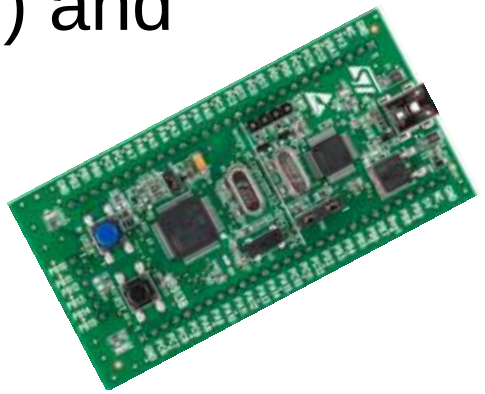
STM32F1xx motor-control firmware for dSPIN

Quick guide

STM32F1xx firmware for dSPIN

- STM32F1xx firmware allows you to control L6470 dSPIN micro-stepping motor driver via STM32F1 MCU.

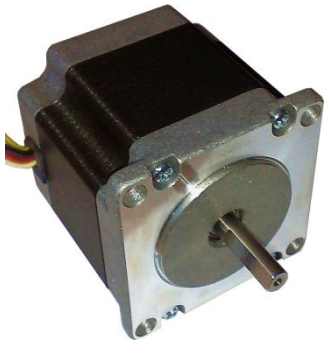
- This firmware supports STEVAL-PCC009V2 + STM32 Value Line Discovery evaluation boards with library configuration (dspin.h) and development tools project files.



Easy portability

- 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…

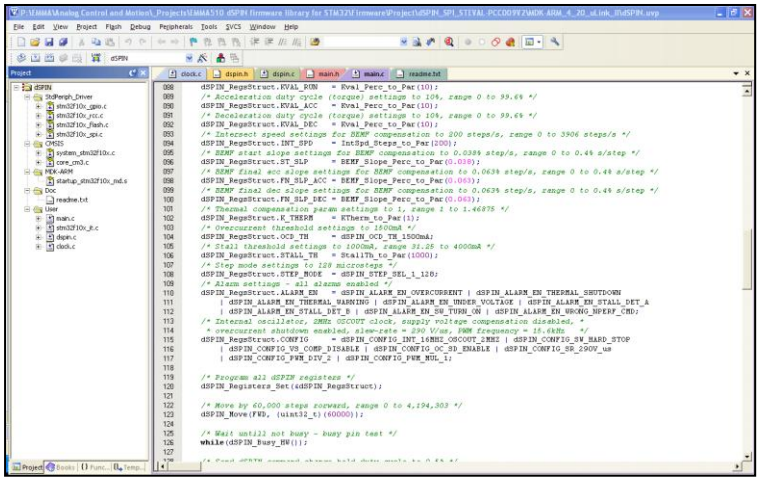
Firmware library structure



- 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



Application commands examples

Library defined constant for run duty cycle

dSPIN_Set_Param(dSPIN_KVAL_RUN, Kval_Perc_to_Par(5));

Sends dSPIN command to change run duty cycle to 5%

Library defined constants for movement direction

dSPIN_Move(FWD, (uint32_t)(60000));

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

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;
```