LVHB_KL25Z_34933EP-EVB_Stepper_GPIO

Example project for Low Voltage H-Bridge SW Driver

Date: *25/05/2017*

Revision: *1.0*

Overview

The purpose of this example project is to demonstrate how to control a stepper motor using dual H-Bridge device, four GPIO MCU pins and Low Voltage H-Bridge (LVHB) SW Driver. The project contains several cases to show how to use driver functions for stepper motor control.

Hardware Requirements

Following is required:

- FRDM-KL25Z (MCU freedom board)
- FRDM-34933-EP-EVB (H-Bridge freedom board)
- Stepper motor
- External Power Source (according to used stepper motor)
- USB Mini B cable

Setting up Hardware

Target platform for this example is FRDM-KL25Z and FRDM-34933-EP-EVB. Note that the driver supports also other LVHB devices and other MCUs. MCUs supported by SDK 2.x can be found in a roadmap on the NXP community. For more information about supported devices refer to LVHB SW driver user guide.

In Figure 1 you can see HW connection of FRDM-34933-EP-EVB freedom board with load. Description of HW connection is in Table 1.

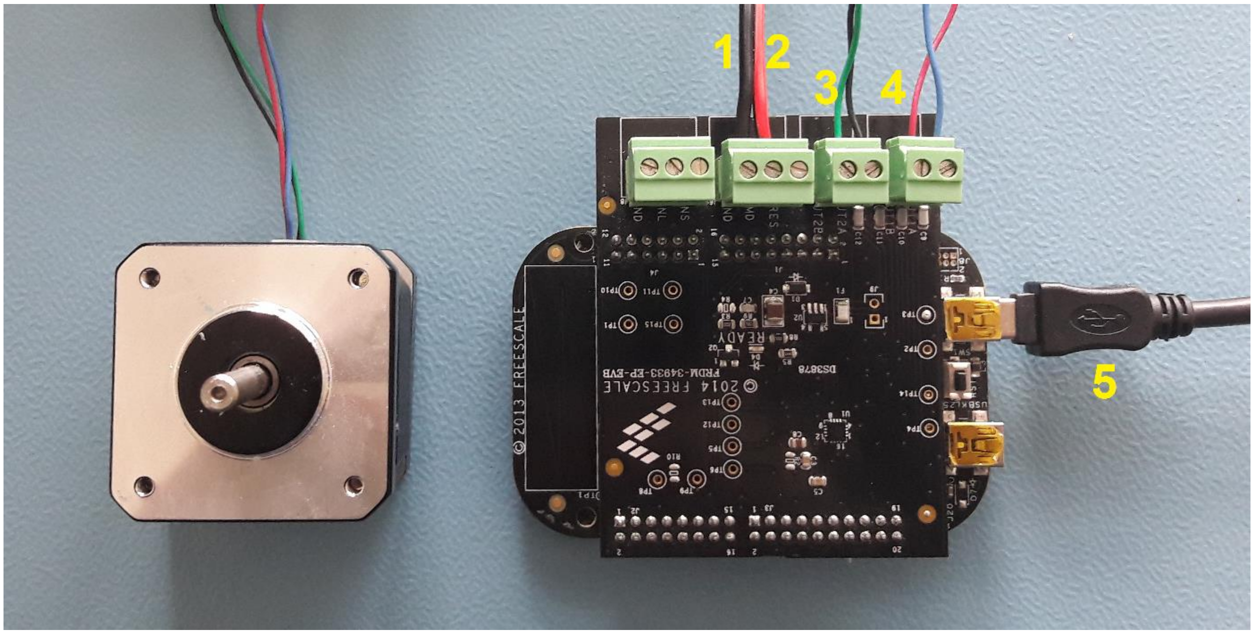


Figure 1. HW connection of FRDM-34933-EP-EVB

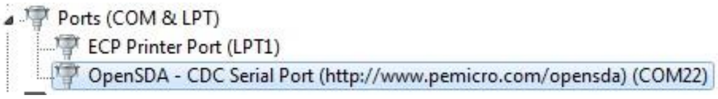
Table 1. Legend for HW connection

Label	Description
1.	DC Power supply (GND)
2.	DC Power supply (+)
3.	First winding connection of stepper motor
4.	Other winding connection of stepper motor
5.	USB mini

Setting up Software

Make sure that you have installed KDS 3.2.0 or newer.

The application uses debug interface with virtual serial port to print user messages. Check that your debug connection has been set up properly. Type of used debug connection depends on used MCU. FRDM-KL25Z uses **OpenSDA**, see Figure 2. Note that number of COM port may differ because of different system resource usage. Baud rate is 115200 Bd.



Ports (COM & LPT)
ECP Printer Port (LPT1)
OpenSDA - CDC Serial Port (http://www.pemicro.com/opensda) (COM22)

Figure 2. OpenSDA virtual port

Description

The purpose of this example project is to show a simple demo intended for demonstration of full-stepping mode without acceleration ramp. Full-stepping signals are generated using timer periphery and GPIO pins.

The project uses the following peripherals:

- GPIO – IN1A, IN1B, IN2A and IN2B pins.
- TPM0 – Timing for stepper motor control.
- UART0 – Print error messages to serial COM console.

Pin selection for all mentioned peripherals follows in Table 2 for selected MCU.

Table 2. Pin selection

Pin Function	FRDM-KL25Z
IN1A	PTD4
IN1B	PTA12
IN2A	PTA4
IN2B	PTA5
UART RX	PTA1
UART TX	PTA2

Application uses virtual serial port to print user messages that describe executed test cases. Serial port settings are following:

- Data width: 8 bits
- Baud rate: 115 200 Bd
- Parity: none

In module *main.c* the board hardware is initialized. Then the configuration structure of LVHB driver is filled in. According to the configuration structure utilized timer periphery and GPIO pins are initialized.

There are several test cases, which demonstrate full-step mode. Function *LVHB_AlignRotor* is called at the start of the example to align rotor to a full-step position. Function *WaitForCompletion* uses polling to wait for completion of motor movement. *BasicDemo* and *AdvancedDemo* functions are called in a loop.

1. **Basic demo** (function *BasicDemo*) demonstrates control of motor in full-step mode. Firstly, full-stepping speed is set by function *LVHB_SetFullStepSpeed*. Then function *LVHB_MoveSteps* moves motor in forward direction by 100 full-steps. There is no way to start new movement while motor is running. Function *WaitForCompletion* is used to detect movement completion.
2. **Continual mode** (function *AdvancedDemo*) shows usage of LVHB driver functions related to continual mode. Full-stepping speed is increased and motor runs in continual mode. Note that it is not possible to change speed while motor is running. The movement is started by function *LVHB_MoveContinual* and continues until function *LVHB_StopContinualMovement* is called. Note that stepper motor does not stop immediately (i.e. when the *LVHB_StopContinualMovement* is called), it can execute several steps. Therefore, motor does not have to stop at the same position where it started. Finally, stepper position counter is restarted by *LVHB_ResetFullStepPosition* and H-Bridge outputs are set to LOW using function *LVHB_DisableMotor*. Note that motor holds position (coils are powered) when movement is complete and function *LVHB_DisableMotor* is not called.

In *main.c* following set of functions is implemented covering LVHB SW driver functionality:

- *GetDefaultConfig* – Fills the driver configuration by default values.
- *ConfigureGpio* – Configures GPIO for usage with the driver.
- *ConfigureTimer* – Configures timer for usage with the driver.
- *Init* – Initializes the device.
- *AlignRotor* – Aligns rotor to the full-step position.
- *MoveSteps* – Moves motor by specified number of full-steps.
- *MoveContinual* – Moves motor continually in full-step mode.
- *StopContinualMovement* – Stop continual movement of stepper motor.
- *DisableMotor* – Disables the stepper motor.
- *GetMotorStatus* – Returns status of stepper motor control.

- *SetFullStepSpeed* – Sets the speed of full-step mode.
- *SetFullStepAcceleration* – Sets the acceleration ramp of full-step mode.
- *ResetFullStepPosition* – Sets the counter of full-steps to zero.
- *GetFullStepPosition* – Returns the current full-step position.

Import the Example Project

The following steps show how to import an example project into KDS 3.2.0.

1. In KDS click on the ***File / Import***.
2. Choose ***General / Existing Projects into Workspace***.
3. Click ***Browse to select root directory*** with your downloaded example projects.
4. ***Select project*** named **LVHB_KL25Z_34933EP-EVB_Stepper_GPIO** and click ***Finish*** to complete the process.
5. Now the example project should be in your workspace and ready to run.

Building and Running the Project

In order to build and run the project you need to ***build*** the project usual way. If the build is successful, ***debug and run*** the project. This can be accomplished in following steps:

1. Click on the **arrow** next to the **debug icon** and select **Debug Configurations**.
2. **Select** one of the existing configurations with **project name** under **PEMicro** group or **create** one by double clicking on this group.
3. Pick up proper **debug interface** and **USB port**.
4. Apply changes and click on **Debug**.

If you have any questions related to how to work with debug configurations, see ***Kinetis Design Studio User's Guide***.