Advanced Control Library for Cortex-M4 Core

User Reference Manual

ACLCM4UG
Rev. 1.1
3/2013

freescale.com

The following revision history table summarizes changes contained in this document.

Table 0-1. Revision History

Date	Revision Label	Description
	0	Initial release
03/2013	1.1	Support of additional compilers added

Chapter 1 License Agreement

FREESCALE SEMICONDUCTOR SOFTWARE LICENSE AGREEMENT. This is a legal agreement between you (either as an individual or as an authorized representative of your employer) and Freescale Semiconductor, Inc. ("Freescale"). It concerns your rights to use this file and any accompanying written materials (the "Software"). In consideration for Freescale allowing you to access the Software, you are agreeing to be bound by the terms of this Agreement. If you do not agree to all of the terms of this Agreement, do not download the Software. If you change your mind later, stop using the Software and delete all copies of the Software in your possession or control. Any copies of the Software that you have already distributed, where permitted, and do not destroy will continue to be governed by this Agreement. Your prior use will also continue to be governed by this Agreement.

OBJECT PROVIDED, OBJECT REDISTRIBUTION LICENSE GRANT. Freescale grants to you, free of charge, the non-exclusive, non-transferable right (1) to reproduce the Software, (2) to distribute the Software, and (3) to sublicense to others the right to use the distributed Software. The Software is provided to you only in object (machine-readable) form. You may exercise the rights above only with respect to such object form. You may not translate, reverse engineer, decompile, or disassemble the Software except to the extent applicable law specifically prohibits such restriction. In addition, you must prohibit your sublicensees from doing the same. If you violate any of the terms or restrictions of this Agreement, Freescale may immediately terminate this Agreement, and require that you stop using and delete all copies of the Software in your possession or control.

COPYRIGHT. The Software is licensed to you, not sold. Freescale owns the Software, and United States copyright laws and international treaty provisions protect the Software. Therefore, you must treat the Software like any other copyrighted material (e.g. a book or musical recording). You may not use or copy the Software for any other purpose than what is described in this Agreement. Except as expressly provided herein, Freescale does not grant to you any express or implied rights under any Freescale or third-party patents, copyrights, trademarks, or trade secrets. Additionally, you must reproduce and apply any copyright or other proprietary rights notices included on or embedded in the Software to any copies or derivative works made thereof, in whole or in part, if any.

SUPPORT. Freescale is NOT obligated to provide any support, upgrades or new releases of the Software. If you wish, you may contact Freescale and report problems and provide suggestions regarding the Software. Freescale has no obligation whatsoever to respond in any way to such a problem report or suggestion. Freescale may make changes to the Software at any time, without any obligation to notify or provide updated versions of the Software to you.

NO WARRANTY. TO THE MAXIMUM EXTENT PERMITTED BY LAW, FREESCALE EXPRESSLY DISCLAIMS ANY WARRANTY FOR THE SOFTWARE. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. YOU ASSUME THE ENTIRE RISK ARISING OUT OF THE USE OR PERFORMANCE OF THE SOFTWARE, OR ANY SYSTEMS YOU DESIGN USING THE SOFTWARE (IF ANY). NOTHING IN THIS AGREEMENT MAY BE CONSTRUED AS A WARRANTY OR REPRESENTATION BY FREESCALE THAT THE SOFTWARE OR ANY DERIVATIVE WORK DEVELOPED WITH OR INCORPORATING THE SOFTWARE WILL BE FREE FROM INFRINGEMENT OF THE INTELLECTUAL PROPERTY RIGHTS OF THIRD PARTIES.

INDEMNITY. You agree to fully defend and indemnify Freescale from any and all claims, liabilities, and costs (including reasonable attorney's fees) related to (1) your use (including your sublicensee's use, if permitted) of the Software or (2) your violation of the terms and conditions of this Agreement.

LIMITATION OF LIABILITY. IN NO EVENT WILL FREESCALE BE LIABLE, WHETHER IN CONTRACT, TORT, OR OTHERWISE, FOR ANY INCIDENTAL, SPECIAL, INDIRECT, CONSEQUENTIAL OR PUNITIVE DAMAGES, INCLUDING, BUT NOT LIMITED TO, DAMAGES FOR ANY LOSS OF USE, LOSS OF TIME, INCONVENIENCE, COMMERCIAL LOSS, OR LOST PROFITS, SAVINGS, OR REVENUES TO THE FULL EXTENT SUCH MAY BE DISCLAIMED BY LAW.

COMPLIANCE WITH LAWS; EXPORT RESTRICTIONS. You must use the Software in accordance with all applicable U.S. laws, regulations and statutes. You agree that neither you nor your licensees (if any) intend to or will, directly or indirectly, export or transmit the Software to any country in violation of U.S. export restrictions.

GOVERNMENT USE. Use of the Software and any corresponding documentation, if any, is provided with RESTRICTED RIGHTS. Use, duplication or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of The Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 or subparagraphs (c)(l) and (2) of the Commercial Computer Software--Restricted Rights at 48 CFR 52.227-19, as applicable. Manufacturer is Freescale Semiconductor, Inc., 6501 William Cannon Drive West, Austin, TX, 78735.

HIGH RISK ACTIVITIES. You acknowledge that the Software is not fault tolerant and is not designed, manufactured or intended by Freescale for incorporation into products intended for use or resale in on-line control equipment in hazardous, dangerous to life or potentially life-threatening environments requiring fail-safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines or weapons systems, in which the failure of products could lead directly to death, personal injury or severe physical or environmental damage ("High Risk Activities"). You specifically represent and warrant that you will not use the Software or any derivative work of the Software for High Risk Activities.

CHOICE OF LAW; VENUE; LIMITATIONS. You agree that the statutes and laws of the United States and the State of Texas, USA, without regard to conflicts of laws principles, will apply to all matters relating to this Agreement or the Software, and you agree that any litigation will be subject to the exclusive jurisdiction of the state or federal courts in Texas, USA. You agree that regardless of any statute or law to the contrary, any claim or cause of action arising out of or related to this Agreement or the Software must be filed within one (1) year after such claim or cause of action arose or be forever barred.

PRODUCT LABELING. You are not authorized to use any Freescale trademarks, brand names, or logos.

ENTIRE AGREEMENT. This Agreement constitutes the entire agreement between you and Freescale regarding the subject matter of this Agreement, and supersedes all prior communications, negotiations, understandings, agreements or representations, either written or oral, if any. This Agreement may only be amended in written form, executed by you and Freescale.

SEVERABILITY. If any provision of this Agreement is held for any reason to be invalid or unenforceable, then the remaining provisions of this Agreement will be unimpaired and, unless a modification or replacement of the invalid or unenforceable provision is further held to deprive you or Freescale of a material benefit, in which case the Agreement will immediately terminate, the invalid or unenforceable provision will be replaced with a provision that is valid and enforceable and that comes closest to the intention underlying the invalid or unenforceable provision.

NO WAIVER. The waiver by Freescale of any breach of any provision of this Agreement will not operate or be construed as a waiver of any other or a subsequent breach of the same or a different provision.

Chapter 2 INTRODUCTION

2.1 Overview

This Reference Manual describes *Advanced Control Library for Cortex-M4 Core* for Freescale 32-bit ARM® Cortex™-M4 based Kinetis family of microcontrollers. This library contains optimized functions for the Kinetis family of microcontrollers. The library is supplied in a binary form, which is unique for its ease of integration with user applications. The *Set of General Math and Motor Control Functions for Cortex-M4 Core* must be included in the application project for correct functionality of the Advanced Motor Control Library.

2.2 References

[1] *MCLIBCORETXM4UG - Set of General Math and Motor Control Functions for Cortex M4 Core, User Reference Manual*, Freescale Semiconductor, 2012

2.3 Supported compilers

Advanced Control Library for Cortex-M4 Core (ACLIB) is written in C language using some of the compiller's intrinsics functions in order to directly utilize specific core instructions. The library was built and tested using following compilers:

1. IAR Embedded Workbench® for ARM v. 4.40.2
2. CodeWarrior for MCU, version 10.3, GCC build tools
3. Keil µVision V4.60.0.0

The library is delivered in the library module *Cortex_M4_XX.a*, together with *Set of General Math and Motor Control Functions for Cortex-M4 Core*. The *XX* denotes the particular compiller (CW, IAR, KEIL). The interfaces to the algorithms included in this library have been combined into a single public interface include file, *aclib.h*. This was done to simplify the number of files required for inclusion by application programs. Refer to the specific algorithm sections of this document for details on the software application programming interface (API), definitions, and functionality provided for the individual algorithms.

2.4 Installation

The ACLIB is delivered as a single executable file, with Set of General Math and Motor Control Functions for Cortex-M4 Core. To install the ACLIB on a user computer, run the installation file *CORTEX_M4_FSLESL.exe* and follow the steps that are listed in the user's manual dedicated to Set of General Math and Motor Control Functions for Cortex-M4 Core [1].

2.5 Library integration into the software project

The procedure to include the library in the software project is described separately for each supported compiler in the user's manual dedicated to Set of General Math and Motor Control Functions for Cortex-M4 Core [1]. Special attention needs to be paid when this library is included in the CodeWarrior 10.3 project with ARM® GCC build tools used.

2.6 API definition

The description of each function described in this Advanced Control Library for Cortex-M4 Core user reference manual consists of a number of subsections:

Synopsis

This subsection gives the header files that should be included within a source file that references the function or macro. It also shows an appropriate declaration for the function or for a function that can be substituted by a macro. This declaration is not included in your program; only the header file(s) should be included.

Prototype

This subsection shows the original function prototype declaration with all its arguments.

Arguments

This optional subsection describes input arguments to a function or macro.

Description

This subsection is a description of the function or macro. It explains algorithms being used by functions or macros.

Return

This optional subsection describes the return value, if any, of the function or macro.

Range issues

This optional subsection specifies the ranges of input variables.

Special issues

This optional subsection specifies special assumptions that are mandatory for correct function calculation; for example, saturation, rounding, and so on.

Implementation

This optional subsection specifies whether a call of the function generates a library function call or a macro expansion.

This subsection also consists of one or more examples of the use of the function. The examples are often fragments of code—not completed programs—for illustration purposes.

See also

This optional subsection provides a list of related functions or macros.

Performance

This section specifies the actual requirements of the function or macro in terms of required code memory, data memory, and number of clock cycles to execute.

2.7 Data types

The 32-bit Cortex-M4 core supports two types of two’s-complement data formats:

- Signed integer
- Unsigned integer

The signed and unsigned integer data types are useful for general-purpose computation; they are familiar to the microprocessor and microcontroller programmers. The calculations performed in the library functions utilize fractional data types:

- Signed fractional
- Unsigned fractional

Fractional data types allow powerful numeric and digital-signal-processing algorithms to be implemented. Even though calulations in these fractional data types are not directly supported by Cortex-M4 core architecture, it is not difficult to emulate the calculations by software, and therefore benefit from using the fractional arithmetic.

2.7.1 Signed integer (SI)

This format is used for processing data as integers. In this format, the N-bit operand is represented using the N.0 format (N integer bits). The signed integer numbers lie in the following range:

$$-2^{[N-1]} \leq SI \leq [2^{[N-1]} - 1] \qquad \textbf{\textit{Eqn. 2-1}}$$

This data format is available for bytes, words, and longs. The most negative signed word that can be represented is –32,768 ($8000), and the most negative signed long word is –2,147,483,648 ($80000000).

The most positive signed word is 32,767 ($7FFF), and the most positive signed long word is 2,147,483,647 ($7FFFFFFF).

2.7.2 Unsigned integer (UI)

The unsigned integer numbers are positive only, and they have nearly twice the magnitude of a signed number of the same size. The unsigned integer numbers lie in the following range:

$$0 \le UI \le [2^{[N-1]} - 1] \qquad \textbf{Eqn. 2-2}$$

The binary word is interpreted as having a binary point immediately to the right of the integer's least significant bit. This data format is available for bytes, words, and long words. The most positive 16-bit, unsigned integer is 65,535 ($FFFF), and the most positive 32-bit, unsigned integer is 4,294,967,295 ($FFFFFFFF). The smallest unsigned integer number is zero ($0000), regardless of size.

2.7.3 Signed fractional (SF)

In this format, the N-bit operand is represented using the 1.[N–1] format (one sign bit, N–1 fractional bits). The signed fractional numbers lie in the following range:

$$-1.0 \le SF \le 1.0 - 2^{-[N-1]} \qquad \textbf{Eqn. 2-3}$$

This data format is available for words and long words. For both word and long-word signed fractions, the most negative number that can be represented is –1.0; its internal representation is $8000 (word) or $80000000 (long word). The most positive word is $7FFF ($1.0 - 2^{-15}$); its most positive long word is $7FFFFFFF ($1.0 - 2^{-31}$).

2.7.4 Unsigned fractional (UF)

The unsigned fractional numbers can only be positive, and they have nearly twice the magnitude of a signed number with the same number of bits. The unsigned fractional numbers lie in the following range:

$$0.0 \le UF \le 2.0 - 2^{-[N-1]} \qquad \textbf{Eqn. 2-4}$$

The binary word is interpreted as having a binary point after the MSB. This data format is available for words and longs. The most positive 16-bit, unsigned number is $FFFF, or $\{1.0 + (1.0 - 2^{-[N-1]})\} = 1.99997$. The smallest unsigned fractional number is zero ($0000).

2.8 User common types

Table 2-1. User-defined typedefs in *MCF51_types.h*

Mnemonics	Size — bits	Description
Word8	8	To represent 8-bit signed variable/value.
UWord8	8	To represent 8-bit unsigned variable/value.
Word16	16	To represent 16-bit signed variable/value.
UWord16	16	To represent 16-bit unsigned variable/value.
Word32	32	To represent 32-bit signed variable/value.
UWord32	32	To represent 32-bit unsigned variable/value.
Int8	8	To represent 8-bit signed variable/value.
UInt8	8	To represent 8-bit unsigned variable/value.
Int16	16	To represent 16-bit signed variable/value.
UInt16	16	To represent 16-bit unsigned variable/value.
Int32	32	To represent 32-bit signed variable/value.
UInt32	32	To represent 32-bit unsigned variable/value.
Frac16	16	To represent 16-bit signed variable/value.
Frac32	32	To represent 32-bit signed variable/value.
NULL	constant	Represents NULL pointer.
bool	16	Boolean variable.
false	constant	Represents false value.
true	constant	Represents true value.
FRAC16()	macro	Transforms float value from <–1, 1) range into fractional representation <–32768, 32767>.
FRAC32()	macro	Transforms float value from <–1, 1) range into fractional representation <–2147483648, 2147483648>.

Table 2-2. User-Defined Typedefs in *mclib_types.h*

Name	Structure members	Description
MCLIB_3_COOR_SYST_T	Frac32 f32A Frac32 f32B Frac32 f32C	three phase system
MCLIB_2_COOR_SYST_T	Frac32 f32A Frac32 f32B	two phase system
MCLIB_2_COOR_SYST_ALPHA_BETA_T	Frac32 f32Alpha Frac32 f32Beta	two phase system — alpha/beta
MCLIB_2_COOR_SYST_D_Q_T	Frac32 f32D Frac32 f32Q	two phase system — generic DQ
MCLIB_ANGLE_T	Frac32 f32Sin Frac32 f32Cos	two phase system — sine and cosine components

2.9 Code size and execution time data

This document provides, for each function, the size of the compiled binary code and the execution time of the algorithms for each compiler. The following table summarizes the settings of each supported compiler.

Table 2-3. Compilers settings

Copolymer	Optimization options
IAR Embedded Workbench	Optimization level “High” Optimize for “Speed” “No size constraints” checked All transformations enabled
CodeWarrior for MCU	Optimize most (-O3)
Keil µVision	Optimization: “Level 3(-O3)” “Optimize for Time” checked

Chapter 3 FUNCTION API

3.1 API Summary

Table 3-1. API functions summary

Name	Arguments	Output	Description
ACLIB_TrackObsrv	*Frac32 f32ThetaErr* *ACLIB_TRACK_OBSRV_T * const pudtCtrl*	void	This function calculates the tracking observer for determination angular speed and position of input error functional signal.
ACLIB_PMSMBemfObsrvDQ	*MCLIB_2_COOR_SYST_D_Q_T *pudtIdq* *MCLIB_2_COOR_SYST_D_Q_T *pudtUdq,Frac32 f32Speed* *ACLIB_BEMF_OBSRV_DQ_T * const pudtCtrl*	void	The function calculates the algorithm of back electro-motive force observer in rotating reference frame.

3.2 ACLIB_PMSMBemfObsrvDQ

The function calculates the algorithm of back electro-motive force observer in rotating reference frame.

3.2.1 Synopsis

```
#include ”aclib.h”
void ACLIB_PMSMBemfObsrvDQ(MCLIB_2_COOR_SYST_D_Q_T *pudtIDQ,
MCLIB_2_COOR_SYST_D_Q_T *pudtUDQ, Frac32 f32Speed, ACLIB_BEMF_OBSRV_DQ_T
*pudtCtrl)
```

3.2.2 Prototype

```
void ACLIB_PMSMBemfObsrvDQFC(MCLIB_2_COOR_SYST_D_Q_T *pudtIDQ,
MCLIB_2_COOR_SYST_D_Q_T *pudtUDQ, Frac32 f32Speed, ACLIB_BEMF_OBSRV_DQ_T
*pudtCtrl)
```

3.2.3 Arguments

Table 3-2. Function arguments

Name	In/ Out	Format	Valid range	Description
*pudtIDQ	in	MCLIB_2_COOR_SYST_D_Q_T	N/A	Pointer to structure which contain input signal of d/q current components.
*pudtUDQ	in	MCLIB_2_COOR_SYST_D_Q_T	N/A	Pointer to structure which contain input signal of d/q voltage components.
f32Speed	in/out	Frac32	N/A	Fraction value of electrical speed.
*pudtCtrl	in/out	ACLIB_BEMF_OBSRV_DQ_T	N/A	Pointer to an observer structure, which contains coefficients.

Table 3-3. User types

Typedef	Name	Format	Valid range	Description
ACLIB_BEMF_OBSRV_DQ_T	udtEObsrv.f32D	Frac32	0x80000000... 0x7FFFFFFF	Estimated back-EMF voltage in d-axis.
	udtEObsrv.f32Q	Frac32	0x80000000... 0x7FFFFFFF	Estimated back-EMF voltage in q-axis.
	udtIObsrv.f32D	Frac32	0x80000000... 0x7FFFFFFF	Estimated current in d-axis.
	udtIObsrv.f32Q	Frac32	0x80000000... 0x7FFFFFFF	Estimated current in q-axis.
	udtCtrl.f32ID_1	Frac32	0x80000000... 0x7FFFFFFF	State variable in alpha part of the observer; integral part at step k-1;
	udtCtrl.f32IQ_1	Frac32	0x80000000... 0x7FFFFFFF	State variable in beta part of the observer; integral part at step k-1;
	udtCtrl.f16PropScaled	Frac16	$8000... $7FFF	Observer proportional gain.
	udtCtrl.i16PropShift	Word16	-F...F	Observer proportional gain shift.
	udtCtrl.f16IntegScaled	Frac16	$8000... $7FFF	Observer integral gain.
	udtCtrl.i16IntegShift	Word16	-F...F	Observer integral gain shift.
	f32Error	Frac32	0x80000000... 0x7FFFFFFF	Estimated phase error between real d/q frame system and estimated d/q reference system.
	f16IScaled	Frac16	$8000... $7FFF	Scaling coefficient for current I_{FRAC}
	f16UScaled	Frac16	$8000... $7FFF	Scaling coefficient for voltage U_{FRAC}
	f16WIScaled	Frac16	$8000... $7FFF	Scaling coefficient for angular speed WI_{FRAC}
	f16EScaled	Frac16	$8000... $7FFF	Scaling coefficient for back-emf E_{FRAC}

3.2.4 Availability

This library module is available in the C-callable interface.

This library module is targeted for Cortex-M4 platforms.

3.2.5 Dependencies

List of all dependent files:

- SWLIBS_Typedefs.h

- SWLIBS_Defines.h
- SWLIBS_Inlines.h
- intrinsic.h
- Cortex_M4_IAR.a library
- ACLIB_PMSMBemfObsrvDQ.h
- aclib.h

3.2.6 Description

The estimation method for the rotor position and angular speed is based on the motor mathematical model of interior PMSM motor with an extended electro-motive force function. This is realized in an estimated quasi synchronous reference frame $\gamma\delta$ as depicted in Figure 3-1.

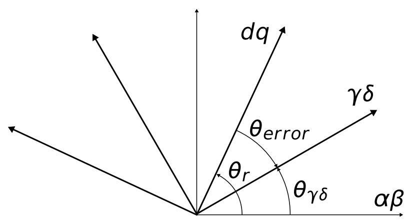

Figure 3-1. Estimated $\gamma\delta$ and real rotor dq synchronous reference frames

The back-EMF observer detects the generated motor voltages induced by the permanent magnets. A tracking observer uses the back-EMF signals to calculate the position and speed of the rotor. The transformed model is then derived as follows:

$$\begin{bmatrix} u_\gamma \\ u_\delta \end{bmatrix} = \begin{bmatrix} R_S + sL_D & -\omega_r L_Q \\ \omega_r L_Q & R_S + sL_D \end{bmatrix} \begin{bmatrix} i_\gamma \\ i_\delta \end{bmatrix} + (\Delta L \cdot (\omega_e i_D - i_Q') + k_e \omega_e) \cdot \begin{bmatrix} -\sin(\theta_{error}) \\ \cos(\theta_{error}) \end{bmatrix} \qquad \textbf{Eqn. 3-1}$$

where

- R_s stator resistance
- L_D, L_Q - D-axis and Q-axis inductance
- k_e back-EMF constant
- ω_e angular electrical speed
- u_D, u_Q stator voltages
- i_D, i_Q stator currents
- s operator of derivative
- i_q' - first derivative of i_q current

The block diagram of the observer in the estimated reference frame is shown in Figure 3-2. The observer compensator is substituted by a standard PI controller. As noted in Figure 3-2, the observer model and the PI controller gains in both axes are identical to each other.

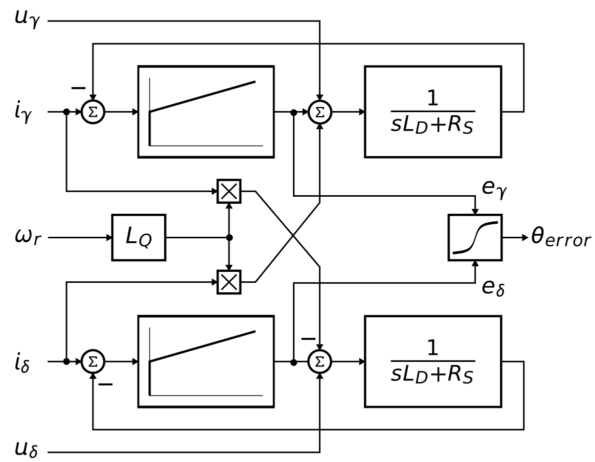

Figure 3-2. Block diagram of proposed Luenberger type stator current observer acting as state filter for back-EMF.

The position estimation can now be performed by extracting the θ_{error} term from the model and adjusting the position of the estimated reference frame to achieve $\theta_{error} = 0$. Because the θ_{error} term is only included in the saliency-based EMF component of both u_γ, u_δ axis voltage equations, the Luenberger based disturbance observer is designed to observe these voltage components u_γ, u_δ. The position displacement information θ_{error} is then obtained from estimated back-EMFs as follows:

$$\theta_{error} = \operatorname{atan}\left(\frac{-u_\gamma}{u_\delta}\right) \qquad \textbf{\textit{Eqn. 3-2}}$$

The estimated position $\hat{\theta}_r$ can be obtained by driving the position of the estimated reference frame to achieve zero displacement $\theta_{error} = 0$. The phase locked loop mechanism can be adopted, where the loop compensator ensures correct tracking of the actual rotor flux position by keeping the error signal θ_{error} to be zeroed, $\theta_{error} = 0$.

A perfect match between the actual and estimated motor model parameters is assumed, and the back-EMF transfer function is simplified as follows

$$\hat{E}_{\gamma\delta}(s) = -E_{\gamma\delta}(s)\cdot\left[\frac{F_c(s)}{sL_D + R_S + F_C(s)}\right] \qquad \textbf{\textit{Eqn. 3-3}}$$

Appropriate dynamic behavior of the back EMFobserver is achieved by placement of the poles of the stator current observer characteristic polynomial. This general method is based on matching the coefficients of the characteristic polynomial with the coefficients of the general second-order system.

The back EMFobserver is a Luenberger type observer with motor model which is realized in fixed point arithmetic and transformed using backward Euler transformation.

$$\boldsymbol{i}_{FRFAC}(k) = U_{FRAC} \cdot \boldsymbol{u}_{FRAC}(k) + E_{FRAC} \cdot \boldsymbol{e}_{FRAC}(k) + WI_{FRAC} \cdot \omega_{eFRAC}(k) \cdot \boldsymbol{i}'_{FRAC}(k) + I_{FRAC} \cdot \boldsymbol{i}_{FRAC}(k-1) \qquad \text{Eqn. 3-4}$$

where

- $\boldsymbol{i}_{FRFAC}(k) = [i_\gamma, i_\delta]$ is fractional representation of stator current vector
- $\boldsymbol{u}_{FRAC}(k) = [u_\gamma, u_\delta]$ is fractional representation of stator voltage vector
- $\boldsymbol{e}_{FRAC}(k) = [e_\gamma, e_\delta]$ is fractional representation of stator back-emf voltage vector
- $\boldsymbol{i}'_{FRFAC}(k) = [i_\delta, -i_\gamma]$ is fractional representation of complementary stator current vector
- $\omega_{FRFAC}(k)$ is fractional representation of angular speed

Scaling coefficients relating to maximal values are expressed as

$$U_{FRAC} = \frac{\Delta T_S}{L_D + \Delta T_S R_S} \cdot \frac{U_{MAX}}{I_{MAX}} \qquad \textbf{Eqn. 3-5}$$

$$E_{FRAC} = \frac{\Delta T_S}{L_D + \Delta T_S R_S} \cdot \frac{E_{MAX}}{I_{MAX}} \qquad \textbf{Eqn. 3-6}$$

$$WI_{FRAC} = \frac{L_Q \cdot \Delta T_S}{L_D + \Delta T_S R_S} \cdot \Omega_{MAX} \qquad \textbf{Eqn. 3-7}$$

$$I_{FRAC} = \frac{L_D}{L_D + \Delta T_S R_S} \qquad \textbf{Eqn. 3-8}$$

where

- ΔT_S sampling time in [sec]
- I_{MAX} maximal peak current in [A]
- E_{MAX} maximal peak back-emf voltage in [V]
- U_{MAX} maximal peak stator voltage in [V]
- Ω_{MAX} maximal angular speed in [rad/sec]

If a Luenberger type stator current observer is properly designed in the stationary reference frame, the back-EMF can be estimated as a disturbance, produced by the observer controller. This is only valid however if the back-EMF term is not included in the observer model. The observer is actually a closed loop current observer so it acts as a state filter for the back-EMF term.

The estimate of extended EMF term can be derived from Equation 3-3 as follows:

$$-\frac{\hat{E}_{\gamma\delta}(s)}{E_{\gamma\delta}(s)} = \frac{sK_P + K_I}{s^2 L_D + sR_S + sK_P + K_I} \qquad \text{Eqn. 3-9}$$

The observer controller can be designed by comparing the closed loop characteristic polynomial with that of a standard second order system as:

$$s^2 + \frac{K_P + R_S}{L_D} \cdot s + \frac{K_I}{L_D} = s^2 + 2\xi\omega_0 s + \omega_0^2 \qquad \textit{Eqn. 3-10}$$

where

- ω_0 is the natural frequency of the closed loop system (loop bandwith)
- ξ is the loop attenuation.

3.2.7 Returns

The function returns a phase error between the real rotating reference frame and the estimated one.

3.2.8 Range issues

The function works with the 32-bit signed fractional values in the range <-1,1).

3.2.9 Special issues

N/A.

3.2.10 Implementation

Example 3-1. Implementation Code

```
#include "gflib.h"
#include "mclib.h"
#include "aclib.h"

MCLIB_2_COOR_SYST_D_Q_T    mcIdq,mcUdq;
ACLIB_BEMF_OBSRV_DQ_T      acBemfObsrv;
Frac32                     f32Speed;

void main (void)
{

acBemfObsrv.udtIObsrv.f32D = FRAC32(0.0);
acBemfObsrv.udtIObsrv.f32Q = FRAC32(0.0);
acBemfObsrv.udtEObsrv.f32D = FRAC32(0.0);
acBemfObsrv.udtEObsrv.f32Q = FRAC32(0.0);
acBemfObsrv.udtCtrl.f32ID_1= FRAC32(0.0);
acBemfObsrv.udtCtrl.f32IQ_1= FRAC32(0.0);
acBemfObsrv.udtCtrl.f16PropScaled= BEMFOBSRV_DQ_PROP_GAIN_SCALED;
acBemfObsrv.udtCtrl.i16PropShift= BEMFOBSRV_DQ_PROP_GAIN_SHIFT;
acBemfObsrv.udtCtrl.f16IntegScaled= BEMFOBSRV_DQ_INTEG_GAIN_SCALED;
acBemfObsrv.udtCtrl.i16IntegShift   = BEMFOBSRV_DQ_INTEG_GAIN_SHIFT;
acBemfObsrv.f16IScaled             = BEMFOBSRV_DQ_I_SCALED;
acBemfObsrv.f16UScaled             = BEMFOBSRV_DQ_U_SCALED;
acBemfObsrv.f16EScaled             = BEMFOBSRV_DQ_E_SCALED;
```

```
acBemfObsrv.f16WIScaled            = BEMFOBSRV_DQ_WI_SCALED;

}

/* Periodical function or interrupt */
void ISR(void)
{
ACLIB_PMSMBemfObsrvDQ(&mcIdq, &mcUdq, f32Speed, &acBemfObsrv);
}
```

3.2.11 Performance

Table 3-4. Performance of ACLIB_PMSMBemfObsrvDQ function

Code size (words) IAR **CW** **Keil**	584 + 196 (GFLIB_AtanYX) + 220 (GFLIB_ATAN) 648 + 364 (GFLIB_AtanYX) + 220 (GFLIB_ATAN) 574 + 264 (GFLIB_AtanYX) + 78 (GFLIB_ATAN)	
Data size (words)	0	
Execution clock [cycles] IAR/CW/Keil	Min	172 / 209 / 188
	Max	290 / 501 / 298

The algorithm test was performed on the MK40X265VMD100 device. The code ran from flash memory, with the CPU speed set to 100 MHz.

3.3 ACLIB_TrackObsrv

The function calculates tracking observer for determination angular speed and position of input error functional signal.

3.3.1 Synopsis

```
#include"aclib.h"
Frac32 ACLIB_TrackObsrv(Frac32 f32Error, ACLIB_TRACK_OBSRV_T *pudtCtrl)
```

3.3.2 Prototype

```
Frac32 ACLIB_TrackObsrvFC(Frac32 f32Error, ACLIB_TRACK_OBSRV_T
*pudtCtrl)
```

3.3.3 Arguments

Table 3-5. Function arguments

Name	In/ Out	Format	Valid range	Description
f32Error	in	Frac32	0x80000000... 0x7FFFFFFF	input signal representing phase error of system to be estimated
*pudtCtrl	in/out	ACLIB_TRACK_OBSRV_T	N/A	pointer to a tracking observer structure ACLIB_TRACK_OBSRV_T, which contains algorithm coefficients

Table 3-6. User type definitions

Typedef	Name	In/ Out	Format	Valid range	Description
ACLIB_TRACK_OBSRV_T	f32Theta	in/out	Frac32	0x80000000... 0x7FFFFFFF	Estimated position as output of the second numerical integrator
	f32Speed	in/out	Frac32	0x80000000... 0x7FFFFFFF	Estimated speed as output of the first numerical integrator
ACLIB_TRACK_OBSRV_T	f32I_1	in/out	Frac32	0x80000000... 0x7FFFFFFF	State variable in controller part of the observer; integral part at step k-1
	f16PropScale	in	Frac16	$8000... $7FFF	Obsrever proportional gain
	i16PropShift	in	Word16	-F...F	Obsrever proportional gain shift
	f16IntegScale	in	Frac16	$8000... $7FFF	Obsrever integral gain
	i16IntegShift	in	Word16	-F...F	Obsrever integral gain shift
	f16ThScaled	in	Frac16	$8000... $7FFF	Scaling coefficient for output integrator of position
	i16ThShift	in	Word16	-F...F	Scaling coefficient shift for output integrator of position

3.3.4 Availability

This library module is available in the C-callable interface.

This library module is targeted for Cortex-M4 platforms.

3.3.5 Dependencies

List of all dependent files:

- SWLIBS_Typedefs.h
- SWLIBS_Defines.h
- SWLIBS_Inlines.h
- Cortex_M4_IAR.a library
- ACLIB_TrackObsrv.h
- aclib.h

3.3.6 Description

This function calculates the tracking observer algorithm where the phase locked loop mechanism is adopted. It is recommended to call this function at every sampling period. It requires a single input argument as phase error. Such phase tracking observer, with standard PI controller used as the loop compensator, is depicted in Figure 3-3.

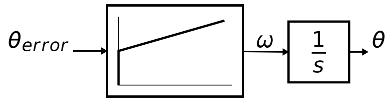

Figure 3-3. Block diagram of proposed PLL scheme for position estimation

Depicted tracking observer structure has the transfer function as

$$\frac{\hat{\theta}(s)}{\theta(s)} = \frac{sK_p + K_i}{s^2 + sK_p + K_i} \qquad \textbf{Eqn. 3-11}$$

where the controller gains K_p and K_i are calculated by comparing the characteristic polynomial of the resulting transfer function to a standard second order system polynomial.

The essential equations for implementation of the tracking observer, according to block scheme in Figure 3-3, are as follows:

$$\begin{aligned} \omega(k) &= K_p \cdot e(k) + \Delta T_S \cdot K_i \cdot e(k) + I(k-1) \\ I(k) &= \Delta T_S \cdot K_i \cdot e(k) + I(k-1) \end{aligned} \qquad \textbf{Eqn. 3-12}$$

$$\theta(k) = \theta(k-1) + \Delta T_S \cdot \omega(k) \qquad \textbf{Eqn. 3-13}$$

In equations Equation 3-12 and Equation 3-13, there are coefficients and quantities that might be greater than one (for example, the actual rotor speed $\omega(k)$) or that are too small to be precisely represented within 16-bit fractional value. Due to this fact, a special transformation has to be carried out in order to be successfully implemented using fractional arithmetic.

$$K_{pFRAC} = \frac{K_p}{\Omega_{MAX}} \qquad \textbf{Eqn. 3-14}$$

$$K_{iFRAC} = \Delta T_S \cdot \frac{K_i}{\Omega_{MAX}} \qquad \textbf{Eqn. 3-15}$$

$$T_{hFRAC} = \Delta T_S \cdot \frac{\Omega_{MAX}}{\Theta_{MAX}} \qquad \textbf{Eqn. 3-16}$$

where the variables of the angle tracking observer are

- $e(k)$ is observer error in step k,
- ΔT_S is the sampling period [s],
- $\omega(k)$ is the actual rotor speed [rad/s] in step k,
- $\theta(k)$ is the actual rotor angle [rad] in step k.

The scaled coefficients which are suitable for implementation on the DSP core are as follows:

$$\mathrm{f16KPScaled} = K_{pFRAC} \cdot 2^{-\mathrm{i16KPShift}} \quad \text{Eqn. 3-17}$$

$$\mathrm{f16KIScaled} = K_{iFRAC} \cdot 2^{-\mathrm{i16KIShift}} \quad \text{Eqn. 3-18}$$

$$\mathrm{f16ThScaled} = T_{hFRAC} \cdot 2^{-\mathrm{i16ThShift}} \quad \text{Eqn. 3-19}$$

3.3.7 Returns

The function returns an estimation of the actual rotor angle as 32-bit fractional value.

3.3.8 Range issues

The function works with the 32-bit signed fractional values in the range <-1,1).

3.3.9 Special issues

N/A.

3.3.10 Implementation

Example 3-2. Implementation Code

```
#include "aclib.h"

ACLIB_TRACK_OBSRV_T acTo;
Frac32           f32ThetaError;
Frac32           f32PositionEstim;

void main (void)
{
acTo.f32Theta     = FRAC32(0.0);
acTo.f32Speed     = FRAC32(0.0);
acTo.f32I_1       = FRAC32(0.0);
acTo.f16PropScale= TRACKOBSRV_PROP_GAIN_SCALED;
acTo.i16PropShift= TRACKOBSRV_PROP_GAIN_SHIFT;
acTo.f16IntegScale= TRACKOBSRV_INTEG_GAIN_SCALED;
acTo.i16IntegShift= TRACKOBSRV_INTEG_GAIN_SHIFT;
acTo.f16ThScaled= TRACKOBSRV_TH_SCALED;
acTo.i16ThShift   = TRACKOBSRV_TH_SHIFT;
}

/* Periodical function or interrupt */
void ISR(void)
{
f32PositionEstim = ACLIB_TrackObsrv(f32ThetaError, &acTo);
}
```

3.3.11 Performance

Table 3-7. Performance of ACLIB_TrackObsrv function

Code size [bytes] IAR/CW/Keil	194 / 228 / 184	
Data size [bytes]	0	
Execution clock [cycles] IAR/CW/Keil	Min	49 / 57 / 52
	Max	49 / 57 / 52

The algorithm test was performed on the MK40X265VMD100 device. The code ran from flash memory, with the CPU speed set to 100 MHz.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

ACLCM4UG
Rev. 1.1, 3/2013