

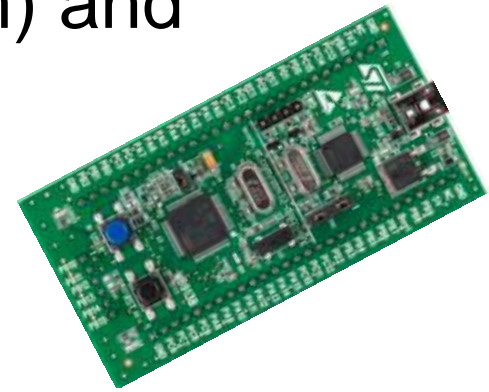


# STM32F1xx motor-control firmware for dSPIN

---

**Quick guide**

- STM32F1xx firmware allows you to control L6470 dSPIN micro-stepping motor driver via STM32F1 MCU.
- This firmware supports STEVAL-PCC009V2 + STM32 Value Line Discovery evaluation boards with library configuration (dspin.h) and development tools project files.

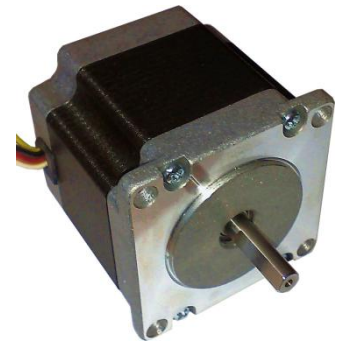


- Easy portability due to ANSI C standard compliance
  - Only HAL (Hardware Abstraction Layer) routines should be modified when used with another platform
  - Simple portability to STM8 families thanks to similar HAL routines for peripherals - SPI, GPIOs...

# Firmware library structure



- `dspin.c`
  - Microcontroller peripherals initialization
  - dSPIN Application commands implementation
  - Library support functions implementation
- `dspin.h`
  - Function prototypes for implemented commands & support functions
  - Register value (options) definition
  - Register masks definition
  - Macros for selected function parameter conversions
  - Evaluation boards related definitions – GPIO signals and peripherals assignment
- `main.c`
  - Example of library usage – system configuration / function calls
- Other microcontroller configuration files
  - `clock.c`, `-.h`, `stm32f10x_conf.h` (peripherals configuration), `stm32f10x_it.c`, `-.h` (interrupt routines) & standard library for GPIO and SPI



# Development tools support



- Library contains project folders (files) for development tools:

- IAR – EWARM v5
  - J-Link for STEVAL-PCC009V2
  - ST-Link for STM32 VL Discovery
- IAR – EWARM v6
  - J-Link for STEVAL-PCC009V2
- KEIL – uVision v4.03, v4.20
  - uLink II for STEVAL-PCC009V2
  - uLink Pro for STEVAL-PCC009V2
  - ST-Link for STM32 VL Discovery
- Raisonance – RIDE v7
  - R-Link for STEVAL-PCC009V2

```
008  gSPIN_RegistStruct_KVAL_RUN = RVAL_Peco_to_Pac(10);
009  /* Acceleration duty cycle (duty) settings to 10%, range 0 to 99.4% */
010  gSPIN_RegistStruct_KVAL_ACC = RVAL_Peco_to_Pac(10);
011  /* Deceleration duty cycle (duty) settings to 10%, range 0 to 99.4% */
012  gSPIN_RegistStruct_KVAL_DEC = RVAL_Peco_to_Pac(10);
013  /* Interscan speed settings for REMP compensation to 200 steps/s, range 0 to 3000 steps/s */
014  gSPIN_RegistStruct_INT_SPO = IntScan_Speed_to_Pac(100);
015  /* REMP start slope settings for REMP compensation to 0.0389 step/s, range 0 to 0.4% a/step */
016  gSPIN_RegistStruct_ST_SLP = REMP_Slope_Peco_to_Pac(0.01);
017  /* REMP start stop slope settings for REMP compensation to 0.0639 step/s, range 0 to 0.4% a/step */
018  gSPIN_RegistStruct_FM_SLP_ACC = REMP_Slope_Peco_to_Pac(0.05);
019  /* REMP start stop slope settings for REMP compensation to 0.0639 step/s, range 0 to 0.4% a/step */
020  gSPIN_RegistStruct_FM_SLP_DEC = REMP_Slope_Peco_to_Pac(0.05);
021  /* Thermal compensation param settings to 1, range 1 to 1.4675 */
022  gSPIN_RegistStruct_R_TRESH = RThrm_to_Pac(1);
023  /* Overcurrent threshold settings to 1500mA */
024  gSPIN_RegistStruct_OCR_TH = gSPIN_OCR_TH_1500mA;
025  /* Stall threshold settings to 4000mA, range 32.25 to 4000mA */
026  gSPIN_RegistStruct_STALL_TH = StallThr_to_Pac(1000);
027  /* Stop mode settings to 120 microseconds */
028  gSPIN_RegistStruct_STOP_MODE = gSPIN_STOP_SEL_110;
029  /* Alarm settings - all alarms enabled */
030  gSPIN_RegistStruct_ALARM_EN = gSPIN_ALARM_EN_OVERCURRENT | gSPIN_ALARM_EN_THERMAL_SHUTDOWN
031  | gSPIN_ALARM_EN_THERMAL_WARNING | gSPIN_ALARM_EN_UNDER_VOLTAGE | gSPIN_ALARM_EN_STALL_DET_A
032  | gSPIN_ALARM_EN_STALL_DET_B | gSPIN_ALARM_EN_SW_TURB_ON | gSPIN_ALARM_EN_UNDER_SPEED_CBD;
033  /* Interscan oscillator, REMP (COOUT_dload, supply voltage compensation disable),
034  * overcurrent shutdown enable, alarm reset = 200 Vsec, REMP frequency = 11.8kHz */
035  gSPIN_RegistStruct_CONF10 = gSPIN_CONF10_INT_148MHz_COOUT_2MHz | gSPIN_CONF10_SW_BARD_STOP
036  | gSPIN_CONF10_SW_COMP_DISABLE | gSPIN_CONF10_SW_20_DISABLE | gSPIN_CONF10_SW_STOP_wd
037  | gSPIN_CONF10_PWR_DIV_2 | gSPIN_CONF10_PWR_MTC_1;
038
039  /* Program all gSPIN registers */
040  gSPIN_Registers_Set(gSPIN_RegistStruct);
041
042  /* Move by 60,000 steps forward, range 0 to 4,124,303 */
043  gSPIN_Move(FWD, (int32_t)60000);
044
045  /* Wait until not busy - busy pin test */
046  while(gSPIN_Busy_RM());
047
```

# Application commands examples



**dSPIN\_Set\_Param(dSPIN\_KVAL\_RUN, Kval\_Perc\_to\_Par(5));**

Sends dSPIN command to change run duty cycle to 5%

Library defined constant for run duty cycle

**dSPIN\_Move(FWD, (uint32\_t)(60000));**

Moves by 60,000 steps forward, range 0 to 4,194,303

Library defined constants for movement direction

**dSPIN\_Run(REV, Speed\_Steps\_to\_Par(50));**

Runs constant speed of 50 steps/s reverse direction

**while(dSPIN\_Busy\_HW());**

Busy pin hardware test (by polling) and wait until released

**dSPIN\_Soft\_Stop();**

Performs SoftStop command

**dSPIN\_rx\_data = dSPIN\_Get\_Status();**

Reads Status register content

# Evaluation board related config (dspin.h)



```
#define STEVAL_PCC009V2
/* #define STM32_VL_Discovery */
```

- The above definition configures all the GPIO signals and SPI peripheral for STEVAL-PCC009V2 evalboard
- No other modification in the firmware is necessary to do in order to configure the target hardware related settings
- Signal / peripheral assignments are fully implemented in the library by preprocessor directive
- Possibility to extend the list by other evaluation boards

# Register bits definition example (dspin.h)



- Example of ALARM\_EN register structure definition in order to have access to each alarm enable bit
- Allows transparent code implementation

```
/* dSPIN ALARM_EN register options */
typedef enum {
    dSPIN_ALARM_EN_OVERCURRENT           =((uint8_t)0x01),
    dSPIN_ALARM_EN_THERMAL_SHUTDOWN      =((uint8_t)0x02),
    dSPIN_ALARM_EN_THERMAL_WARNING       =((uint8_t)0x04),
    dSPIN_ALARM_EN_UNDER_VOLTAGE         =((uint8_t)0x08),
    dSPIN_ALARM_EN_STALL_DET_A           =((uint8_t)0x10),
    dSPIN_ALARM_EN_STALL_DET_B           =((uint8_t)0x20),
    dSPIN_ALARM_EN_SW_TURN_ON            =((uint8_t)0x40),
    dSPIN_ALARM_EN_WRONG_NPERF_CMD       =((uint8_t)0x80)
} dSPIN_ALARM_EN_TypeDef;
```



# Register definition masks example (dspin.h)



- If source of diagnostic needs to be identified, the following mask set can be applied:

```
/* Status Register bit masks */
typedef enum {
    dSPIN_STATUS_HIZ                =(((uint16_t)0x0001)),
    dSPIN_STATUS_BUSY               =(((uint16_t)0x0002)),
    dSPIN_STATUS_SW_F               =(((uint16_t)0x0004)),
    dSPIN_STATUS_SW_EVN            =(((uint16_t)0x0008)),
    dSPIN_STATUS_DIR                =(((uint16_t)0x0010)),
    dSPIN_STATUS_MOT_STATUS         =(((uint16_t)0x0060)),
    dSPIN_STATUS_NOTPERF_CMD        =(((uint16_t)0x0080)),
    dSPIN_STATUS_WRONG_CMD         =(((uint16_t)0x0100)),
    dSPIN_STATUS_UVLO               =(((uint16_t)0x0200)),
    dSPIN_STATUS_TH_WRN             =(((uint16_t)0x0400)),
    dSPIN_STATUS_TH_SD              =(((uint16_t)0x0800)),
    dSPIN_STATUS_OCD                =(((uint16_t)0x1000)),
    dSPIN_STATUS_STEP_LOSS_A        =(((uint16_t)0x2000)),
    dSPIN_STATUS_STEP_LOSS_B        =(((uint16_t)0x4000)),
    dSPIN_STATUS_SCK_MOD            =(((uint16_t)0x8000))
} dSPIN_STATUS_Masks_TypeDef;
```