STM32 F1 firmware library for easySPIN (L6474) motor driver

Quick guide
STM32 F1 firmware library for easySPIN

- The firmware library allows you to control the L6474 easySPIN microstepping motor driver using STM32 F1 microcontrollers.

- In addition to basic easySPIN low-level configuration functions, the library contains a set of handy motion control routines that help to reduce your programming effort when writing an application (e.g. acceleration/deceleration patterns and user-defined speed profiling).

- Designed for STEVAL-PCC009V1/2 and STM32VLDISCOVERY demonstration control boards, but can be easily configured to suit any STM32-based application.
easySPIN evaluation hardware

- **easySPIN** features can be explored using the EVAL6474H board
  - Direct connection to STEVAL-PCC009V1/2 demonstration boards
  - Voltage range from 8 to 45 V
  - Phase current up to 3 A
  - SPI interface with daisy chain feature allows evaluation of the L6474 in multi-motor applications
  - Status LED

**EVAL6474H**
Development tools supported

• Project files are available for the following development environments:
  • Keil µVision v4.50
  • IAR EWARM 6.30
  • Atollic TrueSTUDIO 2.3.0

• By default, all demo projects are configured to use the ST-LINK in-circuit debugger

• The configuration can be easily changed if another tool is used for application debugging
Project setup customization

When using the library with non-default tools/settings, the following points need to be checked/modified:

- Configure debugger options in *Project settings* for the in-circuit debugger used (type, debugging interface, etc.)

- Select appropriate control board and/or define proper MCU pin mapping (*easyspin_target_config.h, easyspin.h*)

- easySPIN configuration values can be tuned in order to suit particular stepper motor characteristics (*easyspin_target_config.h, main.c*):

```c
/* Customize target stepper-motor specific registers at easySPIN module level */
/* TVAL register setup */
easySPIN_SetParam(easySPIN_TVAL, 0x00);

/* T_FAST register setup */
easySPIN_SetParam(easySPIN_STEP_MODE, easySPIN_STEP_SEL_1 | easySPIN_SYNC_SEL_1_2);
```

...
STM32 firmware library structure

The key library components are the following

- **easyspin_target_config.h**
  - Application specific settings
  - Motion dynamics configuration
  - Target board selection

- **easyspin.c/h**
  - Contains definitions of L6474 internal registers, its options and masks
  - Microcontroller peripheral initialization routines
  - *easySPIN Application commands* implementation
  - SPI communication

- **eMotionControl.c/h**
  - Implementation of complex motion control commands
  - Runtime motion control mechanism (step generation)
  - Motion speed profile computation (smooth motion) according to the application needs

- **main.c**
  - Demo program code

- **clock.c/h**
  - System clock setup routines
Speed profile computation

- The firmware library computes speed profiles for smooth motor motion suitable to the application needs (jerky motion prevention)

- User defines motion parameters (acceleration/deceleration, max speed):

  ```
  #define MIN_STEPS_PER_SEC 10 /* [steps/sec] */
  #define MAX_STEPS_PER_SEC 200 /* [steps/sec] */
  #define ACCELERATION_RATE 40 /* [steps/sec^2] */
  #define DECELERATION_RATE 40 /* [steps/sec^2] */
  ```

- Motion control module works to keep the above settings satisfied
  - The appropriate speed profile (step timing sequence) is computed every time a motion command arrives
  - In some cases, a steady speed may not be achieved (see below) to keep motion parameters defined and to avoid rippling in shaft rotation
Motion control API

• System initialization
  
  eMotionControl_Init()
  
  eMotionControl_ResetDevice()

• Motion commands
  
  eMotionControl_Run(direction, speed)
  
  eMotionControl_Move(direction, stepCount)
  
  eMotionControl_GoTo(targetPosition)
  
  eMotionControl_GoHome()
  
  eMotionControl_GoMark()
  
  eMotionControl_ResetPos()

• Program control
  
  eMotionControl_WaitWhileActive()
  
  eMotionControl_GetState()
Brief explanation of commands

- **eMotionControl_Move**(DIR_Forward, 200)
  - Produces given number of steps in given direction
  - The steps are performed providing that the speed profile meets the settings

- **eMotionControl_Run**(DIR_Reverse, 200)
  - This command produces a motion in a given direction at speed (steps per second) given by function parameter
  - If the speed value exceeds the maximum speed defined, then it is clamped accordingly
  - Starting phase of the motion corresponds to the motion dynamics settings

- **eMotionControl_GoTo**(65)
  - Produces steps to reach given absolute position (ABS_POS register value)
  - The steps are performed providing that the speed profile meets the settings
Demo program code sample

- The initialization of the motion control module is straightforward

```c
void main() {
    /* Configure the System clock frequency */
    SetSysClock();

    /* eMotionControl module initialization */
    eMotionControl_Init();

    /* Motion parameters customization can be done here... */

    /* Move command example */
    eMotionControl_Move(DIR_Forward, 200);
    eMotionControl_WaitWhileActive();

    /* GoTo command example */
    eMotionControl_GoTo(65);
    eMotionControl_WaitWhileActive();
    currentPosition = easySPIN_GetParam(easySPIN_ABS_POS);

    /* ... */
}
```