

Fast Current Loop (C28x) Library

This user's guide provides a description of the Fast Current Loop (C28x) software library API (Application Program Interface), which can be used for high bandwidth, inner loop control of AC servo drives with TMS320F2837x, F28004x, F2806x or F28M3x MCUs.

This document also explains the header files that are delivered along with the library. In this version of the library, CLA is not used.

Contents

1	Introduction	2
2	FCL Library Details	2
3	Building and Linking an Application With the Library	6

List of Tables

1	Summary of FCL APIs	2
2	Summary of Common Variables Across the Application and Library	3

Trademarks

Code Composer Studio is a trademark of Texas Instruments.
All other trademarks are the property of their respective owners.

1 Introduction

1.1 Reference Example

An example project using this library, Dual Motor Control Using FCL and Performance Analysis Using SFRA on TMS320F28379D LaunchPad, is available in ControlSUITE that can be referenced for details on how to build and link or integrate the library with an application. The example is built to work with the LAUNCHXL F28379D evaluation platform from TI.

The FCL software library can be found at
controlSUITE\libs\app_libs\motor_control\libs\FCL_SFRA\v01_00_00_00\lib.

The FCL example project can be found at: controlSUITE\libs\app_libs\motor_control\libs\FCL_SFRA\v01_00_00_00\Example

2 FCL Library Details

2.1 API Overview

Table 1 lists the FCL APIs.

Table 1. Summary of FCL APIs

API Function	Description
Uin32 FCL_GetSwVersion(void);	Returns a 32-bit constant; for this version the value returned is 0x00000004
void FCL_initPWM(MOTOR_VARS * m);	Initializes all motor control PWMs for FCL operation, this function is called by the user application during initialization process.
void FCL_PI_Ctrl(MOTOR_VARS * motor);	Performs the PI control as part of the FCL
void FCL_PI_CtrlWrap(MOTOR_VARS * motor);	Wrap-up function called by the user application at the completion of the FCL in PI control mode
void FCL_CC_Ctrl(MOTOR_VARS * motor);	Performs the Complex control as part of the FCL
void FCL_CC_CtrlWrap (MOTOR_VARS * motor);	Wrap-up function called by the user application at the completion of the FCL in Complex control mode
void QepPosEstModule(MOTOR_VARS * motor);	Function to calculate the rotor position from electrical zero of the motor phase A, using QEP data
void FCL_QEP_wrap(MOTOR_VARS * motor);	Function to wrap up the QEP application at the completion of Fast Current Loop routine

2.2 Header Files

2.2.1 Fast_Current_Loop.h

This header file contains general variables and pointers that are used across the application and the library.

Macro FCL_LIB is predefined when building the library and is not defined when the header file is included in the application. This helps applications use the same header file that is used by the library.

For example, in the following pointer declarations, when the header file is included in the library, the pointers are defined as extern, but when the same header file is included in the application the pointers are global. This helps the library work with variables that are common across the application and the software library.

```
#ifndef FCL_LIB
extern
#endif
uint32_t      FF_COMP
= 0
#endif
;

#ifndef FCL_LIB
extern
#endif
uint32_t      COMPLEX_PIC
= 1
#endif
;
```

Table 2 lists the variables needed by the library, which are supposed to be defined by the application. The same information is available in the `Fast_Current_Loop.h` header file delivered with the library. So it is sufficient if applications include the header file.

Table 2. Summary of Common Variables Across the Application and Library

Variable Name	Description or Use
<code>extern uint32_t FF_COMP;</code>	Variable that set up feed forward compensation within <code>FCL_PI_Ctrl()</code>
<code>extern uint32_t COMPLEX_PIC;</code>	Variable that enables complex PI controller within <code>FCL_PI_Ctrl()</code>

2.2.2 motorVars.h

This file also defines the following typedef of a structure that contains all variables, structures and pointers needed to run a motor. It helps to easily pass data between library and example project. The variables of the structure should be initialized by the application before being used by the library.

```
// *****
// Motor variables - for Field Oriented Control
// *****
typedef struct {
    //=====
    // Pointers for various on-chip peripherals used in the library
    //=====
    volatile struct EPWM_REGS * PwmARegs,      // PWM reg for phase A
    * PwmBRegs,      // PWM reg for phase B
    * PwmCRegs;      // PWM reg for phase C

    volatile struct EQEP_REGS * QepRegs;      // QEP reg for QEP sensor

    volatile struct SPI_REGS * SpiRegs;      // SPI reg for DRV830x

    volatile struct CMPSS_REGS * CmpssARegs,   // CMPSS for phase curent A
    * CmpssBRegs,   // CMPSS for phase curent B
    * CmpssCRegs;  // CMPSS for phase curent C

    volatile uint32_t * CurA_PPRESULT, // ADCPPRESULT - phase cur A
    * CurB_PPRESULT, // ADCPPRESULT - phase cur B
    * CurC_PPRESULT; // ADCPPRESULT - phase cur C

    volatile uint16_t * Vdc_AdcResult; // ADC result - DC bus voltage
};
```

```

volatile uint32_t          * pwmCompA,          // CMP reg for phase A pwm
                          * pwmCompB,          // CMP reg for phase B pwm
                          * pwmCompC;          // CMP reg for phase C pwm

//=====================================================

// Transform variables
CLARKE clarke;             // clarke transform
PARK    park;              // park transform
IPARK   ipark;             // inv park transform

// Controller variables
PIDREG3      pid_pos;      // (optional - for eval)
PI_CONTROLLER pi_pos;
PID_CONTROLLER pid_spd;
CURRENT_CONTROLLER cntlr_id;
CURRENT_CONTROLLER cntlr_iq;

FastCurrentLoopPars_t     FCL_Pars;          // FCL params variable

SVGEN svgen;              // SVPWM variable

RMPCTRL rc;               // ramp control

RAMPGEN rg;               // sweep angle generator for forced angle control

PHASEVOLTAGE volt;        // motor voltages

SPEED_MEAS_QEP speed;     // speed calc

QEP qep;                  // QEP variables

DRV8301_Vars drv8301;     // DRV8301 parameters

DRV8305_Vars drv8305;     // DRV8305 parameters

float32 currentAs,        // phase A
        currentBs,        // phase B
        currentCs,        // phase C
        currentSenseScale; // Fbk current sense scale

float32 T;                // sampling time
float32 Speed0;           // prev speed
float32 SFRA_noiseD,      // SFRA_noise D signal
        SFRA_noiseQ,      // SFRA_noise Q signal
        SFRA_noiseW;      // SFRA noise speed signal

float32 BaseInverterVoltage,
        BaseMotorVoltage,
        BaseInverterCurrent,
        BaseMotorCurrent,
        BaseFrequency;

_iq offset_shntA,        // shunt current feedback A - offset @ 0A
     offset_shntB,        // shunt current feedback B - offset @ 0A
     offset_shntC,        // shunt current feedback C - offset @ 0A

VdTesting,               // Vd reference (pu)
VqTesting,               // Vq reference (pu)
IdRef,                   // Id reference (pu)
IqRef,                   // Iq reference (pu)
SpeedRef,                 // speed ref (pu)
angMax,                   // maximum rotation per step
ElecTheta,                // position encoder - elec angle (pu)
MechTheta;                // position encoder - mech angle (pu)

int32 alignCntr,         // rotor alignment time at start up, Id current ramp up

```

```

        alignCnt;    // rotor alignment time cntr

QepStatus_t   lsw;           // Qep status (loop switch)
RunStop_t     RunMotor;     // Motor run/ stop

Uint16  TripFlagDMC,        // motor trip flag
        clearTripFlagDMC,  // clear trip flag
        SpeedLoopPrescaler, // Speed loop pre scalar
        SpeedLoopCount,    // Speed loop counter
        PosSenseReverse,   // position sense reverse flag
        newCmdDRV;         // send new command to DRV
} MOTOR_VARS;

```

2.2.3 FCL_pars.h

This file defines the typedef of a structure that contains all variables needed to set up FCL parameters. The variables of the structure should be initialized by the application before being used by the library.

```

//*****
// FCL parameters
//*****
typedef struct currentLoopPars {
    float32  CARRIER_MID,    // Mid point value of carrier count
            ADC_SCALE,       // ADC conversion scale to pu
            cmidsqrt3;       // internal variable

    float32  tSamp,          // sampling time
            Rd,             // Motor resistance in D axis
            Rq,             // Motor resistance in Q axis
            Ld,             // Motor inductance in D axis
            Lq,             // Motor inductance in Q axis
            Vbase,          // Base voltage for the controller
            Ibase,          // Base current for the controller
            FccD,           // D axis current controller bandwidth in Hz
            FccQ,           // Q axis current controller bandwidth in Hz
            wccD,           // D axis current controller bandwidth
            wccQ,           // Q axis current controller bandwidth
            Vdcbus,         // DC bus voltage
            BemfK,          // Motor Bemf constant
            Wbase;         // Controller base frequency (Motor) in rad/sec
} FastCurrentLoopPars_t;

```

2.3 CLA Resources Used

In this version of the library, CLA is not used.

2.4 Flags Cleared by the Library

The library is constructed as functional module for certain specific task and the task of clearing the flags are outside the scope of the library. The application project using this library should address the task of clearing the flags.

2.5 Application Dependencies

The user application must perform initializations defined in this section for the library to be properly operational.

As shown in the example, all the parameters must be initialized before enabling any interrupts in the application initialization phase.

2.5.1 Initializing Current Loop Parameters for the Library

The following function, provided in the example code, initializes FCL_Pars, referred to in [Section 2.2.3](#).

```
fast_current_loop_vars_init_MOTOR_1();
```

```
fast_current_loop_vars_init_MOTOR_2();
```

2.5.2 Initializing PWM and PWM Access Pointers for the Library

The following code, shown in the example, initializes the PWM modules for the FCL library and sets the PWM access pointers for the library. This makes the library more portable, but it adds a slight cycle count during the execution of the library.

```
motor1.PwmARegs = &EPwm1Regs; // set up EPWM for motor 1 phase A
motor1.PwmBRegs = &EPwm2Regs; // set up EPWM for motor 1 phase B
motor1.PwmCRegs = &EPwm3Regs; // set up EPWM for motor 1 phase C

FCL_initPWM(&motor1);
```

2.5.3 Initializing the ADC Int Flag and ADC PPB Result Register Pointers for the Library

The following code, shown in the example, initializes the PWM modules for Fast control loop library and sets the PWM access pointers for the library. This makes the library more portable but adds a slight cycle count during the execution of library.

```
motor1.CurA_PPRESULT = &(AdccResultRegs.ADCPPB1RESULT.all); //motor 1 current A
motor1.CurB_PPRESULT = &(AdcbResultRegs.ADCPPB1RESULT.all); //motor 1 current B
motor1.CurC_PPRESULT = &(AdcaResultRegs.ADCPPB1RESULT.all); //motor 1 current C
motor1.Vdc_AdcResult = &VFB_DC1;
```

2.5.4 Initializing the EQEP Access Pointer for the Library

The following code, shown in the example, initializes the EQEP registers pointer for the library to access.

```
motor1.QepRegs = &EQep1Regs; // set up EQEP for motor 1
```

3 Building and Linking an Application With the Library

The provided example with the library should help users integrate the library into an application running from flash/RAM. The appropriate linker command files are also provided with the example project.

The library is built with the v17.9.0.STS tool chain and the v210 controlSUITE device support package for the F2837xD, in Code Composer Studio™ v7 IDE.

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ("TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources 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.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products (<http://www.ti.com/sc/docs/stdterms.htm>), [evaluation modules](#), and [samples](http://www.ti.com/sc/docs/sampterm.htm) (<http://www.ti.com/sc/docs/sampterm.htm>).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2018, Texas Instruments Incorporated