Low Voltage H-Bridge Software Driver

Generated by Doxygen 1.8.13

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Module Documentation

4.1 Enum_group

Enumerations

- enum _lvhb_status {
  kStatus_LVHB_TimerInit = MAKE_STATUS(kStatusGroup_LVHB, 0), kStatus_LVHB_InvalidDriverConfig = MAKE_STATUS(kStatusGroup_LVHB, 1), kStatus_LVHB_InvalidArgument = MAKE_STATUS(kStatus↩ Group_LVHB, 2), kStatus_LVHB_TimerFrequency = MAKE_STATUS(kStatusGroup_LVHB, 3),
  kStatus_LVHB_NotInContinualMovement = MAKE_STATUS(kStatusGroup_LVHB, 4), kStatus_LVHB_↩ DeviceBusy = MAKE_STATUS(kStatusGroup_LVHB, 5) }

  Error codes.
- enum lvhb_device_t {
  lvhbDeviceMPC17510 = 0x0U, lvhbDeviceMPC17511 = 0x1U, lvhbDeviceMPC17529 = 0x2U, lvhbDevice↩ MPC17531A = 0x3U,
  lvhbDeviceMPC17533 = 0x4U, lvhbDeviceMPC17C724 = 0x5U, lvhbDeviceMC34933 = 0x6U }

  H-bridge device.
- enum lvhb_bridge_t { lvhbBridge1 = 0x0U, lvhbBridge2 = 0x1U }

  Bridge of H-bridge.
- enum lvhb_motor_type_t { lvhbMotorStepper = 0x0U, lvhbMotorBrushed = 0x1U }

  Motor type.
- enum lvhb_output_direction_t { lvhbDirectionALowBHigh = 0x0U, lvhbDirectionAHighBLow = 0x1U }

  Bridge output direction.
- enum lvhb_recirculation_t { lvhbRecirculationLow = 0x0U, lvhbRecirculationHighImp = 0x1U }

  Recirculation side.
- enum lvhb_output_state_t { lvhbOutputOff = 0x0U, lvhbOutputOn = 0x1U }

  Output state.
- enum lvhb_input_pins_t { lvhbPinsGpio = 0x0U, lvhbPinsGpioPwm = 0x1U, lvhbPinsPwmGpio = 0x2U, lvhb↩ PinsPwm = 0x3U }

  Bridge input pin types.
- enum lvhb_motor_status_t { lvhbStatusRunning = 0x0U, lvhbStatusStop = 0x1U }

  Motor status.
- enum lvhb_stepper_mode_t { lvhbStepperModeMicrostep = 0U, lvhbStepperModeFullstep = 1U, lvhb↩ StepperModeDisabled = 2U }

  Stepper motor mode.
- enum lvhb_stepper_state_t { lvhbStepperStateSleepLoop = 0U, lvhbStepperStateInitCompleted = 1U, lvhb↩ StepperStateSlowStart = 2U }

Stepper motor control states.

- enum lvhb_micro_step_t { lvhbMicroStep2 = LVHB_MICROSTEP_CNT_MAX / 2U, lvhbMicroStep4 = LVHB_MICROSTEP_CNT_MAX / 4U, lvhbMicroStep8 = LVHB_MICROSTEP_CNT_MAX / 8U, lvhbMicroStep16 = LVHB_MICROSTEP_CNT_MAX / 16U, lvhbMicroStep32 = LVHB_MICROSTEP_CNT_MAX / 32U }

  Number of micro-steps per one full-step.

- enum lvhb_ramp_state_t { lvhbRampOff = 0U, lvhbRampUpSpeed = 1U, lvhbRampDownSpeed = 2U, lvhbRampConstSpeed = 3U }

  States of finite state machine which are used for acceleration ramp.

4.1.1 Detailed Description

4.1.2 Enumeration Type Documentation

4.1.2.1 _lvhb_status

`enum _lvhb_status`

Error codes.

Enumerator

kStatus_LVHB_TimerInit	Wrong parameter in tmr_lvhb_config_t or wrong Timer initialization.
kStatus_LVHB_InvalidDriverConfig	Invalid driver configuration.
kStatus_LVHB_InvalidArgument	Invalid function argument.
kStatus_LVHB_TimerFrequency	Frequency of timer can not be out of range for this application.
kStatus_LVHB_NotInContinualMovement	Motor is not running or in continual movement.
kStatus_LVHB_DeviceBusy	Motor is running.

4.1.2.2 lvhb_bridge_t

`enum lvhb_bridge_t`

Bridge of H-bridge.

Enumerator

lvhbBridge1	Bridge 1 - IN1A, IN1B (IN1 and IN2 by MPC1751x) pins.
lvhbBridge2	Bridge 2 - IN2A, IN2B pins.

4.1.2.3 lvhb_device_t

`enum lvhb_device_t`

H-bridge device.

Enumerator

lvhbDeviceMPC17510	MPC17510.
lvhbDeviceMPC17511	MPC17511.
lvhbDeviceMPC17529	MPC17529.
lvhbDeviceMPC17531A	MPC17531A.
lvhbDeviceMPC17533	MPC17533.
lvhbDeviceMPC17C724	MPC17C724.
lvhbDeviceMC34933	MC34933.

4.1.2.4 lvhb_input_pins_t

`enum lvhb_input_pins_t`

Bridge input pin types.

Enumerator

lvhbPinsGpio	Both INxA, INxB (IN1, IN2 by MPC1751x) GPIO - state control only.
lvhbPinsGpioPwm	INxA (IN1 by MPC1751x) GPIO, INxB (IN2 by MPC1751x) PWM.
lvhbPinsPwmGpio	INxA (IN1 by MPC1751x) PWM, INxB (IN2 by MPC1751x) GPIO.
lvhbPinsPwm	Both INxA, INxB (IN1, IN2 by MPC1751x) PWM.

4.1.2.5 lvhb_micro_step_t

`enum lvhb_micro_step_t`

Number of micro-steps per one full-step.

Enumerator

lvhbMicroStep2	Half-step mode. 2 micro-steps per one full-step.
lvhbMicroStep4	4 micro-steps per one full-step.
lvhbMicroStep8	8 micro-steps per one full-step.
lvhbMicroStep16	16 micro-steps per one full-step.
lvhbMicroStep32	32 micro-steps per one full-step.

4.1.2.6 lvhb_motor_status_t

`enum lvhb_motor_status_t`

Motor status.

Enumerator

lvhbStatusRunning	Motor is running.
lvhbStatusStop	Motor is not running.

4.1.2.7 lvhb_motor_type_t

`enum lvhb_motor_type_t`

Motor type.

Enumerator

lvhbMotorStepper	Stepper motor.
lvhbMotorBrushed	Brush motor.

4.1.2.8 lvhb_output_direction_t

`enum lvhb_output_direction_t`

Bridge output direction.

Enumerator

lvhbDirectionALowBHigh	INxA (IN1 by MPC1751x) is Low, INxB (IN2 by MPC1751x) is High.
lvhbDirectionAHighBLow	INxA (IN1 by MPC1751x) is High, INxB (IN2 by MPC1751x) is Low.

4.1.2.9 lvhb_output_state_t

`enum lvhb_output_state_t`

Output state.

Enumerator

lvhbOutputOff	Output turned off.
lvhbOutputOn	Output turned on.

4.1.2.10 lvhb_ramp_state_t

`enum lvhb_ramp_state_t`

States of finite state machine which are used for acceleration ramp.

Enumerator

lvhbRampOff	Motor is not running.
lvhbRampUpSpeed	Acceleration phase.
lvhbRampDownSpeed	Deceleration phase.
lvhbRampConstSpeed	Motor is running at constant speed.

4.1.2.11 lvhb_recirculation_t

`enum lvhb_recirculation_t`

Recirculation side.

Enumerator

lvhbRecirculationLow	Low-side recirculation.
lvhbRecirculationHighImp	High-impedance recirculation.

4.1.2.12 lvhb_stepper_mode_t

`enum lvhb_stepper_mode_t`

Stepper motor mode.

Enumerator

lvhbStepperModeMicrostep	Motor is in micro-stepping mode.
lvhbStepperModeFullstep	Motor is in full-stepping mode.
lvhbStepperModeDisabled	Motor is disabled (no signal generated on IN pins).

4.1.2.13 lvhb_stepper_state_t

```
enum lvhb_stepper_state_t
```

Stepper motor control states.

Enumerator

lvhbStepperStateSleepLoop	Nothing to do.
lvhbStepperStateInitCompleted	First part of full-step or micro-step initialization is completed.
lvhbStepperStateSlowStart	Second part of full-step or micro-step initialization is completed.

4.2 Struct_group

Data Structures

- struct tmr_lvhb_config_t

  Driver specific Timer configuration.
- struct lvhb_brush_config_t

  Brushed motor configuration.
- struct lvhb_stepper_config_t

  Stepper motor configuration.
- struct lvhb_stepper_data_t

  Stepper motor control data structure.
- struct lvhb_device_data_t

  Device configuration.
- struct lvhb_drv_config_t

  Driver configuration.

4.2.1 Detailed Description

4.3 Function_group

Functions

- status_t LVHB_ConfigureGpio (lvhb_drv_config_t *const drvConfig)

  This function configures GPIO for usage with this driver.
- status_t LVHB_ConfigureTimer (lvhb_drv_config_t *const drvConfig, tmr_sdk_config_t *const tmrSdkConfig)

  This function configures Timer for usage with this driver.
- status_t LVHB_GetDefaultConfig (lvhb_drv_config_t *const drvConfig, lvhb_device_t device, lvhb_motor_type_t motorType)

  This function gets a default configuration of the driver for specific device and motor type.
- status_t LVHB_Init (lvhb_drv_config_t *const initConfig)

  This function initializes driver data and output pin values.
- status_t LVHB_Deinit (lvhb_drv_config_t *const initConfig)

  This function deinitializes the driver.
- status_t LVHB_SetMode (lvhb_drv_config_t *const drvConfig, bool active)

  This method sets H-Bridge device mode using enable pin.
- status_t LVHB_SetGateDriver (lvhb_drv_config_t *const drvConfig, bool outputHigh)

  This function controls Gate Driver Input (GIN) pin. It is available for MPC17510 and MPC17511 only.
- status_t LVHB_RotateFull (lvhb_drv_config_t *const drvConfig, lvhb_output_state_t state, lvhb_bridge_t bridge)

  This function spins the motor in desired direction at full speed.
- status_t LVHB_RotateProportional (lvhb_drv_config_t *const drvConfig, uint8_t pwmDuty, lvhb_bridge_t bridge)

  This function spins the motor in desired direction at PWM duty speed.
- status_t LVHB_SetTriState (lvhb_drv_config_t *const drvConfig, lvhb_bridge_t bridge)

  This function sets output of specified H-Bridge to tri-state (high impedance) using input control pins.
- status_t LVHB_SetDirection (lvhb_drv_config_t *const drvConfig, lvhb_output_direction_t direction, lvhb_bridge_t bridge)

  This function sets direction of brush motor at specified H-Bridge interface.
- status_t LVHB_SetRecirculation (lvhb_drv_config_t *const drvConfig, lvhb_recirculation_t side, lvhb_bridge_t bridge)

  This function sets low/high-impedance-side recirculation of the H-Bridge.
- status_t LVHB_SetFullStepSpeed (lvhb_drv_config_t *const drvConfig, uint16_t stepsSec)

  This function sets the speed of full-step mode.
- status_t LVHB_SetMicroStepSpeed (lvhb_drv_config_t *const drvConfig, uint16_t microStepsSec)

  This function sets the speed of micro-step mode.
- status_t LVHB_SetFullStepAcceleration (lvhb_drv_config_t *const drvConfig, uint32_t acceleration)

  This function sets the acceleration ramp of full-step mode.
- status_t LVHB_SetMicroStepAcceleration (lvhb_drv_config_t *const drvConfig, uint32_t acceleration)

  This function sets the acceleration ramp of micro-step mode.
- status_t LVHB_MoveSteps (lvhb_drv_config_t *const drvConfig, bool forward, uint32_t steps)

  This function moves motor by specified number of full-steps.
- status_t LVHB_MoveMicroSteps (lvhb_drv_config_t *const drvConfig, bool forward, uint32_t microSteps)

  This function moves motor by specified number of micro-steps.
- status_t LVHB_MoveContinual (lvhb_drv_config_t *const drvConfig, bool forward)

  This function moves motor continually in full-step mode.
- status_t LVHB_MoveMicroContinual (lvhb_drv_config_t *const drvConfig, bool forward)

  This function moves motor continually in micro-step mode.
- status_t LVHB_StopContinualMovement (lvhb_drv_config_t *const drvConfig)

  This function is intended to stop continual movement of stepper motor.

- lvhb_motor_status_t LVHB_GetMotorStatus (lvhb_drv_config_t *const drvConfig)

  This function returns status of stepper motor control.
- status_t LVHB_AlignRotor (lvhb_drv_config_t *const drvConfig)

  This function aligns rotor to the full-step position.
- status_t LVHB_SetMicroStepSize (lvhb_drv_config_t *const drvConfig, lvhb_micro_step_t microStepsPer↩ Step)

  This function changes the size of micro-step.
- int32_t LVHB_GetFullStepPosition (lvhb_drv_config_t *const drvConfig)

  This function returns the current full-step position.
- int32_t LVHB_GetMicroStepPosition (lvhb_drv_config_t *const drvConfig)

  This function returns the current micro-step position.
- status_t LVHB_ResetFullStepPosition (lvhb_drv_config_t *const drvConfig)

  This function sets the counter of full-steps to zero.
- status_t LVHB_DisableMotor (lvhb_drv_config_t *const drvConfig)

  This function disables the stepper motor.
- void LVHB_OnCounterRestart (lvhb_drv_config_t *const drvConfig)

  Counter restart event handler. This function must be called from counter restart interrupt handler.
- void LVHB_OnActionComplete (lvhb_drv_config_t *const drvConfig)

  Declaration of function that can be used by user for handling the action complete event. This event occurs when "LV↩ HB_MoveSteps", "LVHB_MoveMicroSteps", "LVHB_StopContinualMovement" or "LVHB_AlignRotor" action is done.

4.3.1 Detailed Description

4.3.2 Function Documentation

4.3.2.1 LVHB_AlignRotor()

```
status_t LVHB_AlignRotor (
            lvhb_drv_config_t *const drvConfig )
```

This function aligns rotor to the full-step position.

The method executes 4 full-steps forward (one electrical revolution) at minimum speed (see LVHB_FULLSTEP_↩ SPEED_MIN constant). These steps are not counted to the number of full-steps. The input timer frequency must be between LVHB_FULLSTEP_IN_FREQ_MIN and LVHB_FULLSTEP_IN_FREQ_MAX. Note that the motor must not run when this function is called and you must wait for completion of this action before you can run motor again (use method LVHB_GetMotorStatus or event LVHB_OnActionComplete).

This function may be used for the stepper motors control only.

Parameters

drvConfig	Pointer to driver instance configuration.

Returns

status_t Error code.

4.3.2.2 LVHB_ConfigureGpio()

```
status_t LVHB_ConfigureGpio (
            lvhb_drv_config_t *const drvConfig )
```

This function configures GPIO for usage with this driver.

This function initializes all used GPIO pins. These pins are identified according to the selected device and INx pin types in the driver configuration.

Parameters

drvConfig	Pointer to driver instance configuration.

Returns

status_t Error code.

4.3.2.3 LVHB_ConfigureTimer()

```
status_t LVHB_ConfigureTimer (
            lvhb_drv_config_t *const drvConfig,
            tmr_sdk_config_t *const tmrSdkConfig )
```

This function configures Timer for usage with this driver.

This function initializes Timer device. There are two parameters. lvhb_drv_config_t must be used always with appropriate driver settings but tmr_sdk_config_t could be used for custom user SDK settings (use NULL pointer for default settings). There is no need to call this function if GPIO control of brushed motors is used.

Parameters

drvConfig	Pointer to driver instance configuration.
tmrSdkConfig	SDK specific configuration. Use NULL pointer for default settings.

Returns

status_t Error code.

4.3.2.4 LVHB_Deinit()

```
status_t LVHB_Deinit (
            lvhb_drv_config_t *const initConfig )
```

This function deinitializes the driver.

Parameters

initConfig	Pointer to initial driver configuration.

Returns

status_t Error code.

4.3.2.5 LVHB_DisableMotor()

```
status_t LVHB_DisableMotor (
             lvhb_drv_config_t *const drvConfig )
```

This function disables the stepper motor.

This function disables timer interrupts and sets the INxA and INxB pins output value to LOW. It can be used to stop the stepper motor. Note that default behavior of the motor control is to hold position when a movement is completed.

This function may be used for the stepper motors control only.

Parameters

drvConfig	Pointer to driver instance configuration.

Returns

status_t Error code.

4.3.2.6 LVHB_GetDefaultConfig()

```
status_t LVHB_GetDefaultConfig (
             lvhb_drv_config_t *const drvConfig,
             lvhb_device_t device,
             lvhb_motor_type_t motorType )
```

This function gets a default configuration of the driver for specific device and motor type.

Default device configuration will be stored to the defaultConfig pointer.

Parameters

drvConfig	Pointer to variable where the configuration will be stored.
device	LVHB device.
motorType	Motor type to control by this driver.

Returns

status_t Error code.

4.3.2.7 LVHB_GetFullStepPosition()

```
int32_t LVHB_GetFullStepPosition (
            lvhb_drv_config_t *const drvConfig )
```

This function returns the current full-step position.

Position is set to zero by the LVHB driver initialization. It can be reset by LVHB_ResetFullStepPosition function.

This function may be used for the stepper motors control only.

Parameters

drvConfig	Pointer to driver instance configuration.

Returns

Current rotor position in number of full-steps from initial position.

4.3.2.8 LVHB_GetMicroStepPosition()

```
int32_t LVHB_GetMicroStepPosition (
            lvhb_drv_config_t *const drvConfig )
```

This function returns the current micro-step position.

Size of micro-step depends on current "Micro-steps per Step" setting. Position is set to zero by the LVHB driver initialization.

This function may be used for the stepper motors control with micro-stepping support only - ie. all IN pins are used as timer outputs.

Parameters

drvConfig	Pointer to driver instance configuration.

Returns

Current rotor position in number of micro-steps from initial position.

4.3.2.9 LVHB_GetMotorStatus()

```
lvhb_motor_status_t LVHB_GetMotorStatus (
            lvhb_drv_config_t *const drvConfig )
```

This function returns status of stepper motor control.

Possible returned values are defined in lvhb_motor_status_t enumeration.

This function may be used for the stepper motors control only.

Parameters

drvConfig	Pointer to driver instance configuration.

Returns

lvhb_motor_status_t Motor status.

4.3.2.10 LVHB_Init()

```
status_t LVHB_Init (
            lvhb_drv_config_t *const initConfig )
```

This function initializes driver data and output pin values.

Initialization of all used peripherals and stepper control variables is performed. GIN and OE/EN/PSAVE pin values are set according to the driver configuration. INx signals are set to get LOW at OUTx pins of used H-Bridge.

Parameters

initConfig	Pointer to initial driver configuration.

Returns

status_t Error code.

4.3.2.11 LVHB_MoveContinual()

```
status_t LVHB_MoveContinual (
            lvhb_drv_config_t *const drvConfig,
            bool forward )
```

This function moves motor continually in full-step mode.

The motor can be stopped by LVHB_StopContinualMovement function. When the rotor is not at physical full-step position then the function sets nearest full-step without correction. The input timer frequency must be between LVHB_FULLSTEP_IN_FREQ_MIN and LVHB_FULLSTEP_IN_FREQ_MAX. Note that the motor must not run when this function is called.

This function may be used only for the stepper motors control when the acceleration ramp is disabled (full-step acceleration equals 0).

Parameters

drvConfig	Pointer to driver instance configuration.
forward	Direction of movement.

Returns

status_t Error code.

4.3.2.12 LVHB_MoveMicroContinual()

```
status_t LVHB_MoveMicroContinual (
            lvhb_drv_config_t *const drvConfig,
            bool forward )
```

This function moves motor continually in micro-step mode.

The motor can be stopped by LVHB_StopContinualMovement function. When the rotor is not at physical micro-step position then the function sets nearest micro-step without correction. For example the size is initialized to 32 micro-steps per one full-step and the motor executed three micro-steps. Then user changes micro-step size to 2 micro-steps per one full-step and starts motor movement (previous three micro-steps are not visible). The input timer frequency must be between LVHB_MICROSTEP_IN_FREQ_MIN and LVHB_MICROSTEP_IN_FREQ_MAX. Note that the motor must not run when this function is called.

This function may be used only for the stepper motors control with micro-stepping support only (ie. all IN pins are used as timer outputs) when the acceleration ramp is disabled (micro-step acceleration equals 0).

Parameters

drvConfig	Pointer to driver instance configuration.
forward	Direction of movement.

Returns

status_t Error code.

4.3.2.13 LVHB_MoveMicroSteps()

```
status_t LVHB_MoveMicroSteps (
            lvhb_drv_config_t *const drvConfig,
```

```
            bool forward,
            uint32_t microSteps )
```

This function moves motor by specified number of micro-steps.

When the rotor is not at physical micro-step position then the function sets nearest micro-step without correction. For example the size is initialized to 32 micro-steps per one full-step and the motor executed three micro-steps. Then user changes micro-step size to 2 micro-steps per one full-step and starts motor movement (previous three micro-steps are not visible). The input timer frequency must be between LVHB_MICROSTEP_IN_FREQ_MIN and LVHB_MICROSTEP_IN_FREQ_MAX. Note that the motor must not run when this function is called and you must wait for completion of this action before you can run motor again (use method LVHB_GetMotorStatus or event LVHB_OnActionComplete).

This function may be used for the stepper motors control with micro-stepping support only - ie. all IN pins are used as timer outputs.

Parameters

drvConfig	Pointer to driver instance configuration.
forward	Direction of movement.
microSteps	Number of micro-steps to move (equal or less than LVHB_STEPPER_MAX_STEPS).

Returns

status_t Error code.

4.3.2.14 LVHB_MoveSteps()

```
status_t LVHB_MoveSteps (
            lvhb_drv_config_t *const drvConfig,
            bool forward,
            uint32_t steps )
```

This function moves motor by specified number of full-steps.

When the rotor is not at physical micro-step position then the function sets nearest micro-step without correction. The input timer frequency must be between LVHB_FULLSTEP_IN_FREQ_MIN and LVHB_FULLSTEP_IN_FR↩EQ_MAX. Note that number of steps returned by method LVHB_GetFullStepPosition are updated before they are executed. Note that the motor must not run when this function is called and you must wait for completion of this action before you can run motor again (use method LVHB_GetMotorStatus or event LVHB_OnActionComplete).

This function may be used for the stepper motors control only.

Parameters

drvConfig	Pointer to driver instance configuration.
forward	Direction of movement.
steps	Number of steps to move (equal or less than LVHB_STEPPER_MAX_STEPS).

Returns

status_t Error code.

4.3.2.15 LVHB_OnActionComplete()

```
void LVHB_OnActionComplete (
            lvhb_drv_config_t *const drvConfig )
```

Declaration of function that can be used by user for handling the action complete event. This event occurs when "LVHB_MoveSteps", "LVHB_MoveMicroSteps", "LVHB_StopContinualMovement" or "LVHB_AlignRotor" action is done.

Parameters

drvConfig	Pointer to driver instance configuration.

4.3.2.16 LVHB_OnCounterRestart()

```
void LVHB_OnCounterRestart (
            lvhb_drv_config_t *const drvConfig )
```

Counter restart event handler. This function must be called from counter restart interrupt handler.

This function may be used for the stepper motors control only.

Parameters

drvConfig	Pointer to driver instance configuration.

4.3.2.17 LVHB_ResetFullStepPosition()

```
status_t LVHB_ResetFullStepPosition (
            lvhb_drv_config_t *const drvConfig )
```

This function sets the counter of full-steps to zero.

This function may be used for the stepper motors control only.

Parameters

drvConfig	Pointer to driver instance configuration.

Returns

status_t Error code.

4.3.2.18 LVHB_RotateFull()

```
status_t LVHB_RotateFull (
             lvhb_drv_config_t *const drvConfig,
             lvhb_output_state_t state,
             lvhb_bridge_t bridge )
```

This function spins the motor in desired direction at full speed.

This function serves to output state control of brushed motors, it must not be used for stepper motor control.

Parameters

drvConfig	Pointer to driver instance configuration.
state	State of outputs (turned on/off).
bridge	H-Bridge of device the motor is connected to. Only lvhbBridge1 value is correct when single H-Bridge device is used.

Returns

status_t Error code.

4.3.2.19 LVHB_RotateProportional()

```
status_t LVHB_RotateProportional (
             lvhb_drv_config_t *const drvConfig,
             uint8_t pwmDuty,
             lvhb_bridge_t bridge )
```

This function spins the motor in desired direction at PWM duty speed.

This function serves to speed control of brushed motors. It may be used only for brushed motor control with one non-GPIO pin at least.

Parameters

drvConfig	Pointer to driver instance configuration.
pwmDuty	- PWM duty, Possible values: 0 - 100, where 0 is minimum and 100 is maximum.
bridge	H-Bridge of device the motor is connected to. Only lvhbBridge1 value is correct when single H-Bridge device is used.

Returns

status_t - Error code.

4.3.2.20 LVHB_SetDirection()

```
status_t LVHB_SetDirection (
            lvhb_drv_config_t *const drvConfig,
            lvhb_output_direction_t direction,
            lvhb_bridge_t bridge )
```

This function sets direction of brush motor at specified H-Bridge interface.

In lvhbDirectionAHighBLow direction the first IN pin is set to high (or PWM) and the second IN pin to low. In reverse direction the first IN is set to low and the second IN to high (or PWM). Change of the direction is applied when you start rotation (not when the motor is running). This function may be used in case of brushed motor control only.

Parameters

drvConfig	Pointer to driver instance configuration.
direction	Desired motor direction from lvhb_output_direction_t enumeration.
bridge	H-Bridge of device the motor is connected to. Only lvhbBridge1 value is correct when single H-Bridge device is used.

Returns

status_t Error code.

4.3.2.21 LVHB_SetFullStepAcceleration()

```
status_t LVHB_SetFullStepAcceleration (
            lvhb_drv_config_t *const drvConfig,
            uint32_t acceleration )
```

This function sets the acceleration ramp of full-step mode.

Unit is full-steps per second^2. Put 0 value to disable ramp. Note that it is not allowed to change acceleration ramp while motor is running.

This function may be used for the stepper motors control only.

Parameters

drvConfig	Pointer to driver instance configuration.
acceleration	Motor acceleration in number of full-steps per second^2. Maximal acceleration is LVHB_FULLSTEP_ACCEL_MAX.

Returns

status_t Error code.

4.3.2.22 LVHB_SetFullStepSpeed()

```
status_t LVHB_SetFullStepSpeed (
            lvhb_drv_config_t *const drvConfig,
            uint16_t stepsSec )
```

This function sets the speed of full-step mode.

Unit is number of full-steps per second. Note that it is not allowed to change speed while motor is running.

This function may be used for the stepper motors control only.

Parameters

drvConfig	Pointer to driver instance configuration.
stepsSec	Motor speed in number of full-steps per second. Minimal and maximal speed is defined by LVHB_FULLSTEP_SPEED_MIN and LVHB_FULLSTEP_SPEED_MAX.

Returns

status_t Error code.

4.3.2.23 LVHB_SetGateDriver()

```
status_t LVHB_SetGateDriver (
            lvhb_drv_config_t *const drvConfig,
            bool outputHigh )
```

This function controls Gate Driver Input (GIN) pin. It is available for MPC17510 and MPC17511 only.

Parameters

drvConfig	Pointer to driver instance configuration.
outputHigh	TRUE to set GOUT pin to High, FALSE for Low.

Returns

status_t Error code.

4.3.2.24 LVHB_SetMicroStepAcceleration()

```
status_t LVHB_SetMicroStepAcceleration (
            lvhb_drv_config_t *const drvConfig,
            uint32_t acceleration )
```

This function sets the acceleration ramp of micro-step mode.

Unit is micro-steps per second^2. Put 0 value to disable ramp. Note that it is not allowed to change acceleration ramp while motor is running.

This function may be used for the stepper motors control with micro-stepping support only - ie. all IN pins are used as timer outputs.

Parameters

drvConfig	Pointer to driver instance configuration.
acceleration	Motor acceleration in number of micro-steps per second^2. Maximal acceleration is LVHB_MICROSTEP_ACCEL_MAX.

Returns

status_t Error code.

4.3.2.25 LVHB_SetMicroStepSize()

```
status_t LVHB_SetMicroStepSize (
            lvhb_drv_config_t *const drvConfig,
            lvhb_micro_step_t microStepsPerStep )
```

This function changes the size of micro-step.

Note that the motor must not run when this function is called.

This function may be used for the stepper motors control with micro-stepping support only - ie. all IN pins are used as timer outputs.

Parameters

drvConfig	Pointer to driver instance configuration.
microStepsPerStep	New size from lvhb_micro_step_t enumeration.

Returns

status_t Error code.

4.3.2.26 LVHB_SetMicroStepSpeed()

```
status_t LVHB_SetMicroStepSpeed (
            lvhb_drv_config_t *const drvConfig,
            uint16_t microStepsSec )
```

This function sets the speed of micro-step mode.

Unit is number of micro-steps per second. Size of micro-step depends on current driver setting. Note that it is not allowed to change speed while motor is running.

This function may be used for the stepper motors control with micro-stepping support only - ie. all IN pins are used as timer outputs.

Parameters

drvConfig	Pointer to driver instance configuration.
microStepsSec	Motor speed in number of micro-steps per second. Minimal and maximal speed is defined by LVHB_MICROSTEP_SPEED_MIN and LVHB_MICROSTEP_SPEED_MAX.

Returns

status_t Error code.

4.3.2.27 LVHB_SetMode()

```
status_t LVHB_SetMode (
            lvhb_drv_config_t *const drvConfig,
            bool active )
```

This method sets H-Bridge device mode using enable pin.

This function may be used by devices with EN, OE or PSAVE pin only - ie. all supported devices except MC34933.

Parameters

drvConfig	Pointer to driver instance configuration.
active	H-Bridge mode. Put FALSE to set power save mode, TRUE for normal operational mode.

Returns

status_t Error code.

4.3.2.28 LVHB_SetRecirculation()

```
status_t LVHB_SetRecirculation (
            lvhb_drv_config_t *const drvConfig,
```

```
            lvhb_recirculation_t side,
            lvhb_bridge_t bridge )
```

This function sets low/high-impedance-side recirculation of the H-Bridge.

When Low-side recirculation (freewheeling) is held while a DC motor is spinning, the back-EMF of the motor will cause a recirculating current to flow that will rapidly brake the motor to a stop. This is known as dynamic braking.

Parameters

drvConfig	Pointer to driver instance configuration.
side	Low or High-impedance-side recirculation.
bridge	H-Bridge of device the motor is connected to. Only lvhbBridge1 value is correct when single H-Bridge device is used.

Returns

status_t Error code.

4.3.2.29 LVHB_SetTriState()

```
status_t LVHB_SetTriState (
            lvhb_drv_config_t *const drvConfig,
            lvhb_bridge_t bridge )
```

This function sets output of specified H-Bridge to tri-state (high impedance) using input control pins.

This function may be used in case of brushed motor control only.

Parameters

drvConfig	Pointer to driver instance configuration.
bridge	H-Bridge of device the motor is connected to. Only lvhbBridge1 value is correct when single H-Bridge device is used.

Returns

status_t Error code.

4.3.2.30 LVHB_StopContinualMovement()

```
status_t LVHB_StopContinualMovement (
            lvhb_drv_config_t *const drvConfig )
```

This function is intended to stop continual movement of stepper motor.

This function does not stop motor immediately, motor can execute several steps. In micro-step mode the motor does not have to stop at full-step position (can stop anywhere). Note that you must wait for completion of this action before you can run motor again (use method LVHB_GetMotorStatus or event LVHB_OnActionComplete).

This function may be used for the stepper motors control only.

Parameters

drvConfig	Pointer to driver instance configuration.

Returns

status_t Error code.

Chapter 5

Data Structure Documentation

5.1 lvhb_brush_config_t Struct Reference

Brushed motor configuration.

`#include <lvhb.h>`

Data Fields

- lvhb_recirculation_t recirculation
- lvhb_output_direction_t outputDirection

5.1.1 Detailed Description

Brushed motor configuration.

5.1.2 Field Documentation

5.1.2.1 outputDirection

`lvhb_output_direction_t lvhb_brush_config_t::outputDirection`

Direction of rotation (state of outputs) of used bridges.

5.1.2.2 recirculation

`lvhb_recirculation_t lvhb_brush_config_t::recirculation`

Recirculation side (high-impedance, low) of used bridges.

The documentation for this struct was generated from the following file:

- lvhb/lvhb.h

5.2 lvhb_device_data_t Struct Reference

Device configuration.

`#include <lvhb.h>`

Data Fields

- lvhb_device_t device
- lvhb_motor_type_t motorType
- lvhb_brush_config_t brushConfig [LVHB_BRIDGE_COUNT_MAX]
- uint32_t brushPwmFrequency
- lvhb_stepper_config_t stepperConfig
- lvhb_stepper_data_t stepperData
- bool secondaryBridgeUsed
- bool activeMode
- bool gateDriverOutputHigh

5.2.1 Detailed Description

Device configuration.

This structure contains device configuration.

5.2.2 Field Documentation

5.2.2.1 activeMode

`bool lvhb_device_data_t::activeMode`

Mode of H-Bridge. Used by all H-Bridges except MC34933.

5.2.2.2 brushConfig

`lvhb_brush_config_t lvhb_device_data_t::brushConfig[LVHB_BRIDGE_COUNT_MAX]`

Brush motor configuration.

5.2.2.3 brushPwmFrequency

`uint32_t lvhb_device_data_t::brushPwmFrequency`

PWM frequency for speed control of brushed motors. Maximum value is 20 kHz.

5.2.2.4 device

```
lvhb_device_t lvhb_device_data_t::device
```

H-Bridge device.

5.2.2.5 gateDriverOutputHigh

```
bool lvhb_device_data_t::gateDriverOutputHigh
```

GIN pin value. Used by MPC17510 and MPC17511 only.

5.2.2.6 motorType

```
lvhb_motor_type_t lvhb_device_data_t::motorType
```

Motor type.

5.2.2.7 secondaryBridgeUsed

```
bool lvhb_device_data_t::secondaryBridgeUsed
```

Usage of second bridge in case of brush motor control. Not used by MPC17510 and MPC17511.

5.2.2.8 stepperConfig

```
lvhb_stepper_config_t lvhb_device_data_t::stepperConfig
```

Stepper motor configuration.

5.2.2.9 stepperData

```
lvhb_stepper_data_t lvhb_device_data_t::stepperData
```

Stepper motor data.

The documentation for this struct was generated from the following file:

- lvhb/lvhb.h

5.3 lvhb_drv_config_t Struct Reference

Driver configuration.

```
#include <lvhb.h>
```

Data Fields

- aml_instance_t enPinInstance
- uint8_t enPinIndex
- aml_instance_t ginPinInstance
- uint8_t ginPinIndex
- lvhb_input_pins_t inputPins [LVHB_BRIDGE_COUNT_MAX]
- aml_instance_t inxaPinInstance [LVHB_BRIDGE_COUNT_MAX]
- uint8_t inxaPinIndex [LVHB_BRIDGE_COUNT_MAX]
- aml_instance_t inxbPinInstance [LVHB_BRIDGE_COUNT_MAX]
- uint8_t inxbPinIndex [LVHB_BRIDGE_COUNT_MAX]
- aml_instance_t tmrInstance
- tmr_lvhb_config_t tmrLvhbConfig
- lvhb_device_data_t deviceConfig

5.3.1 Detailed Description

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.

5.3.2 Field Documentation

5.3.2.1 deviceConfig

```
lvhb_device_data_t lvhb_drv_config_t::deviceConfig
```

Device configuration.

5.3.2.2 enPinIndex

```
uint8_t lvhb_drv_config_t::enPinIndex
```

EN (MPC17510, MPC17511) / OE (MPC17529, MPC17533) / PSAVE (MPC17C724, MPC17531A) pin index. Not used by MC34933.

5.3.2.3 enPinInstance

```
aml_instance_t lvhb_drv_config_t::enPinInstance
```

EN (MPC17510, MPC17511) / OE (MPC17529, MPC17533) / PSAVE (MPC17C724, MPC17531A) port instance. Not used by MC34933.

5.3.2.4 ginPinIndex

```
uint8_t lvhb_drv_config_t::ginPinIndex
```

GIN pin index. Used by MPC17510 and MPC17511 only.

5.3.2.5 ginPinInstance

```
aml_instance_t lvhb_drv_config_t::ginPinInstance
```

GIN pin port instance. Used by MPC17510 and MPC17511 only.

5.3.2.6 inputPins

```
lvhb_input_pins_t lvhb_drv_config_t::inputPins[LVHB_BRIDGE_COUNT_MAX]
```

GPIO, GPIO/PWM, PWM/GPIO, or PWM.

5.3.2.7 inxaPinIndex

```
uint8_t lvhb_drv_config_t::inxaPinIndex[LVHB_BRIDGE_COUNT_MAX]
```

IN1/IN1A (first bridge) & IN2A (second bridge) GPIO pin index.

5.3.2.8 inxaPinInstance

```
aml_instance_t lvhb_drv_config_t::inxaPinInstance[LVHB_BRIDGE_COUNT_MAX]
```

IN1/IN1A (first bridge) & IN2A (second bridge) GPIO pin port instance.

5.3.2.9 inxbPinIndex

```
uint8_t lvhb_drv_config_t::inxbPinIndex[LVHB_BRIDGE_COUNT_MAX]
```

IN2/IN1B (first bridge) & IN2B (second bridge) GPIO pin index.

5.3.2.10 inxbPinInstance

```
aml_instance_t lvhb_drv_config_t::inxbPinInstance[LVHB_BRIDGE_COUNT_MAX]
```

IN2/IN1B (first bridge) & IN2B (second bridge) GPIO pin port instance.

5.3.2.11 tmrInstance

```
aml_instance_t lvhb_drv_config_t::tmrInstance
```

Timer instance - only if any PWM pin used.

5.3.2.12 tmrLvhbConfig

```
tmr_lvhb_config_t lvhb_drv_config_t::tmrLvhbConfig
```

Device Timer configuration - only if any PWM pin used.

The documentation for this struct was generated from the following file:

- lvhb/lvhb.h

5.4 lvhb_stepper_config_t Struct Reference

Stepper motor configuration.

```
#include <lvhb.h>
```

Data Fields

- uint32_t fullStepSpeed
- uint32_t fullStepAcceleration
- uint32_t microStepSpeed
- uint32_t microStepAcceleration
- lvhb_micro_step_t microStepPerStep
- uint32_t microStepPwmFrequency

5.4.1 Detailed Description

Stepper motor configuration.

5.4.2 Field Documentation

5.4.2.1 fullStepAcceleration

```
uint32_t lvhb_stepper_config_t::fullStepAcceleration
```

Fluent acceleration to desired speed and decelaration to zero. Put 0 value to disable ramp. Unit is full-steps per second^2.

5.4.2.2 fullStepSpeed

```
uint32_t lvhb_stepper_config_t::fullStepSpeed
```

Motor speed in Full-step mode. Unit is number of full-steps per second. Admissible range is 1 to 5000.

5.4.2.3 microStepAcceleration

```
uint32_t lvhb_stepper_config_t::microStepAcceleration
```

Fluent acceleration to desired speed and decelaration to zero. Put 0 value to disable ramp. Unit is micro-steps per second^2.

5.4.2.4 microStepPerStep

```
lvhb_micro_step_t lvhb_stepper_config_t::microStepPerStep
```

Number of micro-steps per one full-step.

5.4.2.5 microStepPwmFrequency

```
uint32_t lvhb_stepper_config_t::microStepPwmFrequency
```

Desired timer frequency for micro-stepping. Minimum value is 10 kHz and maximum value is 20 kHz.

5.4.2.6 microStepSpeed

```
uint32_t lvhb_stepper_config_t::microStepSpeed
```

Motor speed in Micro-step mode. Unit is number of micro-steps per second. Admissible range is 1 to 5000.

The documentation for this struct was generated from the following file:

- lvhb/lvhb.h

5.5 lvhb_stepper_data_t Struct Reference

Stepper motor control data structure.

```
#include <lvhb.h>
```

Data Fields

- bool forward
- bool continual
- uint32_t stepCntr
- uint32_t steps
- int32_t stepPos
- uint8_t fullStepIdx
- lvhb_stepper_mode_t stepperMode
- lvhb_motor_status_t motorStatus
- lvhb_stepper_state_t controlState
- uint32_t timerOverflows
- uint32_t fullSpeed
- uint8_t fullStepsPerPeriod
- uint32_t currentFullSpeed
- uint32_t microSpeed
- uint8_t microWind1Idx
- uint8_t microWind2Idx
- int8_t microPosInFull
- uint16_t microStepOffset [LVHB_MICROSTEP_CNT_MAX+1]
- lvhb_ramp_state_t rampState
- uint32_t maxSpeedDelay
- uint32_t currentDelay
- uint32_t currentDelayRest
- uint32_t decelStepCnt

5.5.1 Detailed Description

Stepper motor control data structure.

5.5.2 Field Documentation

5.5.2.1 continual

```
bool lvhb_stepper_data_t::continual
```

Continual movement.

5.5.2.2 controlState

```
lvhb_stepper_state_t lvhb_stepper_data_t::controlState
```

States for control generating steps.

5.5.2.3 currentDelay

```
uint32_t lvhb_stepper_data_t::currentDelay
```

Current timer delay.

5.5.2.4 currentDelayRest

```
uint32_t lvhb_stepper_data_t::currentDelayRest
```

Rest of the division in current timer delay computation.

5.5.2.5 currentFullSpeed

```
uint32_t lvhb_stepper_data_t::currentFullSpeed
```

Speed used in current movement.

5.5.2.6 decelStepCnt

```
uint32_t lvhb_stepper_data_t::decelStepCnt
```

Number of steps for deceleration.

5.5.2.7 forward

```
bool lvhb_stepper_data_t::forward
```

Current direction of motor movement.

5.5.2.8 fullSpeed

```
uint32_t lvhb_stepper_data_t::fullSpeed
```

Speed in full-step mode in number of timer clock source ticks (equal to timer period).

5.5.2.9 fullStepIdx

```
uint8_t lvhb_stepper_data_t::fullStepIdx
```

Index to the table with full-step sequence values (used in micro-step and full-step).

5.5.2.10 fullStepsPerPeriod

```
uint8_t lvhb_stepper_data_t::fullStepsPerPeriod
```

Number of full-steps executed until next timer restart event (counting of full-steps).

5.5.2.11 maxSpeedDelay

```
uint32_t lvhb_stepper_data_t::maxSpeedDelay
```

Timer delay when max speed is reached (Size of 1 full-step in seconds).

5.5.2.12 microPosInFull

```
int8_t lvhb_stepper_data_t::microPosInFull
```

No. of taken micro-steps from full-step position (it is zero at physic full-step position).

5.5.2.13 microSpeed

```
uint32_t lvhb_stepper_data_t::microSpeed
```

Speed of micro-stepping in number of timer overflows.

5.5.2.14 microStepOffset

```
uint16_t lvhb_stepper_data_t::microStepOffset[LVHB_MICROSTEP_CNT_MAX+1]
```

Table for tmrInstance offset values for 32 micro-steps per one full-step (in number of ticks).

5.5.2.15 microWind1Idx

```
uint8_t lvhb_stepper_data_t::microWind1Idx
```

Index to the table with micro-step offsets. Intended for first winding.

5.5.2.16 microWind2Idx

```
uint8_t lvhb_stepper_data_t::microWind2Idx
```

Index to the table with micro-step offsets. Intended for second winding.

5.5.2.17 motorStatus

```
lvhb_motor_status_t lvhb_stepper_data_t::motorStatus
```

Current motor state.

5.5.2.18 rampState

```
lvhb_ramp_state_t lvhb_stepper_data_t::rampState
```

State of the acceleration ramp.

5.5.2.19 stepCntr

```
uint32_t lvhb_stepper_data_t::stepCntr
```

Counter of steps.

5.5.2.20 stepperMode

```
lvhb_stepper_mode_t lvhb_stepper_data_t::stepperMode
```

Stepper motor mode (Full-stepping, micro-stepping or disabled).

5.5.2.21 stepPos

```
int32_t lvhb_stepper_data_t::stepPos
```

Actual position of motor in full-steps. Not used in algorithm, only for information. User have to check possible overflow and reset variable by provided function.

5.5.2.22 steps

```
uint32_t lvhb_stepper_data_t::steps
```

Number of steps executed in current movement.

5.5.2.23 timerOverflows

```
uint32_t lvhb_stepper_data_t::timerOverflows
```

Number of timer overflows in current step.

The documentation for this struct was generated from the following file:

- lvhb/lvhb.h

5.6 tmr_lvhb_config_t Struct Reference

Driver specific Timer configuration.

```
#include <lvhb.h>
```

Data Fields

- uint8_t counterWidth
- uint32_t srcClck_Hz
- tmr_clk_prescale_t prescale
- uint8_t inxaChannelNumber [LVHB_BRIDGE_COUNT_MAX]
- uint8_t inxbChannelNumber [LVHB_BRIDGE_COUNT_MAX]

5.6.1 Detailed Description

Driver specific Timer configuration.

5.6.2 Field Documentation

5.6.2.1 counterWidth

```
uint8_t tmr_lvhb_config_t::counterWidth
```

Counter width in bits.

5.6.2.2 inxaChannelNumber

```
uint8_t tmr_lvhb_config_t::inxaChannelNumber[LVHB_BRIDGE_COUNT_MAX]
```

INxA/IN1 channel number - used for PWM/PWM and PWM/GPIO pin control only.

5.6.2.3 inxbChannelNumber

```
uint8_t tmr_lvhb_config_t::inxbChannelNumber[LVHB_BRIDGE_COUNT_MAX]
```

INxB/IN2 channel number - used for PWM/PWM and GPIO/PWM pin control only.

5.6.2.4 prescale

```
tmr_clk_prescale_t tmr_lvhb_config_t::prescale
```

Timer prescale value.

5.6.2.5 srcClck_Hz

```
uint32_t tmr_lvhb_config_t::srcClck_Hz
```

Timer source clock in Hz.

The documentation for this struct was generated from the following file:

- lvhb/lvhb.h

Chapter 6

File Documentation

6.1 lvhb/lvhb.c File Reference

```
#include "lvhb.h"
```

Functions

- status_t LVHB_ConfigureGpio (lvhb_drv_config_t *drvConfig)

  This function configures GPIO for usage with this driver.
- status_t LVHB_ConfigureTimer (lvhb_drv_config_t *const drvConfig, tmr_sdk_config_t *const tmrSdkConfig)

  This function configures Timer for usage with this driver.
- status_t LVHB_GetDefaultConfig (lvhb_drv_config_t *const defaultConfig, lvhb_device_t device, lvhb_motor_type_t motorType)

  This function gets a default configuration of the driver for specific device and motor type.
- status_t LVHB_Init (lvhb_drv_config_t *const initConfig)

  This function initializes driver data and output pin values.
- status_t LVHB_Deinit (lvhb_drv_config_t *const initConfig)

  This function deinitializes the driver.
- status_t LVHB_SetMode (lvhb_drv_config_t *const drvConfig, bool active)

  This method sets H-Bridge device mode using enable pin.
- status_t LVHB_SetGateDriver (lvhb_drv_config_t *const drvConfig, bool outputHigh)

  This function controls Gate Driver Input (GIN) pin. It is available for MPC17510 and MPC17511 only.
- status_t LVHB_RotateFull (lvhb_drv_config_t *const drvConfig, lvhb_output_state_t state, lvhb_bridge_t bridge)

  This function spins the motor in desired direction at full speed.
- status_t LVHB_RotateProportional (lvhb_drv_config_t *const drvConfig, uint8_t pwmDuty, lvhb_bridge_t bridge)

  This function spins the motor in desired direction at PWM duty speed.
- status_t LVHB_SetTriState (lvhb_drv_config_t *const drvConfig, lvhb_bridge_t bridge)

  This function sets output of specified H-Bridge to tri-state (high impedance) using input control pins.
- status_t LVHB_SetDirection (lvhb_drv_config_t *const drvConfig, lvhb_output_direction_t direction, lvhb_bridge_t bridge)

  This function sets direction of brush motor at specified H-Bridge interface.
- status_t LVHB_SetRecirculation (lvhb_drv_config_t *const drvConfig, lvhb_recirculation_t side, lvhb_bridge_t bridge)

This function sets low/high-impedance-side recirculation of the H-Bridge.

- status_t LVHB_SetFullStepSpeed (lvhb_drv_config_t *const drvConfig, uint16_t stepsSec)

  This function sets the speed of full-step mode.
- status_t LVHB_SetMicroStepSpeed (lvhb_drv_config_t *const drvConfig, uint16_t microStepsSec)

  This function sets the speed of micro-step mode.
- status_t LVHB_SetFullStepAcceleration (lvhb_drv_config_t *const drvConfig, uint32_t acceleration)

  This function sets the acceleration ramp of full-step mode.
- status_t LVHB_SetMicroStepAcceleration (lvhb_drv_config_t *const drvConfig, uint32_t acceleration)

  This function sets the acceleration ramp of micro-step mode.
- status_t LVHB_MoveSteps (lvhb_drv_config_t *const drvConfig, bool forward, uint32_t steps)

  This function moves motor by specified number of full-steps.
- status_t LVHB_MoveMicroSteps (lvhb_drv_config_t *const drvConfig, bool forward, uint32_t microSteps)

  This function moves motor by specified number of micro-steps.
- status_t LVHB_MoveContinual (lvhb_drv_config_t *const drvConfig, bool forward)

  This function moves motor continually in full-step mode.
- status_t LVHB_MoveMicroContinual (lvhb_drv_config_t *const drvConfig, bool forward)

  This function moves motor continually in micro-step mode.
- status_t LVHB_StopContinualMovement (lvhb_drv_config_t *const drvConfig)

  This function is intended to stop continual movement of stepper motor.
- lvhb_motor_status_t LVHB_GetMotorStatus (lvhb_drv_config_t *const drvConfig)

  This function returns status of stepper motor control.
- status_t LVHB_AlignRotor (lvhb_drv_config_t *const drvConfig)

  This function aligns rotor to the full-step position.
- status_t LVHB_SetMicroStepSize (lvhb_drv_config_t *const drvConfig, lvhb_micro_step_t microStepsPer↩ Step)

  This function changes the size of micro-step.
- int32_t LVHB_GetFullStepPosition (lvhb_drv_config_t *const drvConfig)

  This function returns the current full-step position.
- int32_t LVHB_GetMicroStepPosition (lvhb_drv_config_t *const drvConfig)

  This function returns the current micro-step position.
- status_t LVHB_ResetFullStepPosition (lvhb_drv_config_t *const drvConfig)

  This function sets the counter of full-steps to zero.
- status_t LVHB_DisableMotor (lvhb_drv_config_t *const drvConfig)

  This function disables the stepper motor.
- void LVHB_OnCounterRestart (lvhb_drv_config_t *const drvConfig)

  Counter restart event handler. This function must be called from counter restart interrupt handler.

6.1.1 Detailed Description

Low voltage H-Bridge driver based on AML layer. Supports boards based on MPC17510, MPC17511, MPC17529, MPC17531A, MPC17533, MPC17C724 and MC34933.

This module is common for all supported models.

6.2 lvhb/lvhb.h File Reference

```
#include "stdbool.h"
#include "stdint.h"
#include "../aml/gpio_aml.h"
#include "../aml/tmr_aml/tmr_aml.h"
```

Data Structures

- struct tmr_lvhb_config_t

  Driver specific Timer configuration.
- struct lvhb_brush_config_t

  Brushed motor configuration.
- struct lvhb_stepper_config_t

  Stepper motor configuration.
- struct lvhb_stepper_data_t

  Stepper motor control data structure.
- struct lvhb_device_data_t

  Device configuration.
- struct lvhb_drv_config_t

  Driver configuration.

Macros

- #define LVHB_BRIDGE_COUNT_MAX 2U

  Maximal number of H-bridge interfaces in supported devices.
- #define LVHB_BRUSH_PWM_FREQ_MAX 20000U

  Maximal PWM frequency for speed control of brushed motors.
- #define LVHB_BRUSH_PWM_FREQ_DEF 10000U

  Speed control PWM frequency of brushed motors set in default configuration.
- #define LVHB_BRUSH_DIR_DEF lvhbDirectionAHighBLow

  Motor direction of brushed motors set in default configuration.
- #define LVHB_BRUSH_REC_DEF lvhbRecirculationLow

  Recirculation side of brushed motors set in default configuration.
- #define LVHB_STEPPER_INIT_PWM_FREQ 1000U

  Initial timer PWM frequency in timer configuration for stepper motors.
- #define LVHB_STEPPER_MAX_STEPS 100000000U

  Maximal number of steps that can be executed.
- #define LVHB_FULLSTEP_IN_FREQ_MIN 131072UL

  Minimal timer input frequency for full-stepping.
- #define LVHB_FULLSTEP_IN_FREQ_MAX 10000000UL

  Maximal timer input frequency for full-stepping.
- #define LVHB_FULLSTEP_SPEED_MIN 1U

  Minimal speed in full-step mode (full-steps per second).
- #define LVHB_FULLSTEP_SPEED_MAX 5000U

  Maximal speed in full-step mode (full-steps per second).
- #define LVHB_FULLSTEP_SPEED_DEF 100U

  Speed in full-step mode set in default configuration.
- #define LVHB_FULLSTEP_ACCEL_MAX 5000U

  Maximal acceleration in full-step mode.
- #define LVHB_FULLSTEP_ACCEL_DEF 0U

  Acceleration in full-step mode set in default configuration.
- #define LVHB_FULLSTEP_CYCLE 4U

  Number of full-steps per one electrical cycle.
- #define LVHB_FULLSTEP_CNT_CONT 1U

  Number of full-steps to next timer restart interrupt in continual mode.
- #define LVHB_FULLSTEP_CH_INACTIVE 0x00U

Output value of channel is Low.

- #define LVHB_FULLSTEP_CH_IN1A 0x01U

  Output value of channel IN1A is High.
- #define LVHB_FULLSTEP_CH_IN1B 0x02U

  Output value of channel IN1B is High.
- #define LVHB_FULLSTEP_CH_IN2A 0x04U

  Output value of channel IN2A is High.
- #define LVHB_FULLSTEP_CH_IN2B 0x08U

  Output value of channel IN2B is High.
- #define LVHB_MICROSTEP_IN_FREQ_MIN 1200000UL

  Minimal timer input frequency for micro-stepping.
- #define LVHB_MICROSTEP_IN_FREQ_MAX 10000000UL

  Maximal timer input frequency for micro-stepping.
- #define LVHB_MICROSTEP_PWM_FREQ_MIN 10000U

  Minimal PWM frequency for micro-stepping.
- #define LVHB_MICROSTEP_PWM_FREQ_MAX 20000U

  Maximal PWM frequency for micro-stepping.
- #define LVHB_MICROSTEP_PWM_FREQ_DEF 20000U

  PWM frequency for micro-stepping set in default configuration..
- #define LVHB_MICROSTEP_SPEED_MIN 1U

  Minimal speed in micro-step mode (micro-steps per second).
- #define LVHB_MICROSTEP_SPEED_MAX 5000U

  Maximal speed in micro-step mode (micro-steps per second).
- #define LVHB_MICROSTEP_SPEED_DEF 100U

  Speed in micro-step mode set in default configuration.
- #define LVHB_MICROSTEP_ACCEL_MAX 5000U

  Maximal acceleration in micro-step mode.
- #define LVHB_MICROSTEP_ACCEL_DEF 0U

  Acceleration in micro-step mode set in default configuration.
- #define LVHB_MICROSTEP_CNT_MAX 32

  Maximal number of micro-steps per full-step.
- #define LVHB_MICROSTEP_PER_FS_DEF lvhbMicroStep2

  Micro-steps per full-step set in default configuration.
- #define LVHB_WIND1_IDX_INIT (LVHB_MICROSTEP_CNT_MAX / 2U)

  Initial value of index for winding 1 that represents physical position of full-step no. 0.
- #define LVHB_WIND2_IDX_INIT (LVHB_MICROSTEP_CNT_MAX + (LVHB_MICROSTEP_CNT_MAX / 2↩ U))

  Initial value of index for winding 2 that represents physical position of full-step no. 0.
- #define LVHB_TIMER_VALUE_MAX(drvConfig) (1U << (drvConfig->tmrLvhbConfig.counterWidth))

  Returns maximal value of counter.
- #define LVHB_FULLSTEP_IN_FREQ(drvConfig) (drvConfig->tmrLvhbConfig.srcClck_Hz >> drvConfig->tmrLvhbConfig.prescale)

  Returns timer input frequency for full-stepping.
- #define LVHB_FULLSTEP_IN_FREQ_0676(drvConfig) ((uint32_t)(LVHB_FULLSTEP_IN_FREQ(drvConfig) * 0.676))

  Returns factored frequency for full-step ramp.
- #define LVHB_GET_FULL_SPEED_TICKS(drvConfig, stepsSec) (LVHB_FULLSTEP_IN_FREQ(drvConfig) / (stepsSec))

  Returns number of timer ticks per timer period.
- #define LVHB_MICROSTEP_IN_FREQ(drvConfig) (drvConfig->tmrLvhbConfig.srcClck_Hz >> drvConfig->tmrLvhbConfig.prescale)

Returns timer input frequency for micro-stepping.

- #define LVHB_MICROSTEP_IN_FREQ_0676(drvConfig) ((uint32_t)(LVHB_MICROSTEP_IN_FREQ(drv↩ Config) ∗ 0.676))

  Returns factored frequency for micro-step ramp.
- #define LVHB_MICROSTEP_PERIOD_TICKS(drvConfig) (LVHB_MICROSTEP_IN_FREQ(drvConfig) / drvConfig->deviceConfig.stepperConfig.microStepPwmFrequency)

  Returns timer period in micro-step mode (ticks per period).
- #define LVHB_GET_MICRO_SPEED(drvConfig, speedOverflows) (((uint32_t)LVHB_MICROSTEP_IN_F↩ REQ(drvConfig) / LVHB_MICROSTEP_PERIOD_TICKS(drvConfig)) / (speedOverflows))

  Returns number of micro-steps per second.
- #define LVHB_GET_MICRO_SPEED_OF(drvConfig, microStepsSec) ((uint32_t)(drvConfig->device↩ Config.stepperConfig.microStepPwmFrequency / (microStepsSec)));

  Returns number of timer overflows according to desired speed.
- #define LVHB_GET_MICRO_OFFSET(drvConfig, activeChannel, fullStepVal, tableIdx) ((((activeChannel) & (fullStepVal)) != LVHB_FULLSTEP_CH_INACTIVE) ? drvConfig->deviceConfig.stepperData.microStep↩ Offset[tableIdx] : 0U)

  Returns offset for specified micro-step.
- #define LVHB_COMPUTE_REMAINDER(dividend, divisor, fraction) (((dividend) < (divisor)) ? (dividend) : (dividend) - ((divisor) ∗ (fraction)))

  Compute remainder after integer division.
- #define LVHB_HAS_TWO_BRIDGES(device) ((device != lvhbDeviceMPC17510) && (device != lvhbDevice↩ MPC17511))

  Checks if the device has two H-Bridges.
- #define LVHB_IS_MPC1751x(device) ((device == lvhbDeviceMPC17510) || (device == lvhbDeviceMP↩ C17511))

  Checks if the device is MPC17510 or MPC17511.
- #define LVHB_CHECK_BRIDGE(drvConfig, bridge)

  Checks if the bridge can be selected in LVHB configuration.
- #define LVHB_HAS_GIN_PIN(device) ((device == lvhbDeviceMPC17510) || (device == lvhbDeviceMP↩ C17511))

  Checks if the device has GIN pin.
- #define LVHB_HAS_EN_PIN(device) (device != lvhbDeviceMC34933)

  Checks if the device has EN/OE/PSAVE pin.
- #define LVHB_HAS_GPIO_IN_PINS(drvConfig) ((drvConfig->inputPins[lvhbBridge1] == lvhbPinsGpio) && (drvConfig->inputPins[lvhbBridge2] == lvhbPinsGpio))

  Checks if all four INxA and INxB pins are set as GPIO pins in the driver configuration.
- #define LVHB_HAS_PWM_IN_PINS(drvConfig) ((drvConfig->inputPins[lvhbBridge1] == lvhbPinsPwm) && (drvConfig->inputPins[lvhbBridge2] == lvhbPinsPwm))

  Checks if all four INxA and INxB pins are set as PWM outputs in the driver configuration.
- #define LVHB_IS_STEPPER_SELECTED(drvConfig) (drvConfig->deviceConfig.motorType == lvhbMotor↩ Stepper)

  Checks if stepper motor is selected in the driver configuration.
- #define LVHB_IS_BRUSHED_MOTOR_SELECTED(drvConfig) (drvConfig->deviceConfig.motorType == lvhbMotorBrushed)

  Checks if brushed motor is selected in the driver configuration.

Enumerations

- enum _lvhb_status { kStatus_LVHB_TimerInit = MAKE_STATUS(kStatusGroup_LVHB, 0), kStatus_LVHB_InvalidDriverConfig = MAKE_STATUS(kStatusGroup_LVHB, 1), kStatus_LVHB_InvalidArgument = MAKE_STATUS(kStatus↩ Group_LVHB, 2), kStatus_LVHB_TimerFrequency = MAKE_STATUS(kStatusGroup_LVHB, 3), kStatus_LVHB_NotInContinualMovement = MAKE_STATUS(kStatusGroup_LVHB, 4), kStatus_LVHB_↩ DeviceBusy = MAKE_STATUS(kStatusGroup_LVHB, 5) }

Error codes.

- enum lvhb_device_t { lvhbDeviceMPC17510 = 0x0U, lvhbDeviceMPC17511 = 0x1U, lvhbDeviceMPC17529 = 0x2U, lvhbDeviceMPC17531A = 0x3U, lvhbDeviceMPC17533 = 0x4U, lvhbDeviceMPC17C724 = 0x5U, lvhbDeviceMC34933 = 0x6U }

  H-bridge device.
- enum lvhb_bridge_t { lvhbBridge1 = 0x0U, lvhbBridge2 = 0x1U }

  Bridge of H-bridge.
- enum lvhb_motor_type_t { lvhbMotorStepper = 0x0U, lvhbMotorBrushed = 0x1U }

  Motor type.
- enum lvhb_output_direction_t { lvhbDirectionALowBHigh = 0x0U, lvhbDirectionAHighBLow = 0x1U }

  Bridge output direction.
- enum lvhb_recirculation_t { lvhbRecirculationLow = 0x0U, lvhbRecirculationHighImp = 0x1U }

  Recirculation side.
- enum lvhb_output_state_t { lvhbOutputOff = 0x0U, lvhbOutputOn = 0x1U }

  Output state.
- enum lvhb_input_pins_t { lvhbPinsGpio = 0x0U, lvhbPinsGpioPwm = 0x1U, lvhbPinsPwmGpio = 0x2U, lvhbPinsPwm = 0x3U }

  Bridge input pin types.
- enum lvhb_motor_status_t { lvhbStatusRunning = 0x0U, lvhbStatusStop = 0x1U }

  Motor status.
- enum lvhb_stepper_mode_t { lvhbStepperModeMicrostep = 0U, lvhbStepperModeFullstep = 1U, lvhbStepperModeDisabled = 2U }

  Stepper motor mode.
- enum lvhb_stepper_state_t { lvhbStepperStateSleepLoop = 0U, lvhbStepperStateInitCompleted = 1U, lvhbStepperStateSlowStart = 2U }

  Stepper motor control states.
- enum lvhb_micro_step_t { lvhbMicroStep2 = LVHB_MICROSTEP_CNT_MAX / 2U, lvhbMicroStep4 = LVHB_MICROSTEP_CNT_MAX / 4U, lvhbMicroStep8 = LVHB_MICROSTEP_CNT_MAX / 8U, lvhbMicroStep16 = LVHB_MICROSTEP_CNT_MAX / 16U, lvhbMicroStep32 = LVHB_MICROSTEP_CNT_MAX / 32U }

  Number of micro-steps per one full-step.
- enum lvhb_ramp_state_t { lvhbRampOff = 0U, lvhbRampUpSpeed = 1U, lvhbRampDownSpeed = 2U, lvhbRampConstSpeed = 3U }

  States of finite state machine which are used for acceleration ramp.

Functions

- status_t LVHB_ConfigureGpio (lvhb_drv_config_t *const drvConfig)

  This function configures GPIO for usage with this driver.
- status_t LVHB_ConfigureTimer (lvhb_drv_config_t *const drvConfig, tmr_sdk_config_t *const tmrSdkConfig)

  This function configures Timer for usage with this driver.
- status_t LVHB_GetDefaultConfig (lvhb_drv_config_t *const drvConfig, lvhb_device_t device, lvhb_motor_type_t motorType)

  This function gets a default configuration of the driver for specific device and motor type.
- status_t LVHB_Init (lvhb_drv_config_t *const initConfig)

  This function initializes driver data and output pin values.
- status_t LVHB_Deinit (lvhb_drv_config_t *const initConfig)

  This function deinitializes the driver.
- status_t LVHB_SetMode (lvhb_drv_config_t *const drvConfig, bool active)

This method sets H-Bridge device mode using enable pin.

- status_t LVHB_SetGateDriver (lvhb_drv_config_t *const drvConfig, bool outputHigh)

  This function controls Gate Driver Input (GIN) pin. It is available for MPC17510 and MPC17511 only.
- status_t LVHB_RotateFull (lvhb_drv_config_t *const drvConfig, lvhb_output_state_t state, lvhb_bridge_t bridge)

  This function spins the motor in desired direction at full speed.
- status_t LVHB_RotateProportional (lvhb_drv_config_t *const drvConfig, uint8_t pwmDuty, lvhb_bridge_t bridge)

  This function spins the motor in desired direction at PWM duty speed.
- status_t LVHB_SetTriState (lvhb_drv_config_t *const drvConfig, lvhb_bridge_t bridge)

  This function sets output of specified H-Bridge to tri-state (high impedance) using input control pins.
- status_t LVHB_SetDirection (lvhb_drv_config_t *const drvConfig, lvhb_output_direction_t direction, lvhb_bridge_t bridge)

  This function sets direction of brush motor at specified H-Bridge interface.
- status_t LVHB_SetRecirculation (lvhb_drv_config_t *const drvConfig, lvhb_recirculation_t side, lvhb_bridge_t bridge)

  This function sets low/high-impedance-side recirculation of the H-Bridge.
- status_t LVHB_SetFullStepSpeed (lvhb_drv_config_t *const drvConfig, uint16_t stepsSec)

  This function sets the speed of full-step mode.
- status_t LVHB_SetMicroStepSpeed (lvhb_drv_config_t *const drvConfig, uint16_t microStepsSec)

  This function sets the speed of micro-step mode.
- status_t LVHB_SetFullStepAcceleration (lvhb_drv_config_t *const drvConfig, uint32_t acceleration)

  This function sets the acceleration ramp of full-step mode.
- status_t LVHB_SetMicroStepAcceleration (lvhb_drv_config_t *const drvConfig, uint32_t acceleration)

  This function sets the acceleration ramp of micro-step mode.
- status_t LVHB_MoveSteps (lvhb_drv_config_t *const drvConfig, bool forward, uint32_t steps)

  This function moves motor by specified number of full-steps.
- status_t LVHB_MoveMicroSteps (lvhb_drv_config_t *const drvConfig, bool forward, uint32_t microSteps)

  This function moves motor by specified number of micro-steps.
- status_t LVHB_MoveContinual (lvhb_drv_config_t *const drvConfig, bool forward)

  This function moves motor continually in full-step mode.
- status_t LVHB_MoveMicroContinual (lvhb_drv_config_t *const drvConfig, bool forward)

  This function moves motor continually in micro-step mode.
- status_t LVHB_StopContinualMovement (lvhb_drv_config_t *const drvConfig)

  This function is intended to stop continual movement of stepper motor.
- lvhb_motor_status_t LVHB_GetMotorStatus (lvhb_drv_config_t *const drvConfig)

  This function returns status of stepper motor control.
- status_t LVHB_AlignRotor (lvhb_drv_config_t *const drvConfig)

  This function aligns rotor to the full-step position.
- status_t LVHB_SetMicroStepSize (lvhb_drv_config_t *const drvConfig, lvhb_micro_step_t microStepsPerStep)

  This function changes the size of micro-step.
- int32_t LVHB_GetFullStepPosition (lvhb_drv_config_t *const drvConfig)

  This function returns the current full-step position.
- int32_t LVHB_GetMicroStepPosition (lvhb_drv_config_t *const drvConfig)

  This function returns the current micro-step position.
- status_t LVHB_ResetFullStepPosition (lvhb_drv_config_t *const drvConfig)

  This function sets the counter of full-steps to zero.
- status_t LVHB_DisableMotor (lvhb_drv_config_t *const drvConfig)

  This function disables the stepper motor.
- void LVHB_OnCounterRestart (lvhb_drv_config_t *const drvConfig)

Counter restart event handler. This function must be called from counter restart interrupt handler.

- void LVHB_OnActionComplete (lvhb_drv_config_t *const drvConfig)

  Declaration of function that can be used by user for handling the action complete event. This event occurs when "LVHB_MoveSteps", "LVHB_MoveMicroSteps", "LVHB_StopContinualMovement" or "LVHB_AlignRotor" action is done.

6.2.1 Detailed Description

Low voltage H-Bridge driver based on AML layer. Supports boards based on MPC17510, MPC17511, MPC17529, MPC17531A, MPC17533, MPC17C724 and MC34933.

This module is common for all supported models.

6.2.2 Macro Definition Documentation

6.2.2.1 LVHB_CHECK_BRIDGE

```
#define LVHB_CHECK_BRIDGE(
               drvConfig,
               bridge )
```

Value:

```
((bridge <= lvhbBridge2) &&                                \
     ((bridge == lvhbBridge1) ||                             \
      (LVHB_HAS_TWO_BRIDGES(drvConfig->deviceConfig.device) && (
      LVHB_IS_STEPPER_SELECTED(drvConfig) || drvConfig->deviceConfig.secondaryBridgeUsed)
      )))
```

Checks if the bridge can be selected in LVHB configuration.

Bridge can be selected from lvhb_bridge_t enumeration only. LvhbBridge1 can be selected everytime. LvhbBridge2 can be selected if device with two H-bridges is chosen and stepper motor or secondaryBridgeUsed variable is set in driver configuration.

Parameters

drvConfig	Pointer to the LVHB driver instance configuration.
bridge	Bridge from lvhb_bridge_t enumeration.

Returns

True if bridge can be selected.

6.2.2.2 LVHB_COMPUTE_REMAINDER

```
#define LVHB_COMPUTE_REMAINDER(
              dividend,
              divisor,
              fraction ) (((dividend) < (divisor)) ?  (dividend) :  (dividend) - ((divisor) *
(fraction)))
```

Compute remainder after integer division.

Remainder = dividend mod divisor = dividend - (divisor * (dividend / divisor)).

Parameters

dividend	Expression dividend.
divisor	Expression divisor.
fraction	Equal to dividend div divisor.

Returns

Remainder after integer division.

6.2.2.3 LVHB_FULLSTEP_IN_FREQ

```
#define LVHB_FULLSTEP_IN_FREQ(
              drvConfig ) (drvConfig->tmrLvhbConfig.srcClck_Hz >> drvConfig->tmrLvhbConfig.↩
prescale)
```

Returns timer input frequency for full-stepping.

Parameters

drvConfig	Pointer to the LVHB driver instance configuration.

Returns

Timer input frequency for full-stepping.

6.2.2.4 LVHB_FULLSTEP_IN_FREQ_0676

```
#define LVHB_FULLSTEP_IN_FREQ_0676(
              drvConfig ) ((uint32_t)(LVHB_FULLSTEP_IN_FREQ(drvConfig) * 0.676))
```

Returns factored frequency for full-step ramp.

Parameters

drvConfig	Pointer to the LVHB driver instance configuration.

Returns

Factored frequency for full-step ramp.

6.2.2.5 LVHB_GET_FULL_SPEED_TICKS

```
#define LVHB_GET_FULL_SPEED_TICKS(
              drvConfig,
              stepsSec ) (LVHB_FULLSTEP_IN_FREQ(drvConfig) / (stepsSec))
```

Returns number of timer ticks per timer period.

Parameters

drvConfig	Pointer to the LVHB driver instance configuration.
stepsSec	Number of steps per second.

Returns

Number of timer ticks.

6.2.2.6 LVHB_GET_MICRO_OFFSET

```
#define LVHB_GET_MICRO_OFFSET(
              drvConfig,
              activeChannel,
              fullStepVal,
              tableIdx ) ((((activeChannel) & (fullStepVal)) != LVHB_FULLSTEP_CH_INACTIVE) ?
drvConfig->deviceConfig.stepperData.microStepOffset[tableIdx] :  0U)
```

Returns offset for specified micro-step.

Parameters

drvConfig	Pointer to the LVHB driver instance configuration.
activeChannel	Channel output value (one of LVHB_FULLSTEP_CH_INxx constants).
fullStepVal	Active channels in current full-step. Value is logic OR of channels output values (LVHB_FULLSTEP_CH_INX constants).
tableIdx	Index to the table with micro-step offsets.

Returns

Channel offset.

6.2.2.7 LVHB_GET_MICRO_SPEED

```
#define LVHB_GET_MICRO_SPEED(
              drvConfig,
              speedOverflows ) (((uint32_t)LVHB_MICROSTEP_IN_FREQ(drvConfig) / LVHB_MICROSTEP↩
_PERIOD_TICKS(drvConfig)) / (speedOverflows))
```

Returns number of micro-steps per second.

Parameters

drvConfig	Pointer to the LVHB driver instance configuration.
speedOverflows	Micro-step speed in number of timer overflows.

Returns

Number of micro-steps per second.

6.2.2.8 LVHB_GET_MICRO_SPEED_OF

```
#define LVHB_GET_MICRO_SPEED_OF(
              drvConfig,
              microStepsSec ) ((uint32_t)(drvConfig->deviceConfig.stepperConfig.microStepPwm↩
Frequency / (microStepsSec)));
```

Returns number of timer overflows according to desired speed.

Parameters

drvConfig	Pointer to the LVHB driver instance configuration.
microStepsSec	Micro-step speed in number of steps per second.

Returns

Number of overflows.

6.2.2.9 LVHB_HAS_EN_PIN

```
#define LVHB_HAS_EN_PIN(
              device ) (device != lvhbDeviceMC34933)
```

Checks if the device has EN/OE/PSAVE pin.

Parameters

device	Device from lvhb_device_t enumeration.

Returns

True if device has EN/OE/PSAVE pin.

6.2.2.10 LVHB_HAS_GIN_PIN

```
#define LVHB_HAS_GIN_PIN(
              device ) ((device == lvhbDeviceMPC17510) || (device == lvhbDeviceMPC17511))
```

Checks if the device has GIN pin.

Parameters

device	Device from lvhb_device_t enumeration.

Returns

True if device has GIN pin.

6.2.2.11 LVHB_HAS_GPIO_IN_PINS

```
#define LVHB_HAS_GPIO_IN_PINS(
              drvConfig ) ((drvConfig->inputPins[lvhbBridge1] == lvhbPinsGpio) && (drvConfig->input↩
Pins[lvhbBridge2] == lvhbPinsGpio))
```

Checks if all four INxA and INxB pins are set as GPIO pins in the driver configuration.

Parameters

drvConfig	Pointer to the LVHB driver instance configuration.

Returns

True if all four INxA and INxB pins are set as GPIO pins.

6.2.2.12 LVHB_HAS_PWM_IN_PINS

```
#define LVHB_HAS_PWM_IN_PINS(
              drvConfig ) ((drvConfig->inputPins[lvhbBridge1] == lvhbPinsPwm) && (drvConfig->input↩
Pins[lvhbBridge2] == lvhbPinsPwm))
```

Checks if all four INxA and INxB pins are set as PWM outputs in the driver configuration.

Parameters

drvConfig	Pointer to the LVHB driver instance configuration.

Returns

True if all four INxA and INxB pins are set as PWM outputs.

6.2.2.13 LVHB_HAS_TWO_BRIDGES

```
#define LVHB_HAS_TWO_BRIDGES(
              device ) ((device != lvhbDeviceMPC17510) && (device != lvhbDeviceMPC17511))
```

Checks if the device has two H-Bridges.

Parameters

device	Device from lvhb_device_t enumeration.

Returns

True if supported device has two H-Bridges.

6.2.2.14 LVHB_IS_BRUSHED_MOTOR_SELECTED

```
#define LVHB_IS_BRUSHED_MOTOR_SELECTED(
              drvConfig ) (drvConfig->deviceConfig.motorType == lvhbMotorBrushed)
```

Checks if brushed motor is selected in the driver configuration.

Parameters

drvConfig	Pointer to the LVHB driver instance configuration.

Returns

True if brushed motor is selected.

6.2.2.15 LVHB_IS_MPC1751x

```
#define LVHB_IS_MPC1751x(
              device ) ((device == lvhbDeviceMPC17510) || (device == lvhbDeviceMPC17511))
```

Checks if the device is MPC17510 or MPC17511.

Parameters

device	Device from lvhb_device_t enumeration.

Returns

True if device is MPC17510 or MPC17511.

6.2.2.16 LVHB_IS_STEPPER_SELECTED

```
#define LVHB_IS_STEPPER_SELECTED(
              drvConfig ) (drvConfig->deviceConfig.motorType == lvhbMotorStepper)
```

Checks if stepper motor is selected in the driver configuration.

Parameters

drvConfig	Pointer to the LVHB driver instance configuration.

Returns

True if stepper motor is selected.

6.2.2.17 LVHB_MICROSTEP_IN_FREQ

```
#define LVHB_MICROSTEP_IN_FREQ(
              drvConfig ) (drvConfig->tmrLvhbConfig.srcClck_Hz >> drvConfig->tmrLvhbConfig.↩
prescale)
```

Returns timer input frequency for micro-stepping.

Parameters

drvConfig	Pointer to the LVHB driver instance configuration.

Returns

Timer input frequency for micro-stepping.

6.2.2.18 LVHB_MICROSTEP_IN_FREQ_0676

```
#define LVHB_MICROSTEP_IN_FREQ_0676(
              drvConfig ) ((uint32_t)(LVHB_MICROSTEP_IN_FREQ(drvConfig) * 0.676))
```

Returns factored frequency for micro-step ramp.

Parameters

drvConfig	Pointer to the LVHB driver instance configuration.

Returns

Factored frequency for micro-step ramp.

6.2.2.19 LVHB_MICROSTEP_PERIOD_TICKS

```
#define LVHB_MICROSTEP_PERIOD_TICKS(
              drvConfig ) (LVHB_MICROSTEP_IN_FREQ(drvConfig) / drvConfig->deviceConfig.↩
stepperConfig.microStepPwmFrequency)
```

Returns timer period in micro-step mode (ticks per period).

Parameters

drvConfig	Pointer to the LVHB driver instance configuration.

Returns

Timer period in micro-step mode (ticks per period).

6.2.2.20 LVHB_TIMER_VALUE_MAX

```
#define LVHB_TIMER_VALUE_MAX(
              drvConfig ) (1U << (drvConfig->tmrLvhbConfig.counterWidth))
```

Returns maximal value of counter.

Parameters

drvConfig	Pointer to the LVHB driver instance configuration.

Returns

Maximal value of counter + 1.

Index