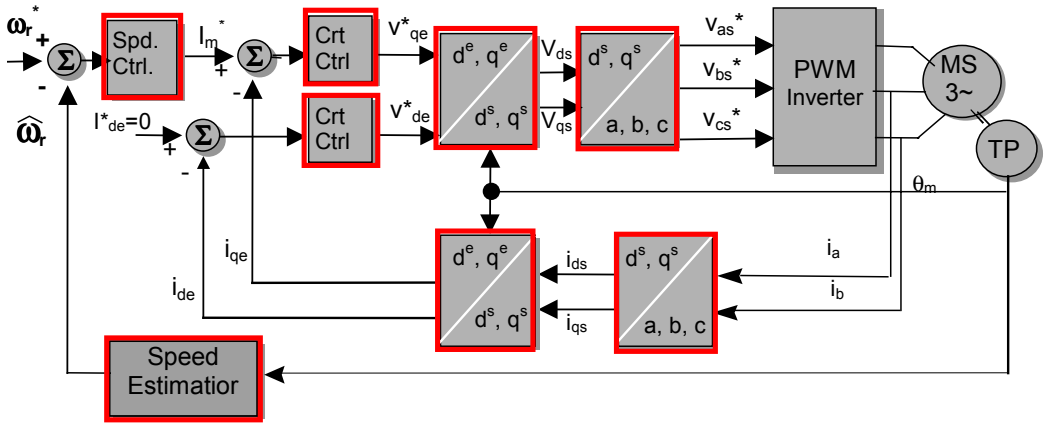
The DMCode-MS(BL) library contains besides the MATLAB system model, complete TMS320F2812 applications structured as projects of the **DMCD-Pro** platform. The complete source files of the application are included in the project structure. The application is the position or the speed control of a brushless motor operating in sinusoidal mode.

The **BL_SPD** application speed control scheme is presented in the figure below. As can be seen, the scheme is based on the measure of two-phase currents and of the motor position. The speed estimator block is a simple encoder position difference block over one sampling period of the speed control loop. The measured phase currents, $\mathbf{i_a}$ and $\mathbf{i_b}$, are transformed into the stator reference frame components, $\mathbf{i_{ds}}$ and $\mathbf{i_{qs}}$. Then, based on the position information, these components are transformed into the direct and quadrature rotor frame components, $\mathbf{i_{de}}$ and $\mathbf{i_{qe}}$. The speed and current controllers are **PI** discrete controllers. The inverse coordinates transformation is used for the computation of the phase voltage references, $\mathbf{v^*_{as}}$, $\mathbf{v^*_{bs}}$ and $\mathbf{v^*_{cs}}$, applied to the inverter, starting from the voltage reference values computed in the *d* and *q* reference frames ($\mathbf{v^*_{de}}$, $\mathbf{v^*_{qe}}$). Thus, the program, based on these reference voltages, directly drives the 6 full-compare PWM outputs of the DSP controller.

The direct current component reference, $\mathbf{i^*_{de}}$, is set to **0** — case corresponding to the motion of the motor in the normal speed range, without considering a possible field weakening operation.

Based on this application that represents a complete ready-to-run motion example, you get all the information you need in order to understand its basic DSP implementation aspects, as well as a convenient starting point for the development of your own applications.

The code is developed only in C language, both for the main structure of the application and for the time-critical parts — as controllers, coordinates transformations, etc.



The complete system can be simulated in the MATLAB-Simulink model provided with the library. The model includes motor, sensors and power converter, as well as digital control part (coordinates transformations, current, speed/position controllers, etc.). For the digital part, IQ-Math fixed-point computations are also included in the model. Blocks that are outlined in the previous figure can be selected to generate the corresponding C-code from MATLAB, and include and execute it in the DSP application project.

Using the advanced features of DMCD-Pro, the **motion reference** can be defined at a high level, from the Windows environment. Calling the **data logger** function allows you to visualize any of the global variables of the program, and to effectively analyze and debug your application (time-critical parts as: controllers, coordinate transformations, etc.).

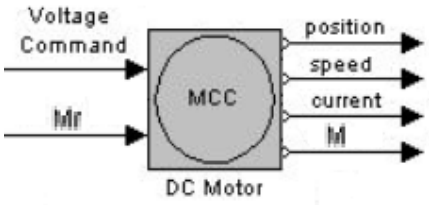
The C code contains all the initialization routines, as well as the main kernel of the application. This offers maximum readability of the software structure. It will be easy for the user to understand the program functionality, as well as to modify, remove or add new functions in this configuration, for the customization of the application for specific cases. The C code also contains all the functions called in the real-time interrupt routines.

Library block description

The Motor Library

Name: DC Motor

Function: Models the operation of a DC (brushed) motor



Parameters:

Input Parameters:

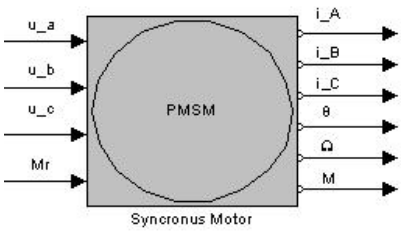
- Voltage command [V]: voltage applied to the motor
- Mr [Nm]: resistive torque applied to the motor

Output Parameters:

- position [rad]: motor position
- speed [rad/s]: motor speed
- current [A]: motor current
- M [Nm]: active motor torque

Name: Synchronous Motor

Function: Models the operation of a 3-phased permanent magnet synchronous motor



Parameters:

Input Parameters:

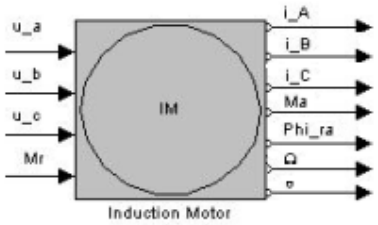
- u_a [V]: supply voltage applied to phase A of the motor
- u_b [V]: supply voltage applied to phase B of the motor
- u_c [V]: supply voltage applied to phase C of the motor
- Mr [V] : resistive torque applied to the motor

Output Parameters:

- i_a [A]: the current on phase A of the motor
- i_b [A]: the current on phase B of the motor
- i_c [A]: the current on phase C of the motor
- θ [rad]: motor position
- Ω [rad/s]: motor speed
- M [Nm]: active motor torque

Name: Induction Motor

Function: Models the operation of a 3-phased induction motor



Parameters:

Input Parameters:

- u_a [V]: supply voltage applied to phase A of the motor
- u_b [V]: supply voltage applied to phase B of the motor
- u_c [V]: supply voltage applied to phase C of the motor
- Mr [V]: resistive torque applied to the motor

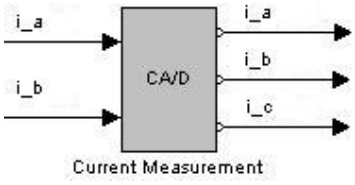
Output Parameters:

- i_a [A]: the current on phase A of the motor
- i_b [A]: the current on phase B of the motor
- i_c [A]: the current on phase C of the motor
- Ma [Nm]: active torque applied to the motor
- Phi_ra [Nm]: rotor flux of the motor
- Ω [rad/s]: motor speed
- θ [rad]: motor position

The AD Library

Name: Current Measurement

Function: Models the operation of the interface measuring the motor currents



Parameters:

Input Parameters:

- i_a [A]: real I_A motor current
- i_b [A]: real I_B motor current

Output Parameters:

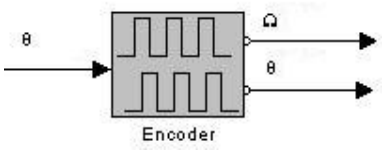
- i_a [bit]: digital value of the I_A current
- i_b [bit]: digital value of the I_B current

- i_c [bit]: digital value of the I_C current

This block simulates the transformation of real currents read from the motor to digital values that are later used in the digital control scheme.

Name: Incremental Encoder

Function: Models the operation of an incremental-encoder type transducer and of the interface measuring the motor position.



Parameters:
Input Parameters:

- θ [rad/s]: real motor position

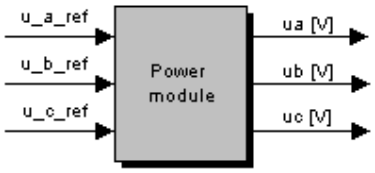
Output Parameters:

- Ω[bit/sampling]: motor speed, estimated from the position variation for 1 sampling period
- θ [bit]: motor position measured in encoder pulses.

This block estimates the motor position measured in encoder pulses, respectively the motor speed measured in encoder pulses, during one sampling period of the speed/position control loop.

Name: Power Module

Function: Models the operation of the power converter used to supply the motor



Parameters:
Input Parameters:

- u_a_ref [bits]: phase A voltage input from the coordinate transformation block TDQ2ABC
- u_b_ref [bits]: phase B voltage input from the coordinate transformation block TDQ2ABC
- u_c_ref [bits]: phase C voltage input from the coordinate transformation block TDQ2ABC

Output Parameters:

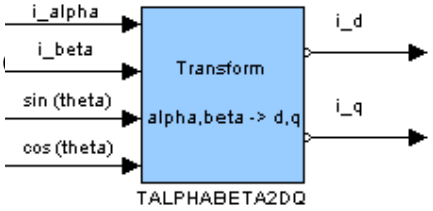
- u_a_ref [V]: real voltage applied to motor phase A
- u_b_ref [V]: real voltage applied to motor phase B
- u_c_ref [V]: real voltage applied to motor phase C

This block simulates the simplified model of the power converter used to supply the motors.

Coordinate Transformation Block Library

Name: TALPHABETA2DQ

Function: Converts the motor current components from (α, β) coordinates to (**d,q**) coordinates.



Parameters:
Input Parameters:

- i_alpha [bit]: current component on the α axis
- i_beta [bit]: current component on the β axis
- sin(theta) [bit]: sine of electric angle θ
- cos(theta) [bit]: cosine of electric angle θ

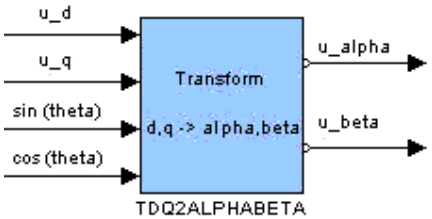
Output Parameters:

- i_d [bit]: current component on the **d** axis
- i_q [bit]: current component on the **q** axis

This block represents the model of current transformation from the fixed (α,β) system to the floating **d-q** system.

Name: TDQ2ALPHABETA

Function: Converts the voltage components from (**d,q**) coordinates to (α,β) coordinates.



Parameters:
Input Parameters:

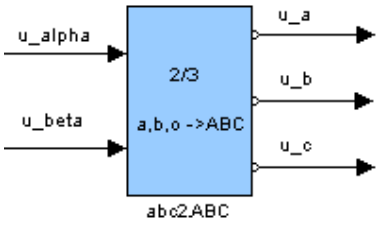
- u_d [bit]: the reference voltage component on the **d** axis
- u_q [bit]: the reference voltage component on the **q** axis
- sin(theta) [bit]: sine of electric angle θ
- cos(theta) [bit]: cosine of electric angle θ

Output Parameters:
- u_alpha [bit]: voltage component on the α axis
- u_beta [bit]: voltage component on the β axis.

This block represents the transformation model of input voltages from the **d-q** floating system to the fixed (α, β) system.

Name: Tabo2ABC

Function: Converts the voltage components from (α,β) coordinates to (**a,b,c**) coordinates.



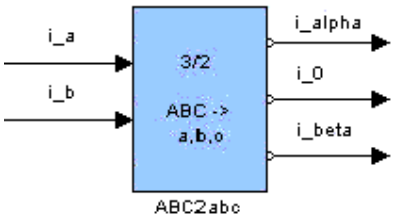
Parameters:
Input Parameters:
- u_alpha [bit]: voltage component on the α axis.
- u_beta [bit]: voltage component on the β axis.

Output Parameters:
- u_a [b]: reference voltage on phase A applied to the voltage amplifier
- u_b [b]: reference voltage on phase B applied to the voltage amplifier
- u_c [b]: reference voltage on phase C applied to the voltage amplifier.

This block represents the voltage transformation model from the fixed (α,β) system to the fixed 3-phased system **a,b,c**.

Name: TABC2abo

Function: Converts the current components from (**a,b,c**) coordinates to (α,β) coordinates.



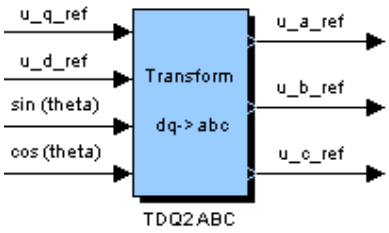
Parameters:
Input Parameters:
- i_a [bit]: current measured on motor phase A
- i_b [bit]: current measured on motor phase B.

Output Parameters:
- i_alpha [bit]: α axis current component
- i_0 [bit]: homopolar current component
- i_beta [bit]: β axis current component

This block represents the current transformation model from the fixed 3-phased **a,b,c** system, to the fixed (α, β) system.

Name: TDQ2ABC

Function: Converts the voltage components from (**d,q**) coordinates to (**a,b,c**) coordinates.



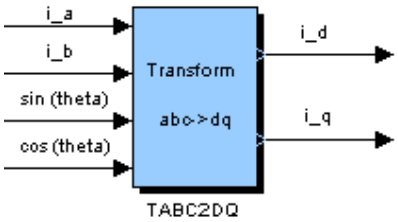
Parameters:
Input Parameters:
- u_q_ref [bit]: reference voltage component on the **q** axis
- u_d_ref [bit]: reference voltage component on the **d** axis.
- sin(theta) [bit]: sine of electric angle θ.
- cos(theta) [bit]: cosine of electric angle θ.

Output Parameters:
- u_a_ref [V]: voltage on phase A applied to the voltage amplifier
- u_b_ref [V]: voltage on phase B applied to the voltage amplifier
- u_c_ref [V]: voltage on phase C applied to the voltage amplifier

This block represents the voltage transformation model from the **d-q** rotor system, to the fixed **a,b,c** system.

Name: TABC2DQ

Function: Converts the current components from (**a,b,c**) coordinates to (**d,q**) coordinates.



Parameters:

Input Parameters:

- i_a [bit]: current on phase A of the motor
- i_b [bit]: current on phase B of the motor
- sin(theta) [bit]: sine of electric angle ***θ***
- cos(theta) [bit]: cosine of electric angle ***θ***

Output Parameters:

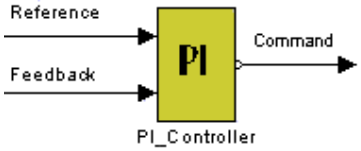
- i_d [b]: **d** axis current component
- i_q [b]: **q** axis current component

This block represents the coordinate transformation model from the fixed 3-phased **a,b,c** system, to the **d-q** rotor system.

Controller Library

Name: PI_controller

Function: Digital controller of the PI type (used for speed or current control, on **d** or **q** axes)



Parameters:

Input Parameters:

- Reference [bit]: imposed reference value. Depending on the controller's use, it can be: position, speed, or current in **d** or **q** axis.
- Feedback [bit]: measured value of the controlled variable. Depending on the controller's use, it can be: position, speed, or current in **d** or **q** axis.

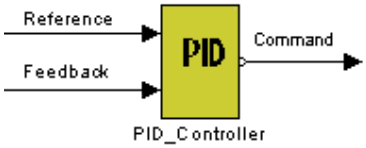
Output Parameters:

- Command [bit]: This parameter can be:
  - A speed reference if the PI controller is used as a position controller
  - A current reference (q component) if the PI is used as a speed controller
  - A voltage reference (d or q) if the PI is used as a current controller (d or q)

This block implements a PI controller. The controller's variable parameters are: **Kp** (proportional component), **Ki** (integral component), as well as the saturation parameters of the controller output.

Name: PID_controller

Function: Digital controller of the PID type (used for position control)



Parameters:

Input Parameters:

- Reference [bit]: imposed reference value
- Feedback [bit]: measured motor position

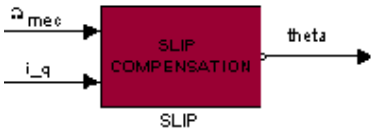
Output Parameters:

- Command [bit]: This parameter can be:
  - speed reference for the speed controller
  - current reference for the current controller

This block implements a PID controller. The controller's variable parameters are: **Kd** (derivative component), **Kp** (proportional component), **Ki** (integral component), sampling rate, integral limit, as well as the saturation parameters of the controller output.

Name: SLIP

Function: Compute the slip between the stator and rotor fields



Parameters:

Input Parameters:

- Ω_{mec} [bit]: motor speed
- i_q [bit]: the current on the q axis of the motor.

Output Parameters:

- theta [bit]: electric angle between phase A and the d axis of the rotor flux.

This block is only used for induction motor control schemes. The adjustable parameters of this block are: **c_slip** (slip increment) and **c_spd** (speed increment).