Document Number: KTTWRMCMVHB1EVBUG
Rev. 2.0, 9/2015

TWR–MC–MVHB1EVB Tower System Platform

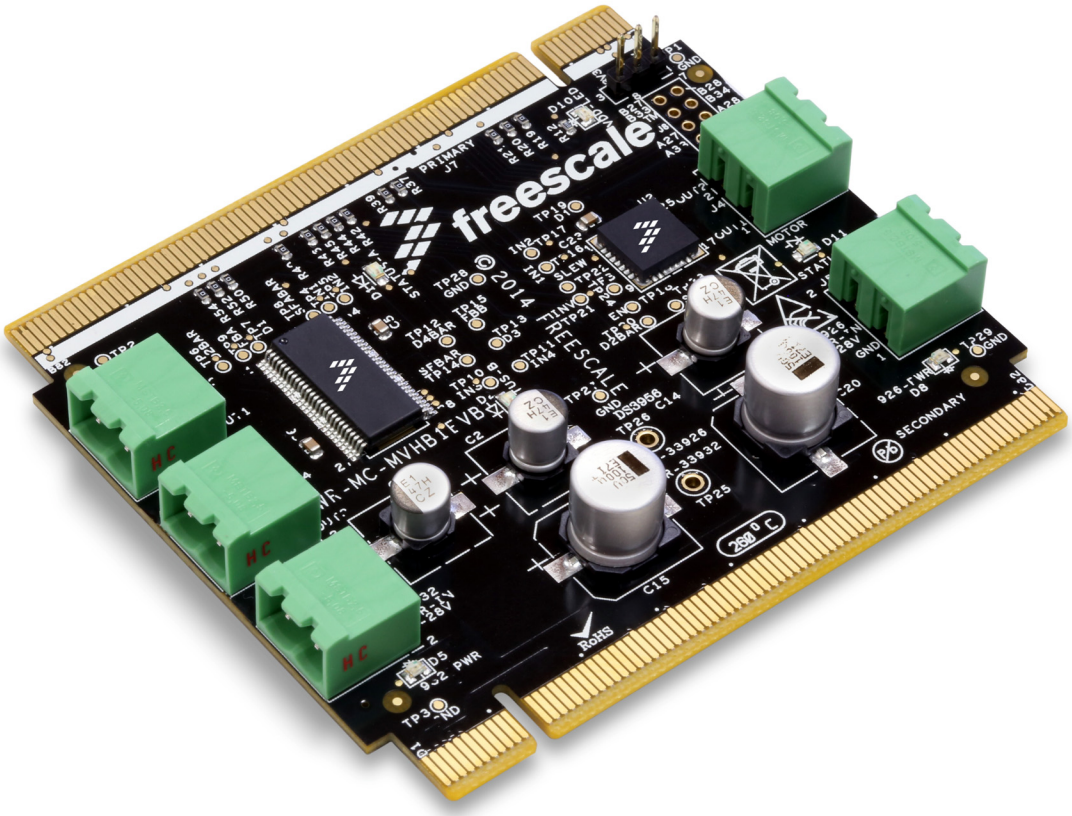


Figure 1. TWR–MC–MVHB1EVB

Contents

1 Important Notice

Freescale provides the enclosed product(s) under the following conditions:

This evaluation kit is intended for use of ENGINEERING DEVELOPMENT OR EVALUATION PURPOSES ONLY. It is provided as a sample IC pre-soldered to a printed circuit board to make it easier to access inputs, outputs, and supply terminals. This evaluation board may be used with any development system or other source of I/O signals by simply connecting it to the host MCU or computer board via off-the-shelf cables. This evaluation board is not a Reference Design and is not intended to represent a final design recommendation for any particular application. Final device in an application will be heavily dependent on proper printed circuit board layout and heat sinking design as well as attention to supply filtering, transient suppression, and I/O signal quality.

The goods provided may not be complete in terms of required design, marketing, and or manufacturing related protective considerations, including product safety measures typically found in the end product incorporating the goods. Due to the open construction of the product, it is the user's responsibility to take any and all appropriate precautions with regard to electrostatic discharge. In order to minimize risks associated with the customers applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards. For any safety concerns, contact Freescale sales and technical support services.

Should this evaluation kit not meet the specifications indicated in the kit, it may be returned within 30 days from the date of delivery and will be replaced by a new kit.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typical”, must be validated for each customer application by customer’s technical experts.

Freescale does not convey any license under its patent rights nor the rights of others. Freescale products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale product could create a situation where personal injury or death may occur.

Should the Buyer purchase or use Freescale products for any such unintended or unauthorized application, the Buyer shall indemnify and hold Freescale and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges Freescale was negligent regarding the design or manufacture of the part. Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

2 Getting Started

2.1 Kit Contents/Packing List

The **TWR-MC-MVHB1EVB** contents include:

- Assembled and tested evaluation board/module in anti-static bag
- Quick Start Guide, Analog Tools
- Warranty card

2.2 Jump Start

Freescale's analog product development boards help to easily evaluate Freescale products. These tools support analog mixed signal and power solutions including monolithic ICs using proven high-volume SMARTMOS mixed signal technology, and system-in-package devices utilizing power, SMARTMOS and MCU dies. Freescale products enable longer battery life, smaller form factor, component count reduction, ease of design, lower system cost and improved performance in powering state of the art systems.

- Click here: www.freescale.com/TWR-MC-MVHB1EVB
- Review your Tool Summary Page
- Look for

- Download documents, software and other information

Once the files are downloaded, review the user guide in the bundle. The user guide includes setup instructions, BOM and schematics. Jump start bundles are available on each tool summary page with the most relevant and current information. The information includes everything needed for design.

2.3 Required Equipment and Software

To use this kit, you need:

- Power supply 8.0 V—36 V with current limit set initially to max 5.0 A.
- Typical loads: DC brushed motor, stepper motor,
- Wire cables for power supply and load connection.
- Other Tower modules (MCU, ELEV etc): www.freescale.com/tower
- (Optional) MVHBRIDGE-PEXPERT: (See www.freescale.com/MVHBRIDGE-PEXPERT)

2.4 System Requirements

The kit requires the following to function properly with the software:

- Windows® XP, Windows 7, or Vista in 32- and 64-bit versions

3 Understanding the Freescale Tower System Platform

Freescale's Tower System peripheral module is designed to be used in conjunction with other Tower System modules.

The Freescale Tower System platform is a modular development environment for 8-, 16- and 32-bit MCUs and MPUs that enables advanced development through rapid prototyping. Featuring more than fifty development boards or modules, the Tower System platform provides designers with building blocks for entry-level to advanced MCU development.

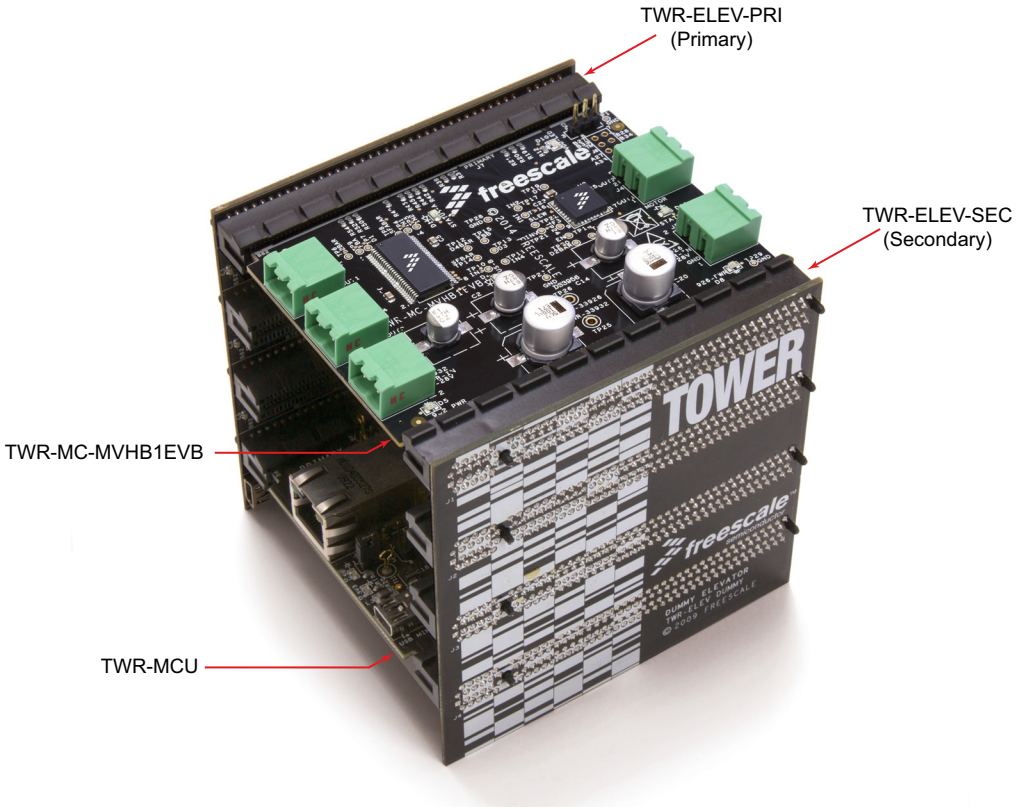


Figure 2. Tower System Overview

Table 1. Description

Name	Description
TWR-MC-MVHB1EVB	TWR-MC-MVHB1EVB Evaluation Board
Tower MCU Board	Additional Freescale Tower boards (Optional)
TWR-ELEV-PRI	Tower System Elevator Primary Module
TWR-ELEV-SEC	Tower System Elevator Secondary Module

3.1 Block Diagram

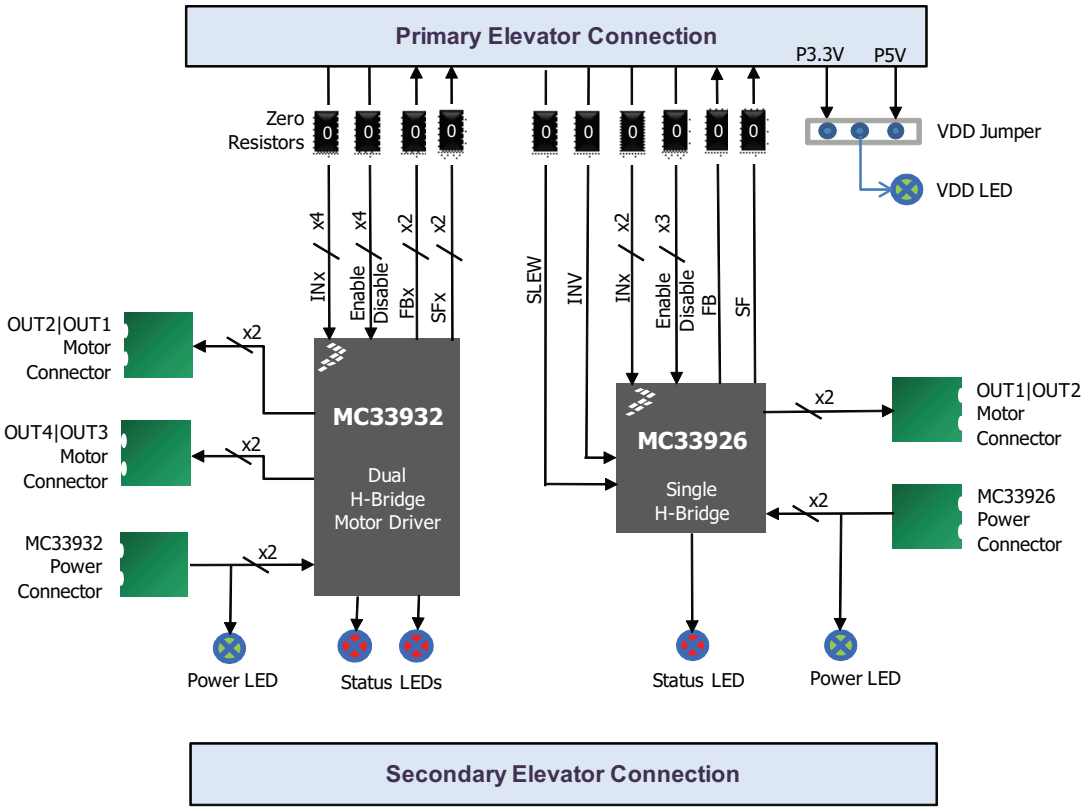


Figure 3. TWR-MC-MVHB1EVB Block Diagram

Table 2. Device Features

Device	Description	Features
MC33932[1]	MC33932 is a monolithic H-bridge Power IC in a robust thermally enhanced package with two independent H-bridges.	• 5.0 V to 28 V continuous operation (transient operation from 5.0 V to 40 V) • 235 mΩ maximum $R_{DS(on)}$ at 150 °C (each H-bridge MOSFET) • 3.0 V and 5.0 V TTL / CMOS logic compatible inputs • Output short-circuit protection (short to VPWR or GND) • Overcurrent limiting (regulation) via internal constant-off-time PWM • Temperature dependant current limit threshold reduction to allow for continuous operation without shutdown • Sleep mode with current draw < 50 µA (each half with inputs floating or set to match default logic states)
MC33926	MC33926 is a monolithic H-bridge Power IC in a robust thermally enhanced package with single H-bridge	• 5.0 V to 28 V continuous operation (transient operation from 5.0 V to 40 V) • 235 mΩ maximum $R_{DS(on)}$ at 150 °C (each H-bridge MOSFET) • 3.0 V and 5.0 V TTL / CMOS logic compatible inputs • Output short-circuit protection (short to VPWR or GND) • Overcurrent limiting (regulation) via internal constant-off-time PWM • Temperature dependant current limit threshold reduction to allow for continuous operation without shutdown • Sleep mode with current draw < 50 µA (each half with inputs floating or set to match default logic states) • Fast or slow slew rate adjustment

Note
1. For MC33931, MC34931, and MC34931S software development, use one half of the MC33932. For MC34932, use MC33932.

4 Getting to Know the Hardware

4.1 Board Overview

The TWR-MC-MVHB1EVB is a Tower System peripheral module circuit board allowing the user to exercise all the functions of two H-bridge motor driver ICs (MC33932 and MC33926.)

4.2 Board Features

The board features are as follows:

- Compatibility with Freescale's Tower Platform. (A zero Ω resistor is placed between the tower connector and each control pin of the device for maximum flexibility.)
- LEDs to indicate power supply and error status
- Transient voltage suppressor to handle system level transients
- Test points to allow probing of signals

4.3 Board Description

The TWR–MC–MVHB1EVB consists of the following components:

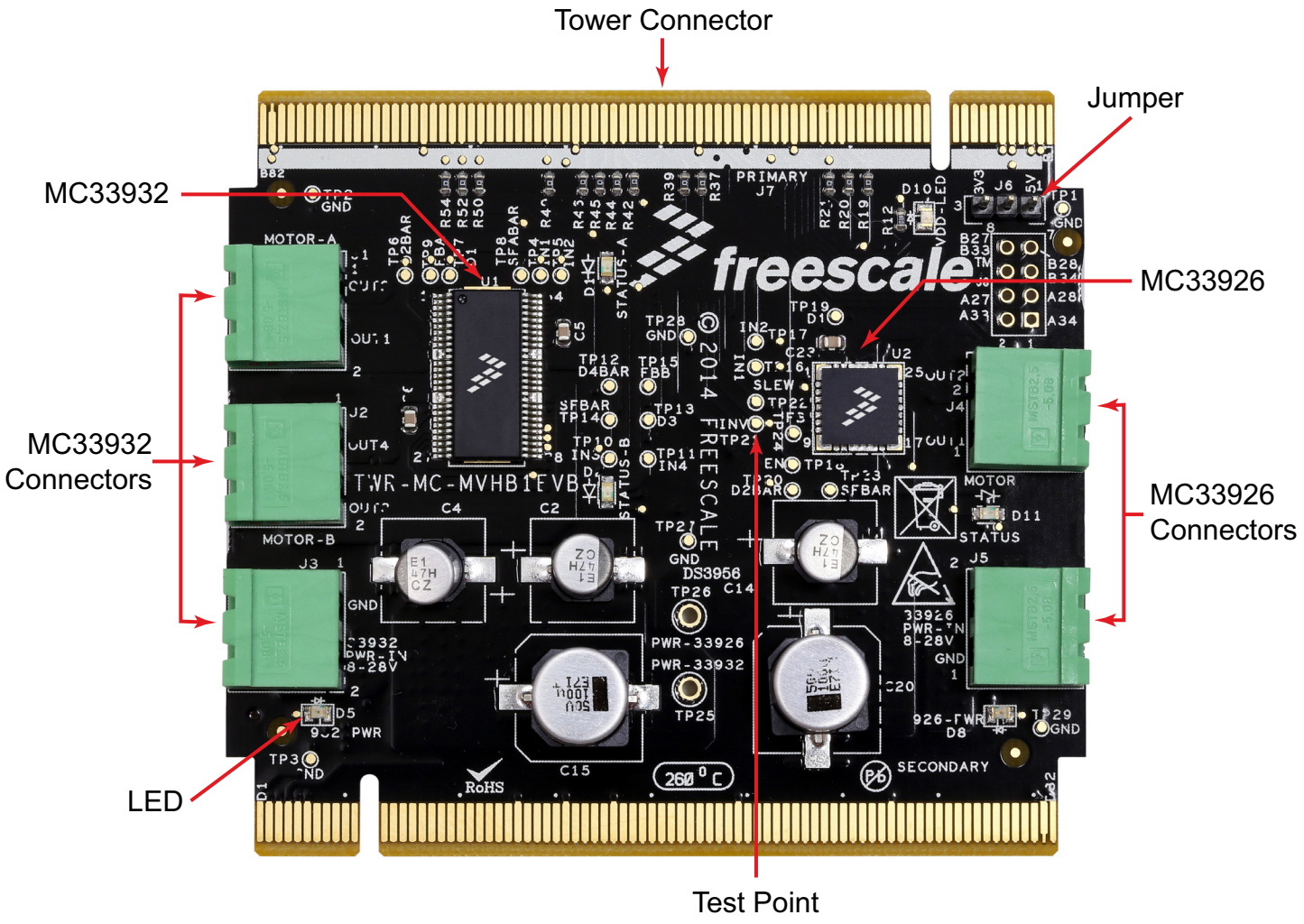


Figure 4. Board Description

Table 3. Board Description

Name	Description
Jumper	VDD power selection. Select 3.3 V / 5.0 V power from tower elevator (based on what MCU tower board is connected)
LEDs	Indicate power supply and fault detection status
Test Points	Allow signal probing
Tower Connectors	Primary and secondary connectors that plug into corresponding Tower Elevator modules
MC33932	H-bridge Power IC with two independent H-bridges
MC33926	H-bridge Power IC with single H-bridge
MC33932 Connectors	MC33932 outputs and power/ground inputs
MC33926 Connectors	MC33926 outputs and power/ground inputs

4.4 LED Display

The following LEDs are provided as visual output devices for the TWR-MC-MVHB1EVB evaluation board:

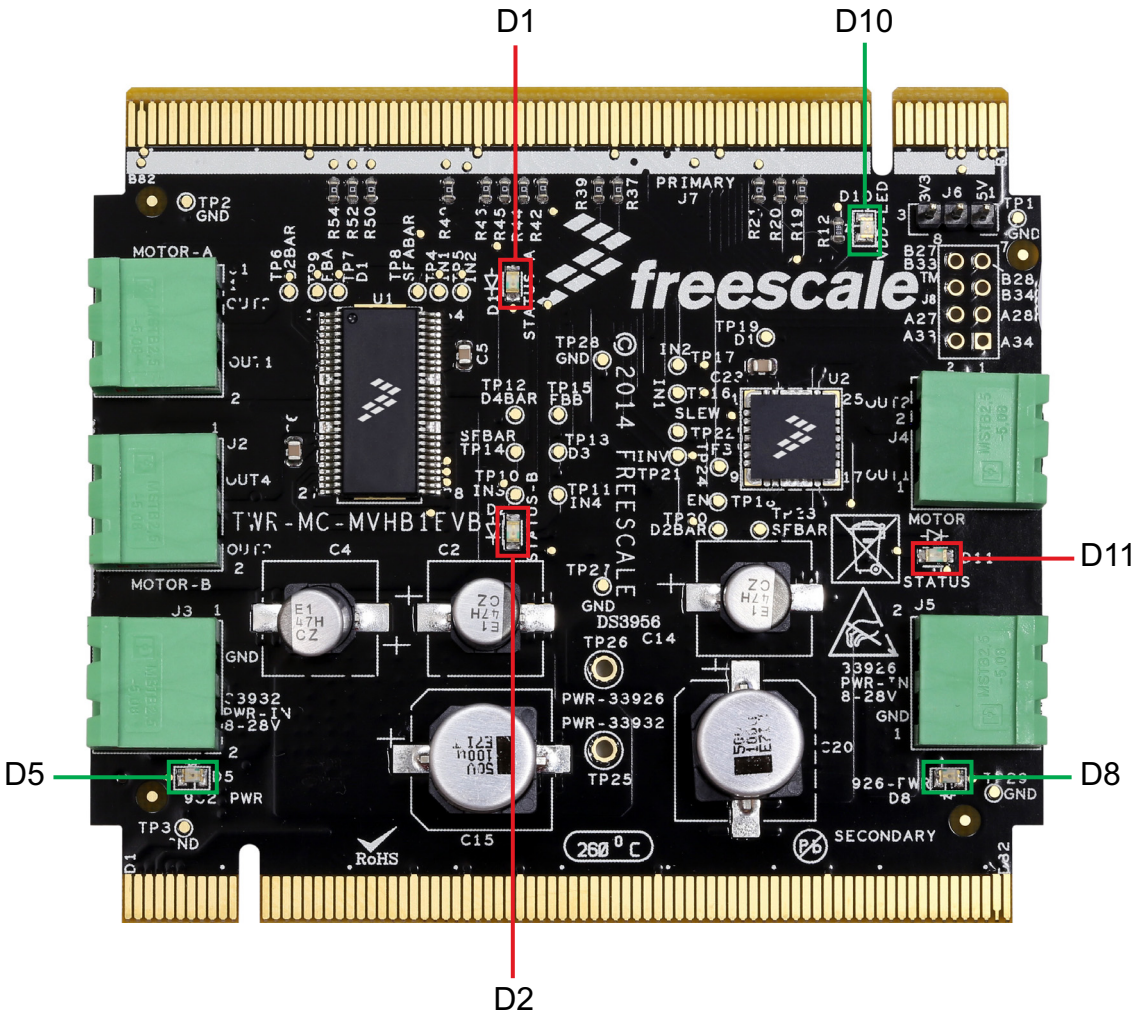


Figure 5. LED

Table 4. LEDs

Schematic Label	Name	Description
D1	Green LED	Indicates when power supply is connected to MC33932
D2	Red LED	Illuminates when a fault is detected in H-bridge Channel B of MC33932
D5	Green LED	Indicates when power supply is connected to MC33932
D8	Green LED	Indicates when power supply is connected to MC33926
D10	Green LED	Indicates when power supply is connected to VDD (Digital power from primary connector)
D11	Red LED	Illuminates when a fault is detected for H-bridge of MC33926

4.5 Connectors

Input/output connectors provide the following signals:

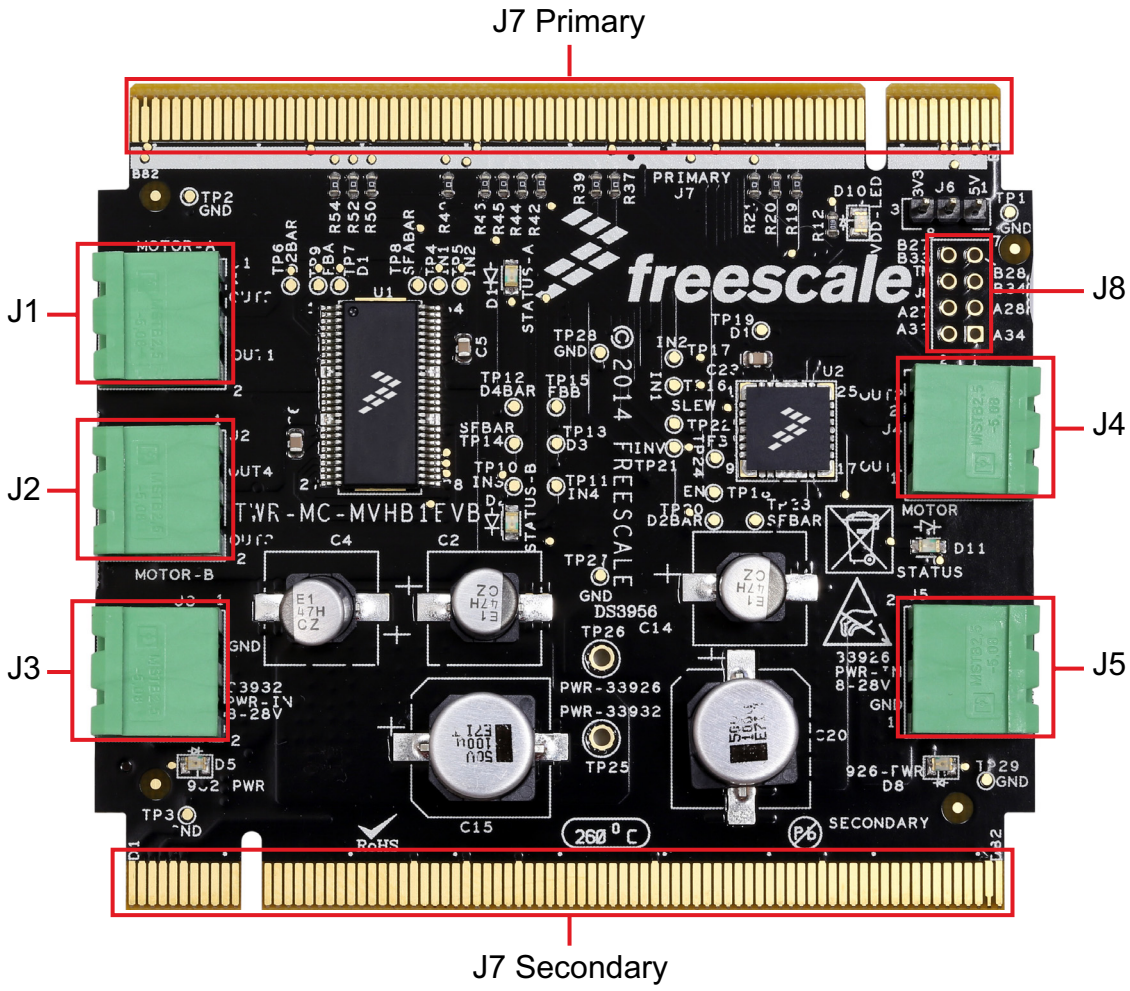


Figure 6. Connectors

Table 5. Connectors

Schematic Label	Name	Description
J1	Motor - A	Motor connector for H-bridge Channel A of MC33932
J2	Motor - B	Motor connector for H-bridge Channel B of MC33932
J3	33932 PWR - IN 8 - 28V	Power supply for MC33932 MCU
J4	Motor	Motor connector for H-bridge Channel A of MC33926
J5	33926 PWR - IN 8 - 28V	Power supply for MC33926 MCU
J7 Primary	Primary Tower Platform Connector	Plugs into primary Tower Elevator connector TWR-ELEV-PRI
J7 Secondary	Secondary Tower Platform Connector	Plugs into secondary Tower Elevator connector TWR-ELEV-SEC
J8	Reserved Connector	Reserved for MCU ADC/PWM interface

4.6 Test Point Definitions

The following test-point jumpers provide access to signals on the TWR-MC-MVHB1EVB IC:

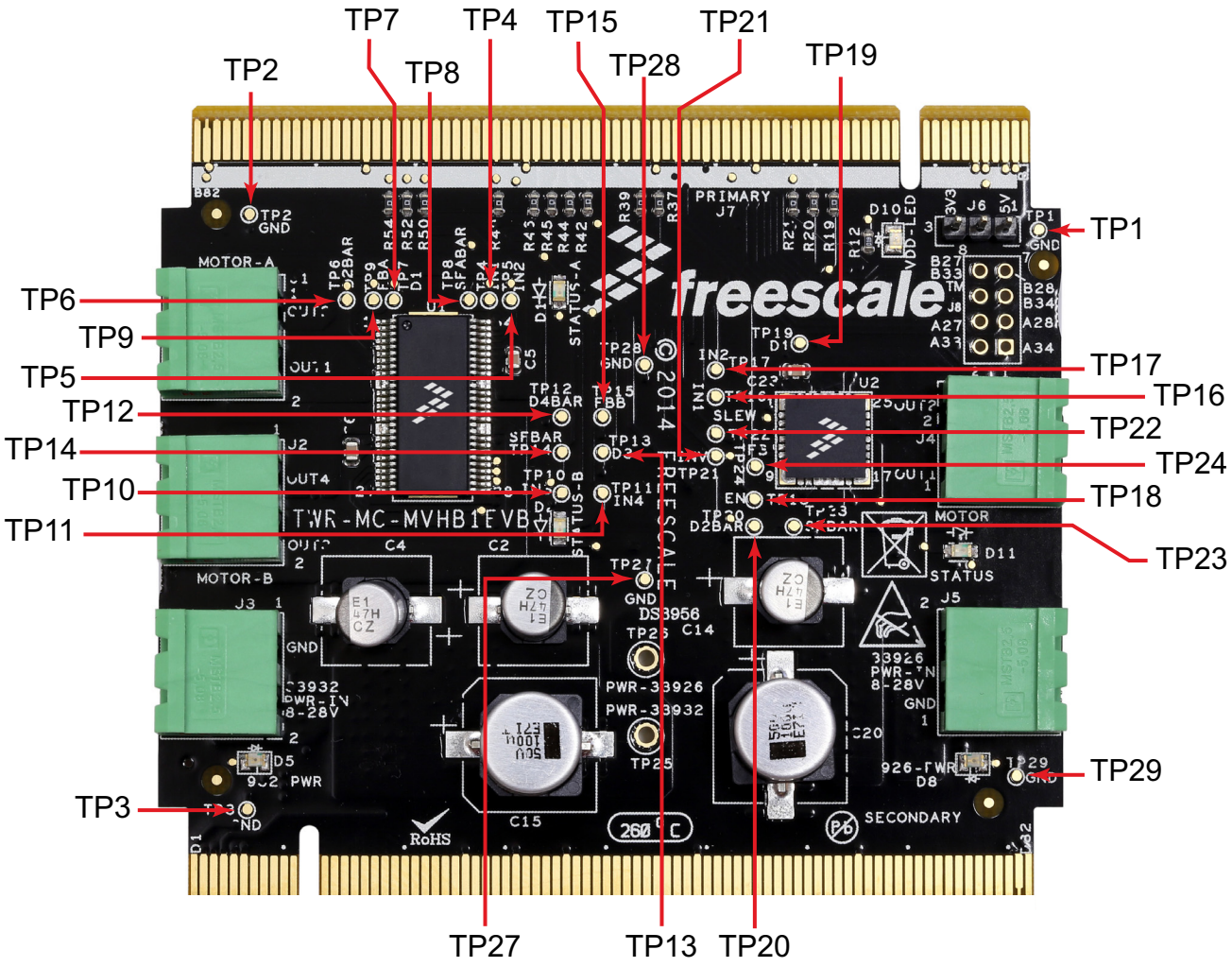


Figure 7. Test Points

Table 6. Test Points

Schematic Label	Signal Name	Description
TP2	GND	Ground
TP3	GND	Ground
TP4	IN1	Logic input control for OUT1 - MC33932
TP5	IN2	Logic input control for OUT2 - MC33932
TP6	EN/D2_bar	Enable input 1 - MC33932
TP7	D1	Disable input 1 - MC33932
TP8	SFA_bar	Status flag A - MC33932
TP9	FBA	Feedback A - MC33932
TP10	IN3	Logic input control for OUT3 - MC33932
TP11	IN4	Logic input control for OUT4 - MC33932

Table 6. Test Points (continued)

Schematic Label	Signal Name	Description
TP12	EN/D4_bar	Enable input 2 - MC33932
TP13	D3	Disable input 2 - MC33932
TP14	SF_bar	Status flag B - MC33932
TP15	FBB	Feedback B - MC33932
TP16	IN1	Logic input control for OUT1 - MC33926
TP17	IN2	Logic input control for OUT2 - MC33926
TP18	EN	Enable input - MC33926
TP19	D1	Disable input 1 - MC33926
TP20	D2_bar	Disable input 2 - MC33926
TP21	INV	Inverter input - MC33926
TP22	SLEW	Slew rate control - MC33926
TP23	SF_bar	Status flag - MC33926
TP24	FB	Feedback - MC33926
TP27	GND	Ground
TP28	GND	Ground
TP29	GND	Ground

4.7 Jumper Definitions

The following table defines the evaluation board jumper position and explains its function:

Figure 8. Jumpers

.

Table 7. Jumpers

Jumper	Description	Setting	Connection
J6	VDD power supply selection	**1-2**	VDD connected to 3.3 V (from Tower Elevator)
		2-3	VDD connected to 5.0 V (from Tower Elevator)

4.8 Tower Elevator Connections

TWR-MC-MVHB1EVB features two expansion card edge connectors that interface to elevator boards in a Tower System: the Primary and Secondary Elevator Connectors. Table 8 provides the pinouts for the Primary Elevator Connector (TWR-ELEV-PRI). There are no electrical connections to the Secondary Elevator Connector (TWR-ELEV-SEC.)

Table 8. Primary Elevator Connector Pinouts

Side B					Side A				
Pin #	**Name**	**Group**	**Usage**	**Jumper**	**Pin #**	**Name**	**Group**	**Usage**	**Jumper**
B1	**5V**	**Power**	**5.0V Power**		A1	**5V**	**Power**	**5.0V Power**	
B2	**GND**	**Power**	**Ground**		A2	**GND**	**Power**	**Ground**	
B3	**3.3V**	**Power**	**3.3V Power**		A3	**3.3V**	**Power**	**3.3V Power**	
B4	ELE_PS_SENSE	Power	Elevator Power Sense		A4	**3.3V**	**Power**	**3.3V Power**	
B5	**GND**	**Power**	**Ground**		A5	**GND**	**Power**	**Ground**	
B6	**GND**	**Power**	**Ground**		A6	**GND**	**Power**	**Ground**	
B7	SDHC_CLK / SPI1_CLK	SDHC / SPI 1			A7	SCL0	I2C 0		
B8	SDHC_CS1_D3 / SPI1_CS1	SDHC / SPI 1			A8	SDA0	I2C 0		
B9	SDHC_CS0_D3 / SPI1_CS0	SDHC / SPI 1			A9	GPIO9 / CTS1	GPIO / UART	MC33932_D1	(2)
B10	SDHC_CMD / SPI1_MOSI	SDHC / SPI 1			A10	GPIO8 / SDHC_D2	GPIO / SDHC	MC33932_EN/ D2bar	(2)
B11	SDHC_D0 / SPI1_MISO	SDHC / SPI 1			A11	GPIO7 / SD_WP_DET	GPIO / SDHC	MC33932_D3	(2)
Mechanical Key									
B12	ETH_COL	Ethernet			A12	ETH_CRS	Ethernet		
B13	ETH_RXER	Ethernet			A13	ETH_MDC	Ethernet		
B14	ETH_TXCLK	Ethernet			A14	ETH_MDIO	Ethernet		
B15	ETH_TXEN	Ethernet			A15	ETH_RXCLK	Ethernet		
B16	ETH_TXER	Ethernet			A16	ETH_RXDV	Ethernet		
B17	ETH_TXD3	Ethernet			A17	ETH_RXD3	Ethernet		
B18	ETH_TXD2	Ethernet			A18	ETH_RXD2	Ethernet		
B19	ETH_TXD1	Ethernet			A19	ETH_RXD1	Ethernet		
B20	ETH_TXD0	Ethernet			A20	ETH_RXD0	Ethernet		
B21	GPIO1 / RTS1	GPIO / UART	MC33926_D1	(2)	A21	SSI_MCLK	SSI		
B22	GPIO2 / SDHC_D1	GPIO / SDHC	MC33926_D2bar	(2)	A22	SSI_BCLK	SSI		
B23	GPIO3	GPIO	MC33926_D2bar	(2)	A23	SSI_FS	SSI		
B24	CLKIN0	Clock			A24	SSI_RXD	SSI		
B25	CLKOUT1	Clock			A25	SSI_TXD	SSI		
B26	**GND**	**Power**	**Ground**		A26	**GND**	**Power**	**Ground**	
B27	AN7	ADC	Reserved 7		A27	AN3	ADC	Reserved 3	
B28	AN6	ADC	Reserved 6		A28	AN2	ADC	Reserved 2	
B29	AN5	ADC			A29	AN1	ADC	MC33932_FBB	(2)
B30	AN4	ADC	MC33926_FB	(2)	A30	AN0	ADC	MC33932_FBA	(2)
B31	**GND**	**Power**	**Ground**		A31	**GND**	**Power**	**Ground**	
B32	DAC1	DAC			A32	DAC0	DAC		
B33	TMR3	Timer	Reserved 5		A33	TMR1	Timer	Reserved 1	
B34	TMR2	Timer	Reserved 4		A34	TMR0	Timer	Reserved 0	
B35	GPIO4	GPIO			A35	GPIO6	GPIO	MC33932_EN/ D4bar	(2)
B36	**3.3V**	**Power**	**3.3V Power**		A36	**3.3V**	**Power**	**3.3V Power**	
B37	PWM7	PWM			A37	PWM3	PWM	MC33932_IN4	(2)
B38	PWM6	PWM			A38	PWM2	PWM	MC33932_IN3	(2)
B39	PWM5	PWM	MC33926_IN2	(2)	A39	PWM1	PWM	MC33932_IN2	(2)

Table 8. Primary Elevator Connector Pinouts

Side B					Side A				
Pin #	Name	Group	Usage	Jumper	Pin #	Name	Group	Usage	Jumper
B40	PWM4	PWM	MC33926_IN1	(2)	A40	PWM0	PWM	MC33932_IN1	(2)
B41	CANRX	CAN			A41	RXD0	UART 0		
B42	CANTX	CAN			A42	TXD0	UART 0		
B43	1WIRE	1-Wire			A43	RXD1	UART 1	MC33932_EN/ D4bar	(2)
B44	SPI0_MISO	SPI 0	MC33926_EN	(2)	A44	TXD1	UART 1	MC33932_D1	(2)
B45	SPI0_MOSI	SPI 0	MC33926_SLEW	(2)	A45	GPIO10	GPIO	VSSA	
B46	SPI0_CS0	SPI 0	MC33926_INV	(2)	A46	GPIO11	GPIO	VDDA	
B47	SPI0_CS1	SPI 0			A47	GPIO12	GPIO		
B48	SPI0_CLK	SPI 0	MC33926_SFbar	(2)	A48	GPIO13	GPIO		
B49	**GND**	**Power**	**Ground**		A49	**GND**	**Power**	**Ground**	
B50	SCL1	I2C 1			A50	GPIO14	GPIO	MC33932_EN/ D4bar	(2)
B51	SDA1	I2C 1			A51	GPIO15	GPIO	MC33932_D1	(2)
B52	GPIO5 / SD_CARD_DET	GPIO/ SDHC	MC33926_SFbar		A52	GPIO16	GPIO		
B53	USB0_DP_PDOWN	USB 0			A53	GPIO17	GPIO		
B54	USB0_DM_PDOWN	USB 0			A54	USB0_DM	USB 0		
B55	IRQ_H	Interrupt			A55	USB0_DP	USB 0		
B56	IRQ_G	Interrupt			A56	USB0_ID	USB 0		
B57	IRQ_F	Interrupt			A57	USB0_VBUS	USB 0		
B58	IRQ_E	Interrupt			A58	TMR7	Timer		
B59	IRQ_D	Interrupt	MC33926_SFbar	(2)	A59	TMR6	Timer		
B60	IRQ_C	Interrupt			A60	TMR5	Timer		
B61	IRQ_B	Interrupt	MC33932_SFBbar	(2)	A61	TMR4	Timer		
B62	IRQ_A	Interrupt	MC33932_SFAbar	(2)	A62	RSTIN_b	Reset		
B63	EBI_ALE/EBI_CS1_b	EBI			A63	RSTOUT_b	Reset		
B64	EBI_CS0_b	EBI			A64	CLKOUT0	Clock		
B65	**GND**	**Power**	**Ground**		A65	**GND**	**Power**	**Ground**	
B66	EBI_AD15	EBI			A66	EBI_AD14	EBI		
B67	EBI_AD16	EBI			A67	EBI_AD13	EBI		
B68	EBI_AD17	EBI			A68	EBI_AD12	EBI		
B69	EBI_AD18	EBI			A69	EBI_AD11	EBI		
B70	EBI_AD19	EBI			A70	EBI_AD10	EBI		
B71	EBI_R/W_b	EBI			A71	EBI_AD9	EBI		
B72	EBI_OE_b	EBI			A72	EBI_AD8	EBI		
B73	EBI_D7	EBI			A73	EBI_AD7	EBI		
B74	EBI_D6	EBI			A74	EBI_AD6	EBI		
B75	EBI_D5	EBI			A75	EBI_AD5	EBI		
B76	EBI_D4	EBI			A76	EBI_AD4	EBI		
B77	EBI_D3	EBI			A77	EBI_AD3	EBI		
B78	EBI_D2	EBI			A78	EBI_AD2	EBI		
B79	FB_D1	Flexbus			A79	FB_AD1	Flexbus		
B80	FB_D0	Flexbus			A80	FB_AD0	Flexbus		
B81	**GND**	**Power**	**Ground**		A81	**GND**	**Power**	**Ground**	
B82	**3.3V**	**Power**	**3.3V Power**		A82	**3.3V**	**Power**	**3.3V Power**	

Notes

2. 0 Ω resistor is connected between the pin and the connector to create a flexible connection.

5 Setting up the Hardware

When configured as a Freedom Tower platform module, the TWR-MC-MVHB1EVB must be used in conjunction with another Tower MCU evaluation board (available at www.freescale.com/tower.)

The following procedure describes how to set up the hardware when the TWR-MC-MVHB1EVB is configured to drive two DC brushed motors. (The MC33932 can only support a single stepper motor, attached to either the Channel A or the Channel B connector.)

1. Assemble the Freescale Tower platform by sliding the TWR-MC-MVHB1EVB elevator connectors into the top slots on the Tower Elevator modules. Insert the Tower MCU evaluation board in the Tower Elevator modules in a set of slots below the TWR-MC-MVHB1EVB.
2. Connect the USB cable between the PC and the USB port on the Tower MCU evaluation board.
3. Connect the MC33932 Channel A load to connector J1 (OUT2 & OUT1) and the MC33932 Channel B load to connector J2 (OUT4 and OUT3) on the TWR-MC-MVHB1EVB evaluation board.
4. Connect the MC33932 8.0 V—28 V DC power supply to connector J3 (PWR-IN) on the evaluation board.
5. Connect the MC33926 load to connector J4 (OUT2 & OUT1) on the TWR-MC-MVHB1EVB evaluation board.
6. Connect the MC33926 8.0 V—28 V DC power supply to connector J5 (PWR-IN) on the evaluation board.
7. Launch the software application used to communicate with the board.

Figure 9 illustrates the procedure.

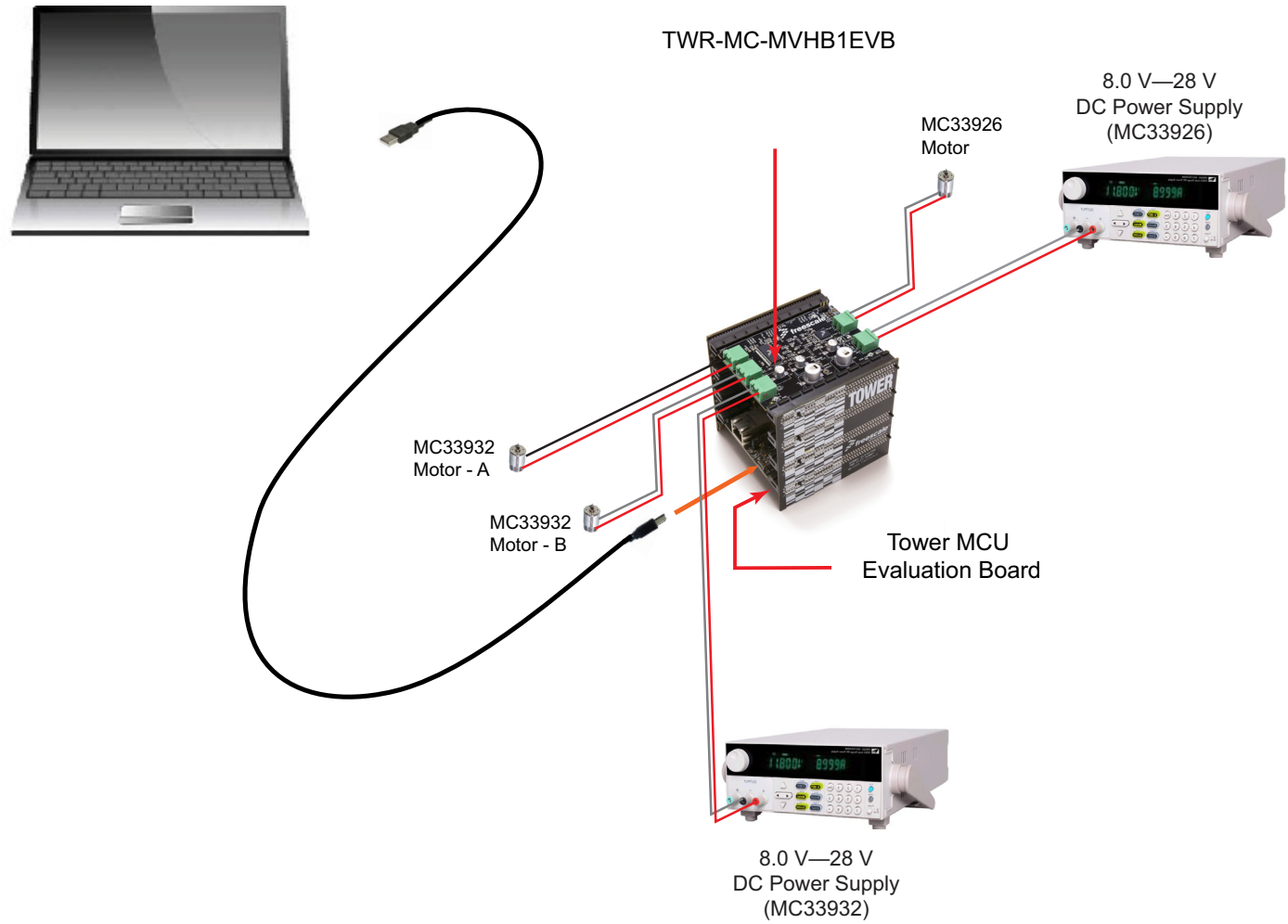


Figure 9. TWR-MC-MVHB1EVB Tower System Hardware Configuration

6 Installing Processor Expert Software

6.1 Installing CodeWarrior on your Computer

This procedure explains how to obtain and install the latest version of CodeWarrior (version 10.6 in this guide).

NOTE

The sample software in this kit requires CodeWarrior 10.6 or newer. The component and some examples in the component package are intended for Kinetis Design Studio 3.0.0. If you have CodeWarrior 10.6 and Kinetis Design Studio 3.0.0 already installed on your system, skip this section.

1. Obtain the latest CodeWarrior installer file from the Freescale CodeWarrior website here: www.freescale.com/webapp/sps/site/homepage.jsp?code=CW_HOME&tid=vanCODEWARRIOR.
2. Run the executable file and follow the instructions.
3. In the **Choose Components** window, select the Kinetis component and click on **Next** to complete the installation.

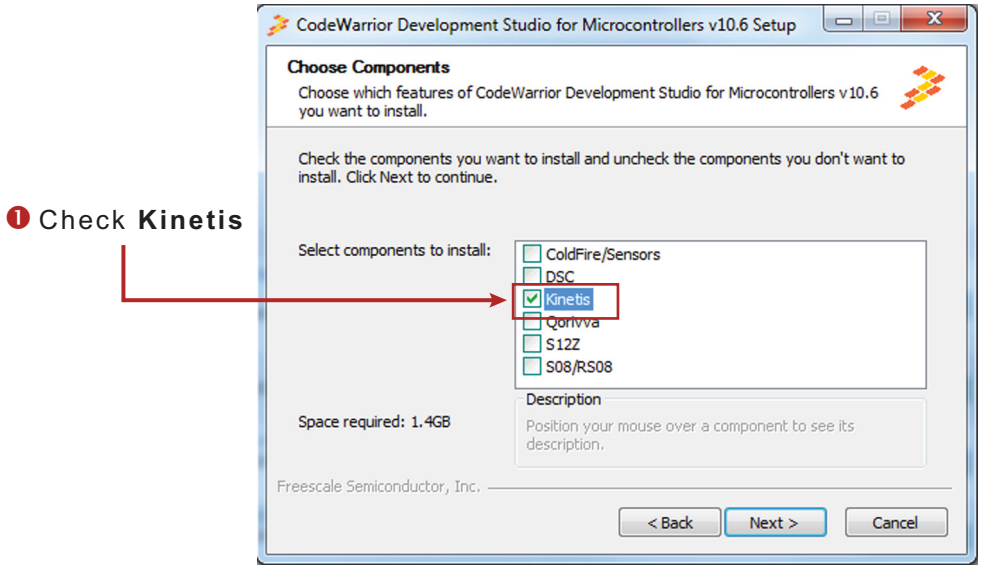


Figure 10. Choose CodeWarrior Components Screen

6.2 Downloading the MVHBridge Component and Example Projects

The examples used in this section are based on a pre-configured CodeWarrior project. You must first download the project and its associated components:

1. Go to the Freescale website www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MVHBRIDGE-PEXPERT
2. Download example projects and H-Bridge component zip file.
3. Unzip the downloaded file and check that the folder contains the files listed in Table 9.

Table 9. MVHBridge Example Project and Components

Folder Name	Folder Contents
CodeWarrior_Examples	Example project folder for CodeWarrior
MVH_K20D72M_brushed	Example project for DC brush motor control
MVH_K20D72M_brushed_FreeMaster	Example project intended for control of brushed motor using FreeMaster tool. Latest Freemaster installation package: www.freescale.com/freemaster
MVH_K20D72M_step_FreeMaster	Example project intended for control of stepper motor using FreeMaster tool
MVH_K20D72M_stepper	Example project for stepper motor control using full-stepping and micro-stepping mode
MVH_K20D72M_stepper_fullstep	Example project for stepper motor control demonstrating full-step mode
MVH_K20D72M_stepper_ramp	Example project for stepper motor control demonstrating acceleration and deceleration ramp
MVH_K64F120M_brushed_2component	Example project for DC brush motor control using two H-Bridges (i.e. MC33932 and MC33926)
MVH_K70F120M_brushed	Example project for TWR-K70F120M with DC brushed motor control.
MVH_K70F120M_stepper	Example project for TWR-K70F120M with stepper motor control using full-stepping and micro-stepping mode
MVH_KL25Z48M_brushed_2component	Example project for DC brushed motor control using a dual H-Bridge device (e.g. MC33932 and 33926)
MVH_KL25Z48M_fullstep_ramp	Example project for stepper motor control demonstrates acceleration and deceleration ramp in full-step mode
Component	Processor Expert component folder
DriverSuite_Examples	Example project folder for Driver Suite
MVH_K20D72M_stepper	Example project for stepper motor control uses full-stepping and micro-stepping mode
KDS_Examples	Example project folder for Kinetis Design Studio
MVH_K20D72M_stepper	Example project for stepper motor control, which uses full-stepping and micro-stepping mode
MVH_K20D72M_stepper_ramp	Example project for stepper motor control demonstrating usage of acceleration and deceleration ramp
FRDM34931SEVB_Examples	Example project folder for CodeWarrior and H-Bridge board FRDM-34931SEVB
MVH_KL25Z_brushed	Example project for DC brush motor control

6.2.1 Importing the MVHBridge Component into the Processor Expert Library

1. Launch CodeWarrior by clicking on the CodeWarrior icon (located on your desktop or in **Program Files** -> **Freescale Codewarrior** folder.)
2. When the CodeWarrior IDE opens, go to the menu bar and click **Processor Expert** -> **Import Component(s)**.
3. In the pop-up window, locate the component file (.PEupd) in the example project folder MVHBridge_PEx_SW\Component. Select **MVHBridge_b**xxx**.PEupd** and **ChannelAllocator_b**xxx**.PEupd** files then click **Open** (see Figure 11).

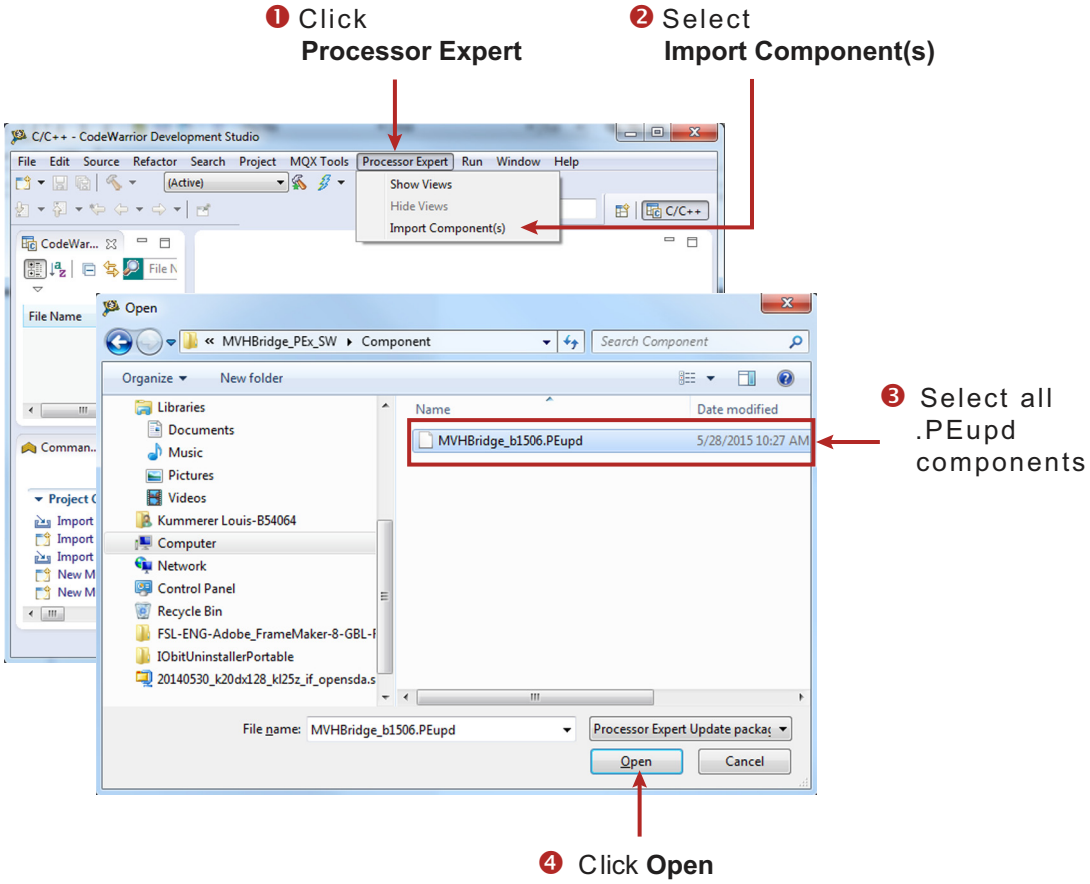


Figure 11. Importing the MVHBridge Component

4. If the import is successful, the MVHBridge component appears in **Components Library** -> **SW** -> **User Component** (see Figure 12). The MVHBridge component is ready to use.

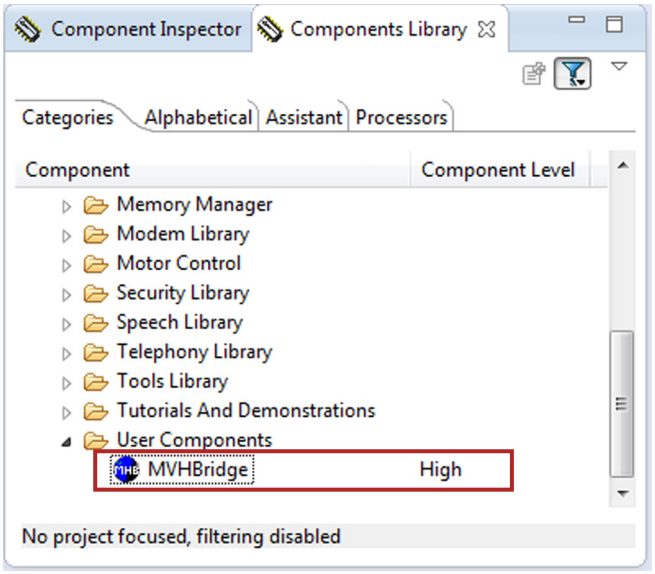


Figure 12. MVHBridge Component Location After Importing to CodeWarrior

6.2.2 Importing an Example Project into the Processor Expert Library

The following steps show how to import an example from the downloaded zip file into CodeWarrior.

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

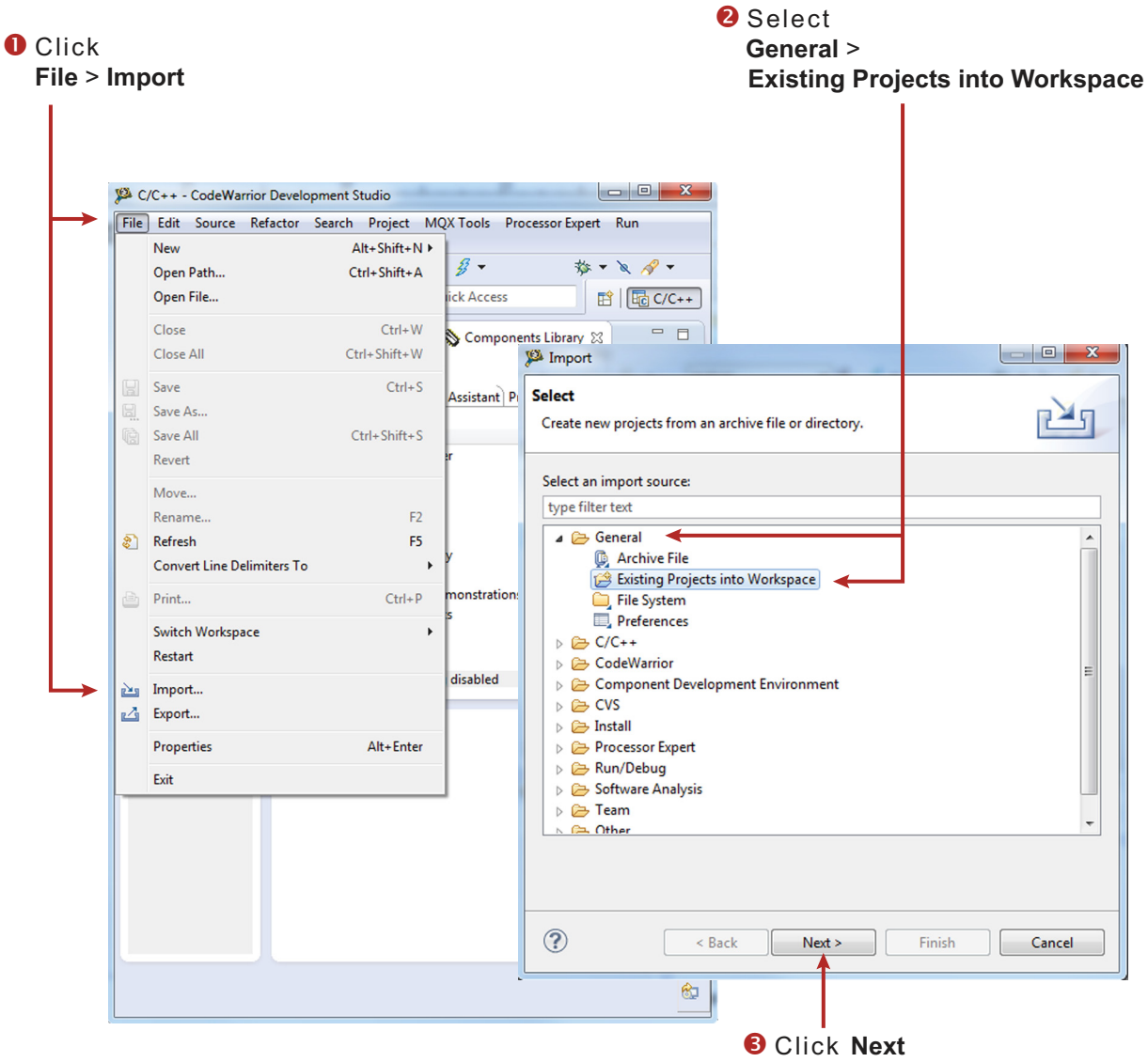


Figure 13. Importing an Example File (a)

2. Click **Browse** and locate the folder where you unzipped the downloaded example files. Find the folder MVHBridge_PEx_SW\CodeWarrior_Examples and select a project to import. (see Figure 14, which shows MVH_K20D72M_step_FreeMaster as the imported project). Then click **OK**.

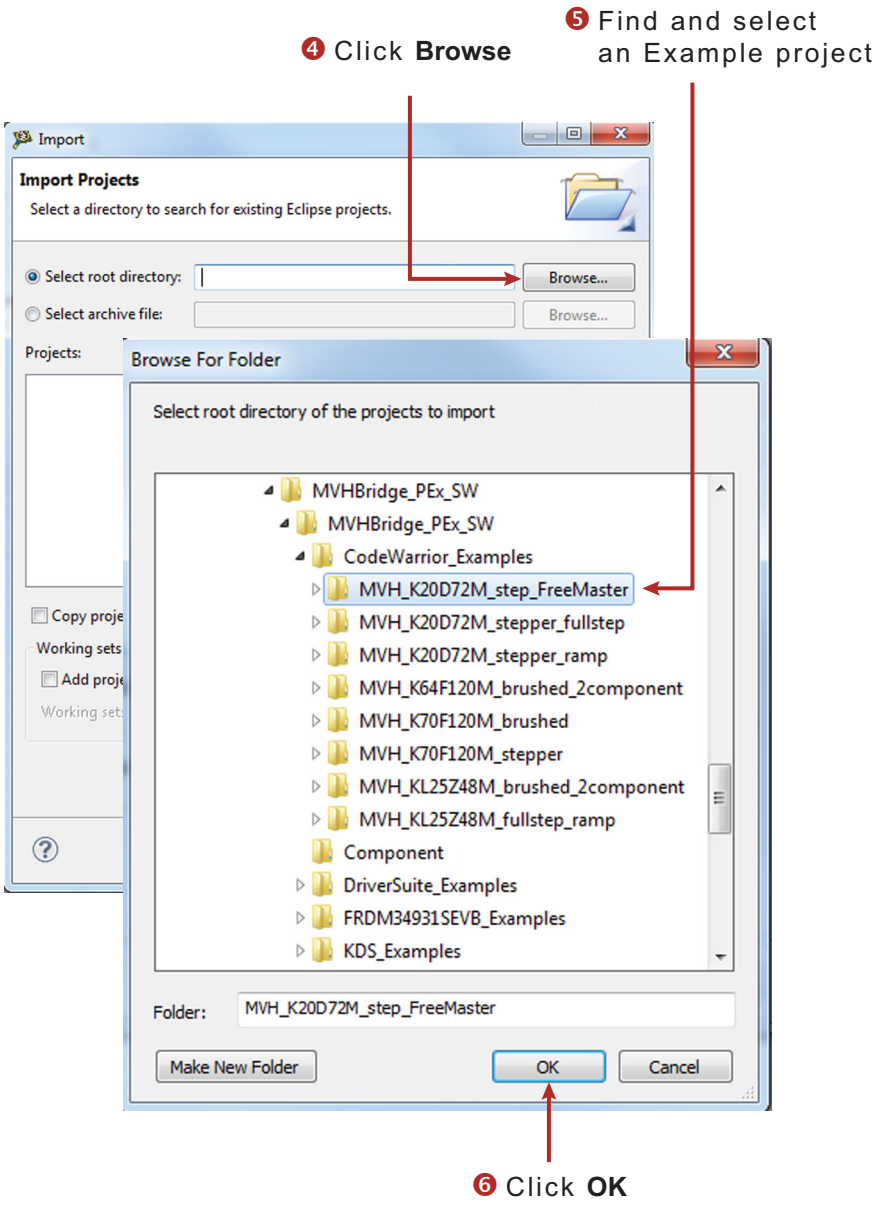


Figure 14. Importing an Example File (b)

3. With your project now loaded in the **Select root directory** box, click on the **Copy projects into workspace** checkbox. Then click **Finish**. Figure 15 shows the CodeWarrior **Projects** panel and the **Components** panel after the project has been successfully imported.

The project is now in the CodeWarrior workspace where you can build and run it.

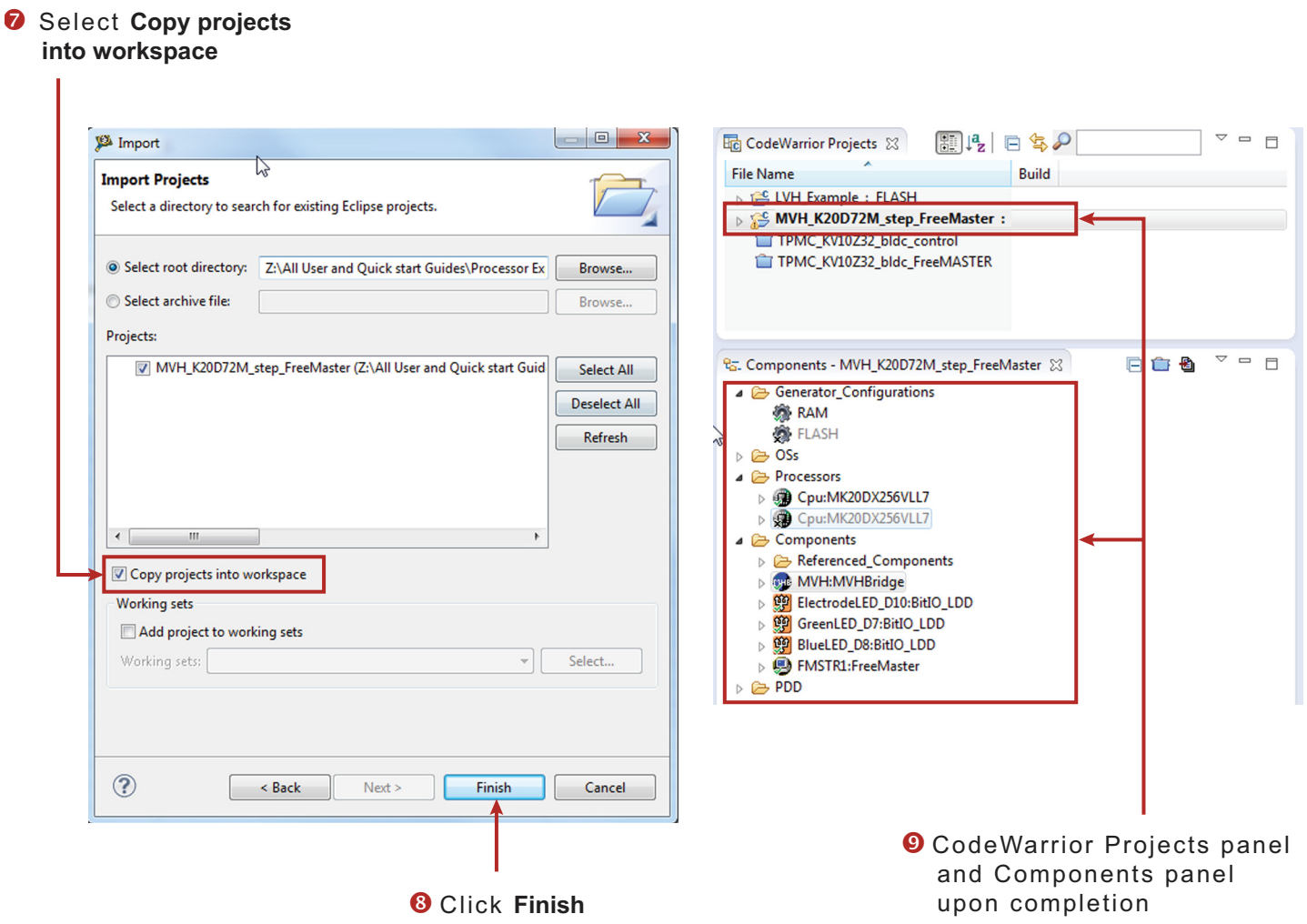


Figure 15. Importing an Example File (c)

6.3 Creating a New Project with Processor Expert and the MVHBridge Component

If you choose not to use the example project, the following instructions describe how to create and setup a new project that uses the MVHBridge component. If you do not have the MVHBridge component in the Processor Expert Library, please follow steps in Importing the MVHBridge Component into the Processor Expert Library on page 18.

To create a new project do the following:

1. In the CodeWarrior menu bar, select **File** -> **New** -> **Bareboard Project**. When the **New Bareboard Project** dialog box opens, enter a project name into the text box and then click **Next**. (see Figure 16).

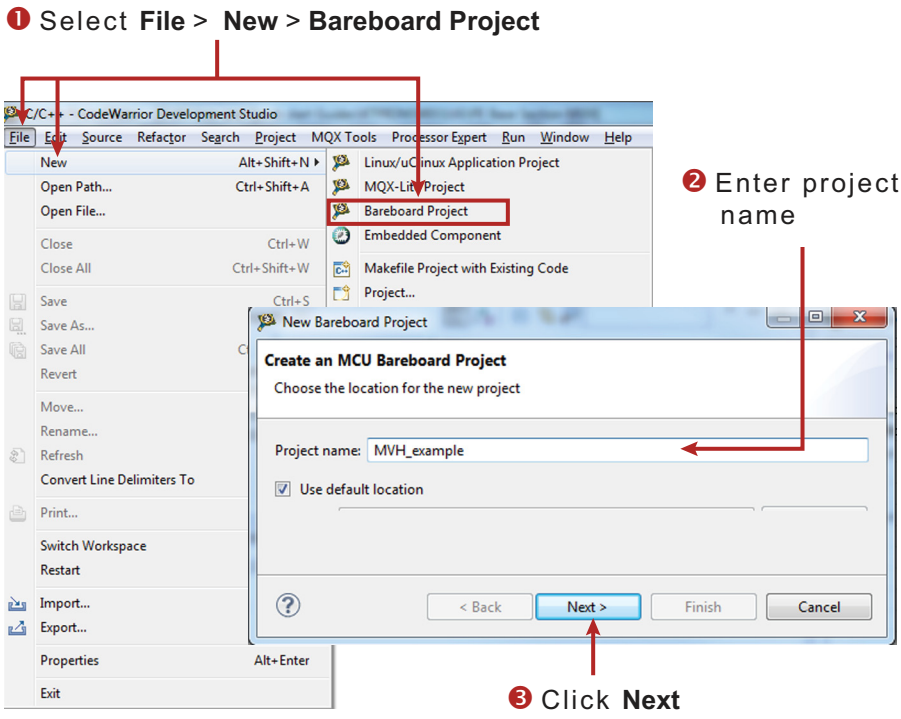


Figure 16. Creating an MCU Bare-board Project

2. In the **Devices** dialog box, select the MCU class your project is using in the MCU board (In Figure 17, MK20DX256 has been selected). Then click **Next**.
3. In the **Connections** dialog box, select the type of connection your project uses. (In Figure 17 **P&E USB Multilink Universal [FX]/USB MultiLink** has been selected). Then click **Next**.

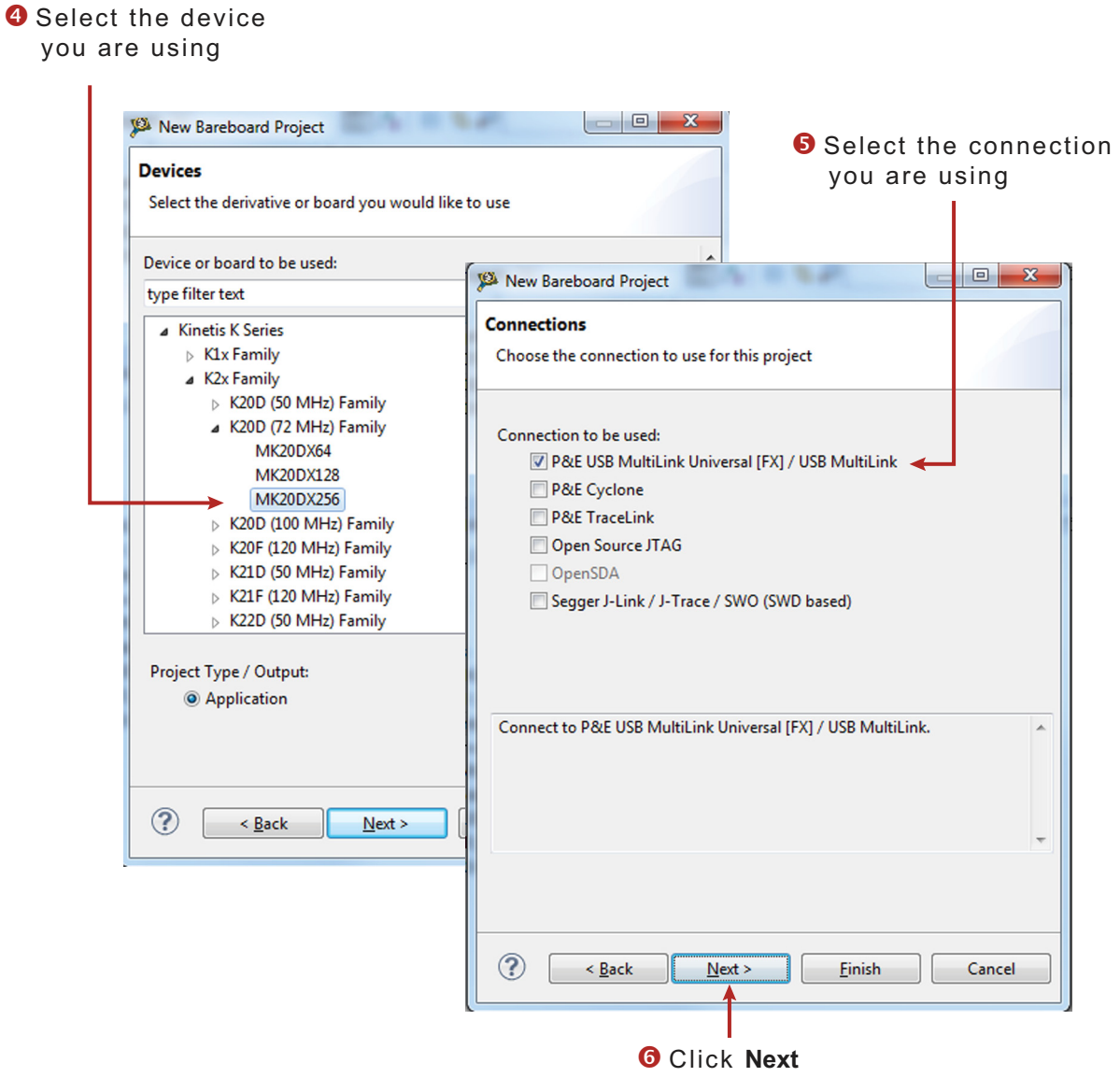


Figure 17. Selecting a Device and a Connection

4. In the **Language and Build Tools Options** dialog box, select the options that apply to your project. (In Figure 18, the default options are selected.) Then click **Next**.
5. In the **Rapid Application Development** dialog box, make sure that the **Processor Expert** button is selected. Then click **Finish**

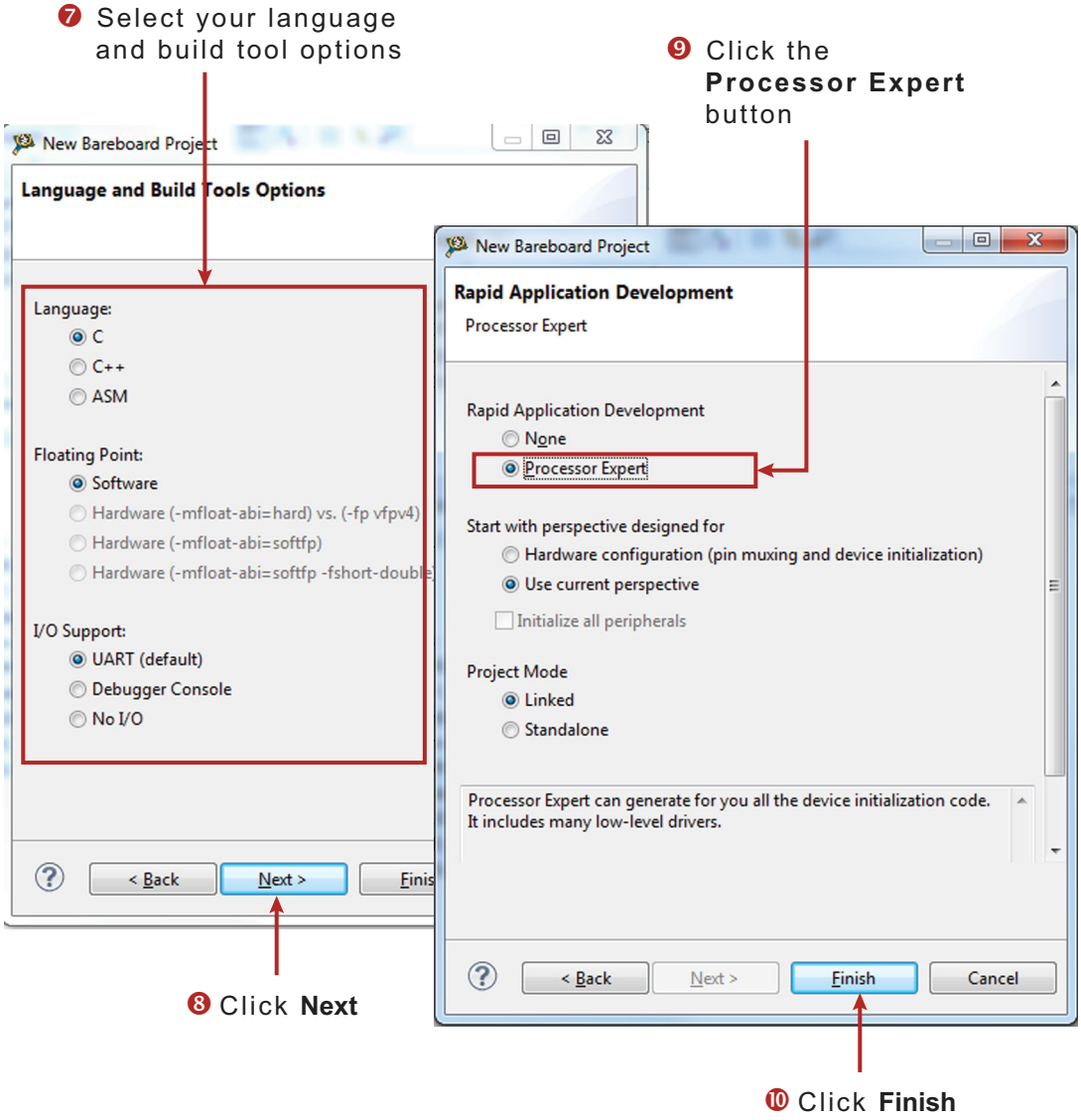


Figure 18. Selecting the Language, Build Tools, and the Rapid Application Development Options

6. Figure 19 shows the CodeWarrior Projects panel and the Components panel after the project has been successfully created.

Before you can build and run your project, you must add the MVHBridge component (imported in Importing the MVHBridge Component into the Processor Expert Library on page 18) into your project. Adding the MVHBridge Component into the Project on page 27 outlines this procedure.

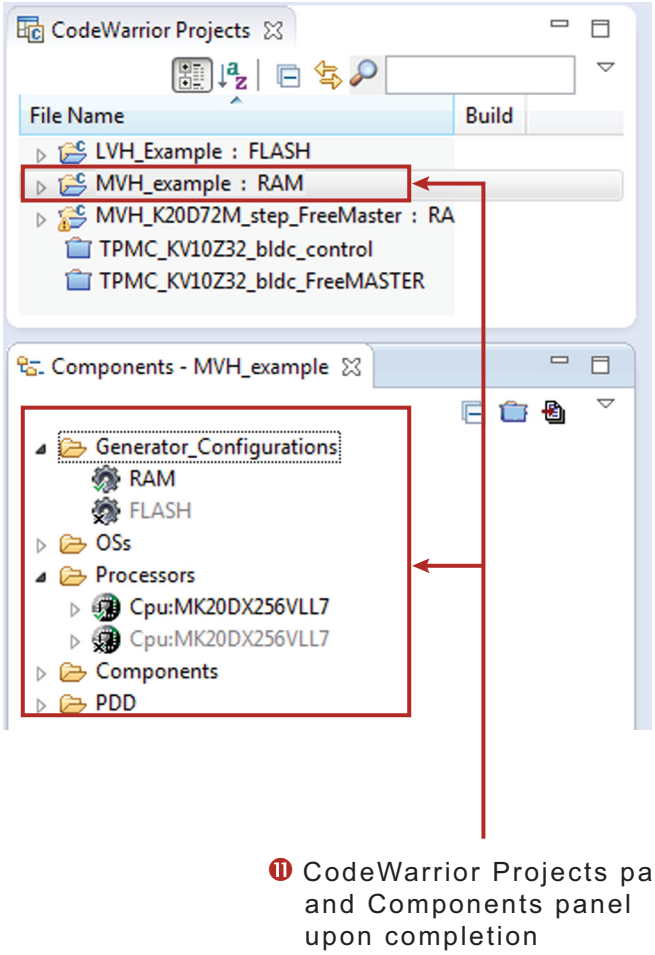


Figure 19. CodeWarrior Projects and Components Panels with Project Created

6.3.1 Adding the MVHBridge Component into the Project

1. Find MVHBridge in the Components Library and add it into your project (see Figure 20).

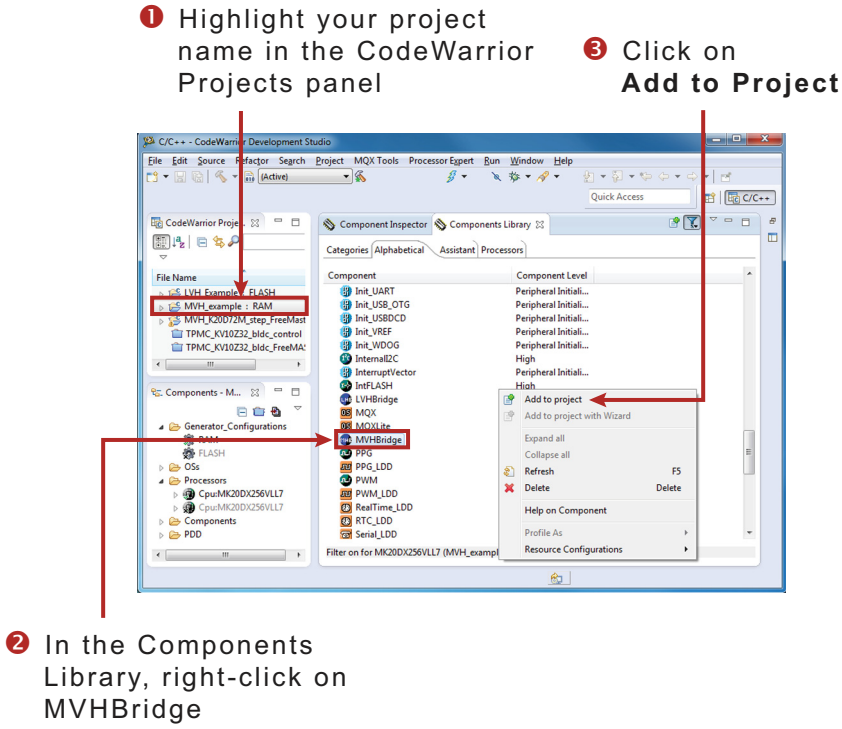


Figure 20. Add the MVHBridge Component to the Project

2. Figure 21 shows the Components panel after the component has been added. To view the Component Inspector options, double click on the MVHBridge component in the Components panel.

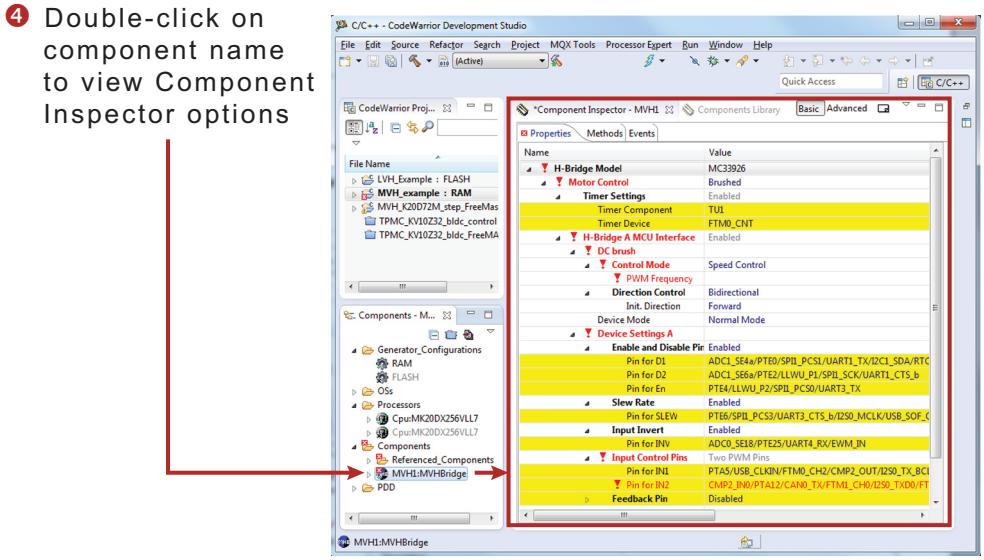


Figure 21. Select the component

6.3.2 General Settings of MVHBridge Component

The Component Inspector view provides a means of accessing and modifying component properties. When CodeWarrior is set to the **Classic** view, properties in the Component Inspector are arranged in a collapsible tree-structure. Property names appear in the **Name** column. The **Values** column lists the current value assigned to the property. Values that are not greyed-out in this column may be modified. The **Details** column contains additional information (including error conditions) about the selected property. (If you have CodeWarrior preferences set to the **Tab** view, properties will be arranged differently in the Component Inspector; However, the same definitions apply.)

Figure 22 shows typical Component Inspector properties for a project using a DC brushed motor and an MC33926 device with a single H-Bridge. Different components and settings may apply when other types of motors and MCU's are used.

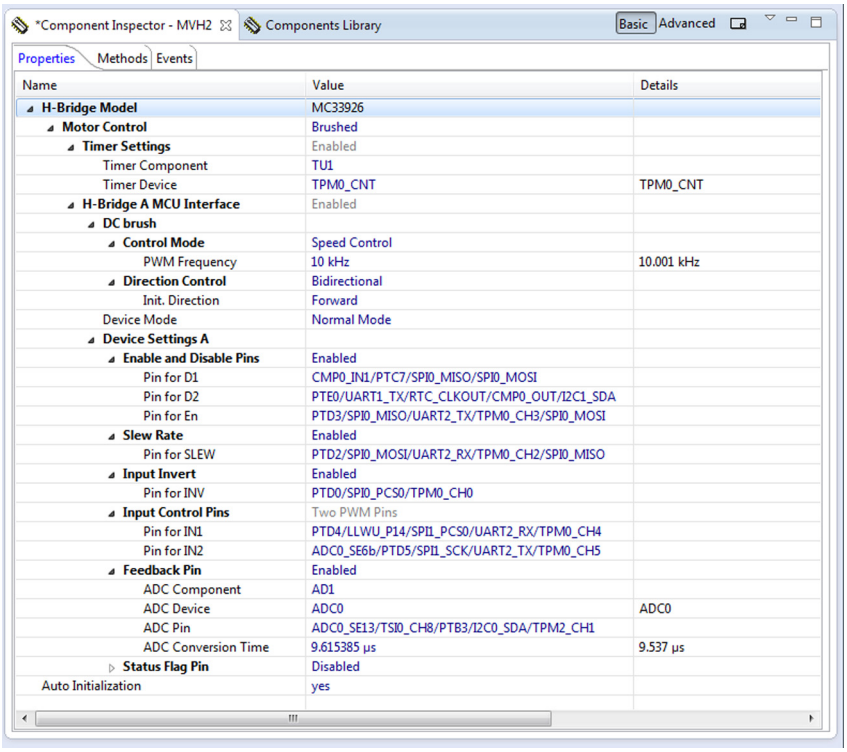


Figure 22. Component Inspector - Brushed DC Motor Project

For the project in Figure 22 the **H-Bridge Model** is the top node in the tree structure. A drop-down menu in the **Value** column allows you to select the H-Bridge model your project uses.

The **Motor Control** group is directly below the **H-Bridge Model** node. The group contains two child nodes: **Timer Setting** and **H-Bridge A MCU Interface**. An MCU with dual H-Bridges would have an **H-Bridge B MCU Interface** group with settings similar to H-Bridge A. The settings in each of these groups are detailed below:

Timer Setting when enabled, defines timer settings for the project. (For the MC33926 used in this example, the timer is enabled by default.) The group contains the following settings:

- Timer Component defines the name of the linked **TimerUnit_LDD Component**.
- Timer Device defines the name of the hardware timer being used.

H-Bridge A MCU Interface defines H-Bridge interface setting. The group contains three child nodes:

DC Brush allows you to select the motor control mode and the motor direction:

Control Mode allows you to select whether your settings control the motor speed (**Speed Control**) or whether the motor is controlled by GPIO pin signals (**State Control**).

- **PWM Frequency** sets the Pulse Width Modulation frequency.

Direction Control determines in which direction the motor is allowed to rotate. **Forward** means the motor can rotate only in the clockwise direction. **Reverse** allows movement in the counterclockwise direction only. **Bidirectional** allows the motor to rotate in either direction.

- **Init Direction** determines which direction (forward or reverse) the motor moves at startup.

Device Mode defines the H-Bridge operational mode for the selected device. The mode specifics depend on the device, but Normal, Sleep, and Stand-by are typical. For more information, see the data sheet for your device. **Device Mode** is controlled by enabling and disabling pins. The mode can be changed in your C code using the **SetMode** method.

Device Settings A associates each of the output pins with a corresponding input pin name.

Enable and Disable Pins settings control the Device Mode. The number and the names of pins in this group depends on the H-Bridge model you have selected. In all cases, you must assign the appropriate value to each pin name in the group.

Slew Rate enables the selection of fast or slow slew rate.

- **Pin for Slew** defines the component setting associated with the device's SLEW pin.

Input Invert allows you to set the device's INV pin, subsequently causing IN1 and IN2 to be set to LOW = TRUE (see the device's data sheet for additional information on the Input Invert (INV) pin).

- **Pin for INV** defines the component setting associated with the device's INV pin.

Input Control Pins settings define H-Bridge outputs. These pins are controlled by timer channels or by GPIO pins according to other settings in the component.

Feedback Pin settings define current measurements on the feedback pin. H-Bridge feedback provides ground-referenced 0.24% of the high side output current.

- **ADC Component** sets the name of the linked ADC_LDD component.
- **ADC Device** defines the device used for current measurement.
- **ADC Pin** defines the pin used for ADC current sensing.
- **ADC Conversion Time** specifies the time interval in micro-seconds allowed for a single analog to digital conversion.

Status Flag Pin allows tracking of the H-Bridge status flag. Method **GetStatusFlag** provides current device status. Method **ClearStatusFlag** clears the status flag. Use Event **OnStatusFlagA** or **OnStatusFlagB** (depending on the H-Bridge interface) to handle errors indicated by the status flag.

Auto Initialization when set, causes Processor Expert to automatically make an initialization call. If this option is not set, your code must make the Init call.

6.3.3 Setting up a Project to Control a DC Brushed Motor

1. Select the H-Bridge model you want to configure and set the **Motor Control** property to **Brushed**.

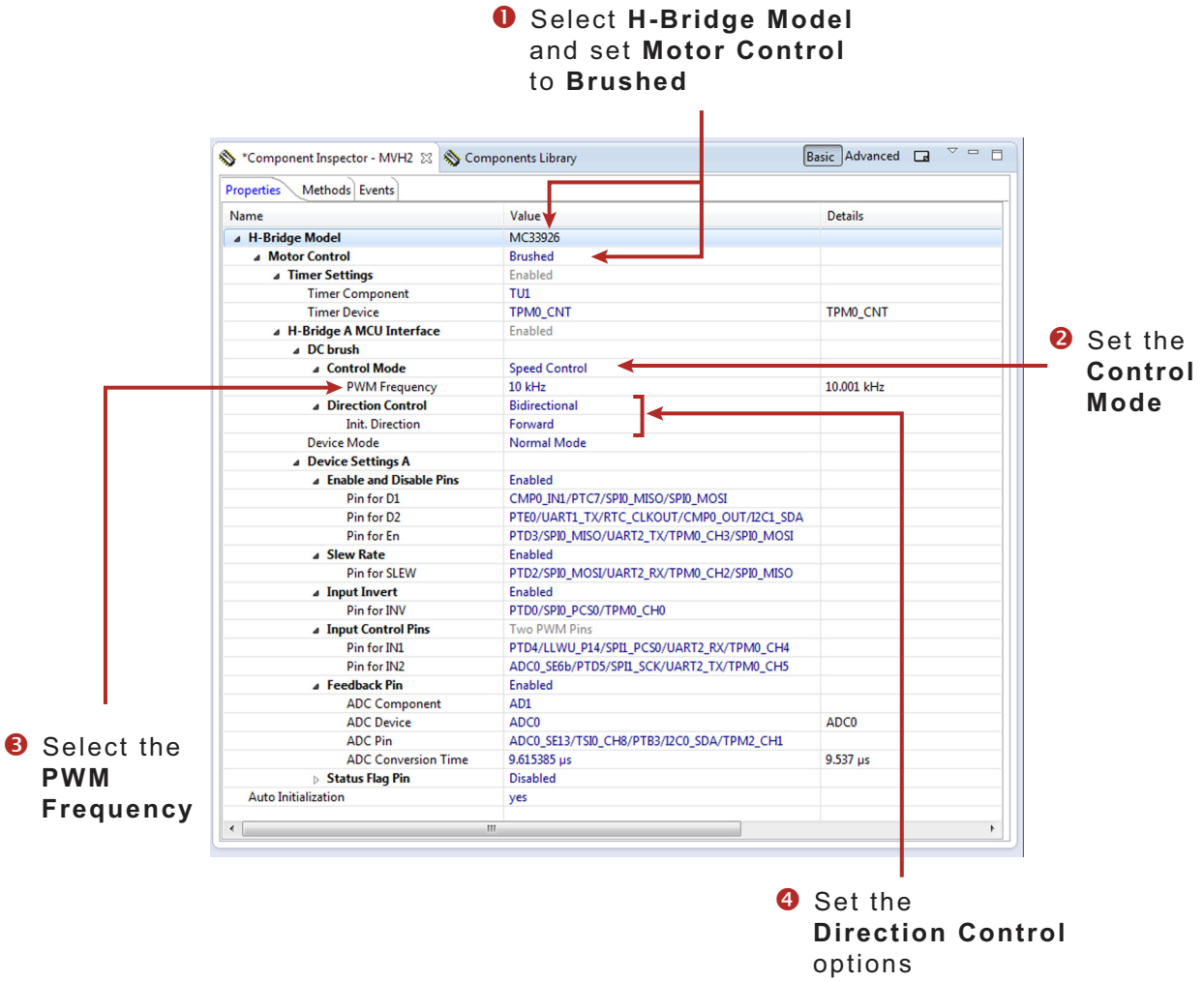


Figure 23. Brushed Motor Control Setup

2. Set the **Control Mode** property. There are two ways to control the DC brushed motor:

   Speed Control - motor speed is controlled by your settings. The **TimerUnit_LDD** component is used to generate the PWM signal. The **PWM Frequency** property is visible in this mode only. If you set the **Speed Control** mode on both interfaces (i.e. Interface A and Interface B), the **PWM Frequency** property on Interface B will be set automatically to the same value as Interface A (because Interface B uses the same timer.)

   State Control - motor is controlled by GPIO pins (**BitIO_LDD** components). This means you can switch the motor on or off without speed adjustments. The advantage of this mode is that you do not need timer channels. If you set **State Control** on both interfaces or you have only a single H-Bridge model (one interface) with **State Control**, the T**imerUnit_LDD** component is not required by the MVHBridge component and you can remove it from the project.

3. Set the **PWM Frequency**.

4. Set the **Direction Control** property. The **Direction Control** property determines what direction the motor is allowed to move in. Setting the property to **Forward** restricts the motor's movement to the forward direction only. Setting the property to **Reverse** restricts movement to the reverse direction only. A **Bidirectional** setting allows the motor to move in either direction. The Bidirectional mode requires two timer channels. **Forward** or **Reverse** requires only one timer channel and one GPIO port. This setting is available only when **Speed Control** mode is set in the **Control Mode** property.

6.3.4 Setting up a Project to Control a Stepper Motor

1. Select the dual H-Bridge model you want to configure and set **Stepper** in the **Motor Control** property. Notice that you must use a dual H-Bridge model because a two phase bipolar stepper motor has four inputs.

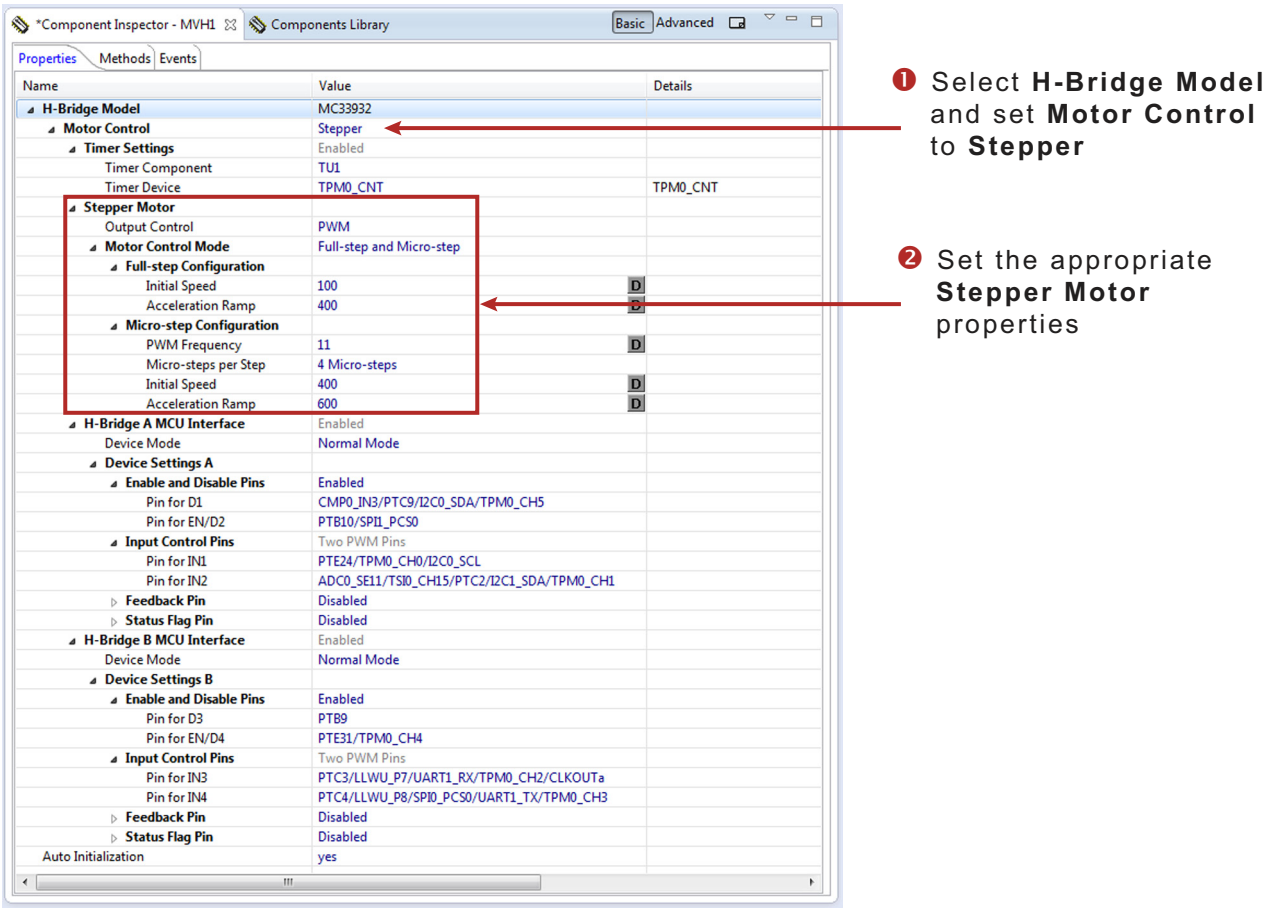


Figure 24. Stepper Motor Control Setup

2. In the **Stepper Motor** group, set the properties that apply to your environment.

   Output Control property defines the control method for the H-Bridge outputs.

   With **PWM** selected the component utilizes four channels of a timer to control the stepper motor. Signals are generated in hardware and micro-step mode is available.

   With **GPIO** selected, GPIO pins instead of timer channels control the motor and only full-step mode is available. Micro-step mode is not available. This mode consumes more MCU clock cycles because all signals are generated in interrupts using four GPIO pins.

   Manual Timer Setting (only visible when **Advanced** is selected) is designed to change the **Counter frequency** property of the linked **TimerUnit_LDD** component. By default **Counter frequency** is set automatically by the MVHBridge component. In some cases the frequency value does not have to be set appropriately (for example, when you want to set a different value or when an error has occurred). For more information see Stepper Motor Speed on page 32.

   Motor Control Mode allows you to select the step mode. The allowable values are **Full-step**, **Micro-step** and **Full-step and Micro-step**. Selecting **Full-step and Micro-step** allows you to switch between full-stepping and micro-stepping in C code.

   The **Full-step Configuration** group contains speed and acceleration settings. Code for the acceleration and deceleration ramp is generated when the Acceleration property is set to a value greater than zero. Note that acceleration is always the same as deceleration. An example of an acceleration ramp is depicted in Figure 25. The acceleration setting is 400, as shown in Figure 24.

Initial Speed is set to 100 full-steps per second. This value can be changed in C code.

Acceleration Ramp defines the ramp speed for both acceleration and deceleration. In this example the value is set to 400 full-steps per second^2. Note that the motor reaches the speed in 0.25 second (desired_speed / acceleration = 100 / 400 = 0.25).

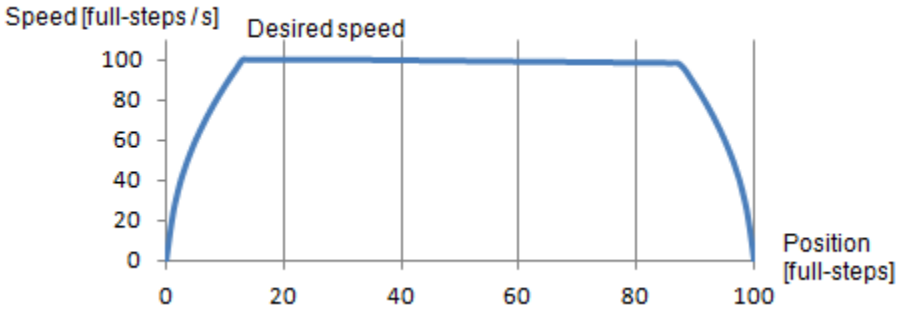


Figure 25. Acceleration and Deceleration Ramp

Micro-step Configuration group settings are similar to those of the **Full-step Configuration**. **PWM Frequency** is the frequency of the micro-step PWM signal. **Micro-step per Step** is the number of micro-steps per one full-step.

6.3.5 Stepper Motor Speed

This defines the stepper motor's minimum and maximum speed. These limit values are used by various component methods. Minimum and maximum speeds depend on the MVHBridge component settings. Counter input frequency does not influence micro-stepping spend, but could affect full-stepping speed.

Table 10: Minimum and maximum stepper motor speed

Mode Description	MVHBridge component properties			Stepper motor speed [steps per second]	
	Timer Device	Output Control	Motor Control Mode	Minimum Speed [steps/sec]	Maximum Speed [steps/sec]
Full-step mode	FTM	PWM	Full-step	Depends on Timer Input Frequency	11 000
Micro-step mode	FTM or TPM	PWM	Micro-step	1	5 000
Full-step mode (using TPM timer)	TPM	PWM	Full-step	1	11 000
Full-step mode (SW control)	FTM or TPM	GPIO	Full-step	1	11 000

Possible values for the timer input frequency (Counter frequency property in TimerUnit_LDD) are in Table 11. Input frequency values depend on MVHBridge component settings. Note that, in one case, two frequency values are needed in **Full-step and Micro-step** mode when FTM timer is used (the MVHBridge component switches in runtime between these two values).

Table 11. Minimum and maximum timer input frequency per stepper control mode

Mode Description	MVHBridge component properties			Stepper motor speed [steps per second]		
	Timer Device	Output Control	Motor Control Mode	Number of values	Minimum	Maximum
Full-step mode	FTM or TPM	PWM	Full-step	1	131 kHz	1 MHz
Micro-step mode	FTM or TPM	PWM	Micro-step	1	1.2 MHz	10 MHz
Full-step and Micro-step mode	FTM	PWM	Full-step and Micro-step	2	1st value for Full-step: 131 kHz	1st value for Full-step: 1 MHz
					2nd value for Micro-step: 1.2 MHz	2nd value for Micro-step: 10 MHz
Full-step and Micro-step mode (using TPM timer)	TPM	PWM	Full-step and Micro-step	1	1.2 MHz	10 MHz
Full-step mode (SW control)	FTM or TPM	GPIO	Full-step	1	131 kHz	1 MHz

6.3.5.1 Computation of Minimum Full-stepping Speed

The minimum full-stepping speed depends on the timer input frequency only when the Timer Device is set to FTM (FTM0_CNT, or FTM1_CNT, etc.), and **Output Control** is set to **PWM**. The Full-step signal is generated by a timer while channels toggle on compare (See Figure 26).

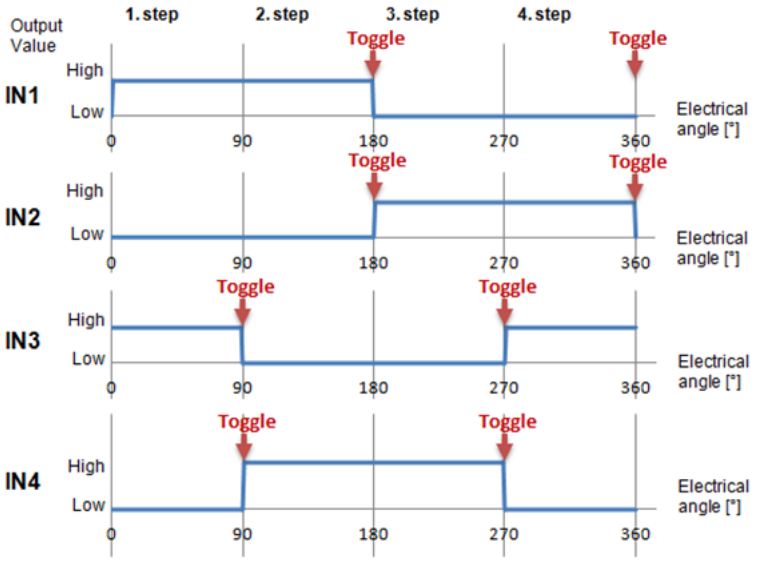


Figure 26. Generating the Full-step Control Signal

The Full-step minimum speed is derived from the input frequency of the timer device (the **Counter frequency** property of the **TimerUnit_LDD** component being used). You can find minimum values for speed in the MVHBridge header file (see constant <component_name>_MIN_FULLSTEP_ SPEED). The formula for calculation of this value is as follows:

$$Speed_{min} = \frac{2 \times Counter\_frequency}{65536} + 1$$

where:

Counter_frequency = input frequency of the timer device

65536 = maximum value of **TimerUnit_LDD** counter (16-bit counter).

Adding 1 ensures that the 16-bit counter does not overflow (which is the point of the formula).

For example if the **Counter frequency** is set to 187,500 Hz, the minimum speed is:

$$Speed_{min} = \frac{2 \times Counter\_frequency}{65536} + 1 = \frac{2 \times 187500}{65536} + 1 = 6.72$$

The MCU rounds the value down, so the result is 6 full-steps per second.

6.3.5.2 Setting the Minimum Full-stepping Speed

This section describes how to change the input frequency of the **TimerUnit_LDD** component.

1. Launch Processor Expert and select the MVHBridge component.
2. In the Processor Expert menu bar, set component visibility to **Advanced**.
3. In the Properties tab, find the **Manual Timer setting** property and set the value to **Enabled**. If you do not see this property, make sure that component visibility is set to **Advanced** (see Figure 27).

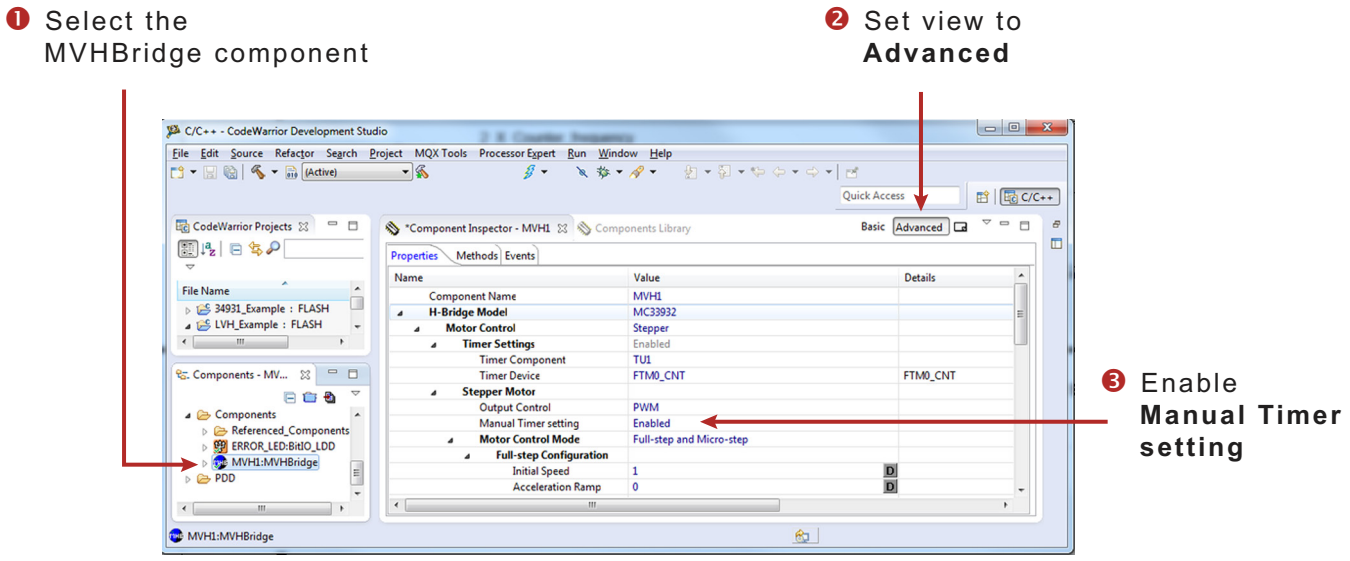


Figure 27. Enabling the Manual Frequency Setting

4. In the Components panel, expand the **Referenced_Components** folder and double-click on the component **TU1:TimerUnit_LDD**
5. Click the **Counter frequency** value field. An ellipsis button appears at the right edge of the cell.
6. Click on the ellipsis button.

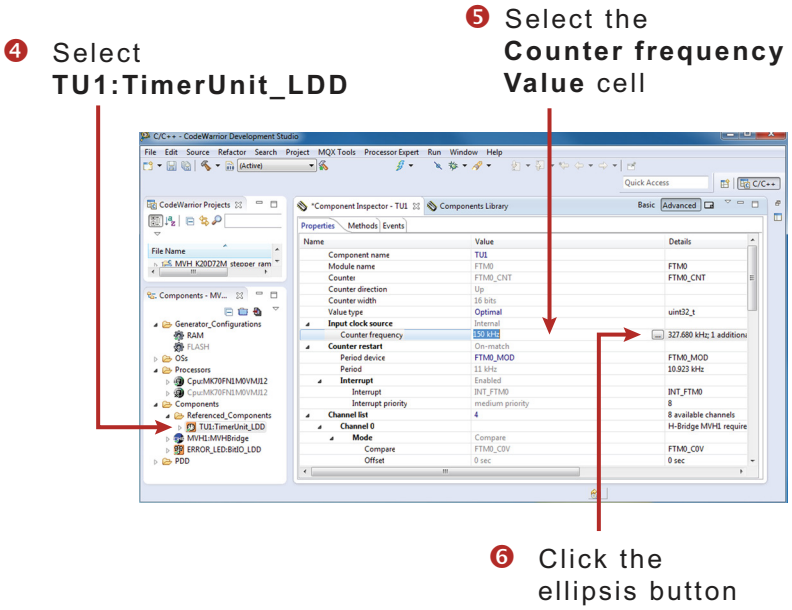


Figure 28. Component TimerUnit_LDD Timing Dialog

7. Set the frequency value (163.84 kHz in illustration). The list of available frequencies depends on the CPU component settings (with an external crystal as the clock source and a core clock of 48 MHz).
8. Set the **Allowed Error** value at 10% (see Figure 29).
9. Click **OK** to complete the process.

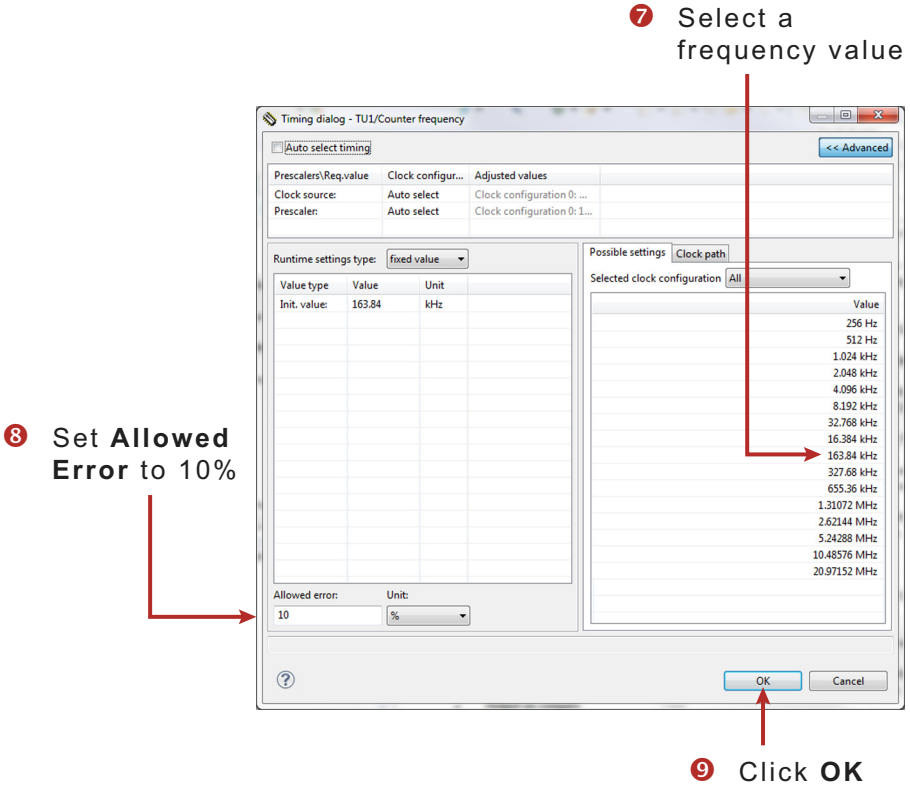


Figure 29. Component TimerUnit_LDD Timing Dialog—Select Input Frequency

6.3.6 Generating Driver Source Code

After you have completed configuring the components, you are ready to generate the driver code that will be incorporated into your application. The process is as follows

1. Click on the **Generate Processor Expert Code** icon in the upper right corner of the **Components** panel.

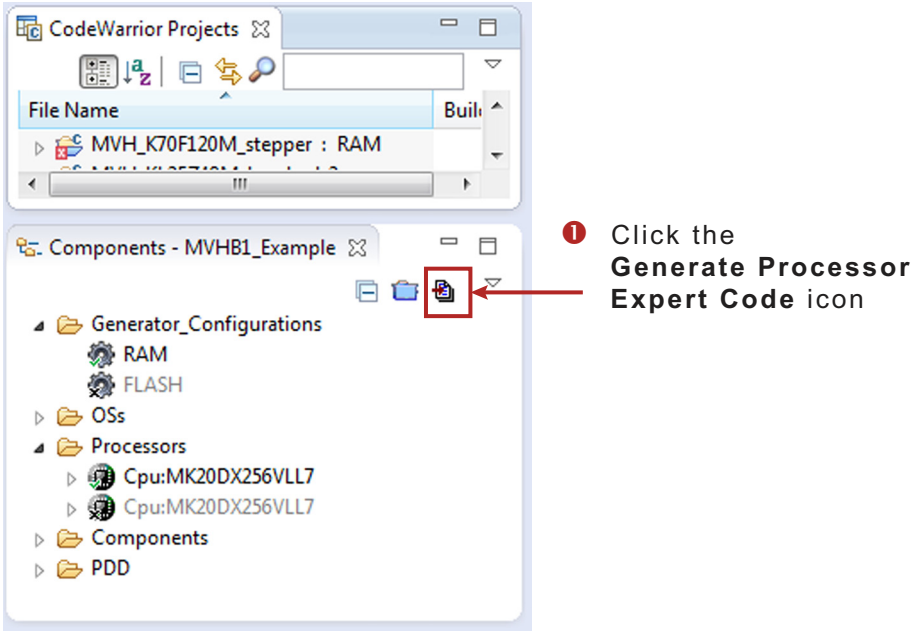


Figure 30. Generating the Source Code

2. The driver code for the H-Bridge device is generated into the **Generated_Code** folder in the **Project** panel. The component only generates the driver code. It does not generate application code. Figure 31 shows the locations of the generated driver source and the application code.

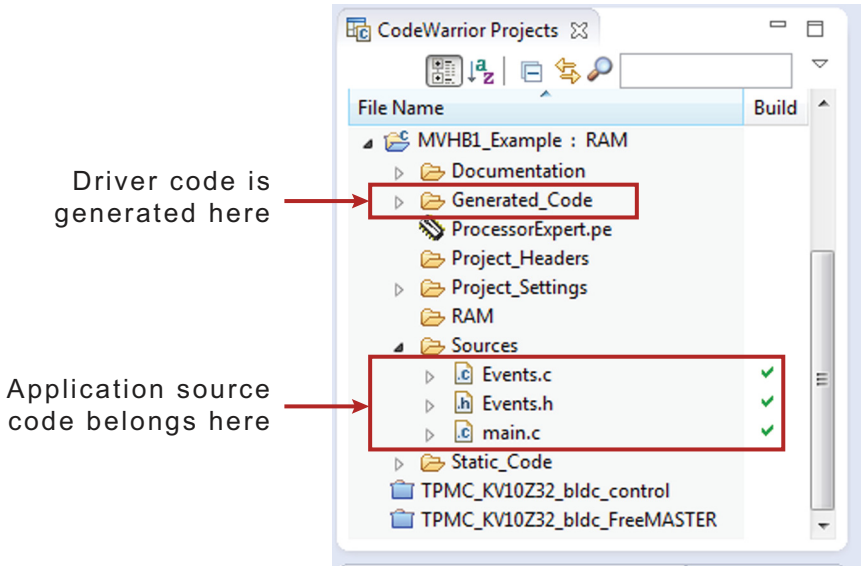


Figure 31. Source Code Locations

6.3.7 Developing Application Code in Processor Expert

Processor Expert allows you to write application code, add component methods, and build your application without leaving the CodeWarrior environment.

6.3.7.1 Writing your Application Code

All of your application code must reside in the **Sources** folder in your project directory. You may modify the code in **main.c** and **Events.c**, but retain the original comments related to usage directions.

6.3.7.2 Adding Component Methods

To add a component method into your application source code:

1. In the **Components** panel for your project, click on **Components**. Find the method you wish to add to your code.
2. Drag and drop the method directly into the source code panel
3. Add the appropriate parameters to the method. (Hovering your mouse over the method displays a a list of the required parameters.)

For example, you can open the MVHBridge component method list, drag and drop **RotateProportional** to **main.c** and add the necessary parameters (see Figure 32).

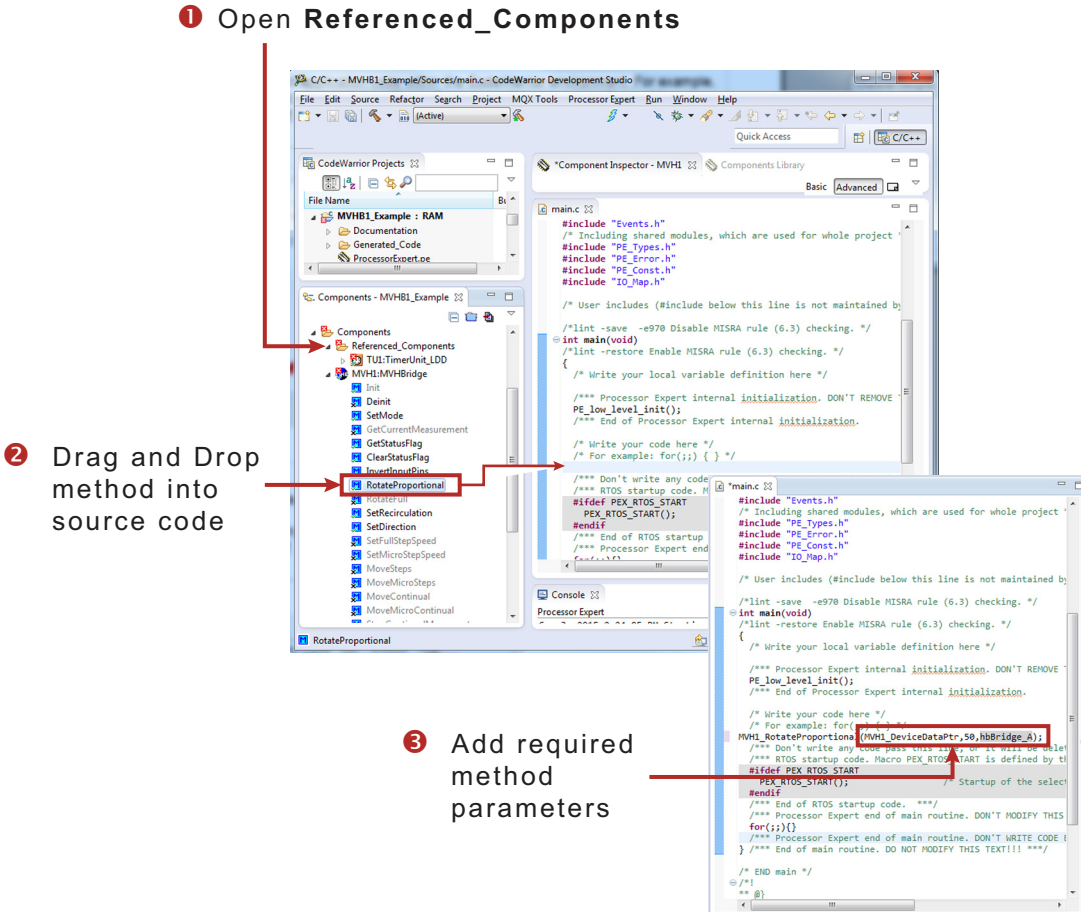


Figure 32. Adding Component Methods

6.3.7.3 Finding Descriptions of the MVHBridge Methods

Hovering your mouse over any of the methods displays a description of the method, including a list of required parameter. See Figure 33.

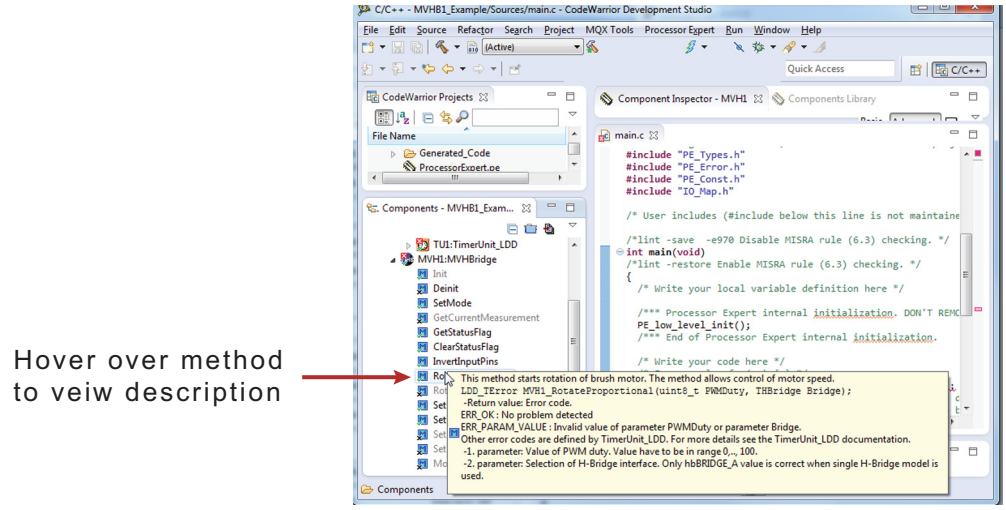


Figure 33. MVHBridge Method Descriptions

6.3.7.4 Jumping into Function Source Code

CodeWarrior is based on the Eclipse IDE which allows you to jump directly into the source code of a function from within the main routine while you are editing. To do so, move your mouse cursor over the function name and click. The source code appears in the edit window.

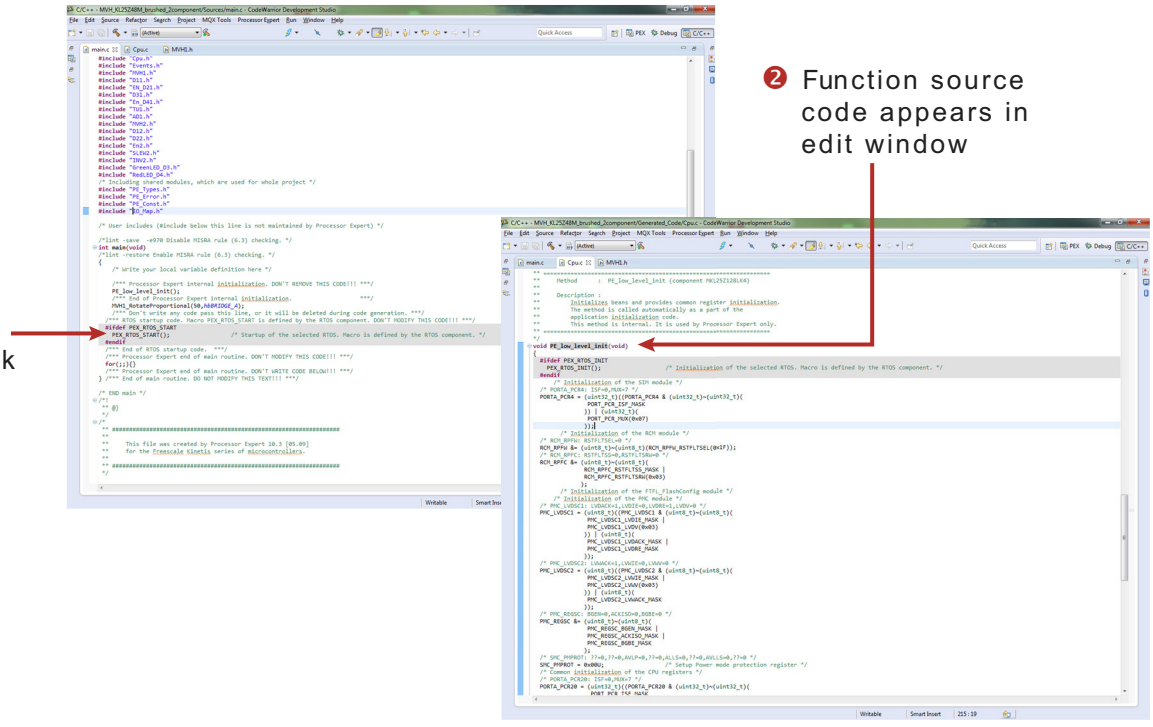


Figure 34. Jumping into a Function's Source Code

6.3.7.5 Compiling, Downloading and Debugging

To compile, download and debug on board, click compile, then click the debug icon in the toolbar. CodeWarrior will download and launch the program on board (see Figure 35).

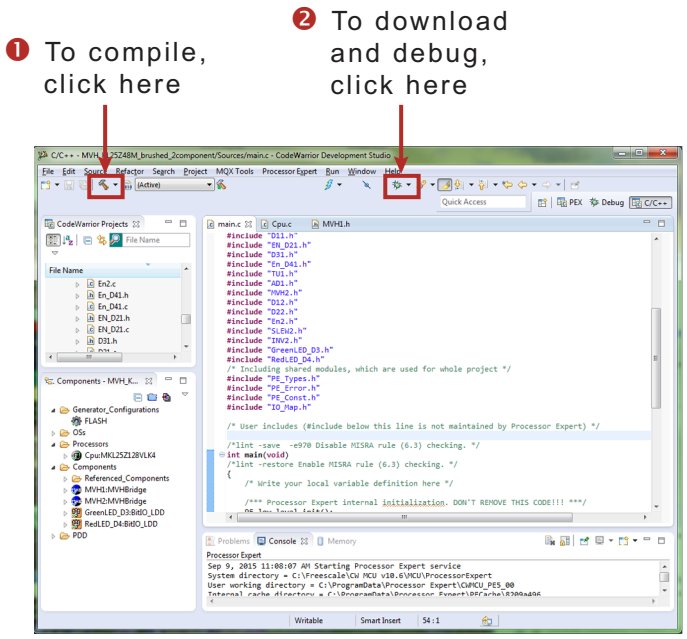


Figure 35. Compiling and Downloading the Application

6.4 Stepper Motor Control Application Notes

The MVHBridge component 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 can be achieved by using PWM to control coil current (open loop control). This method is called micro-stepping (available in the MVHBridge component.)

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 MVHBridge component was tested with a core clock frequency ranging from 20 MHz (minimal value) to 120 MHz.
- Do not change the settings of the timer device (**TimerUnit_LDD**) linked by the MVHBridge component. The component sets the timer device automatically.
- 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 method **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 36. IN1 through IN4 are the input pins of the H-Bridge device which control H-Bridge outputs. These pins input to the stepper motor. You must connect the stepper motor to output pins OUT1-OUT4 and select control input pins on your MCU in the component settings.
- The FTM or TPM timer device is needed by stepper control logic.
- The **AlignRotor** method affects the position of the motor. This method executes four full-steps. It is available only when full-step mode is enabled.

6.4.1 Full-step Control Mode

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

As mentioned in Setting up a Project to Control a Stepper Motor on page 31, you can 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 36. The voltage levels applied to the coils of the stepper motor are depicted in Figure 37. Note that the voltage is applied to both coils at the same time.

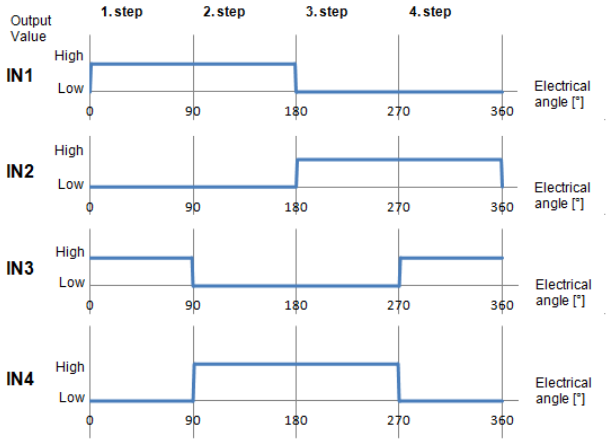


Figure 36. Signals of Logic Input Pins Generated by the MCU in Full-step Mode

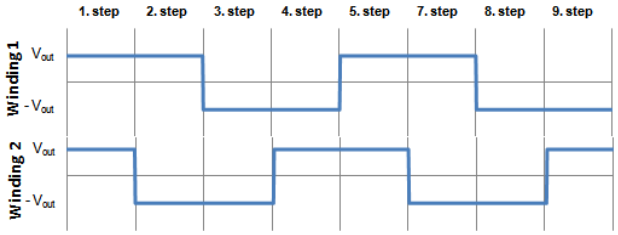


Figure 37. Output of the H-Bridge Device in Full-step Mode

6.4.2 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 MVHBridge component offers 2, 4, 8, 16 and 32 micro-steps per full-step. The micro-step size is defined by the property **Micro-steps per Step** and can be changed later in C code.

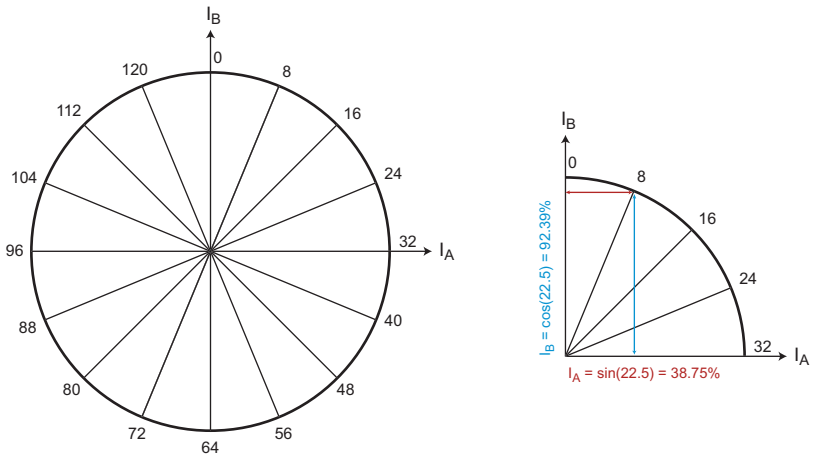


Figure 38. Micro-stepping Phase Diagram

The micro-stepping signal is generated using four timer channels (see Figure 39). Output from logic analyzer in Figure 40 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 41.

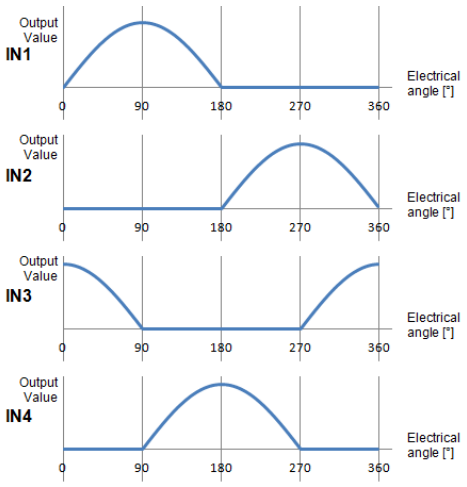


Figure 39. Logic Input Pin Signals Generated by the MCU in Micro-step Mode

Figure 40. Logic Analyzer output

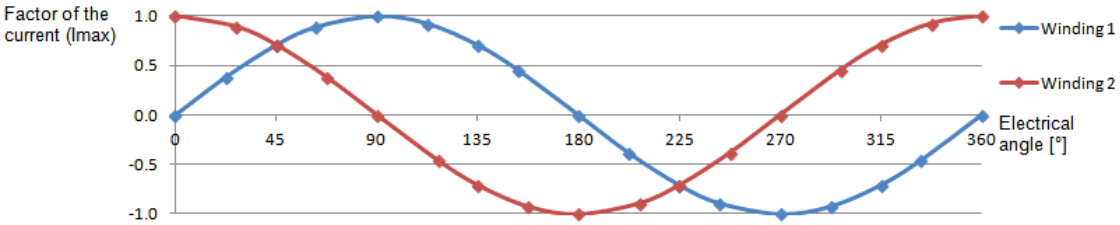


Figure 41. H-Bridge device output in Micro-step mode

6.5 Frequently Asked Questions

Q: Why do I occasionally unexpected behavior in my stepper or DC brushed motor?

A: Check the value of the signals on the enable and disable pins (D1, EN/D2, D3, EN/D4). These signals affect the H-Bridge device mode. To provide a wider range of MCU compatibility, some pins are wired to more than one MCU board pin using 0 Ω resistors. Check your schematic and remove resistors as needed to disconnect unused pins.

Q: How do I set up the MVHBridge component when two or more components with conflicting values are configured to control brushed motors? (See Figure 42)

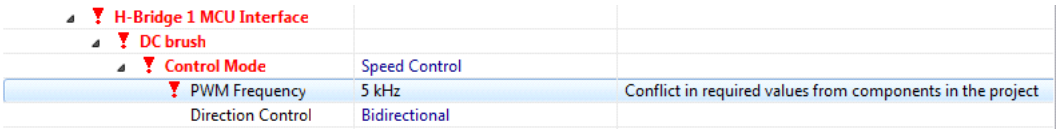


Figure 42. Conflict in the Required Values for Components in the Project

A: You can use more than one MVHBridge components in same project. These components can share the same timer device in brushed motor control mode, but the **PWM Frequency** and **Timer Device** properties must conform in all of the components.

Q: Can I use both a stepper motor and a brushed DC motor on a single timer?

A: The stepper motor control needs a dedicated timer because the timer period can be dynamically changed. Using a stepper motor and a brushed DC motor on the same timer pins is possible only when the **Control Mode** property of the brushed DC motor is set to **State Control**.

Q: The **TimerUnit_LDD** component used by MVHBridge is not set properly and shows some errors.

A: The reason may be that the **TimerUnit_LDD** component channels are not allocated correctly. Change some MVHBridge component properties to force channel allocations. If you are configuring a stepper motor (**Motor Control** property set to **Stepper**), change the Output Control property to **GPIO** then back to **PWM**. For brushed DC motors (**Motor Control** property set to **Brushed**), change the **Control Mode** property to **State Control** and back to **Speed Control** on interface A or interface B.

Q: I sometimes get the following unexpected error while generating Processor Expert code: "Generator: FAILURE: Unexpected status of script: Drivers\Kinetis\TimerUnit_LDD.drv, please contact Freescale support". What causes this?

A: Occasionally, when you enable the MVHBridge component in your project, the **TimerUnit_LDD** component channels have not been allocated. If this occurs, changing certain MVHBridge properties will force allocation of the channels. If you are configuring a stepper motor (**Motor Control** property set to **Stepper**), try changing the **Output Control** property to **GPIO** and then back to **PWM**. If you are configuring a brushed motor (**Motor Control** property set to **Brushed**), change the **Control Mode** property to **State Control** and then back to **Speed Control** on interface 1 or interface 2.

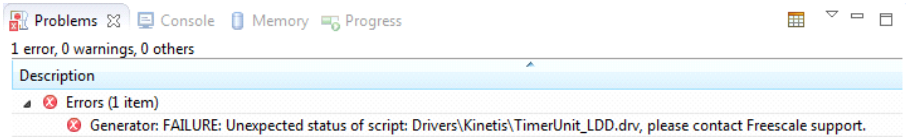


Figure 43. Unexpected error related to the MVHBridge TimerUnit_LDD component

Q: I have set up several CPU clock configurations (via the Clock configurations property of the CPU component). Sometimes during runtime, when I switch between these configuration (using the CPU **SetClockConfiguration** method), the speed of the stepper motor appears to be inaccurate. Why does this occur?

A: Switching to a different configuration results in the use of a different input frequency by a timer device. The MVHBridge component may not pick up the new value and continues to use the previous value in its calculations.

7 Schematic

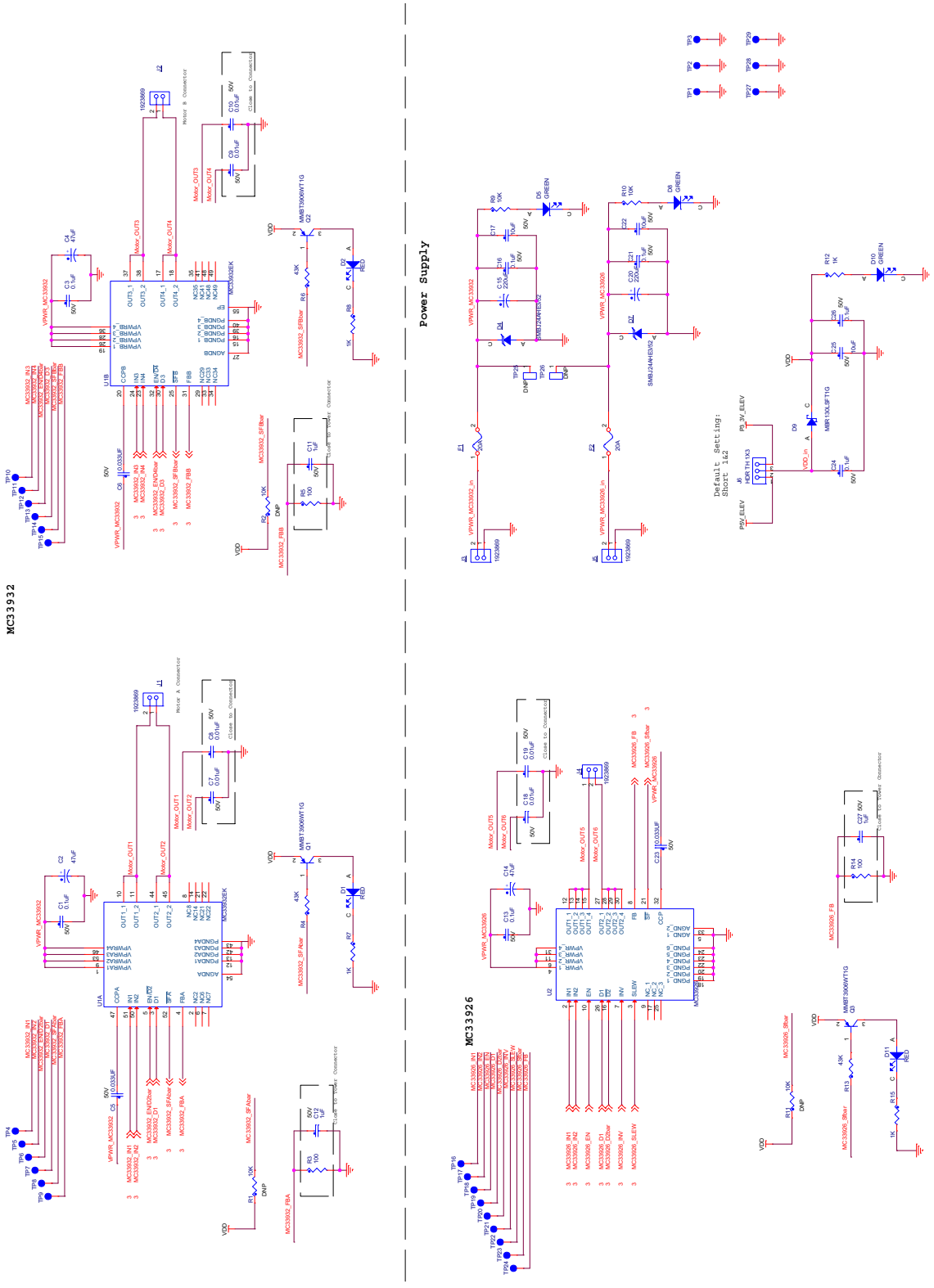


Figure 44. Evaluation Board Schematic, Part 1

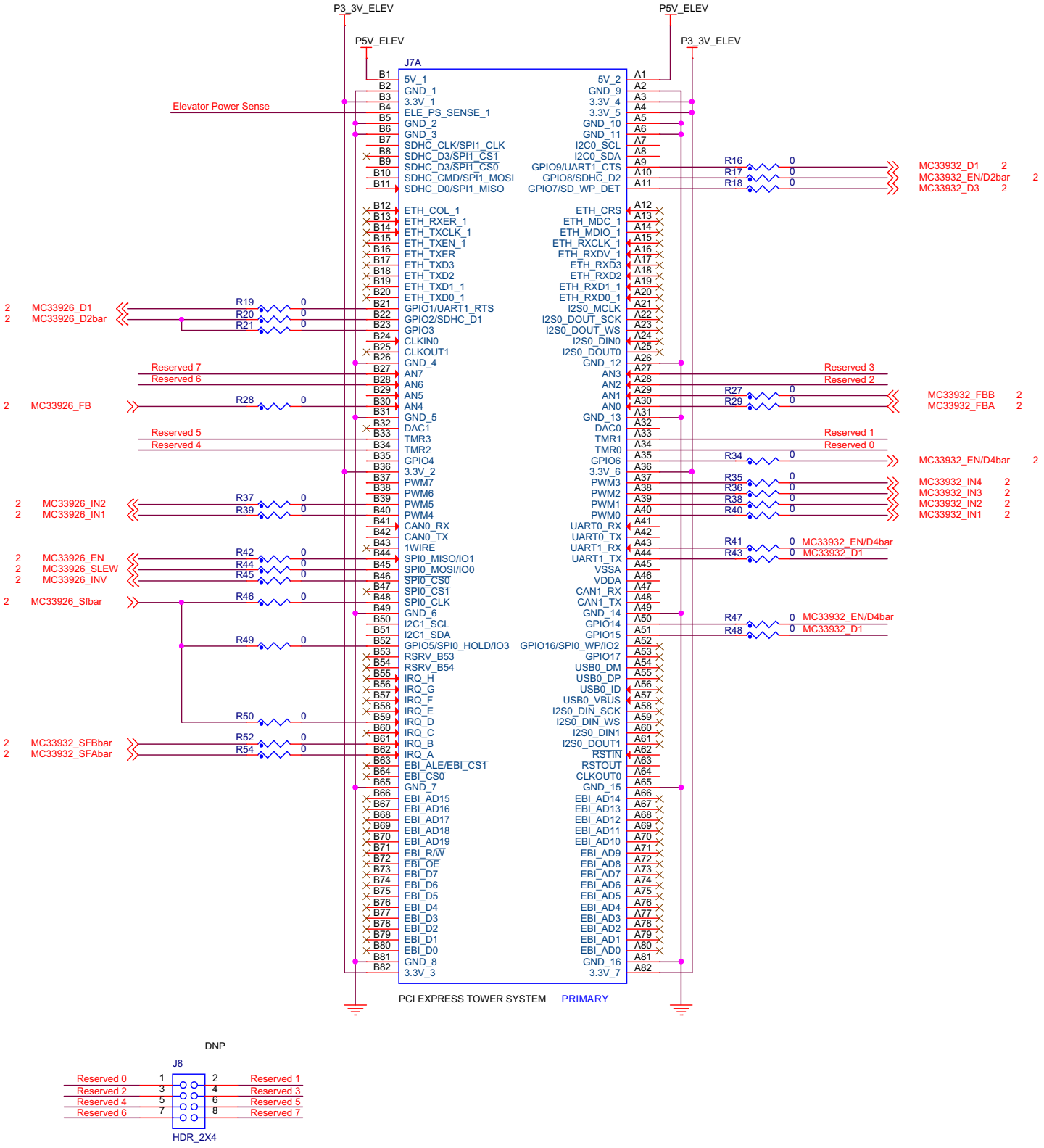


Figure 45. Evaluation Board Schematic, Part 2

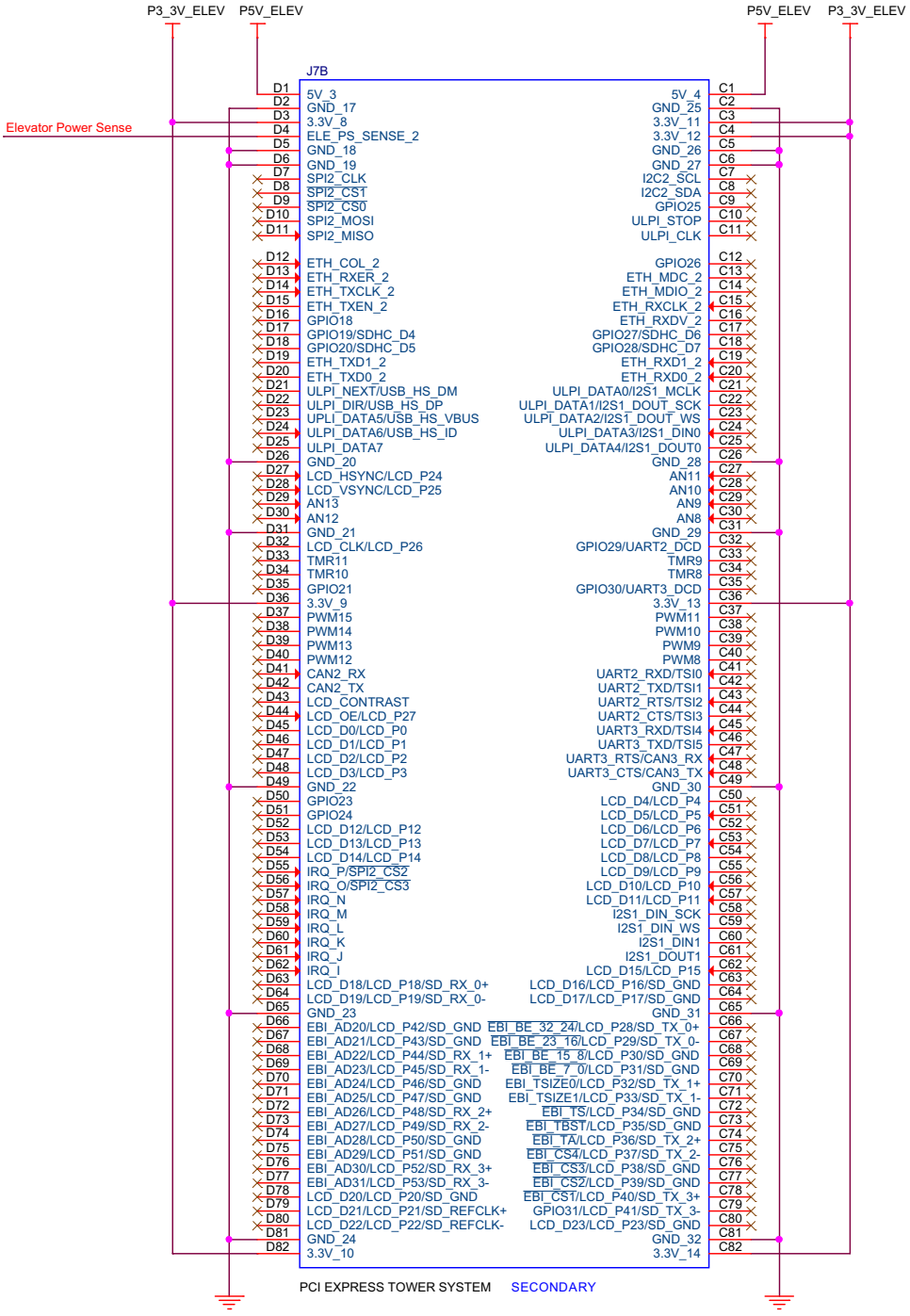
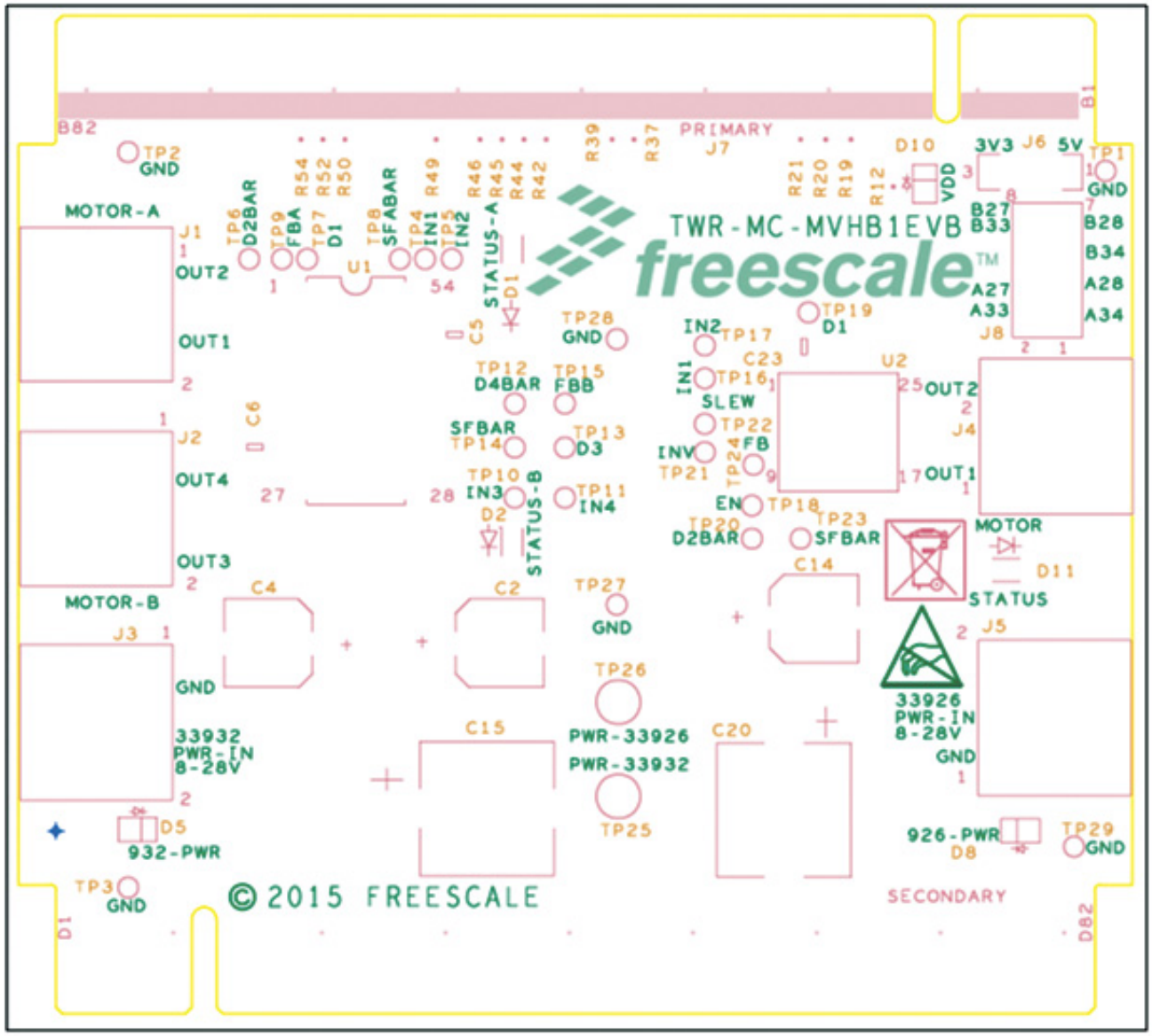


Figure 46. Evaluation Board Schematic, Part 3

8 Board Layout

8.1 Silkscreen



9 Board Bill of Materials

Table 12. Bill of Materials [3]

Item	Qty	Schematic Label	Value	Description	Part Number	Assy Opt
Freescale Components						
1	1	U1		IC THROTTLE CONTROL DUAL H-BRIDGE 8-28 V SOIC54-EP	MC33932EK	
2	1	U2		IC THROTTLE CONTROL H-BRIDGE 8.0-28 V PQFN32	MC33926PNB	
Diodes & Transistors						
3	3	D1,D2,D11	RED	LED RED CLEAR SGL 30MA 0805	LTST-C170EKT	
4	2	D4,D7		DIODE TVS UNIDIR 600 W 24 V AEC-Q101 SMB	SMBJ24AHE3/52	
5	3	D5,D8,D10	GREEN	LED GRN SGL 30MA SMT 0805	LTST-C171KGKT	
6	1	D9		DIODE SCH PWR RECT 1 A 30 V SOD-123	MBR130LSFT1G	
7	3	Q1-Q3		TRAN PNP GEN AMP 200 mA 40 V AEC-Q101 SC70	MMBT3906WT1G	
Capacitors						
8	7	C1,C3,C13,C16,C21,C24,C26	0.1 µF	CAP CER 0.1 µF 50 V 10% X7R 0805	CC0805KRX7R9BB104	
9	3	C11,C12,C27	1 µF	CAP CER 1 µF 50 V 10% X7R AEC-Q200 0805	GCM21BR71H105KA03	
10	2	C15,C20	220 µF	CAP ALUM 220 µF 50 V 20% SMD 12.5MMX13MM	MAL214099111E3	
11	3	C17,C22,C25	10 µF	CAP CER 10 µF 50 V 10% X7S AEC-Q200 1210	GCM32EC71H106KA03	
12	3	C2,C4,C14	47 µF	CAP ALEL 47 µF 50 V 20% AEC-Q200 SMD	MAL214699101E3	
13	3	C5,C6,C23	0.033 µF	CAP CER 0.033 µF 50 V 10% X7R 0805	08055C333KAT2A	
14	6	C7-C10,C18,C19	0.01 µF	CAP CER 0.01 µF 50 V 10% X8R 0603	C1608X8R1H103K	
Resistors						
15	3	R1,R2,R11	10 KΩ	RES MF 10 KΩ 1/10 W 5% AEC-Q200 0603	ERJ-3GEYJ103V	[4]
16	28	R16-R21,R27-R29,R34-R50,R52, R54	0 Ω	RES MF ZERO Ω 1/10 W -- 0603	CRCW06030000Z0EA	
17	3	R3,R5,R14	100 Ω	RES MF 100 Ω 1/10 W 1% 0603	RC0603FR-07100RL	
18	3	R4,R6,R13	43 KΩ	RES MF 43 KΩ 1/10 W 1% 0603	RK73H1JTTD4302F	
19	4	R7,R8,R12,R15	1 KΩ	RES MF 1 KΩ 1/10 W 1% 0603	AR03FTNX1001	
20	2	R9,R10	10 KΩ	RES MF 10 KΩ 1/10 W 5% AEC-Q200 0603	ERJ-3GEYJ103V	
Switches, Connectors, Jumpers and Test Points						
21	2	F1,F2	20 A	FUSE CERAMIC FAST 20 A 32 V 1206	0501020.WRS	
22	5	J1-J5		CON 1X2 SHRD RA TH 5.08 MM 16A SP 339H SN 137L	1923869	
23	1	J6		HDR 1X3 TH 100 MIL SP 339H AU 100L	TSW-103-07-G-S	
24	1	J7		CON DUAL 2X82 Edge PCI Express SMT 1MM SP 591H FOR TOWER SYSTEM NOT A PART TO ORDER	EDGE PCI EXPRESS 164	
25	1	J8		HDR 2X4 TH 100 MIL CTR 330H AU 100L	TSW-104-07-G-D	[4]
26	27	TP1-TP24,TP27-TP29		TEST POINT PAD 40 MIL DIA SMT, NO PART TO ORDER		
27	2	TP25,TP26		TEST POINT BLK 70X220 MIL TH	5006	[4]

Notes

3. Freescale does not assume liability, endorse, or warrant components from external manufacturers are referenced in circuit drawings or tables. While Freescale offers component recommendations in this configuration, it is the customer's responsibility to validate their application.
4. Do not populate

10 References

Freescale.com Support Pages	Description	URL
TWR-MC-MVHB1EVB	Tool Summary Page	www.freescale.com/TWR-MC-MVHB1EVB
MC33932	Product Summary Page	www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MC33932
MC33926	Product Summary Page	www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MC33926
Processor Expert	Tool Summary Page	www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MVHBRIDGE-PEXPERT
CodeWarrior	Tool Summary Page	www.freescale.com/webapp/sps/site/homepage.jsp?code=CW_HOME&tid=vanCODEWARRIOR
Processor Expert Code Model	Code Walkthrough Video	www.freescale.com/video/processor-expert-code-model-codewarrior-code-walkthrough:PROEXPCODMODCW_VID

10.1 Support

Visit www.freescale.com/support for a list of phone numbers within your region.

10.2 Warranty

Visit www.freescale.com/warranty to submit a request for tool warranty.

11 Revision History

Revision	Date	Description of Changes
1.0	7/2015	• Initial Release
2.0	9/2015	• Added Processor Expert section
	9/2015	• Fixed invalid Section reference • Added Processor Expert, CodeWarrior, Kinetis to tradmark citations in last page

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: KTTWRMCMVHB1EVBUG
Rev. 2.0
9/2015