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

List of Tables

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 ControlSUITEthat 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
Uint32 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.

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

#ifdef FCL_LIB
extern
#endif
uint32_t         COMPLEX_PIC
#ifndef FCL_LIB
= 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
extern uint32_t FF_COMP;	Variable that set up feed forward compensation within FCL_PI_Ctrl()
extern uint32_t COMPLEX_PIC;	Variable that enables complex PI controller within FCL_PI_Ctrl()

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_PPBRESULT,  // ADCPPBRESULT - phase cur A
                               * CurB_PPBRESULT,  // ADCPPBRESULT - phase cur B
                               * CurC_PPBRESULT;  // ADCPPBRESULT - 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

    RMPCNTL 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 initializationsdefined 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_PPBRESULT = &(AdccResultRegs.ADCPPB1RESULT.all); //motor 1 current A
motor1.CurB_PPBRESULT = &(AdcbResultRegs.ADCPPB1RESULT.all); //motor 1 current B
motor1.CurC_PPBRESULT = &(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.