

Texas Instruments, Inc.
C2000 Systems and Applications

Digital Motor Control

Software Library:
eSMO



2013

V13.1

Contents

INTRODUCTION 3

eSMO..... 4

Introduction

The digital motor control library is composed of C functions (or macros) developed for C2000 motor control users. These modules are represented as modular blocks in C2000 literature in order to explain system-level block diagrams clearly by means of software modularity. The DMC library modules cover nearly all of the target-independent mathematical macros and target-specific peripheral configuration macros, which are essential for motor control.

In the DMC library, each module is separately documented with source code, use, and background technical theory. The DMC library components have been used by TI to provide system-level motor control examples. In the motor control code, all DMC library modules are initialized according to the system specific parameters, and the modules are inter-connected to each other. At run-time the modules are called in order. Each motor control system is built using an incremental build approach, which allows some sections of the code to be built at a time, so that the developer can verify each section of the application one step at a time. This is critical in real-time control applications, where so many different variables can affect the system, and where many different motor parameters need to be tuned.

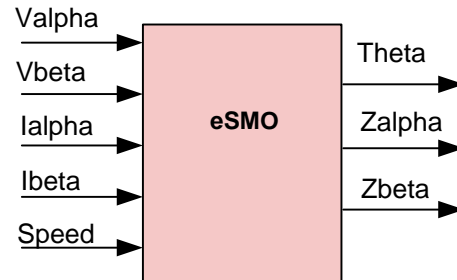
All DMC modules allow users to quickly build, or customize their own systems. The library supports three principal motor types (induction motor, BLDC and PM motors) but is not limited to these motors.

All these modules are provided in the downloadable version of ControlSuite. In addition to these, few additional modules are developed by TI related to the control PMSM motors, for superior performance. The original SMO for sensorless control of PMSM has some scope for improvements and these are covered by enhanced SMO module (eSMO) that provides the following features

1. SMO filter angle compensation
2. Bidirectional speed control

Description

This software module implements a rotor position estimation algorithm for Permanent-Magnet Synchronous Motor (PMSM) based on Sliding-Mode Observer (SMO) and angle compensation due to sliding filter. For detailed information, contact TI sales channel.

**Availability**

C interface version

Module Properties

Type: Target Independent

Target Devices: 28x Fixed or Floating Point

C Version File Names: esmopos.h

IQmath library files for C: IQmathLib.h, IQmath.lib

C Interface

Object Definition

The structure of ESMOPOS object is defined by following structure definition

```
typedef struct {
    _iq Valpha;    // Input: Stationary alpha-axis stator voltage
    _iq Ealpha;    // Variable: Stationary alpha-axis back EMF
    _iq Zalpha;    // Output: Stationary alpha-axis sliding control
    _iq Gsmopos;   // Parameter: Motor dependent control gain
    _iq Estlalpha; // Variable: Estimated stationary alpha-axis stator current
    _iq Fsmopos;   // Parameter: Motor dependent plant matrix
    _iq Vbeta;     // Input: Stationary beta-axis stator voltage
    _iq Ebeta;     // Variable: Stationary beta-axis back EMF
    _iq Zbeta;     // Output: Stationary beta-axis sliding control
    _iq Estlbeta;  // Variable: Estimated stationary beta-axis stator current
    _iq Ialpha;    // Input: Stationary alpha-axis stator current
    _iq IalphaError; // Variable: Stationary alpha-axis current error
    _iq Kslide;    // Parameter: Sliding control gain
    _iq Ibeta;     // Input: Stationary beta-axis stator current
    _iq IbetaError; // Variable: Stationary beta-axis current error
    _iq Kslf;     // Parameter: Sliding control filter gain
    _iq Theta;    // Output: Compensated rotor angle

    _iq E0;       // Input: Current threshold
    _iq smoFreq;  // Input: SMO filter corner frequency
    _iq smoShift; // Output: Compensated rotor angle
    _iq runSpeed; // Input: Running speed of motor
    _iq cmdSpeed; // Input: Commanded speed of motor
    _iq base_wTs; // Input: Incremental angle at base frequency
} ESMOPOS;
```

Module Terminal Variables

Item	Name	Description	Format	Range(Hex)
Inputs	Valpha	stationary d-axis stator voltage	GLOBAL_Q	80000000-7FFFFFFF
	Vbeta	stationary q-axis stator voltage	GLOBAL_Q	80000000-7FFFFFFF
	Ialpha	stationary d-axis stator current	GLOBAL_Q	80000000-7FFFFFFF
	Ibeta	stationary q-axis stator current	GLOBAL_Q	80000000-7FFFFFFF
	E0	Current threshold for smo	GLOBAL_Q	80000000-7FFFFFFF
	runSpeed	Running speed of motor	GLOBAL_Q	80000000-7FFFFFFF
	cmdSpeed	Command speed of motor	GLOBAL_Q	80000000-7FFFFFFF
	base_wTs	Incremental angle at base freq	GLOBAL_Q	80000000-7FFFFFFF
Outputs	Theta	rotor position angle	GLOBAL_Q	00000000-7FFFFFFF (0 – 360 degree)
	Zalfa	stationary d-axis sliding control	GLOBAL_Q	80000000-7FFFFFFF
	Zbeta	stationary q-axis sliding control	GLOBAL_Q	80000000-7FFFFFFF
	smoFreq	Smo filter corner frequency	GLOBAL_Q	80000000-7FFFFFFF
	smoShift	Phase shift due to smo filter	GLOBAL_Q	80000000-7FFFFFFF (0 – 360 degree)
SMOPOS parameter	Fsmopos	$Fsmopos = \exp(-Rs \cdot T/Ls)$	GLOBAL_Q	80000000-7FFFFFFF
	Gsmopos	$Gsmopos = (Vb/Ib) \cdot (1 - \exp(-Rs \cdot T/Ls)) / Rs$	GLOBAL_Q	80000000-7FFFFFFF

eSMO			C Interface	
Internal	Kslide	sliding mode control gain	GLOBAL_Q	80000000-7FFFFFFF
	Kslf	sliding control filter gain	GLOBAL_Q	80000000-7FFFFFFF
	Ealpha	stationary d-axis back EMF	GLOBAL_Q	80000000-7FFFFFFF
	Ebeta	stationary q-axis back EMF	GLOBAL_Q	80000000-7FFFFFFF
	Estlalpha	stationary d-axis estimated current	GLOBAL_Q	80000000-7FFFFFFF
	Estlbeta	stationary q-axis estimated current	GLOBAL_Q	80000000-7FFFFFFF
	lalphaError	stationary d-axis current error	GLOBAL_Q	80000000-7FFFFFFF
	lbetaError	stationary q-axis current error	GLOBAL_Q	80000000-7FFFFFFF

GLOBAL_Q valued between 1 and 30 is defined in the IQmathLib.h header file.

Special Constants and Data types

ESMOPOS

The module definition is created as a data type. This makes it convenient to instance an interface to the sliding-mode rotor position observer of Permanent-Magnet Synchronous Motor module. To create multiple instances of the module simply declare variables of type ESMOPOS.

ESMOPOS_DEFAULTS

Structure symbolic constant to initialize ESMOPOS module. This provides the initial values to the terminal variables as well as method pointers.

Module Usage

Instantiation

The following example instances two ESMOPOS objects
ESMOPOS esmo1, esmo2;

Initialization

To Instance pre-initialized objects
ESMOPOS fe1 = ESMOPOS_DEFAULTS;
ESMOPOS fe2 = ESMOPOS_DEFAULTS;

Invoking the computation macro

```
eSMO_MODULE(&esmo1);
eSMO_MODULE (&esmo2);
```

Example

The following pseudo code provides the information about the module usage.

```
main()
{
    esmo1.Fsmopos = parem1_1;           // Pass parameters to esmo1
    esmo1.Gsmopos = parem1_2;           // Pass parameters to esmo1
    esmo1.Kslide = parem1_3;            // Pass parameters to esmo1
    esmo1.Kslf = parem1_4;              // Pass parameters to esmo1
    esmo2.Fsmopos = parem2_1;           // Pass parameters to esmo2
    esmo2.Gsmopos = parem2_2;           // Pass parameters to esmo2
    esmo2.Kslide = parem2_3;            // Pass parameters to esmo2
    esmo2.Kslf = parem2_4;              // Pass parameters to esmo2
}
```

```
void interrupt_periodic_interrupt_isr()
{
    esmo1.Valpha = voltage_dq1.d;           // Pass inputs to esmo1
    esmo1.Vbeta = voltage_dq1.q;           // Pass inputs to esmo1
    esmo1.Ialpha =current_dq1.d;           // Pass inputs to esmo1
    esmo1.Ibeta =current_dq1.q;           // Pass inputs to esmo1
    esmo1.runSpeed =current_speed1;        // Pass inputs to esmo1
    esmo1.cmdSpeed = command_speed1;       // Pass inputs to esmo1

    esmo2.Valpha = voltage_dq2.d;           // Pass inputs to esmo2
    esmo2.Vbeta = voltage_dq2.q;           // Pass inputs to esmo2
    esmo2.Ialpha =current_dq2.d;           // Pass inputs to esmo2
    esmo2.Ibeta =current_dq2.q;           // Pass inputs to esmo2
    esmo2.runSpeed =current_speed2;        // Pass inputs to esmo2
    esmo2.cmdSpeed = command_speed2;       // Pass inputs to esmo2

    eSMO_MODULE(&esmo1)                   // Call compute macro for esmopos1
    eSMO_MODULE(&esmo2);                   // Call compute macro for esmopos2

    angle1 = esmo1.Theta;                  // Access the outputs of esmopos1
    angle2 = esmo2.Theta;                  // Access the outputs of esmopos2
}
```

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.