LVHBRIDGESDKUG

Low voltage H-bridge software driver

Rev. 1.0 — 27 June 2017 User guide

1 Overview

This documentation describes how to install and use the low voltage H-bridge (LVHB) software driver. This driver is capable of controlling both DC brushed motors and stepper motor. The driver provides an API layer between the hardware and the user application. It makes application development less time consuming by offering an easy-to-use interface for setting motor control options.

Two sets of functions are associated with the LVHB software driver, one for DC brushed motor control and one for stepper motor control.

A single H-bridge device is sufficient for controlling brushed DC motor. LVHB software driver provides two ways of controlling the motor. The first approach uses a timer periphery that initiates on/off switching and controls motor speed. The second approach uses the GPIO pins only to enable basic motor state control and to switch the motor on and off.

Stepper motors require a dual H-bridge model to generate four output signals needed to control the motor. The LVHB software driver allows the motor to be controlled in either full-stepping mode or in micro-stepping control mode (which enables more precise control of the motor).

This driver supports and provides high-level software solution for these analog parts:

- MPC17C724: 0.4 A dual H-bridge motor driver
- MPC17510: 1.2 A 15 V H-bridge motor driver
- MPC17511: 1.0 A 6.8 V H-bridge motor driver
- MPC17529: 0.7 A dual H-bridge motor driver with 3.0 V/5.0 V compatible logic I/O
- MPC17531A: 0.7 A dual H-bridge motor driver with 3.0 V compatible logic I/O
- MPC17533: 0.7 A 6.8 V dual H-bridge motor driver
- MC34933: 1.4 A dual H-bridge driver compatible with 3.0 V logic

NXP offers following board solutions based on these chips:

- FRDM-17C724-EVB
- FRDM-17C724EVB
- FRDM-17510EJ-EVB
- FRDM-17510EVB
- FRDM-17511EP-EVB
- FRDM-17511EV-EVB
- FRDM-17511EVB
- FRDM-17529EV-EVB
- FRDM-17529EVB
- FRDM-17531AEJEVB
- FRDM-17531AEPEVB
- FRDM-17531EP-EVB

- FRDM-17531EV-EVB
- FRDM-17533EV-EVB
- FRDM-34933EVB
- FRDM-34933EP-EVB

See the related user guide and datasheet for detailed information.

2 MCU compatibility

2.1 Peripheral requirements

Peripherals and resource requirements critical to the MCU's ability to handle a given part are as follows:

- TPM/FTM timer is required in case of speed control of DC brushed motors and in case of stepper motor control. Moreover, all pins used as timer channel outputs must be connected to the same timer instance.
- GPIOs are required for the device enable pins (EN/OE/PSAVE), gate driver control (GIN pin) and for output signal generation (in GPIO mode).

Depending on the user application, other resources may be required. See the example projects provided.

2.2 Supported devices

The LVHB software driver supports the following devices: MPC17C724, MPC17510, MPC17511, MPC17529, MPC17531A, MPC17533, and MC34933. All these ICs incorporate internal control logic, a charge pump, gate drive, and high current, low $R_{DS(on)}$ MOSFET output circuitry.

Common device features are as follows:

- PWM frequencies up to 200 kHz
- Undervoltage shutdown
- Cross-conduction suppression

Table 1. Differences between particular devices

Device number	Number of H-bridges	Power supply [V]		Gate driver signal to external MOSFET switch	Output current [A]	
		Logic circuit power supply	Motor driver power supply		Continuous	Peak
MPC17510	1	5	15	yes	1.2	3.8
MPC17511	1	5	5	yes	1	3
MPC17529	2	5	5	no	0.7	1.4
MPC17531A	2	3	5	no	0.7	1.4
MPC17533	2	5	5	no	0.7	1.4
MC34933	2	3	5	no	1	1.4
MPC17C724	2	3	3	no	0.4	0.8

2.3 Supported MCUs

The low voltage H-bridge software driver supports various Kinetis MCUs; some of the MCU evaluation boards are listed in Table 2. The listed MCUs are a subset of MCUs supported by the Kinetis-SDK (Software Development Kit) layer.

Note that a lot of LVHB and MCU freedom board combinations are supported by the driver only partially. INx pins of LVHB chips are often not connected to the same timer instance of MCUs. Therefore, micro-stepping control mode is available only when the Freedom boards are interconnected manually, by wires, to achieve the INx pin connection to the different channels of the same timer instance.

This driver is built on AML (analog middleware layer), which creates API abstraction layer for used SDK. This allows to additionally extend support for different layer similar to SDK without need to change the low voltage H-bridge software driver itself. Current implementation includes abstraction for KSDK 2.0 and S32 SDK 1.0.0.

Table 2. Supported MCU evaluation boards with SDK release number

Supported evaluation boards	SDK Release
FRDM-KL25Z	KSDK 2.0
FRDM-KL27Z	KSDK 2.2
FRDM-K64F	KSDK 2.1

3 Low voltage H-bridge software driver

This section provides an overview of the functionality, settings and usage of the driver (configuration and functions). For additional information, see the *API programmer's guide* (included in the LVHB software driver zip file) and the comments embedded in the code.

3.1 Configuring the driver

The configuration structure shown in Figure 1 is the user interface available for configuring the driver and the device. User must add the appropriate values into the *lvhb_device_data_t* structure. This device data structure is a part of driver configuration structure (*lvhb_drv_config_t*), which also contains used MCU peripherals and pin settings. The driver configuration structure is passed to the initialization function at start up.

```
/*!
 * @brief Brushed motor configuration.
 */
typedef struct {
    lvhb_recirculation_t recirculation;       /*!< Recirculation side (high-impedance, low) of used bridges. */
    lvhb_output_direction_t outputDirection;  /*!< Direction of rotation (state of outputs) of used bridges. */
} lvhb_brush_config_t;

/*!
 * @brief Stepper motor configuration.
 */
typedef struct {
    /* Full-step configuration */
    uint32_t fullStepSpeed;             /*!< Motor speed in Full-step mode. Unit is number of full-steps per secc
    uint32_t fullStepAcceleration;      /*!< Fluent acceleration to desired speed and decelaration to zero. Put 6
    /* Micro-step configuration */
    uint32_t microStepSpeed;            /*!< Motor speed in Micro-step mode. Unit is number of micro-steps per se
    uint32_t microStepAcceleration;     /*!< Fluent acceleration to desired speed and decelaration to zero. Put 6
    lvhb_micro_step_t microStepPerStep; /*!< Number of micro-steps per one full-step. */
    uint32_t microStepPwmFrequency;     /*!< Desired timer frequency for micro-stepping. Minimum value is 10 kHz
} lvhb_stepper_config_t;

/*!
 * @brief Device configuration.
 *
 * This structure contains device configuration.
 */
typedef struct {
    lvhb_device_t device;                                   /*!< H-Bridge device. */
    lvhb_motor_type_t motorType;                            /*!< Motor type. */
    lvhb_brush_config_t brushConfig[LVHB_BRIDGE_COUNT_MAX]; /*!< Brush motor configuration. */
    uint32_t brushPwmFrequency;                             /*!< PWM frequency for speed control of brushed motor
    lvhb_stepper_config_t stepperConfig;                    /*!< Stepper motor configuration. */
    lvhb_stepper_data_t stepperData;                        /*!< Stepper motor data. */
    bool secondaryBridgeUsed;                               /*!< Usage of second bridge in case of brush motor cc
    bool activeMode;                                        /*!< Mode of H-Bridge. Used by all H-Bridges except M
    bool gateDriverOutputHigh;                              /*!< GIN pin value. Used by MPC17510 and MPC17511 onl
} lvhb_device_data_t;

/*!
 * @brief Driver configuration.
 *
 * This structure contains all information needed for proper functionality of the driver,
 * such as used peripherals configurations, control pins configuration and device configuration.
 */
typedef struct
{
    /* EN pin settings. */
    aml_instance_t enPinInstance;                           /*!< EN (MPC17510, MPC17511) / OE (MPC17529, MPC1753
    uint8_t enPinIndex;                                     /*!< EN (MPC17510, MPC17511) / OE (MPC17529, MPC1753
    /* GIN pin settings. */
    aml_instance_t ginPinInstance;                          /*!< GIN pin port instance. Used by MPC17510 and MPC1
    uint8_t ginPinIndex;                                    /*!< GIN pin index. Used by MPC17510 and MPC17511 onl
    /* INx, INxx pins settings */
    lvhb_input_pins_t inputPins[LVHB_BRIDGE_COUNT_MAX];     /*!< GPIO, GPIO/PWM, PWM/GPIO, or PWM. */
    /* IN1/IN1A (first bridge) & IN2A (second bridge) GPIO pin settings. */
    aml_instance_t inxaPinInstance[LVHB_BRIDGE_COUNT_MAX];  /*!< IN1/IN1A (first bridge) & IN2A (second bridge) (
    uint8_t inxaPinIndex[LVHB_BRIDGE_COUNT_MAX];            /*!< IN1/IN1A (first bridge) & IN2A (second bridge) (
    /* IN2/IN1B (first bridge) & IN2B (second bridge) GPIO pin settings. */
    aml_instance_t inxbPinInstance[LVHB_BRIDGE_COUNT_MAX];  /*!< IN2/IN1B (first bridge) & IN2B (second bridge) (
    uint8_t inxbPinIndex[LVHB_BRIDGE_COUNT_MAX];            /*!< IN2/IN1B (first bridge) & IN2B (second bridge) (
    /* Timer settings */
    aml_instance_t tmrInstance;                             /*!< Timer instance - only if any PWM pin used. */
    tmr_lvhb_config_t tmrLvhbConfig;                        /*!< Device Timer configuration - only if any PWM pir
    /* Device settings. */
    lvhb_device_data_t deviceConfig;                        /*!< Device configuration. */
} lvhb_drv_config_t;
```

Figure 1. Driver configuration

Note that LVHB driver supports both stepper and DC brushed motor control and a lot of LVHB chips (devices). Therefore, not all the items need to be filled in the configuration structure for each type of usage. Several devices do not have Enable pin (EN/OE/PSAVE) or Gate driver input pin (GIN). Therefore, configuration of these pins in *lvhb_drv_config_t* structure as well as configuration of their state (*activeMode* and *gateDriverOutputHigh* in *lvhb_device_data_t* structure) or configuration of IN2A and IN2B pins is LVHB chip specific.

In order to use LVHB SW driver for one/two DC brushed motor control, the *lvhbMotorBrushed* value must be set to *motorType* item in *lvhb_device_data_t*

structure. Other required settings are *brushConfig* for all used H-bridge channels, *secondaryBridgeUsed* item in case of dual H-bridge device and *brushPwmFrequency* in case of the speed control.

In order to use LVHB software driver for stepper motor control, the *lvhbMotorStepper* value must be set to *motorType* item in *lvhb_device_data_t* structure. Other required settings are in *lvhb_stepper_config_t* structure. If the stepper is controlled by four PWM pins, the micro-stepping configuration in *lvhb_stepper_config_t* structure must contain correct setting, although micro-stepping is not used by user. Structure lvhb_stepper_data_t is used internally only and it need not be changed by the user. Note that the stepper motor cannot be controlled by device with one H-bridge interface only.

Pins intended to control the motor can be either PWM channel outputs or GPIO pins. This option must be set into *inputPins* item in *lvhb_drv_config_t* structure. Appropriate GPIO pin instances and indexes must be stored in the same structure. If PWM is used, appropriate timer channels must be stored in *tmr_lvhb_config_t* structure.

There are following limitations for PWM/GPIO control of INx pins:

- **DC brushed motor (state control):** there is no limitation for pin-muxing
- **DC brushed motor (speed control):** at least one pin must be controlled by PWM
- **Stepper motor (full-stepping mode):** four GPIO pins or four PWM timer output pins
- **Stepper motor (micro-stepping mode):** all four pins must be PWM timer output pins

Timer periphery is used in case of stepper motor control or when at least one output pin of DC brushed motor is controlled by PWM. Utilized timer instance, timer channels and other timer settings should be stored in *tmrInstance* item and *tmr_lvhb_config_t* structure. Timer prescale should be chosen to ensure the minimal (131 kHz for full-stepping; 1.2 MHz for micro-stepping) and maximal (10 MHz) counter frequency. Note that the timer input frequency affects the minimal stepper speed.

For more detailed description of user configuration structure, see programmer's guide.

3.2 Driver API

This low voltage H-bridge software driver provides API that can be used for dynamic real-time configuration of device in user code. For summary of available functions, see Table 3.

Table 3. Low voltage H-bridge software driver API

Function	Description
GetDefaultConfig	Gets the default configuration of the driver for specific LVHB device and motor type
Init	Initializes driver data and output pin values
Deinit	Deinitializes the driver
SetMode	Sets H-bridge device mode using enable pin
SetGateDriver	Controls Gate Driver Input (GIN) pin. Available only for MPC17510 and MPC17511.
DC Brushed motor	
RotateFull	Spins the motor in desired direction at full speed

Function	Description
RotateProportional	Spins the motor in desired direction at PWM duty speed. This function can be used only when the motor is controlled by at least 1 PWM pin.
SetTriState	Sets output of specified H-bridge to tri-state (high impedance) using INx pins
SetDirection	Sets direction of brush motor at specified H-bridge interface
SetRecirculation	Sets low/high-impedance-side recirculation of the H-bridge
Stepper motor (controlled by either four GPIO or four PWM pins)	
SetFullStepSpeed	Sets the speed of full-step mode
SetFullStepAcceleration	Sets the acceleration ramp of full-step mode
MoveSteps	Moves motor by specified number of full-steps
MoveContinual	Moves motor continually in full-step mode
StopContinualMovement	Stops continual movement of stepper motor
GetMotorStatus	Returns status of stepper motor control
AlignRotor	Aligns rotor to the full-step position
GetFullStepPosition	Returns the current full-step position
ResetFullStepPosition	Sets the counter of full-steps to zero
DisableMotor	Disables the stepper motor
OnCounterRestart	Counter restart event handler
OnActionComplete	Function that can be used by user for handling the action complete event. This event occurs when MoveSteps, MoveMicroSteps, StopContinualMovement or AlignRotor action is done.
Stepper motor (controlled by four PWM pins)	
SetMicroStepSpeed	Sets the speed of micro-step mode
SetMicroStepAcceleration	Sets the acceleration ramp of micro-step mode
MoveMicroSteps	Moves motor by specified number of micro-steps
MoveMicroContinual	Moves motor continually in micro-step mode
SetMicroStepSize	Changes size of the micro-step
GetMicroStepPosition	Returns the current micro-step position
MCU peripherals configuration	
ConfigureGpio	Configures GPIO for usage with this driver
ConfigureTimer	Configures Timer for usage with this driver

For more detailed description of software driver API (function signatures, parameters) see programmer's guide (included in the low voltage H-bridge software driver zip file).

3.3 Low-level drivers

This section describes the low-level MCU peripheral drivers used by the low voltage H-bridge software driver. The LVHB software driver depends on these low-level drivers for functions such as communication and control.

Because the LVHB software driver is built on AML, it has access to a subset of low-level drivers supported by the underlying SDK. Most analog devices require that some of the SPI, I^2C, Timer, ADC and GPIO drivers are configured on the MCU side. The AML layer unifies the API for these selected drivers, making the embedded software driver portable across SDKs.

Figure 2 illustrates the software driver architecture in terms of communication, control, and other functions. The timer/GPIO usage for input pins (IN1/IN1A, IN2/IN1B, IN2A, IN2B) depends on the user pin configuration.

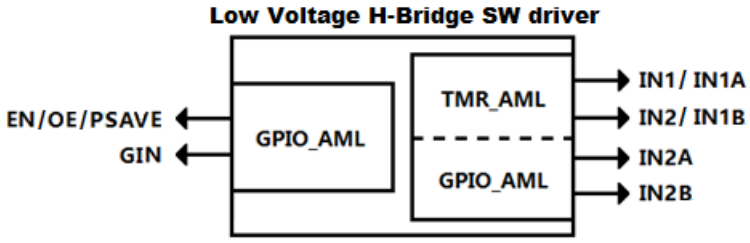


Figure 2. Driver architecture

3.4 Required driver setup

In order to execute correctly, the LVHB software driver requires the following:

- In the file **aml/common_aml.h**, the **SDK_VERSION** parameter must indicate which SDK is being used.
- All pins and peripherals need to be set in the driver configuration structure in **lvhb/lvhb.h**.

3.5 Implementation notes

The low voltage H-bridge software driver is designed to control a two phase bipolar stepper motor. Because a stepper motor uses electrical commutation to rotate, it requires a dual H-bridge device. The basic control method is full-stepping which fully powers each coil in sequence. Increased precision is achieved by using the PWM to control coil current (open loop control). This method is called micro-stepping.

In both micro-step and full-step mode, you can control motor speed, direction, acceleration and deceleration and the position of the stepper motor.

The following application notes apply to stepper motor control:

- The low voltage H-bridge software driver was tested with a core clock frequency ranging from 20 MHz (minimum value) to 120 MHz.
- The acceleration and deceleration ramp of the stepper motor is computed in real-time using integer arithmetic. This solution is based on the article “Generate stepper-motor speed profiles in real time" (Austin, David. 2005).

- The stepper motor holds its position (coils are powered) after motor movement is completed. Use function *DisableMotor* to set H-bridge outputs to LOW (coils are not powered).
- Forward motor direction indicates that steps are executed in the order depicted in Figure 3. IN1 through IN4 are the input pins of the H-bridge device which control H-bridge outputs.
- The FTM or TPM timer device is needed by the stepper control logic.
- The *AlignRotor* function affects the position of the motor. This function executes four full-steps.

3.5.1 Full-step control mode

The component uses normal drive mode where two coils are powered at the same time.

It is possible to generate a full-stepping signal either by using four channels of a timer or by using four GPIO pins. The signal generated by the MCU (inputs of H-bridge device) using four timer channels is shown in Figure 3. The voltage levels applied to the coils of the stepper motor are depicted in Figure 4. Note that the voltage is applied to both coils at the same time.

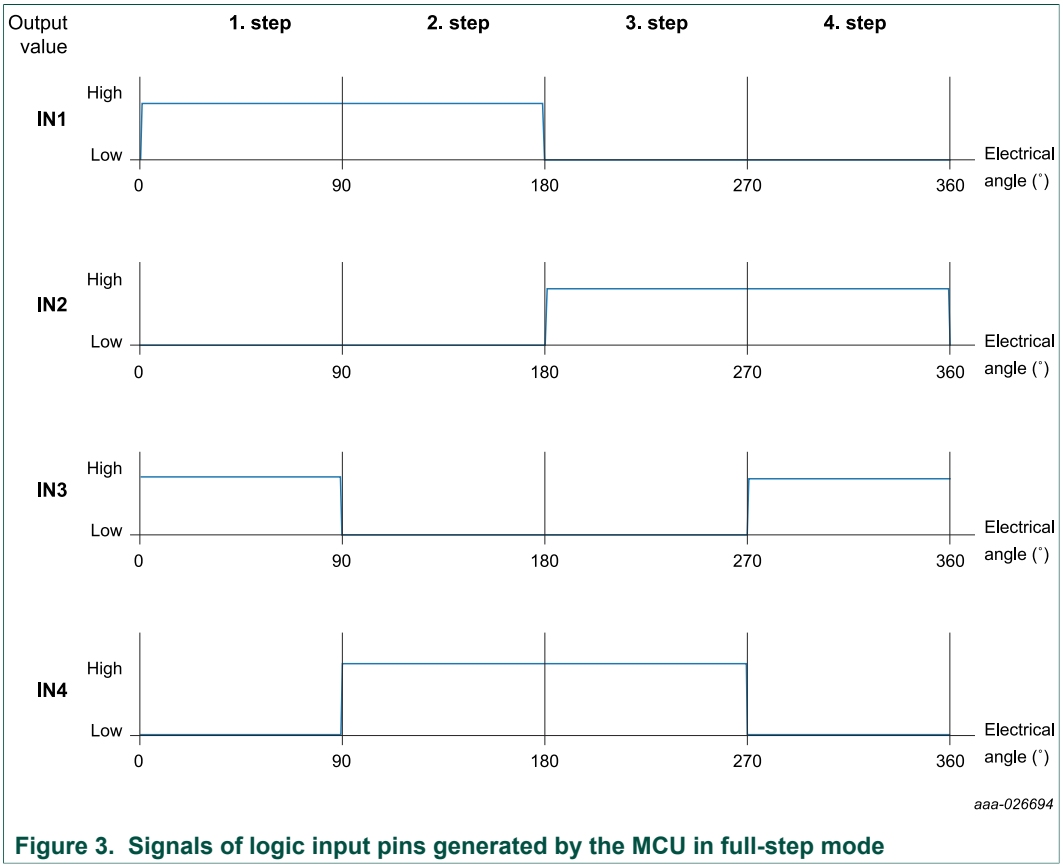


Figure 3. Signals of logic input pins generated by the MCU in full-step mode

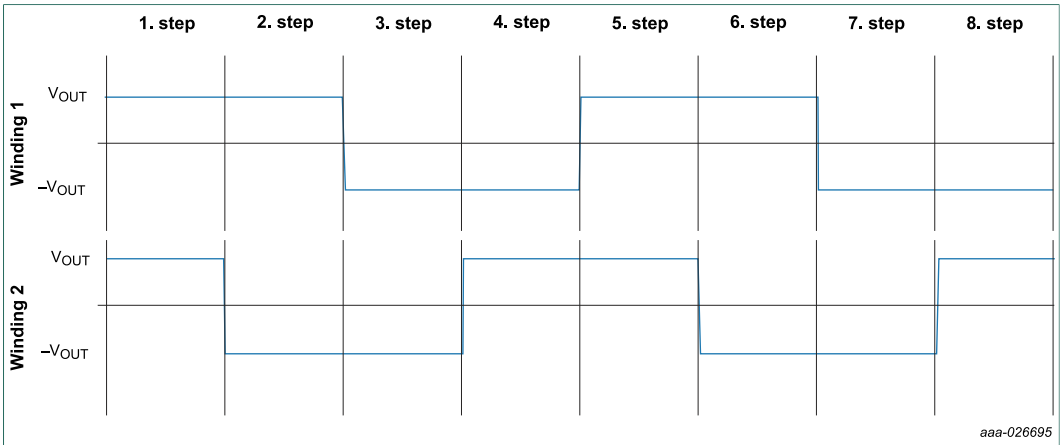


Figure 4. Output of the H-bridge device in full-step mode

3.5.2 Micro-step control mode

Micro-stepping allows for smoother motor movement and increased precision. The current varies in motor windings A and B depending on the micro-step position. A PWM signal is used to reach the desired current value (see the following equations). This method is called sine cosine micro-stepping.

$I_A = I_{MAX} X \sin(\theta)$

$I_B = I_{MAX} X \cos(\theta)$

where:

I_A = the current in winding A

I_B = the current in winding B

I_{MAX} = the maximum allowable current

θ = the electrical angle

In micro-step mode, a full-step is divided into smaller steps (micro-steps). The low voltage H-bridge component offers 2, 4, 8, 16 and 32 micro-steps per full-step. The micro-step size is defined by the property microStepsPerStep in lvhb_stepper_config_t structure and can be changed later in C code by SetMicroStepSize function.

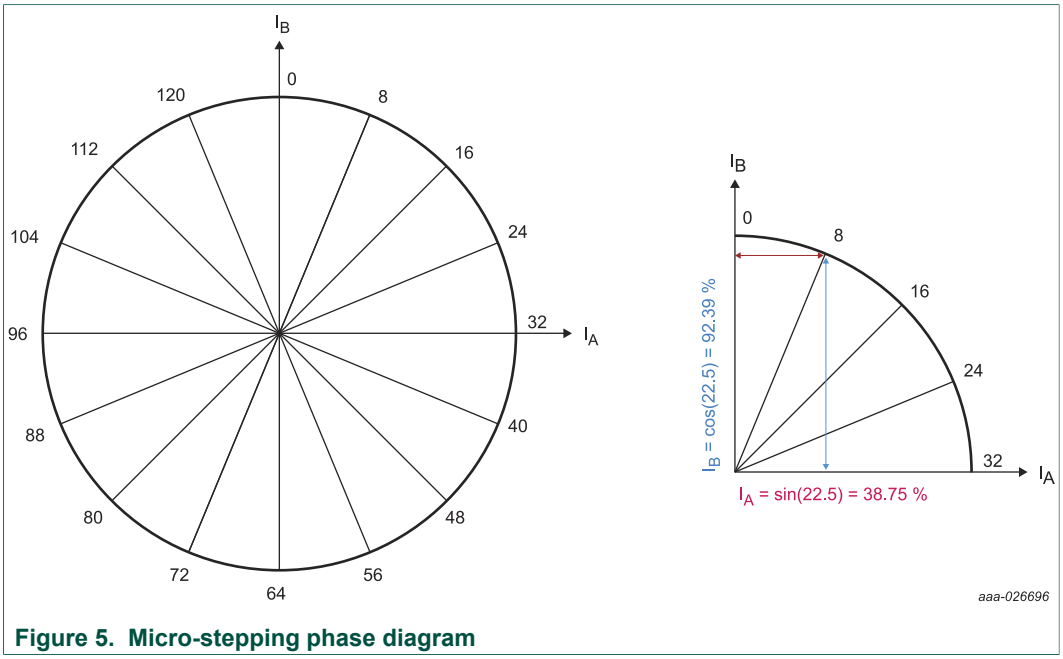


Figure 5. Micro-stepping phase diagram

Table 4. Micro-step phase

Micro-step size [1]					Angle [°]	I [% of I_{MAX}]		Micro-step size					Angle [°]	I [% of I_{MAX}]	
1/2	1/4	1/8	1/16	1/32		A	B	1/2	1/4	1/8	1/16	1/32		A	B
0	0	0	0	0	0.0	0	100	4	8	16	32	64	180	0	−100
				1	2.8	4.91	99.88					65	182.8	−4.91	−99.88
			1	2	5.6	9.8	99.52				33	66	185.6	−9.8	−99.52
				3	8.4	14.67	98.92					67	188.4	−14.67	−98.92
		1	2	4	11.3	19.51	98.08			17	34	68	191.3	−19.51	−98.08
				5	14.1	24.3	97					69	194.1	−24.3	−97
			3	6	16.9	29.03	95.69				35	70	196.9	−29.03	−95.69
				7	19.7	33.69	94.15					71	199.7	−33.69	−94.15
	1	2	4	8	22.5	38.27	92.39		9	18	36	72	202.5	−38.27	−92.39
				9	25.3	42.76	90.4					73	205.3	−42.76	−90.4
			5	10	28.1	47.14	88.19				37	74	208.1	−47.14	−88.19
				11	30.9	51.41	85.77					75	210.9	−51.41	−85.77
		3	6	12	33.8	55.56	83.15			19	38	76	213.8	−55.56	−83.15
				13	36.6	59.57	80.32					77	216.6	−59.57	−80.32
			7	14	39.4	63.44	77.3				39	78	219.4	−63.44	−77.3
				15	42.2	67.16	74.1					79	222.2	−67.16	−74.1
1	2	4	8	16	45	70.71	70.71	5	10	20	40	80	225	−70.71	−70.71
				17	47.8	74.1	67.16					81	227.8	−74.1	−67.16
			9	18	50.6	77.3	63.44				41	82	230.6	−77.3	−63.44
				19	53.4	80.32	59.57					83	233.4	−80.32	−59.57
		5	10	20	56.3	83.15	55.56			21	42	84	236.3	−83.15	−55.56
				21	59.1	85.77	51.41					85	239.1	−85.77	−51.41
			11	22	61.9	88.19	47.14				43	86	241.9	−88.19	−47.14
				23	64.7	90.4	42.76					87	244.7	−90.4	−42.76
	3	6	12	24	67.5	92.39	38.27		11	22	44	88	247.5	−92.39	−38.27
				25	70.3	94.15	33.69					89	250.3	−94.15	−33.69
			13	26	73.1	95.69	29.03				45	90	253.1	−95.69	−29.03
				27	75.9	97	24.3					91	255.9	−97	−24.3
		7	14	28	78.8	98.08	19.51			23	46	92	258.8	−98.08	−19.51

Micro-step size [1]					Angle [°]	I [% of I_{MAX}]	
1/2	1/4	1/8	1/16	1/32		A	B
				29	81.6	98.92	14.67
			15	30	84.4	99.52	9.8
				31	86.4	99.8	6.3
2	4	8	16	32	90	100	0.00
				33	92.8	99.88	−4.91
			17	34	95.6	99.52	−9.8
				35	98.4	98.92	−14.67
		9	18	36	101.3	98.08	−19.51
				37	104.1	97	−24.3
			19	38	106.9	95.69	−29.03
				39	109.7	94.15	−33.69
	5	10	20	40	112.5	92.39	−38.27
				41	115.3	90.4	−42.76
			21	42	118.1	88.19	−47.14
				43	120.9	85.77	−51.41
		11	22	44	123.8	83.15	−55.56
				45	126.6	80.32	−59.57
			23	46	129.4	77.3	−63.44
				47	132.2	74.1	−67.16
3	6	12	24	48	135	70.71	−70.71
				49	137.8	67.16	−74.1
			25	50	140.6	63.44	−77.3
				51	143.4	59.57	−80.32
		13	26	52	146.3	55.56	−83.15
				53	149.1	51.41	−85.77
			27	54	151.9	47.14	−88.19
				55	154.7	42.76	−90.4
	7	14	28	56	157.5	38.27	−92.39
				57	160.3	33.69	−94.15
			29	58	163.1	29.03	−95.69
				59	165.9	24.3	−97
		15	30	60	168.8	19.51	−98.08
				61	171.6	14.67	−98.92
			31	62	174.4	9.8	−99.52
				63	176.4	6.3	−99.8
4	8	16	32	64	180	0.00	−100

Micro-step size					Angle [°]	I [% of I_{MAX}]	
1/2	1/4	1/8	1/16	1/32		A	B
				93	261.6	−98.92	−14.67
			47	94	264.4	−99.52	−9.8
				95	266.4	−99.8	−6.3
6	12	24	48	96	270	−100	0.00
				97	272.8	−99.88	4.91
			49	98	275.6	−99.52	9.8
				99	278.4	−98.92	14.67
		25	50	100	281.3	−98.08	19.51
				101	284.1	−97	24.3
			51	102	286.9	−95.69	29.03
				103	289.7	−94.15	33.69
	13	26	52	104	292.5	−92.39	38.27
				105	295.3	−90.4	42.76
			53	106	298.1	−88.19	47.14
				107	300.9	−85.77	51.41
		27	54	108	303.8	−83.15	55.56
				109	306.6	-80.32	59.57
			55	110	309.4	−77.3	63.44
				111	312.2	−74.1	67.16
7	14	28	56	112	315	−70.71	70.71
				113	317.8	−67.16	74.1
			57	114	320.6	−63.44	77.3
				115	323.4	−59.57	80.32
		29	58	116	326.3	−55.56	83.15
				117	329.1	−51.41	85.77
			59	118	331.9	−47.14	88.19
				119	334.7	−42.76	90.4
	15	30	60	120	337.5	−38.27	92.39
				121	340.3	−33.69	94.15
			61	122	343.1	−29.03	95.69
				123	345.9	−24.3	97
		31	62	124	348.8	−19.51	98.08
				125	351.6	−14.67	98.92
			63	126	354.4	−9.8	99.52
				127	356.4	−6.3	99.8
8	16	32	64	128	360	0.00	100

[1] Shaded rows indicate one quarter step of the motor

The micro-stepping signal is generated using four timer channels (see Figure 6). Output from logic analyzer in Figure 7 shows the change of PWM duty with respect to the micro-step position. Current values applied to the stepper motor coils are depicted in Figure 8.

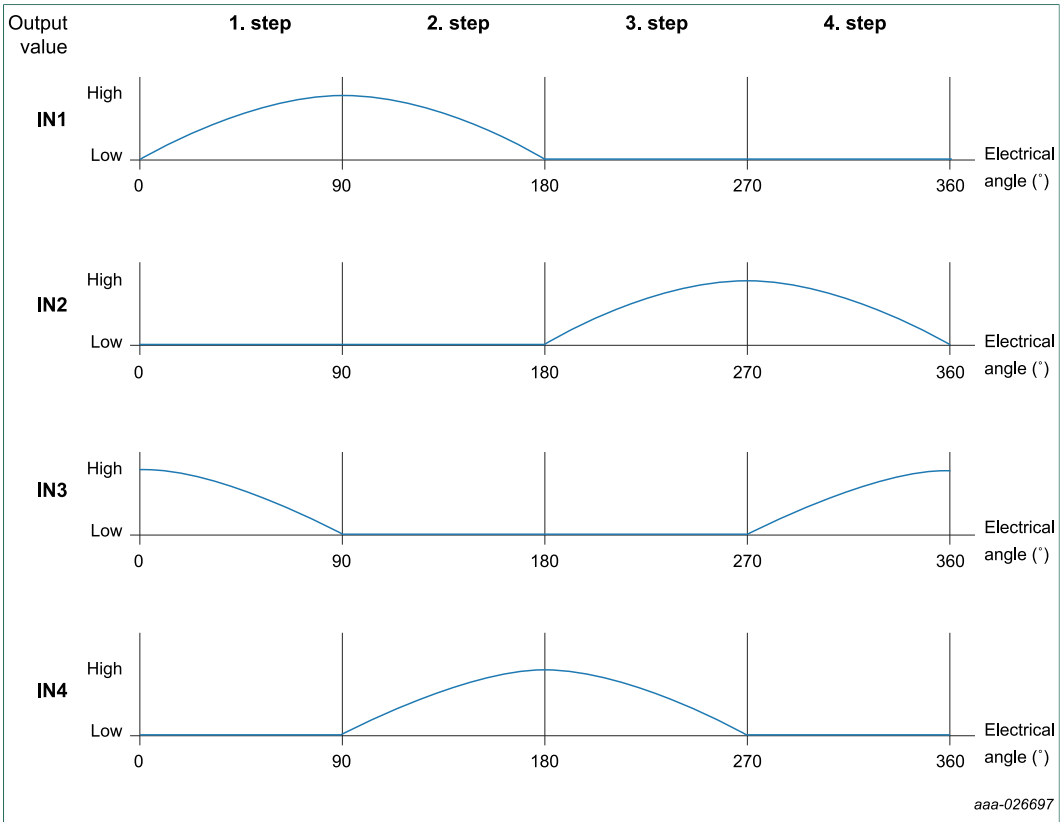


Figure 6. Logic input pin signals generated by the MCU in micro-step mode

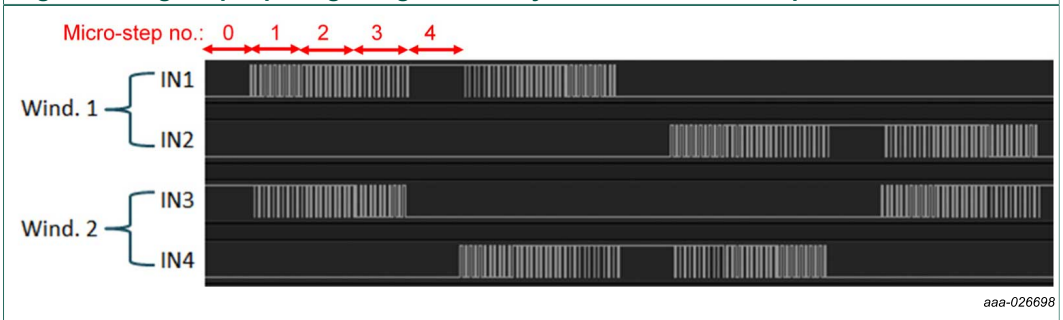


Figure 7. Logic Analyzer output

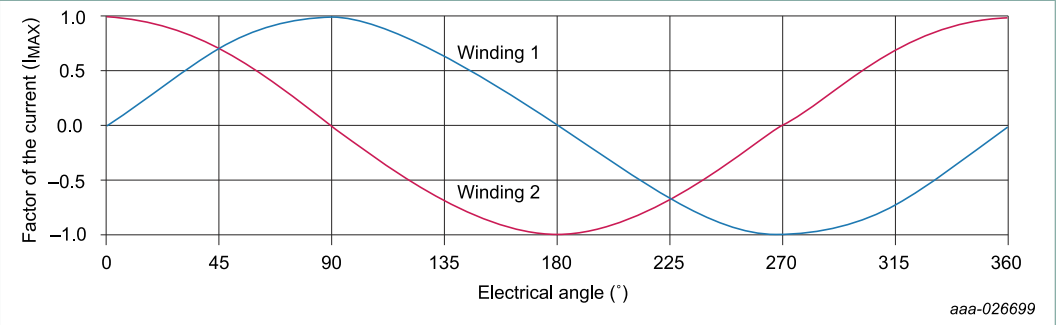


Figure 8. H-bridge device output in micro-step mode

3.5.3 DC brushed motor control

The low voltage H-bridge software driver design can be customized by the user. Driver allows several different pins configuration for motor control using PWM or GPIO. Speed control is done using PWM and state control using GPIO. The PWM and GPIO pins could be combined for saving timer resources. Balance between saving resources and motor control limitation is set by the user. Speed control without limitation needs at least one PWM controlled pin per motor.

Figure 9 describes motor control approach. It shows how motor runs in “Forward” and “Reverse” direction and how motor breaks in “Free-Wheel High Impedance” or “Free-Wheel Low”.

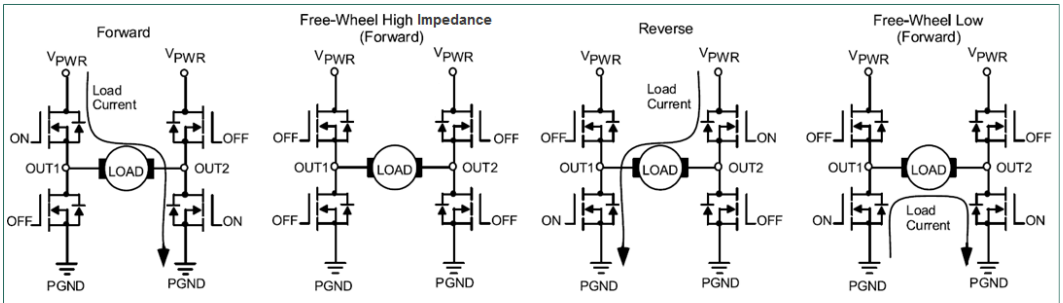


Figure 9. Motor Control using OUT1 and OUT2

Table 5 summarizes possible Free-Wheeling settings according to driver configuration, where Max is max offset of counter period. PWM is offset for PWM duty. H is GPIO set to High. L is GPIO set to low. Zero is 0 counter offset.

Note low logic level at IN1 and IN2 produces high impedance and high logic level produces low at OUT1 OUT2 using MPC17510 and MPC17511. Therefore, the control implementation is different for these two devices.

Table 5. Free-Wheeling using PWM generation in Speed Control mode (for devices except MPC1751x)

Pin configuration	Direction							
	OUT1 high, OUT2 low		OUT1 low, OUT2 high		OUT1 low, OUT2 low (low-side)		OUT1 high, OUT2 high (high impedance side)	
	OUT1	OUT2	OUT1	OUT2	OUT1	OUT2	OUT1	OUT2
GPIO – PWM	H	Max – PWM	L	PWM	L	0	H	Max
PWM – GPIO	PWM	L	Max – PWM	H	0	L	Max	H
PWM – PWM low-side	PWM	0	0	PWM	0	0	—	—
PWM – PWM high impedance	Max	Max – PWM	Max – PWM	Max	—	—	Max	Max

4 Installing the software

This section describes installation of Kinetis Design Studio and how to use this software driver built on SDK for application development.

Kinetis-SDK is a comprehensive collection of software enablements that support NXP microcontroller development. For more information, click www.nxp.com/KSDK.

4.1 Installing Kinetis Design Studio

This procedure explains how to obtain and install the latest version of Kinetis Design Studio (version 3.2.0 in this guide).

Note:

The driver and some examples in the driver package are intended for Kinetis Design Studio 3.2.0 (and above). If Kinetis Design Studio 3.2.0 is already installed on the system, skip this section.

1. Obtain the latest Kinetis Design Studio 3.2.0 installer file from the NXP website: www.nxp.com/KDS.
2. Run the executable file and follow the instructions.

4.2 Downloading the Kinetis SDK library

This step is necessary only when you create a new Kinetis-SDK project, the selected MCU's low-level drivers must be copied into the project's working directory. The example projects listed in Table 6 include the required drivers. In addition, the S32 Design Studio SDK library is distributed with a complete IDE and requires no manual driver additions.

The MCUXpresso SDK Builder is a tool that allows you to download a customized Kinetis SDK based on your specific evaluation platform or Kinetis MCU. To download the Kinetis SDK, click http://mcuxpresso.nxp.com/.

For a list of all MCU's supported by KSDK, click http://community.nxp.com/docs/DOC-329783.

4.3 Downloading the software driver and example projects

To download the latest version of the low voltage H-bridge software driver and the example projects, do the following:

1. Go to the NXP website http://www.nxp.com/LOW-VOLTAGE-HBRIDGE-SDK.
2. Download the zip file containing the software driver and the example projects.
3. Unzip the file and check to see that the folder contains the files listed in Table 6.

Table 6. Content of the zip file

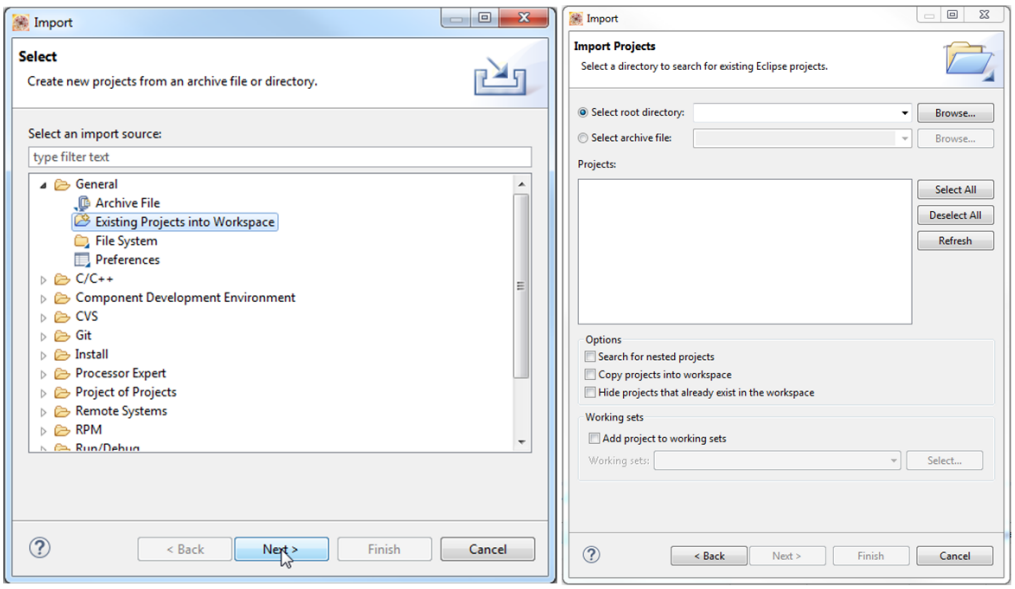
Folder name	Folder contents
KDS_Examples	**Example project folder for Kinetis Design Studio 3.2.0 or newer**
LVHB_K64F_17510EJ-EVB_Brush	The purpose of this example project is to demonstrate how to control a DC brushed motor using single H-bridge device, FRDM-K64F and the low voltage H-bridge software driver.
LVHB_K64F_1751xEVB_Brush	The purpose of this example project is to demonstrate how to control a DC brushed motor using single H-bridge device, FRDM-K64F and the low voltage H-bridge software driver.
LVHB_K64F_34933EP-EVB_Brush	The purpose of this example project is to demonstrate how to control two DC brushed motor using dual H-bridge device, FRDM-K64F and the low voltage H-bridge software driver.
LVHB_K64F_34933EP-EVB_Stepper_GPIO	The purpose of this example project is to demonstrate how to control a stepper motor via four GPIO MCU pins using dual H-bridge device, FRDM-K64F and the low voltage H-bridge software driver.
LVHB_K64F_34933EVB_Brush	The purpose of this example project is to demonstrate how to control two DC brushed motor using dual H-bridge device, FRDM-K64F and the low voltage H-bridge software driver.
LVHB_KL25Z_17510EJ-EVB_Brush	The purpose of this example project is to demonstrate how to control a DC brushed motor using single H-bridge device, FRDM-KL25Z and the low voltage H-bridge software driver.
LVHB_KL25Z_17510EJ-EVB_Brush_FreeMASTER	The purpose of this example project is to demonstrate how to use FreeMASTER application along with the low voltage H-bridge software driver to control DC brushed motor.
LVHB_KL25Z_34933EP-EVB_Brush	The purpose of this example project is to demonstrate how to control two DC brushed motor using dual H-bridge device, FRDM-KL25Z and the low voltage H-bridge software driver.
LVHB_KL25Z_34933EP-EVB_Stepper_GPIO	The purpose of this example project is to demonstrate how to control a stepper motor via four GPIO MCU pins using dual H-bridge device, FRDM-KL25Z and low voltage H-bridge software driver.
LVHB_KL27Z_34933EVB_Brush	The purpose of this example project is to demonstrate how to control two DC brushed motor using dual H-bridge device, FRDM-KL27Z and the low voltage H-bridge software driver.
LVHB_KL27Z_34933EVB_Stepper_FreeMASTER	The purpose of this example project is to demonstrate how to use FreeMASTER application along with low voltage H-bridge software driver to control a stepper motor.
LVHB_KL27Z_VariousEVB_Stepper	The purpose of this example project is to demonstrate how to control a stepper motor using dual H-bridge device, FRDM-KL27Z and low voltage H-bridge software driver.
LVHB_KL27Z_VariousEVB_Stepper_Ramp	The purpose of this example project is to demonstrate how to control a stepper motor with enabled acceleration ramp using dual H-bridge device, FRDM-KL27Z and low voltage H-bridge software driver.
Programmer_Guide	This folder contains detailed API documentation of this software driver.

Folder name	Folder contents
SDK_SW_Driver	Low voltage H-bridge SDK software driver folder. This folder contains two subfolders **aml** and **lvhb**. Note that files in lvhb folder depends relatively on files in aml folder.

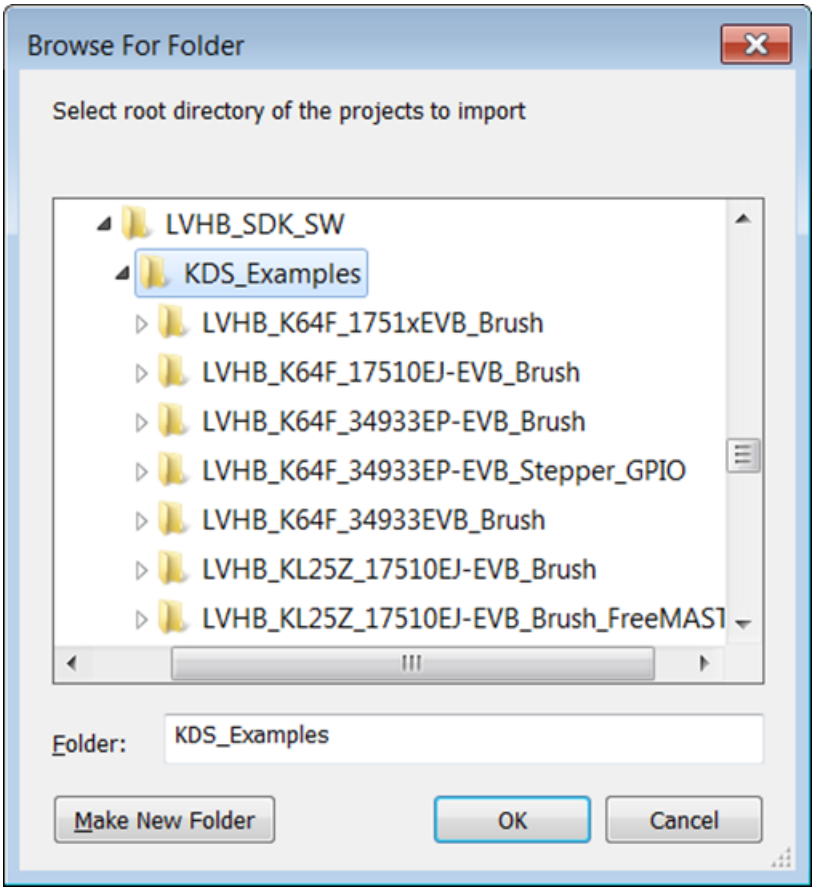
4.4 Import an example project into Kinetis Design Studio

The following steps show how to import an example from the zip file into Kinetis Design Studio.

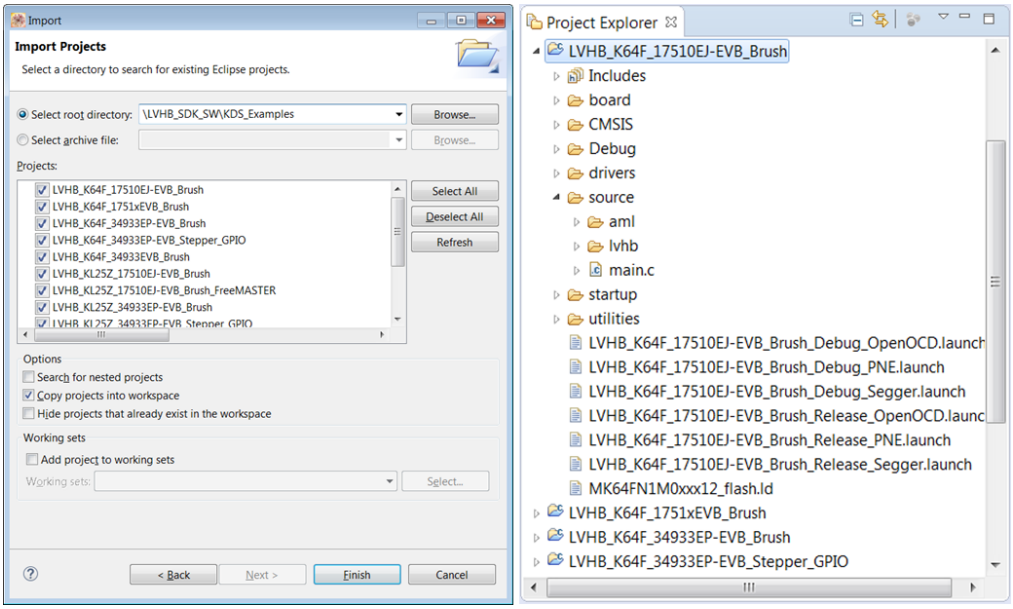
1. In Kinetis Design Studio menu bar, click **File->Import…** In the pop-up window, select **General->Existing Projects into Workspace**, and then click **Next** .



2. Click **Browse** and locate the folder where you unzipped the example files. Find the folder **KDS_Examples** and select a project to import. Then click **OK**.



3. The project is now in the Kinetis Design Studio workspace where you can build and run it. Note that *LVHB_K64F_1751xEVB_Brush*, *LVHB_KL27Z_VariousEVB_Stepper* and *LVHB_KL27Z_VariousEVB_Stepper_Ramp* example projects support more low voltage H-bridge devices and appropriate *main.c* file must be chosen (renamed) for a successful build.

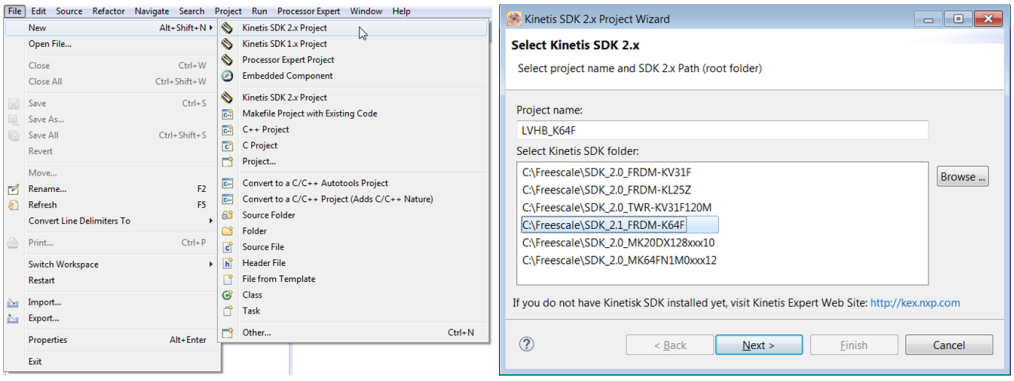


4.5 Creating a new project with low voltage H-bridge software driver

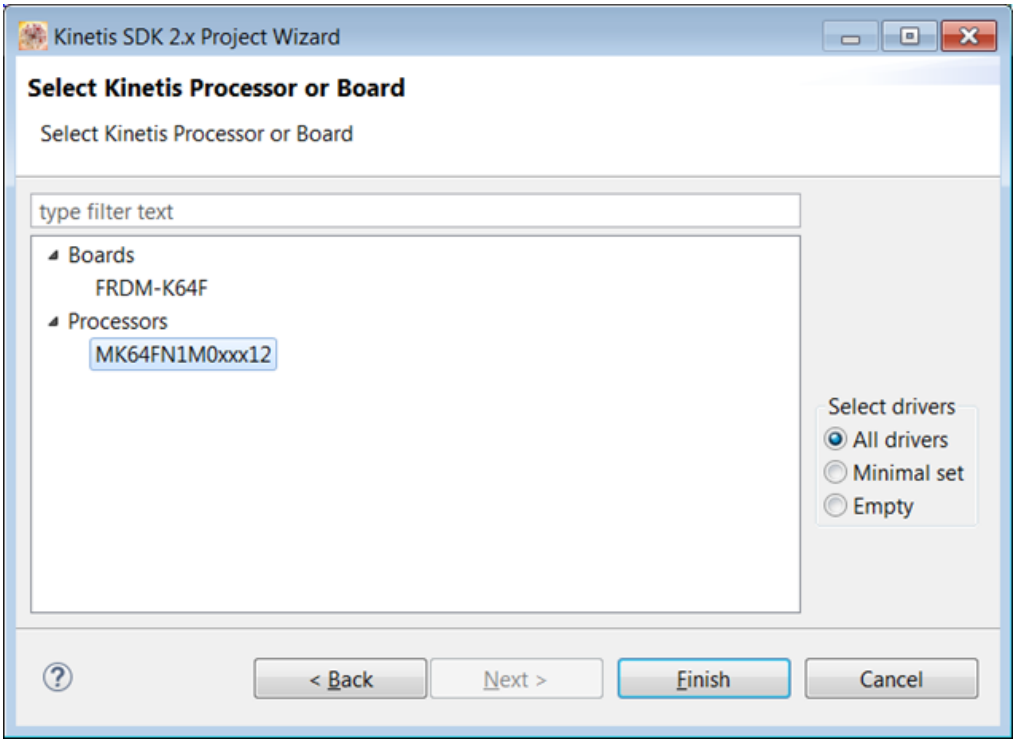
If you choose not to use the example projects, the following instructions describe how to create and setup a new project that uses low voltage H-bridge SDK software driver.

To create a new project, do the following:

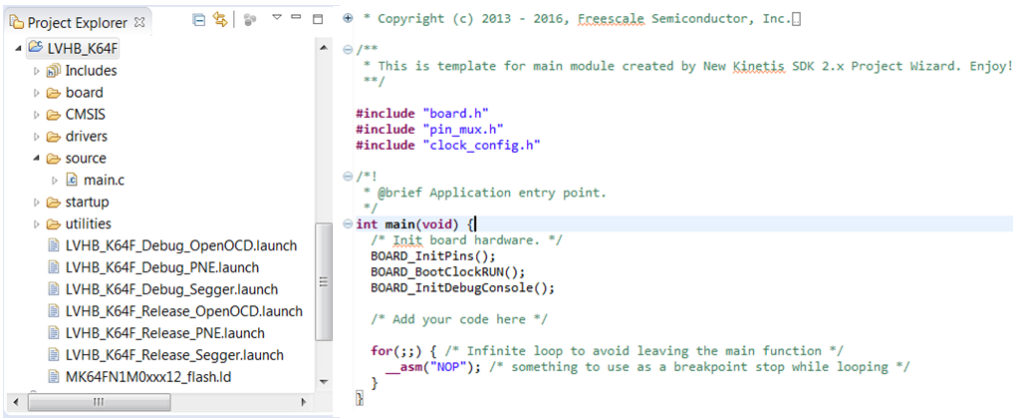
1. In the Kinetis Design Studio menu bar, select **File** -> **New** -> **Kinetis SDK 2.x Project**. When the **Select Kinetis SDK 2.x** box opens, enter a project name in the text box, select path to SDK 2.x library, and then click **Next**. Before creating a new project, make sure to download SDK package for your MCU (see Section 4.2 "Downloading the Kinetis SDK library").



2. In the **Select Kinetis Processor or board** dialog box, select the MCU class or board your project is using (MK64FN1M0xxx12 is selected). Then click **Next**.



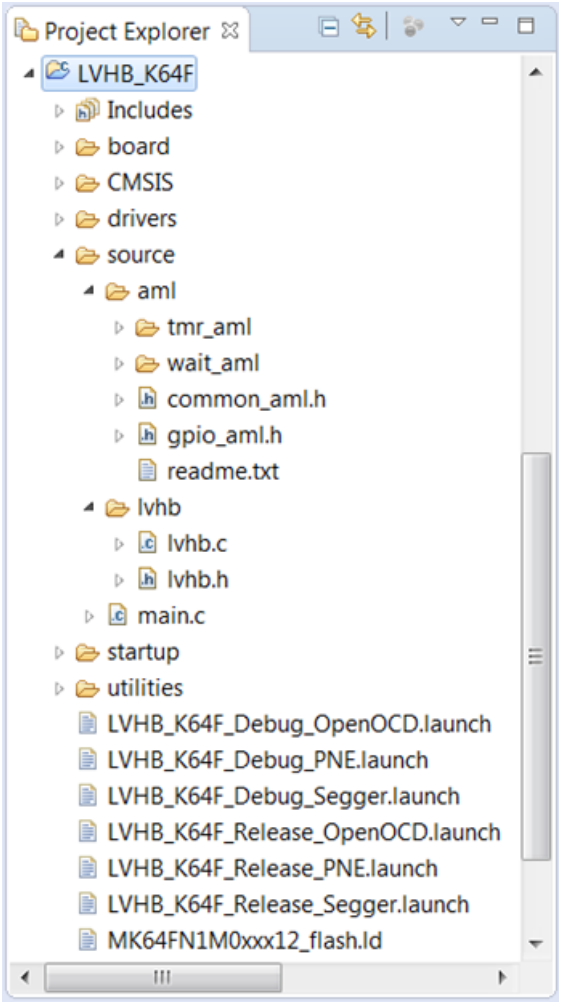
3. The following figure shows the Project Explorer panel and **main.c** content after creation of a new project. All of the needed files are placed in dedicated folders like board, CMSIS, drivers, source, startup and utilities. For more detailed description of SDK project folder structure, see KSDK 2.x user guide.



4.5.1 Adding the low voltage H-bridge software driver to the project

This section describes how to add the low voltage H-bridge software driver to the project.

1. Copy the content of **SDK_SW_Driver** to the **Sources** folder in your newly created project. The list of copied files is shown in the following figure.



2. In **main.c**, add the line highlighted in the following figure to get access to the low voltage H-bridge software driver in user code.

```
/* Copyright (c) 2013 - 2016, Freescale Semiconductor, Inc.
/**
 * This is template for main module created by New Kinetis SDK 2.x Project Wizard. Enjoy!
 **/

#include "board.h"
#include "pin_mux.h"
#include "clock_config.h"

#include "lvhb/lvhb.h"

/*!
 * @brief Application entry point.
 */
int main(void) {
  /* Init board hardware. */
  BOARD_InitPins();
  BOARD_BootClockRUN();
  BOARD_InitDebugConsole();

  /* Add your code here */

  for(;;) { /* Infinite loop to avoid leaving the main function */
    __asm("NOP"); /* something to use as a breakpoint stop while looping */
  }
}
```

4.5.2 Setting up the project

Once the new project has been created and the low voltage H-bridge software driver has been added to it, the project must be set up.

1. In order to get the low voltage H-bridge software driver to run on the selected MCU, you must edit the **board/pin_mux.c** file to configure the timer and GPIO pins being used. This entails making the correct MCU pin selections (see Section 2.3 "Supported MCUs") and then muxing them as needed. Use one of the example projects as a template for this step. The pin muxing on K64F when FRDM-17510EJ-EVB board is connected and IN1 is controlled by GPIO and IN2 by a timer is shown in the following figure.

```
#include "fsl_common.h"
#include "fsl_port.h"
#include "pin_mux.h"

#define SOPT5_UART0TXSRC_UART_TX      0x00u   /*!< UART 0 transmit data source select: UART0_TX pin */

/*FUNCTION**********************************************************************
 *
 * Function Name : BOARD_InitPins
 * Description   : Configures pin routing and optionally pin electrical features.
 *
 *END**********************************************************************/
void BOARD_InitPins(void) {
    /* Ungate the port clock. */
    CLOCK_EnableClock(kCLOCK_PortA);
    CLOCK_EnableClock(kCLOCK_PortB);

    PORT_SetPinMux(PORTB, 16u, kPORT_MuxAlt3);           /* UART0_RX - PTB16 */
    PORT_SetPinMux(PORTB, 17u, kPORT_MuxAlt3);           /* UART0_TX - PTB17 */
    SIM->SOPT5 = ((SIM->SOPT5 &
      (~(SIM_SOPT5_UART0TXSRC_MASK)))                    /* Mask bits to zero which are setting */
       | SIM_SOPT5_UART0TXSRC(SOPT5_UART0TXSRC_UART_TX)  /* UART 0 transmit data source select: UART0_TX pin */
    );

    PORT_SetPinMux(PORTB, 9u, kPORT_MuxAsGpio);          /* IN1 - PTB9 */
    PORT_SetPinMux(PORTA, 1u, kPORT_MuxAlt3);            /* IN2 - PTA1/FTM0_CH6 */
    PORT_SetPinMux(PORTB, 18u, kPORT_MuxAsGpio);         /* EN - PTB18 */
    PORT_SetPinMux(PORTB, 19u, kPORT_MuxAsGpio);         /* GIN - PTB19 */
}
```

2. Define **LVHB_OnActionComplete** function. It is an event handler for several stepper control functions of the LVHB driver. The body of the function can be empty.
3. Create a variable of type **lvhb_drv_config_t** that is passed to all used functions. This variable stores MCU peripheral configuration and low voltage H-bridge software driver configuration. This variable must be accessible during run-time and should be declared either in the **main()** function or as a global variable.
4. Configure the low voltage H-bridge. With the exception of the **LVHB_GetDefaultConfig** function, change individual items as needed.
5. Set up the MCU peripherals to be used with the low voltage H-bridge software driver. There are two options:
   - Use the provided **LVHB_ConfigureTimer** and **LVHB_ConfigureGpio** functions for necessary configuration
   - Handle peripheral initialization

   In either case, the configuration structure to indicate the instances of utilized peripherals still need to be modified. The following figure shows typical settings for the K64F when FRDM-17510EJ-EVB is being used.

```
1 DEFAULT CONFIGURATION
    /* Initialize used peripherals with default configuration. */
    LVHB_ConfigureGpio(&drvConfig);
    LVHB_ConfigureTimer(&drvConfig, NULL);

2 USER CONFIGURATION (for example FTM)
    /* Get default configuration. */
    tmr_sdk_config_t tmrConfig;
    FTM_GetDefaultConfig(&(tmrConfig.tmrConf));
    /* Change parameters. */
    tmrConfig.tmrConf.prescale = kFTM_Prescale_Divide_8;
    ...
    ...
    /* Pass specific configuration to AML. */
    LVHB_ConfigureTimer(&drvConfig, &tmrConfig);
```

6. The low voltage H-bridge initialization function should be called in the end. User must pass reference to MCU peripherals, device and motor control configuration, which are part of driver data structure.

```
#include "board.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "fsl_debug_console.h"

#include "lvhb/lvhb.h"

/* On LVHB action complete event handler. */
void LVHB_OnActionComplete(lvhb_drv_config_t* const drvConfig)
{
    AML_UNUSED(drvConfig);
}

/*!
 * @brief Application entry point.
 */
int main(void) {
    lvhb_drv_config_t drvConfig = {0};

    /* Init board hardware. */
    BOARD_InitPins();
    BOARD_BootClockRUN();
    BOARD_InitDebugConsole();
```

Device and motor control configuration

```
    /* Fill LVHB driver configuration. */
    LVHB_GetDefaultConfig(&drvConfig, lvhbDeviceMPC17510, lvhbMotorBrushed);

    drvConfig.deviceConfig.activeMode = true;
    drvConfig.deviceConfig.gateDriverOutputHigh = false;
    drvConfig.deviceConfig.brushPwmFrequency = 5000;
```

```
    /* Timer settings */
    drvConfig.tmrInstance = 0;                                  /* FTM0 */
    drvConfig.tmrLvhbConfig.counterWidth = 16;
    drvConfig.tmrLvhbConfig.prescale = tmrPrescDiv_8;
    drvConfig.tmrLvhbConfig.srcClck_Hz = CLOCK_GetFreq(kCLOCK_BusClk);

    /* IN1, IN2 settings. */
    drvConfig.inputPins[lvhbBridge1] = lvhbPinsGpioPwm;         /* IN1 (GPIO) + IN2 (PWM) */
    drvConfig.inxaPinInstance[lvhbBridge1] = instanceB;         /* IN1 - PTB9 */
    drvConfig.inxaPinIndex[lvhbBridge1] = 9U;
    drvConfig.tmrLvhbConfig.inxbChannelNumber[lvhbBridge1] = 6; /* IN2 - PTA1/FTM0_CH6 */

    /* EN, GIN settings. */
    drvConfig.enPinInstance = instanceB;                        /* EN - PTB18 */
    drvConfig.enPinIndex = 18U;
    drvConfig.ginPinInstance = instanceB;                       /* GIN - PTB19 */
    drvConfig.ginPinIndex = 19U;
```

Peripherals configuration

```
    /* Initialize used peripherals. */
    LVHB_ConfigureGpio(&drvConfig);
    LVHB_ConfigureTimer(&drvConfig, NULL);

    /* Initialize driver. */
    LVHB_Init(&drvConfig);
```

Peripherals and driver initialization

7. If the low voltage H-bridge software driver is used to control the stepper motor, overflow interrupt of appropriate timer instance should be enabled after peripherals configuration and driver initialization.

4.5.3 Writing an application code

All application code must reside in the **source** folder in the project directory. The code in **main.c** can be modified but the original comments related to usage directions must be retained.

When the low voltage H-bridge SDK software driver is configured properly, the prepared functions can be used to construct an application.

See the *API programmer's guide* (included in the low voltage H-bridge software driver zip file) for function signatures and required parameters. Also review the **source/lvhb/lvhb.h header** file, which contains prototypes for all available functions.

Figure 10 provides an example of user application code after the above procedure has been completed. The example assumes that the user has already configured pin muxing, MCU peripherals and the low voltage H-bridge device itself (part of the code from BOARD_InitPins to LVHB_Init). The code comments indicate what occurs at each step of the execution flow.

For more complex low voltage H-bridge use cases, see example projects.

```
#include "board.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "fsl_debug_console.h"

#include "lvhb/lvhb.h"

/* On LVHB action complete event handler. */
void LVHB_OnActionComplete(lvhb_drv_config_t* const drvConfig)
{
    AML_UNUSED(drvConfig);
}

/*!
 * @brief Application entry point.
 */
int main(void) {
    lvhb_drv_config_t drvConfig = {0};

    /* Init board hardware. */
    BOARD_InitPins();
    BOARD_BootClockRUN();
    BOARD_InitDebugConsole();
    ...
    peripherals initialization
    ...
    /* Initialize driver. */
    LVHB_Init(&drvConfig);

    while (true)
    {
        /* Turn on the motor with 60% PWM duty. */
        LVHB_RotateProportional(&drvConfig, 60, lvhbBridge1);
        for(i=0;i<1000000;i++);

        /* Turn off the motor. */
        LVHB_RotateProportional(&drvConfig, 0, lvhbBridge1);
        for(i=0;i<1000000;i++);
    }
}
```

Figure 10. Example user application code with low voltage H-bridge software driver

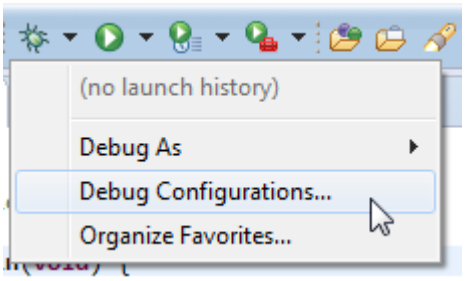
4.5.4 Compiling, downloading and debugging

To compile a project, click the compile icon in the toolbar (see Figure 11).

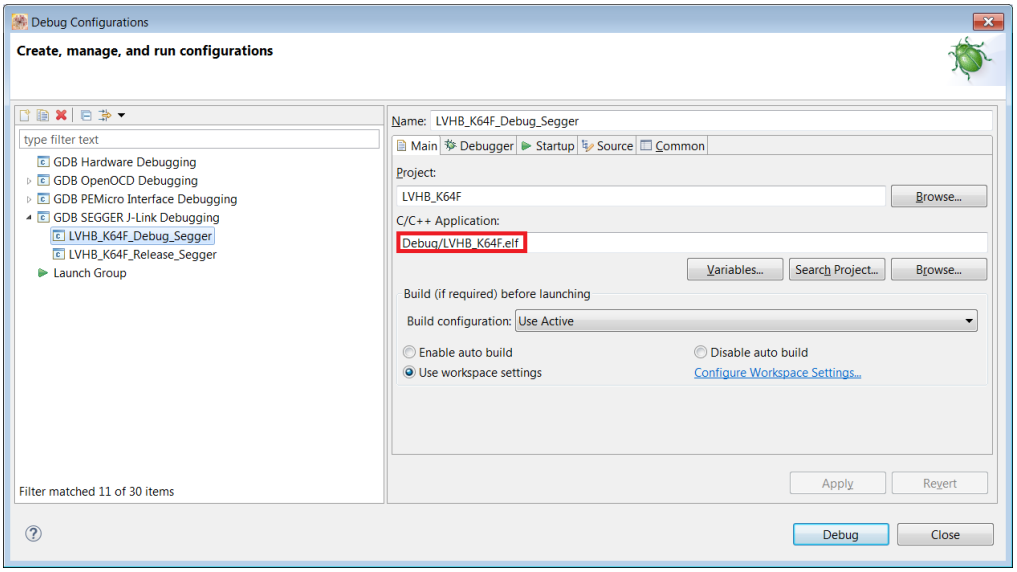
Figure 11. Compiling the project

The process for downloading an application on board in Kinetis Design Studio may differ according to used MCU board. If you have any question, see Kinetis Design Studio user guide. To download and debug on FRDM-K64F MCU board, do the following:

1. Click the arrow next to the debug icon in the toolbar and select **Debug Configurations…**.



2. In the **Debug Configurations** dialog box, click **LVHB_K64F_Debug_Segger** under **GDB SEGGER J-Link Interface Debugging**.
3. Make sure that **C/C++ Application** contains path to .elf file of the project.



4. Then click **Debug**. Kinetis Design Studio downloads and launches the program on the board.

5 References

The following URLs reference related NXP products and application solutions:

NXP.com support pages	Description	URL
FRDM-17C724-EVB	Tool summary page	http://www.nxp.com/FRDM-17C724-EVB
FRDM-17C724EVB	Tool summary page	http://www.nxp.com/FRDM-17C724EVB
FRDM-17510EJ-EVB	Tool summary page	http://www.nxp.com/FRDM-17510EJ-EVB
FRDM-17510EVB	Tool summary page	http://www.nxp.com/FRDM-17510EVB
FRDM-17511EP-EVB	Tool summary page	http://www.nxp.com/FRDM-17511EP-EVB
FRDM-17511EV-EVB	Tool summary page	http://www.nxp.com/FRDM-17511EV-EVB
FRDM-17511EVB	Tool summary page	http://www.nxp.com/FRDM-17511EVB
FRDM-17529EV-EVB	Tool summary page	http://www.nxp.com/FRDM-17529EV-EVB
FRDM-17529EVB	Tool summary page	http://www.nxp.com/FRDM-17529EVB
FRDM-17531AEJEVB	Tool summary page	http://www.nxp.com/FRDM-17531AEJEVB
FRDM-17531AEPEVB	Tool summary page	http://www.nxp.com/FRDM-17531AEPEVB
FRDM-17531EP-EVB	Tool summary page	http://www.nxp.com/FRDM-17531EP-EVB
FRDM-17531EV-EVB	Tool summary page	http://www.nxp.com/FRDM-17531EP-EVB
FRDM-17533EV-EVB	Tool summary page	http://www.nxp.com/FRDM-17533EV-EVB
FRDM-34933EVB	Tool summary page	http://www.nxp.com/FRDM-34933EVB
FRDM-34933EP-EVB	Tool summary page	http://www.nxp.com/FRDM-34933EP-EVB
Low voltage H-bridge software driver	Tool summary page	http://www.nxp.com/LOW-VOLTAGE-HBRIDGE-SDK
MCUXpresso SDK Builder	MCUXpresso SDK Builder	https://mcuxpresso.nxp.com/en/builder
Kinetis Design Studio IDE	Tool summary page	http://www.nxp.com/KDS
S32 Design Studio IDE	Tool summary page	http://www.nxp.com/S32DS

6 Contact information

Visit http://www.nxp.com/support for a list of phone numbers within your region.
Visit http://www.nxp.com/warranty to submit a request for tool warranty.

7 Revision history

Revision history

Rev	Date	Description
v.1.0	20170627	Initial version

8 Legal information

8.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

8.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors. In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory. Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification. Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products. NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Suitability for use in automotive applications — This NXP Semiconductors product has been qualified for use in automotive applications. Unless otherwise agreed in writing, the product is not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer. In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages. Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

8.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

Kinetis — is a trademark of NXP B.V.

NXP — is a trademark of NXP B.V.

Tables

Figures

Contents

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 27 June 2017