Fast Current Loop Library

Ramesh T Ramamoorthy, Santosh Kumar Athuru

ABSTRACT

This reference guide provides a description of the fast current loop software library API (application program interface), which can be used for high-bandwidth, inner-loop control of AC servo drives with TMS320F2837x MCUs.

This document also explains the header files that are delivered with the library, and provides information on which CLA resources are used by the library and which PIE flags are cleared by the library.

Contents

List of Tables

1 Introduction

1.1 Reference Example

Use this guide in conjunction with the *High Bandwidth Current Control of Sensored FOC of 3-Phase PMSM USING F2837x* evaluation guide. This guide is in the controlSUITE™ at \controlSUITE\libs\app_libs\motor_control\libs\FCL\v02_00_00_00\Examples\IDDKv2_PM_Servo_F2837x_FCL_Example01\~Docs\FCL_Sensored FOC of PMSM_IDDK_v2.pdf.

An example is provided with the library 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 IDDK evaluation platform and F2837x Control Card from TI.

When the FCL software package is installed, the FCL software library can be found at controlSUITE\libs\app_libs\motor_control\libs\FCL\v02_00_00_00\lib The FCL example project can be found at: controlSUITE\libs\app_libs\motor_control\libs\FCL\ v02_00_00_00\Examples

2 FCL Library Details

2.1 API Overview

Table 1 lists the FCL APIs.

Table 1. Summary of FCL APIs

API Function	Description
Uint32 FCL_GetSwVersion(void);	Returns a 32-bit constant; for this version the value returned is 0x00000002
void FCL_Complex_Ctrl(void);	Performs the Complex control as part of the FCL
void FCL_PI_Ctrl(void);	Performs the PI control as part of the FCL
void FCL_PI_CtrlWrap(void);	Wrap-up function called by the user application at the completion of the FCL in PI control mode
void FCL_QEP_wrap(void);	Called by the user application to handle the QEP feedback completion. This function is used only in FCL_LEVEL2.
void FCL_Complex_CtrlWrap (void);	Wrap-up function called by the user application at the completion of the FCL in Complex control mode
void FCL_initPWM(volatile struct EPWM_REGS *ePWM);	Initializes PWMs for the FCL operation, this function is called by the user application during the initialization or setup process.
void FCL_ControllerReset(void);	Called to reset the FCL variables and is useful when the user wants to stop and restart the motor.

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.

controlSUITE, Code Composer Studio are trademarks of Texas Instruments.
All other trademarks are the property of their respective owners.

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.

```
#ifdef FCL_LIB
extern
#endif
volatile struct EPWM_REGS *PwmARegs, *PwmBRegs, *PwmCRegs;
```

This file also defines the following typedef of a structure used by the library, but the variables of the structure are initialized by the application, as shown in the provided example.

```
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
            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;
```

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
extern volatile struct EPWM_REGS *PwmARegs, *PwmBRegs, *PwmCRegs;	Pointers to the Motor A, B, and C phase controlling PWM channels on the IDDK. These pointers must be initialized by the application before using the library.
extern volatile union ADCINTFLG_REG *AdcIntFlag;	Pointer to the current sensing ADC INT FLAG register. This pointer must be initialized by the application, as shown in the example, before using the library.
extern volatile union ADCPPB1RESULT_REG *CurA_PPBRESULT, *CurB_PPBRESULT;	Pointers to the current sensing ADC PPB Result registers. These pointers must be initialized by the application, as shown in the example, before using the library.
extern Uint16 lsw;	Loop switch information controlled by both the library and the application
extern QEP qep1;	QEP feedback information accessed by both the application and the library
extern FCL_PI_CONTROLLER pi_iq;	PI IQ controller information accessed and handled by the CLA tasks and CPU inside the library and by CPU in the application
extern FCL_PI_CONTROLLER pi_id;	PI ID controller information accessed by both the library and the application
extern SVGEN svgen1;	Space Vector variables generated by the library are stored here.
extern RAMPGEN rg1;	—
extern SPEED_MEAS_QEP speed1;	—
extern FastCurrentLoopPars_t FCL_Pars;	Current Loop parameter constants that are to be initialized by the application. A reference function is provided in the example provided with the library.

2.2.2 Fast_Current_Loop_qep.h

This file defines the following typedef of a CLA pointer used in the library, and also defines the pointer variable as extern.

```
typedef union{
volatile struct EQEP_REGS *ptr; //Aligned to lower 16-bits
Uint32 pad; //32-bits
}CLA_QEP_PTR;
```

This file should be included in the user application with the Fast_Current_Loop.h file. Table 3 defines the common CLA variables across the library and the application.

Table 3. Summary of Common CLA Variables Across the Application and Library

Variable Name	Description or Use
extern CLA_QEP_PTR ClaQep;	Pointer to the EQEP module registers used by the application for feedback control This pointer is to be initialized by the application before using the library, as shown in the provided example.

2.2.3 fcl_PI.h

This file defines the following typedef of PI variables used in the library.

```
typedef struct {  float  Ref;            // Input: reference set-point
                  float  Fbk;            // Input: feedback
                  float  Err;            // Output : error
                  float  Out;            // Output: controller output
                  float  CarryOver;      // Output : carrier over for next iteration
                  float  Kp;             // Parameter: proportional loop gain
                  float  Ki;             // Parameter: integral gain
                  float  Kerr;           // Parameter: gain for latest error
                  float  KerrOld;        // Parameter: gain for prev error
                  float  Umax;           // Parameter: upper saturation limit
                  float  Umin;           // Parameter: lower saturation limit
                } FCL_PI_CONTROLLER;
```

2.3 *CLA Resources Used*

In this version of the library, the CLA resources in Table 4 are used and are unavailable for the user applications when using the provided software library.

Table 4. Summary of CLA Resources Used by the Library

FLC Controller	CLA Tasks Used
PI controller	CLA TASK1, CLA TASK2, and CLA TASK4
Complex controller	CLA TASK1, CLA TASK3, and CLA TASK4

2.3.1 CLA Task Prototypes

```
__interrupt void Cla1Task1();
__interrupt void Cla1Task2();
__interrupt void Cla1Task3();
__interrupt void Cla1Task4();
```

All the above tasks are declared and defined in the library. The assignment of the tasks to the appropriate CLA vectors is done in the user application.

The example provided with the library shows how to assign the tasks. The relevant code snippet follows.

```
{
    Cla1Regs.MVECT1 = (uint16_t)(&Cla1Task1);
    Cla1Regs.MVECT2 = (uint16_t)(&Cla1Task2);
    Cla1Regs.MVECT3 = (uint16_t)(&Cla1Task3);
    Cla1Regs.MVECT4 = (uint16_t)(&Cla1Task4);
}
```

User applications are free to use the remaining CLA tasks, but these tasks are reserved according to Table 4, depending on the FCL controller option chosen.

2.4 Flags Cleared by the Library

Because the library uses the previously mentioned CLA tasks, it also clears the respective PIE IFR flag bits associated with the tasks.

2.5 Application Dependencies

The user application must initialize and clear the flags 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 2.2.1 and Table 2.

```
fast_current_loop_vars_init();
```

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.

```
    PwmARegs = &EPwm1Regs;
    PwmBRegs = &EPwm2Regs;
    PwmCRegs = &EPwm3Regs;

    FCL_initPWM(PwmARegs);
    FCL_initPWM(PwmBRegs);
    FCL_initPWM(PwmCRegs);
```

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.

```
AdcIntFlag = &AdcaRegs.ADCINTFLG;
CurA_PPBRESULT = &AdcaResultRegs.ADCPPB1RESULT;
CurB_PPBRESULT = &AdcbResultRegs.ADCPPB1RESULT;
```

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.

```
ClaQep.ptr = &EQep1Regs;
```

2.5.5 Configuring and Clearing the CLA TASK1 Trigger

User applications, as shown in the provided example, must be configured to trigger the CLA TASK1 by the same event that triggers the ADC SOC.

The following code in the example shows the initializing of the CLA and setting up of the CLA TASK1 trigger. This must be performed before enabling the PWM clocks.

```
    //initialize CLA for FCL library
    main_cla();

    EALLOW;
    DmaClaSrcSelRegs.CLA1TASKSRCSEL1.bit.TASK1 = CLA_TRIG_EPWM1INT;
```

Similarly, the user application must also clear the event that triggers the CLA task in the user code. This is also shown in the example provided with the library.

```
PwmARegs->ETCLR.bit.INT = 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.

Because the library uses CLA, RAM must be shared across the CPU and CLA. The provided example shows how to do this as well.

The library is built with the v16.12.0 tool chain and the v210 controlSUITE device support package for the F2837xD, in Code Composer Studio™ v7 IDE.