Getting started with the AEK-MOT-WINH92 window lift DC board driver based on the L99H92 gate driver and an external MCU

Introduction

Most modern vehicles feature electric window lifters controlled using buttons in door control panels.

These electric window lifters are fitted with convenience functions that include automatic full up or down of the window lifter when the switch is pressed once, link of the window lifter to the central locking so that, when the car is locked, all windows close automatically, and safety functions, such as antipinch.

The antipinch safety function senses if a motor is running against an obstacle and prevents any injuries to people/objects or damage to the motor itself. When a resistance is registered during closing, the movement of the window is reversed automatically so that it reopens immediately.

The antipinch sensing is usually performed by current monitoring, Hall sensors or a combination of both. This safety system checks continuously the variation of the current and the variation of the velocity. Both are compared with a threshold and if they simultaneously report the presence of an obstacle, the motor is blocked.

Our AEK-MOT-WINH92 evaluation board key objective is to drive a DC motor for car window lifters, ensuring high safety levels.

This board perfectly fits in the automotive market trend, which goes towards the evolution of the window lift application, offering a new antipinch mechanism without leveraging on motor encoders but exploiting the board three different types of current sensing (in-line, low-side, and high-side).

The board can be configured to work with a single H-bridge for a bidirectional DC motor or two independent half-bridges for two unidirectional DC motors.

Thanks to its compact size, the board can be connected to any microcontroller easily, exploiting separate and dedicated connectors for SPI communication, current sensing, and basic motor commands interface with fault detection capabilities. Another key feature is the adjustable gain for antipinch and for fulfilling any of the car window operative conditions (the current range levels could change when opening the window with adverse weather conditions, for example in case of ice).

The board also offers three diagnostic features to detect potential short-to-ground, short-to-battery, and open load conditions when the device is in the *off* state.

In case of failures, the DIAG pin alerts the external MCU, ensuring a prompt intervention in case of open-load, short-to-ground, and short-to-battery, thermal warning and overtemperature shutdown, under/overvoltage and overcurrent protection. For system debugging during the development phase, a fail-safe button was added to the board layout.

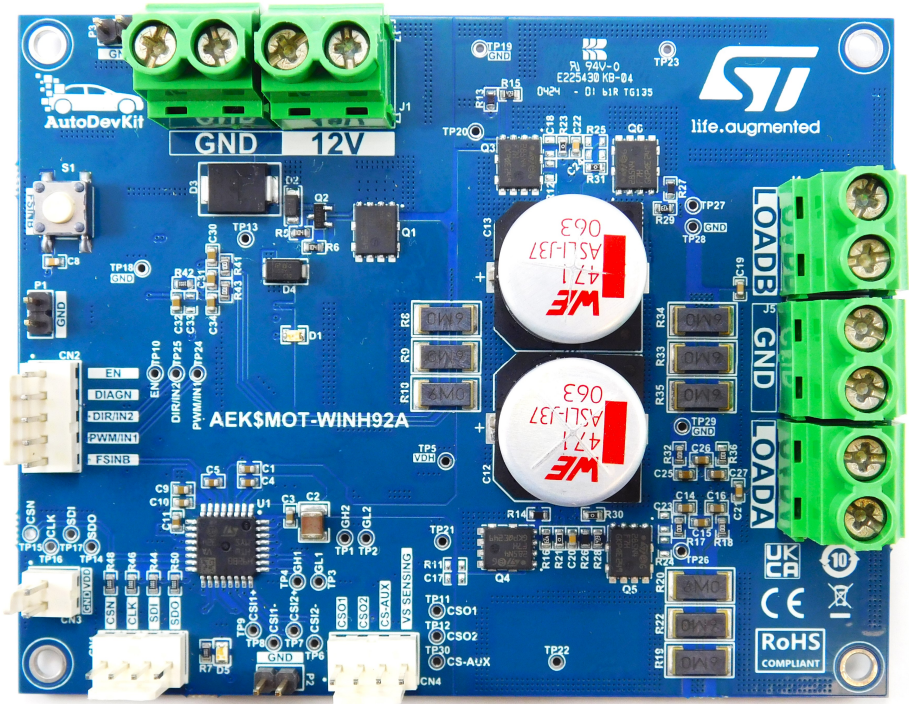
To allow the board robust protection and prevent damage due to the inversion of the power supply polarity, reverse polarity protection has been implemented.

Four external MOSFETs allow current flowing up to 50 A, in line with the board power dissipation capabilities.

The board design is based on the L99H92 H-bridge gate driver for automotive applications, which features flexible current sensing to offer different tools for advanced antipinching algorithm development. For systems requiring a higher safety level, you can enable a watchdog with a configurable time window.

Warning: *The AEK-MOT-WINH92 evaluation board has not to be used in a vehicle as it is designed for R&D laboratory use only.*

Figure 1. **AEK-MOT-WINH92 evaluation board**



1 Hardware overview

1.1 Board main components

1. 12 V DC power supply connector
2. Connector for DC motors
3. Current sensing network based on the internal operational amplifier of the L99H92 driver
4. External current sensing network based on the TSC103
5. Fail-safe input negative button
6. R_{SENSE} for the current sensing networks
7. Connector for the L99H92 driver control
8. Connector for the supply of the L99H92 I/Os and current sense amplifier output stage
9. SPI connector
10. Sensing (current sensing and supply sensing) connector
11. L99H92 driver
12. STL285N4F7AG external MOSFET for full-bridge/half-bridge configuration
13. TSC103 operational amplifier for the high-side current sensing
14. External reverse battery protection

Figure 2. AEK-MOT-WINH92 evaluation board, top view: main components

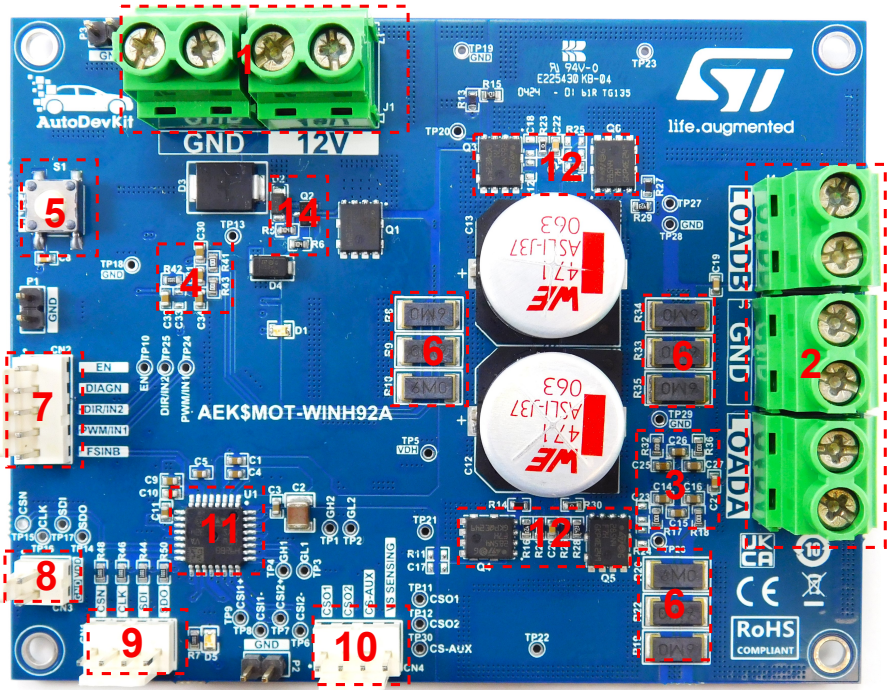
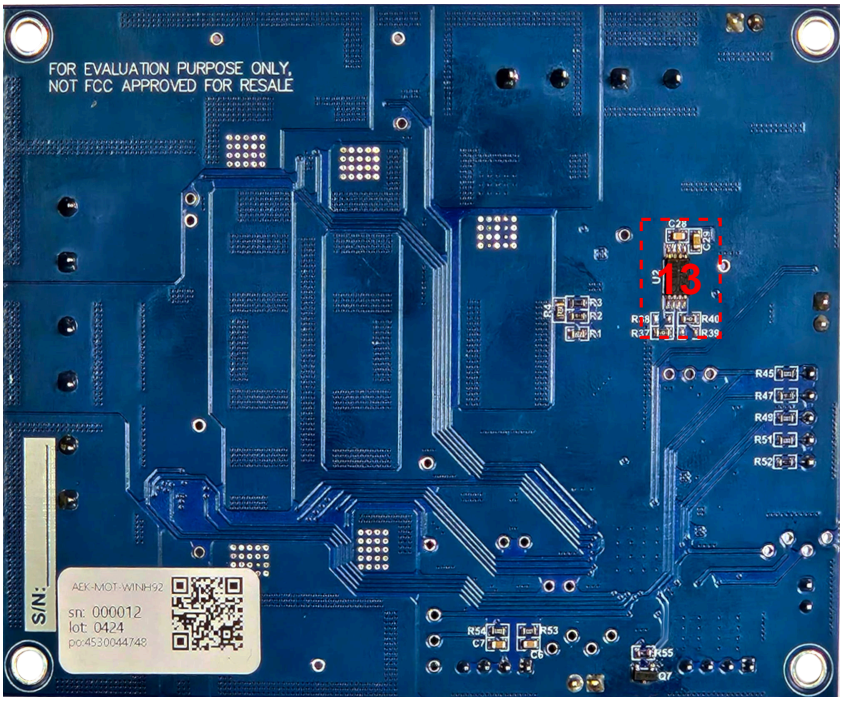


Figure 3. **AEK-MOT-WINH92 evaluation board, bottom view: main components**



1.2 L99H92

The L99H92 is a gate driver, which is programmed through a microcontroller. It is designed to drive four external N-channel MOSFET transistors in a single H-bridge or dual independent half-bridge configuration for DC motor control in automotive applications. Moreover, two independent current sense amplifiers are available, used to detect the current in in-line and low-side parts.

The L99H92 features available in the AEK-MOT-WINH92 are:

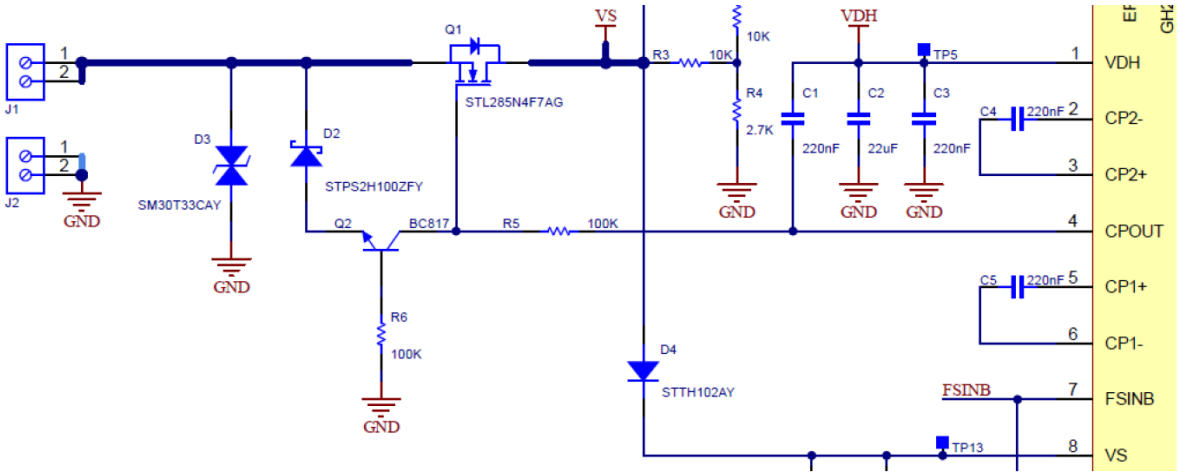
- A dual half-bridge driver used to control four external N-channel MOSFET transistors
- Two independent current sense amplifiers
  - Independently programmable gain (10x, 20x, 50x, 100x)
  - Offset on operational amplifier output centered at VDD/2 or VDD/22
- All the actuator outputs come with the following protection and supervisor features:
  - Current monitoring (in-line and low side only)
  - Open-load, short-to-ground, and short-to-battery conditions
  - Thermal warning and overtemperature shutdown
  - Under/overvoltage protections
  - Overcurrent protection
  - Charge pump monitoring
- Charge pump output available for driving an external reverse battery MOSFET protection
- Asynchronous and logic independent fail-safe input to switch off all the MOSFETs
- DIAGN output pin, which is faster than SPI communication, to alert the microcontroller of a fault occurred in the device
- Configurable window watchdog

Note: For further information on the L99H92, see the related datasheet.

1.3 External reverse battery protection

The external reverse battery protection represents robust protection for the electrical system. A danger that all electrical systems face is reversed polarity from the power source. This event can be caused by a short-circuit or, more often, by simply switching the power and ground terminals when connecting a power supply. This can lead to several damages if the system is not protected.

Figure 4. Reverse battery protection with BJT and NMOS



In the nominal condition, since the gate voltage is not higher than the source voltage (in this case, VBAT that is usually 12 V) the NMOS cannot turn on (i.e. $V_G > V_S$). To allow conduction of the supply voltage, a charge pump (CPOUT) supplies an additional voltage that ensures a gate voltage greater than the source voltage, switching the NMOS on (i.e. V_G = VDH+12 V), when the charge pump is on. So, we have V_G = VDH+12 V > V_S.

1.3.1 Reverse protection

If the supply voltage is reversed (that is, the polarity of the power source is reversed), the bipolar junction transistor (BJT) puts the NMOS gate to GND. When the power source is correctly connected, the base voltage is lower than the collector-emitter voltage and then the BJT is turned off.

So, in the case of reverse polarity, the power source creates a positive voltage between the emitter and the base. This voltage conducts the BJT that will put the NMOS gate to GND. The D2 diode and the R6 resistor have the function, respectively, to block the current input to the BJT under normal operation and to limit the current at the BJT base.

1.4 Current sensing monitoring networks

In addition to the two internal current sensings of the L99H92, the AEK-MOT-WINH92 hosts another current sensing network. So, there are three different current sensing networks: two of them are internal to the L99H92 driver and one is external to the driver. The latter is implemented through the TSC103 high current sense amplifier. These sensing networks detect the current that the DC motors absorb when connected to the AEK-MOT-WINH92 bridges.

The motor current information can be used for rough control and obstacle detection during the motor actuation.

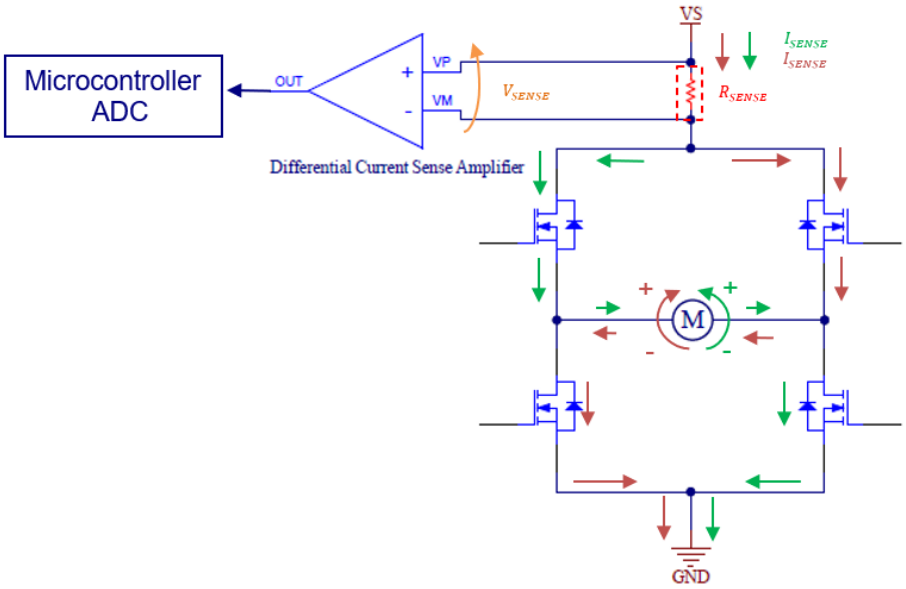
The board features three types of current sensing networks: high, in-line, and low side. The output voltage of the available current sensing networks will be acquired from the microcontroller ADC.

The next paragraphs show the three current sensing networks with the motor polarities that consider the different current directions, represented by arrows of different colors.

1.4.1 High side

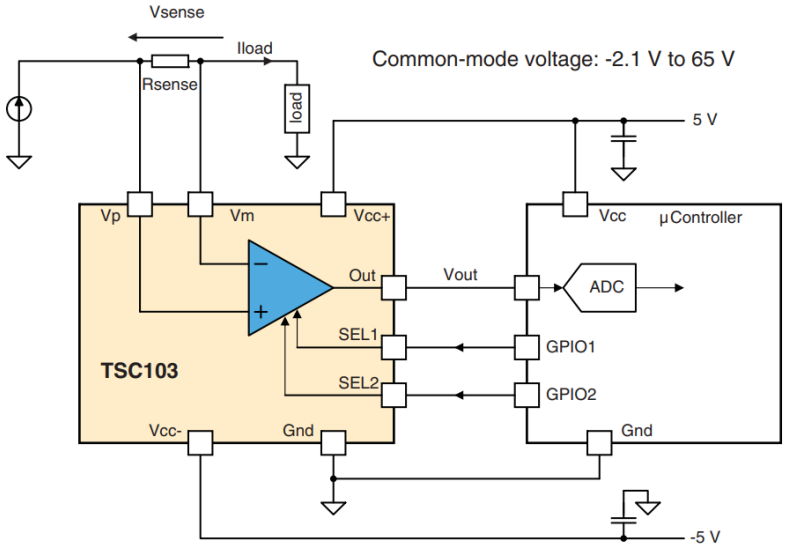
For the high-side part, the current sensing network is based on the high-side current sensing topology, where the sensing resistor is located between the power supply and the load.

Figure 5. External current sense amplifier



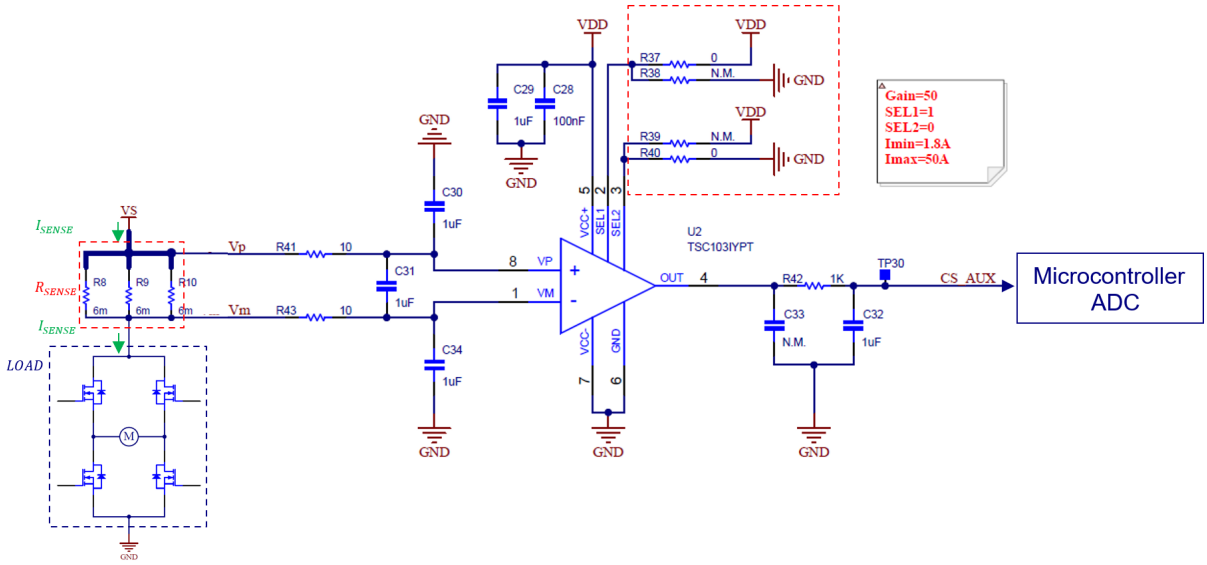
The following figure shows the TSC103 operational amplifier circuitry. It allows measuring a small differential voltage on a high-side shunt resistor and translates it into a ground-referenced output voltage. This voltage permits measuring, with high precision, the current flowing in our application.

Figure 6. TSC103 current sense amplifier



The following figure shows the diagram of the current sensing network implemented, based on the TSC103 current sense amplifier.

Figure 7. TSC103 current sense amplifier



The current amplifier gives the possibility of selecting four different gain levels:

- 20
- 25
- 50
- 100

In this application, the gain is fixed at 50 as a compromise between information accuracy and current absorbed, considering the power the resistors dissipated.

The current sensing network on the board has been developed for a DC motor that can absorb from 2 A up to 50 A.

To get a reliable measurement of the current, we suggest connecting a motor that can absorb at least 2 A.

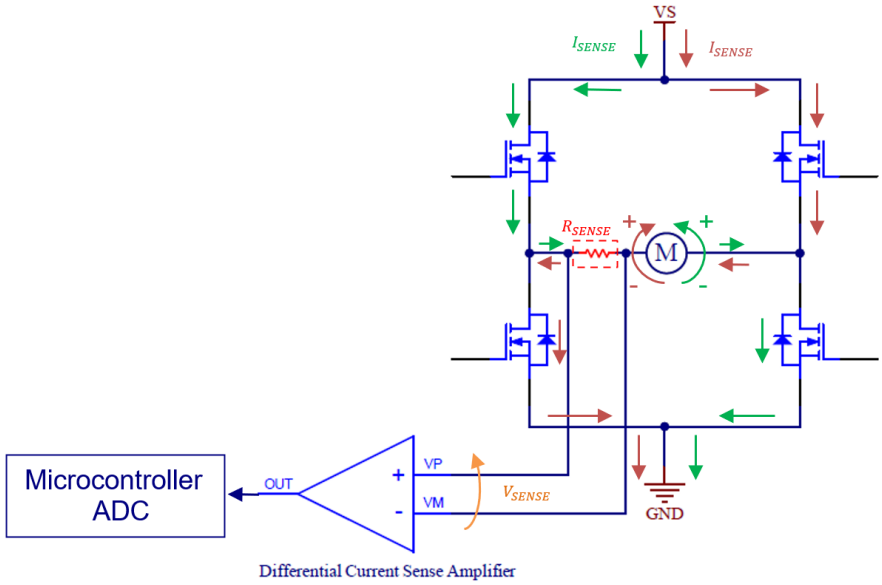
The computed R_{SENSE} soldered on the board is 0.002 Ω.

Note: Section 1.11: How to implement the current sensing network describes how to implement the sensing network according to the chosen motor characteristics.

1.4.2 In-line side

For the in-line side part, the current sensing network is based on the in-line current sensing topology, where the sensing resistor is located between the half-bridge and the load.

Figure 8. Internal current sense amplifier



The current amplifier gives the possibility of selecting four different gain levels:

- 10
- 20
- 50
- 100

In this application, the gain is fixed at 50 as a compromise between information accuracy and current absorbed, considering the power the resistors dissipated.

The current sensing network on the board has been developed for a DC motor that can absorb from 2 A up to 50 A.

To get a reliable measurement of the current, we suggest connecting a motor that can absorb at least 2 A.

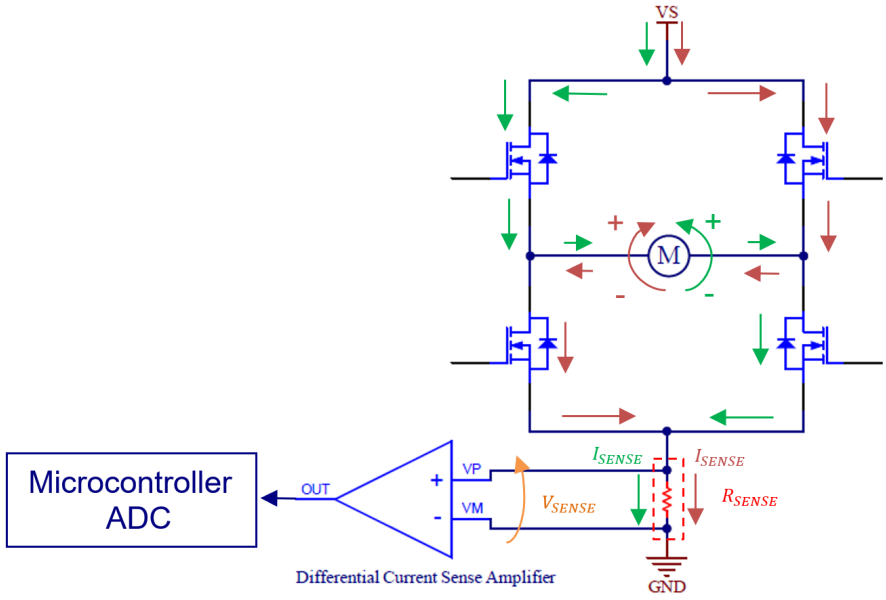
The computed R_{SENSE} soldered on the board is 0.002 Ω.

Note: Section 1.11: How to implement the current sensing network describes how to implement the sensing network according to the chosen motor characteristics.

1.4.3 Low side

For the low-side part, the current sensing network is based on the low current sensing topology, where the sensing resistor is located between the load and the GND.

Figure 9. Internal current sense amplifier



The current amplifier gives the possibility of selecting four different gain levels:

- 10
- 20
- 50
- 100

In this application, the gain is fixed at 50 as a compromise between information accuracy and current absorbed, considering the power the resistors dissipated.

The current sensing network on the board has been developed for a DC motor that can absorb from 2 A up to 50 A.

To get a reliable measurement of the current, we suggest connecting a motor that can absorb at least 2 A.

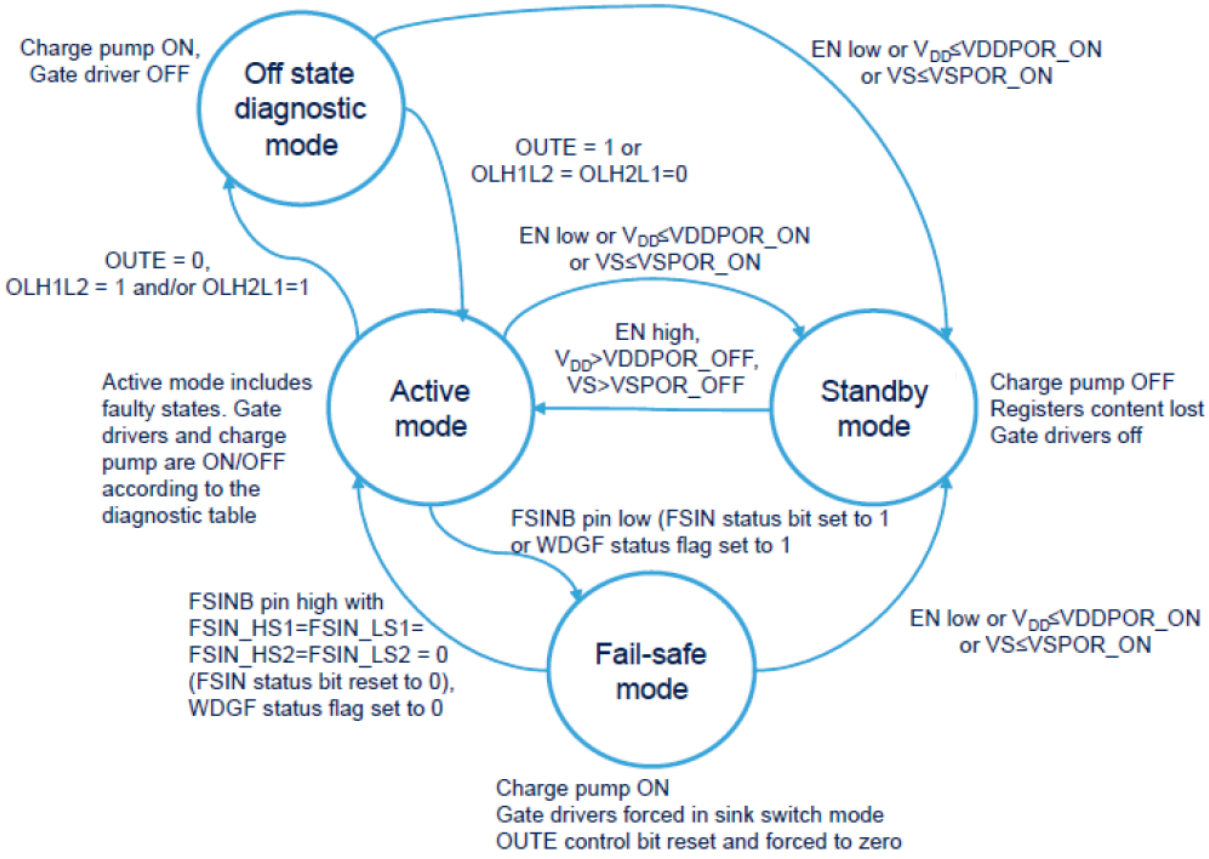
The computed R_{SENSE} soldered on the board is 0.002 Ω.

Note: Section 1.11: How to implement the current sensing network describes how to implement the sensing network according to the chosen motor characteristics.

1.5 L99H92 state machine

The figure below shows the L99H92 finite state machine (FSM).

Figure 10. L99H92 FSM



The main states are:

- **Off state diagnostic mode**: the device enters this state when the OLH1L2 control bit or the OLH2L1 control bit is set to one, while the device OUTE control bit is set to zero and CSA is disabled. To exit from this off-state diagnostic mode, either the OUTE control bit is set to one or both the OLH1L2 and the OLH2L1 are set to zero. In this state, we can detect possible system fault conditions such as open load, short-to-ground, and short-to-battery.
- **Active mode**: to keep the device in the active state, the MCU activates a watchdog that monitors the communication between the microcontroller and the chip. In this state the turned on MOSFETs are controlled through the combination of the PWM/IN1, DIR/IN2 input signals and the INPMODE, AFWE and FWS control bits. In active mode, the device diagnostics is available.
- **Standby mode**: the transition from the active mode to standby mode is enabled/disabled by pulling the EN input pin high/low. Here, the device enters the low power mode when the current consumption is less than 5 µA.
- **Fail-safe mode**: In fail-safe mode, the OUTE control bit is reset, and the gate drivers are forced in sink switch mode to switch off actively all the MOSFETs with the maximum available current.

1.6 Watchdog scheme

The L99H92 state machine performs a state transition from the active mode to the fail-safe mode when the continuity of communication between the device and the microcontroller is lost. This continuity of communication must be guaranteed by writing SPI messages into a special register (WDGTRDIS) that toggles a specific bit.

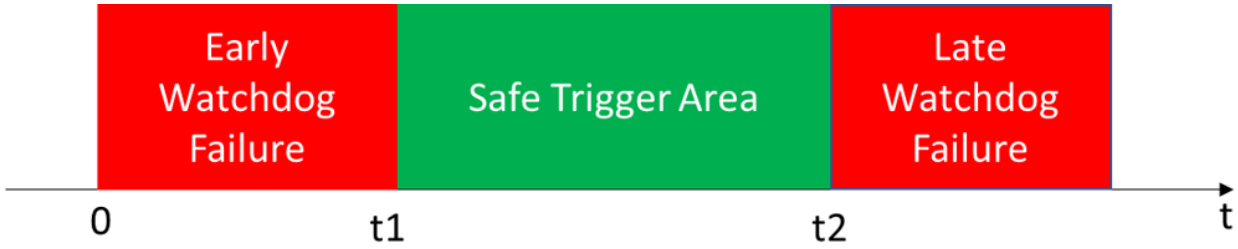
After the power-on or standby mode, the watchdog must start within a maximum timeout (long open window TLW). This window gives the microcontroller the time to run its own setup before starting the watchdog.

From now on, the microcontroller must serve the watchdog within a safe triggering time range. The trigger time window is configurable by SPI.

The watchdog failure happens if the watchdog trigger occurs before t1, in the “early write" window, or after t2, in the "late write" window. In case of watchdog failure, the device sets the WDGF flag, stops the watchdog and puts the device in fail-safe mode.

For debug purposes, it is possible to disable the watchdog by writing a specific key, consisting in two consecutive valid SPI frames to be sent in a specific sequence, to control register WDGTRDIS before the end of any long open window.

Figure 11. Watchdog timing



Note: For further details, refer to the L99H92 documentation on www.st.com.

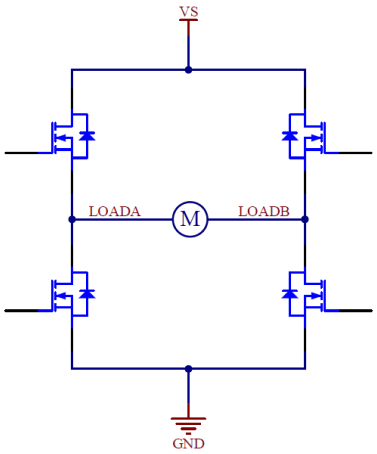
1.7 Board outputs

The AEK-MOT-WINH92 has three outputs for motor drive: LOADA, GND, LOADB.Depending on the value of the INPMODE control bit (DCR register), the device can work as an H-bridge driver or a dual half-bridge driver.

H-BRIDGE

If INPMODE = 0 (default value), the device works in H-bridge mode. In this case, the active H-bridge diagonal, fixing the motor's rotational direction, is selected by DIR/IN2 input while the driving PWM signal must be applied to PWM/IN1 input. Moreover, we have four different freewheeling selectable strategies, thanks to the enable control bit value (AFWE) and the freewheeling selection control bit value (FWS), both available in the DCR register. In the H-bridge configuration, we will have exclusively a connection between LOADA and LOADB.

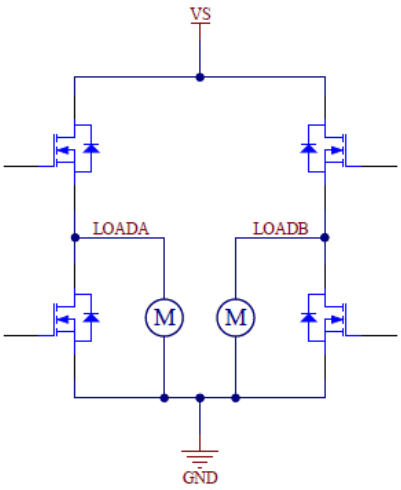
Figure 12. H-bridge mode



HALF-BRIDGE

If INPMODE = 1, the device works in dual half-bridge mode. The two half-bridges can be driven separately by IN1 and IN2 input pins and can be individually disabled through DIS1 and DIS2 control bits (DCR register). In the half-bridge configuration, we will have exclusively a connection between LOADA and GND or LOADB and GND.

Figure 13. Half-bridge mode



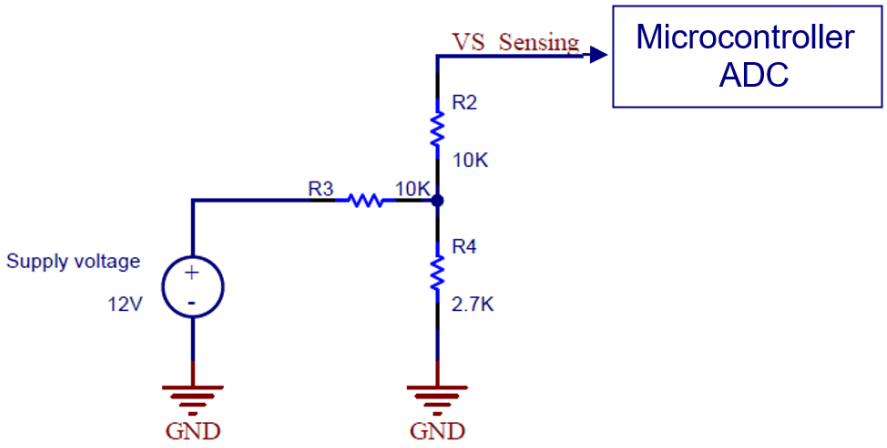
1.8 Output sensing

As specified in Section 1.4: Current sensing monitoring networks, the output currents (LOADA, LOADB, GND) can be acquired from the various current sensing networks available on the AEK-MOT-WINH92 board. These currents will be available in the board outputs (respectively, CSO1, CSO2, and CS-AUX).

After the voltage acquisition, through the *AEK_MOT_WINH92_get_ADC_meas()* callback (see Table 5 for further information), we acquire the current data and, on these outputs, the selected current sensing. The current monitoring can allow determining the motor working state (inrush current, normal, end-limit switch detection, or simple absorption).

Another output sensing is the VS-Sensing output, which allows us to measure the supply voltage of the board as shown in the figure below.

Figure 14. Vs sensing acquisition



The voltage acquired thanks to the callback seen before allows acquiring a precise input voltage.

1.9 Fail-safe input negative button (FSINB)

The board features a fail-safe input negative active button (FSINB) that works as an asynchronous, logic independent fail-safe input. Since the FSINB button is an active low button, and as long as the FSINB input pin is low, the gate drivers are forced in sink switch mode and the OUTE control bit is forced to zero. This means that all the external MOSFETs will be switched off. To reactivate the external MOSFETs, the FSINB should become high and all the FSINHSx and the FSINLSx status bits have to go back to zero (i.e. they should be cleared), and the OUTE control bit should be high. All settings should be done through SPI.

1.10 Diagnostic not output (DIAGN)

The board also features a diagnostic negative active output (DIAGN), which is used to detect device faults, including device power-on-reset event. The purpose of the DIAGN output pin is to warn immediately the microcontroller that a new fault has been detected by the device, avoiding cyclic on-demand failure check via SPI. When a new fault occurs, the blue LED D5 will turn on. All the errors will be mapped in the DIAGCR control register.

1.11 How to implement the current sensing network

The current sensing network implemented in the AEK-MOT-WINH92 is designed to be used with a motor that can manage up to 50 A.

To get an accurate acquisition from the current sensing, we should set the minimum value of the current that we can acquire. Indeed, if zero current flows through the $R_{SENSE,}$ it becomes saturated, and the output is clamped to the V_{OL} (offset on op amp output). Unfortunately, this phenomenon also holds true with a positive V_{OS} (input offset voltage) and low current. In such a case, a voltage drop through the R_{SENSE} (lower than the V_{OS} of the current sensing) gives an incorrect output value.

Through all the formulas below, we can reach the value of the R_{SENSE} resistor in the three sides.

The following steps show how the current sensing network has been dimensioned according to the board requirements (up to 50 A).

Step 1. Define the board requirements (up to 50 A).

Step 2. Check how the ADC reference voltage has been configured in AutoDevKit. In this case, it is 5 V. This means that the voltage range that the ADC can read is 0 V- 5 V.

Step 3. Calculate the R_{SENSE} resistor. In the following formula, V_{OUT} is the maximum value that the ADC can read, GAIN is fixed at 50, and I_{MAX} is the maximum value of the current that the board can manage.

$$R_{SENSE} = \frac{V_{OUT}}{GAIN} \cdot \frac{1}{I_{MAX}} = \frac{5}{50} \cdot \frac{1}{50} = 0.002\Omega$$

According to the formula, and with the constraint that V_{OUT} must not exceed 5 V, the op amp gain has been set to 50. By considering $V_{OS\ MAX}$ and V_{OL}, we can compute the minimum current, which is 227 mV for the internal current sensing and 125 mV for the external current sensing.

Internal

$$I_{MIN} = \frac{\frac{V_{OL}}{GAIN} + V_{OS\ MAX}}{R_{SENSE}} = \frac{\frac{0.227}{50} + 0.003}{0.002} = 3.77A$$

$$I_{MIN} = \frac{\frac{V_{OL}}{GAIN} - V_{OS\ MAX}}{R_{SENSE}} = \frac{\frac{0.227}{50} - 0.003}{0.002} = 0.77A$$

External

$$I_{MIN} = \frac{\frac{V_{OL}}{GAIN} + V_{OS\ MAX}}{R_{SENSE}} = \frac{\frac{0.125}{50} + 0.0011}{0.002} = 1.8A$$

$$I_{MIN} = \frac{\frac{V_{OL}}{GAIN} + V_{OS\ MAX}}{R_{SENSE}} = \frac{\frac{0.125}{50} - 0.0011}{0.002} = 0.7A$$

Considering the variation of V_{OS} and after laboratory tests, we determined that a good value of current to make the current sensing work well is 2 A.
In this manner, the V_{OUT} obtained will be greater than the output stage low-saturation voltage (i.e. $V_{OUT} > V_{OL}$).
Where

$$V_{OUT} = (I \cdot R_{SENSE} \pm V_{OS\ MAX}) \cdot \text{GAIN} > V_{OL}$$

Note: For further details, refer to the L99H92, TSC103, and AN4366 documentation on www.st.com.

1.12 Power dissipation in the AEK-MOT-WINH92 evaluation board

Our application example demonstrates the actual dissipation that occurs in the AEK-MOT-WINH92 when an absorption of high current occurs. For simplicity, consider a resistive load and various temperature zones where the temperature will be acquired.

Figure 15. **Temperature zones**

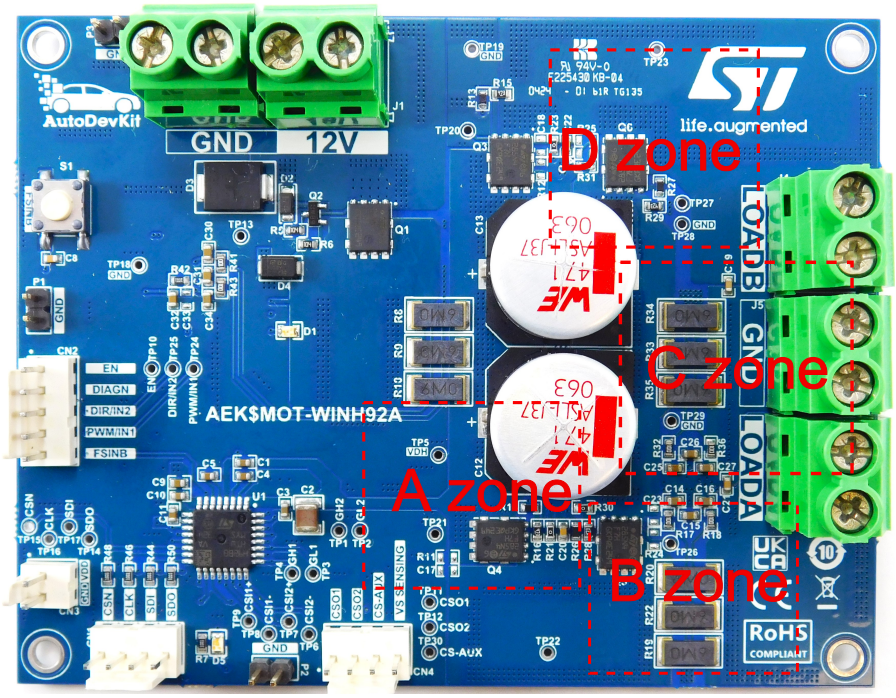
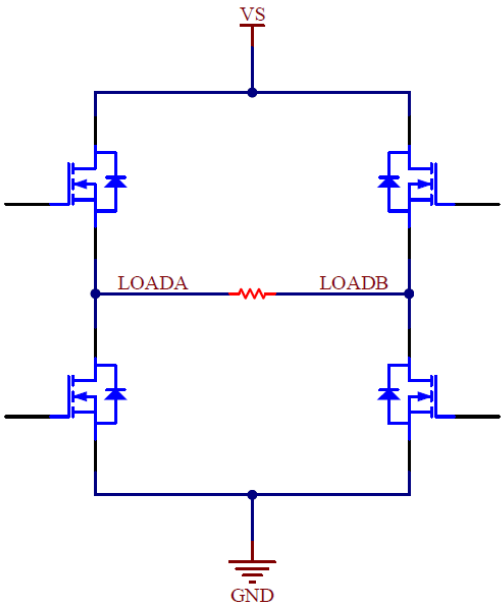


Figure 16. **H-bridge mode with a resistive load**



For the test setup, we attached a resistive load to the board, measuring it in continuous mode for 3 minutes. We used a resistive load with high-power dissipation and several resistances in parallel to achieve a value of the load equal to 0.28 Ω. From an ideal point of view, with a 12 V DC power supply we should obtain:

$$I = \frac{V}{R} = \frac{12}{0.28} = 43.64A$$

However, from our test acquisitions, we got a current of 39.15 A, due to the R_{DSON} of the MOSFETs, the cables connecting the load to the board, and losses of the board itself because of the copper used to supply the high theorized current.

To demonstrate the different heat dissipations that occur on the board and, thus, the different temperatures, we compared two cases with an absorption of ~40 A and an absorption of ~10 A. To measure dissipated temperature, we used a thermal imaging camera. We can observe four zones of dissipated power.

Figure 17. Temperature acquisitions with ~10 A

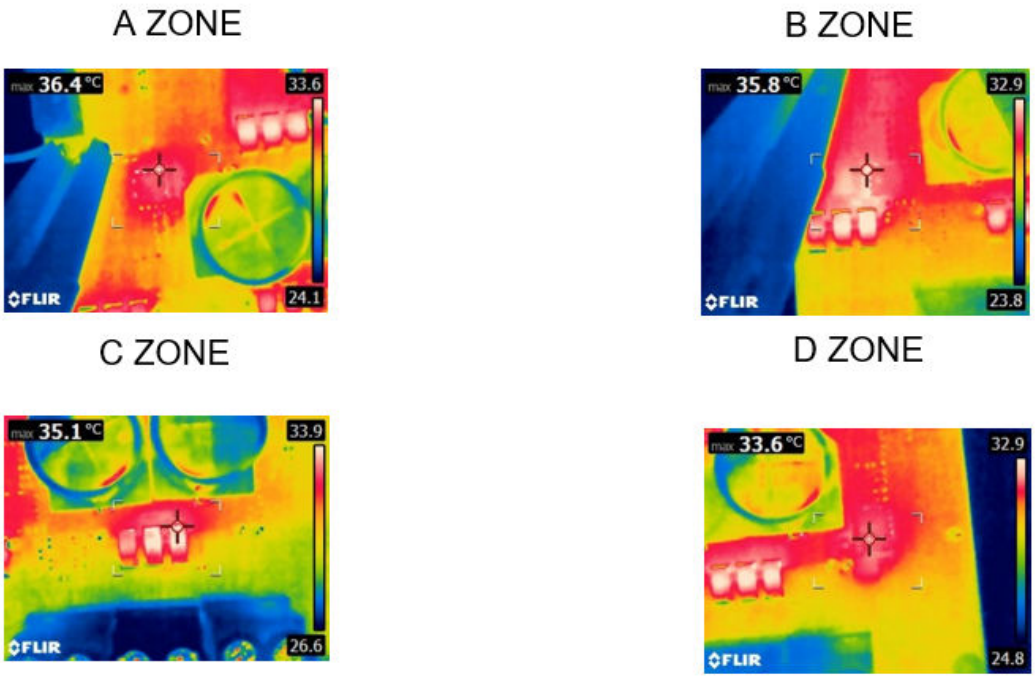
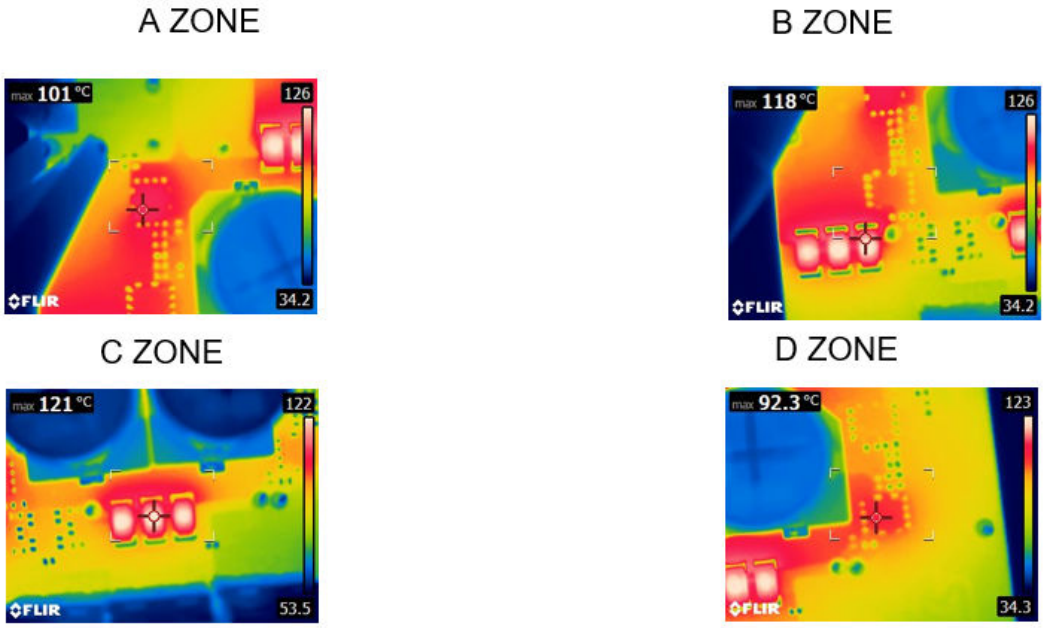


Figure 18. **Temperature acquisitions with ~40 A**



As we can see, the highest value in the reached temperature is present in the power resistance that represents the most critical part to dissipate the heat. Moreover, the only temperature limitation is due to the layout of the board, which respects the specification expected. Indeed, with ~50 A, we do not exceed the internal temperature limitations of the ICs.

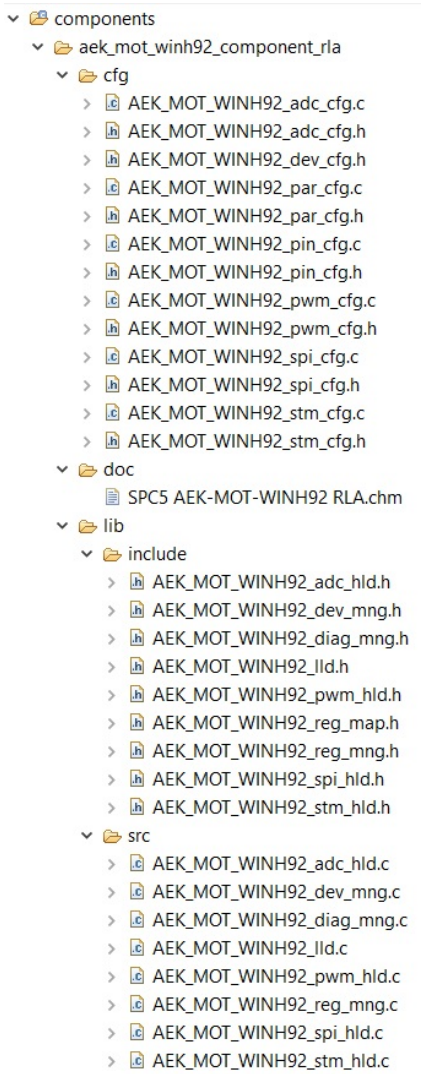
2 AutoDevKit ecosystem

The application development employing the AEK-MOT-WINH92 takes full advantage of the AutoDevKit ecosystem, whose basic components are:

- AutoDevKit Studio IDE (STSW-AUTODEVKIT)
- PLS UDE and OpenOCD programmers and debuggers

2.1 aek_mot-winh92_component_rla folder structure

Figure 19. AEK-MOT-WINH92 component folder structure



- The cfg folder contains all the configuration files generated, based on the GUI.
- The doc folder contains the doxygen documentation.
- The lib folder contains the high-level drivers, low-level drivers, and register map of the AEK-MOT-WINH92.
  - ADC_hld /PWM_hld /SPI_hld and STM_hld files contain peripheral specific high-level drivers
  - The AEK_MOT_WINH92_diag_mng.c contains diagnosis APIs
  - The AEK_MOT_WINH92_lld contains all the APIs:
    - To configure the AEK-MOT-WINH92 working mode
    - To configure driving PWM frequency and duty for the motors
    - To drive motors based on their configuration
    - To configure parameters for current sensing
    - To set parameters related to AFW and VDS thresholds

2.2 Using AEK-MOT-WINH92 in AutoDevKit

In this example, we created an application for the AEK-MOT-WINH92 configured in full-bridge mode. We used the AEK-MCU-C4MLIT1 as microcontroller board.

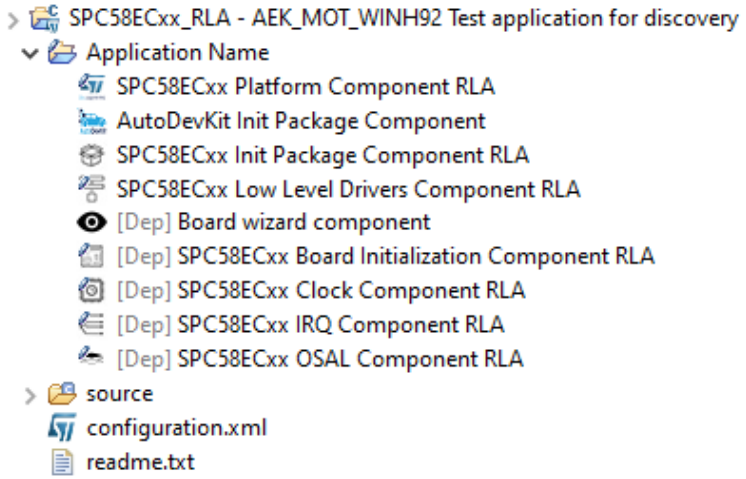
To recreate this scenario, follow the procedure below.

Step 1. Create a new SPC5-STUDIO application for the SPC58EC series microcontroller and add the following components:

- SPC58ECxx Init Package Component RLA
- SPC58ECxx Low Level Drivers Component RLA
- AutoDevKit Init Package Component

These components need to be added immediately, or the other components will not be visible.

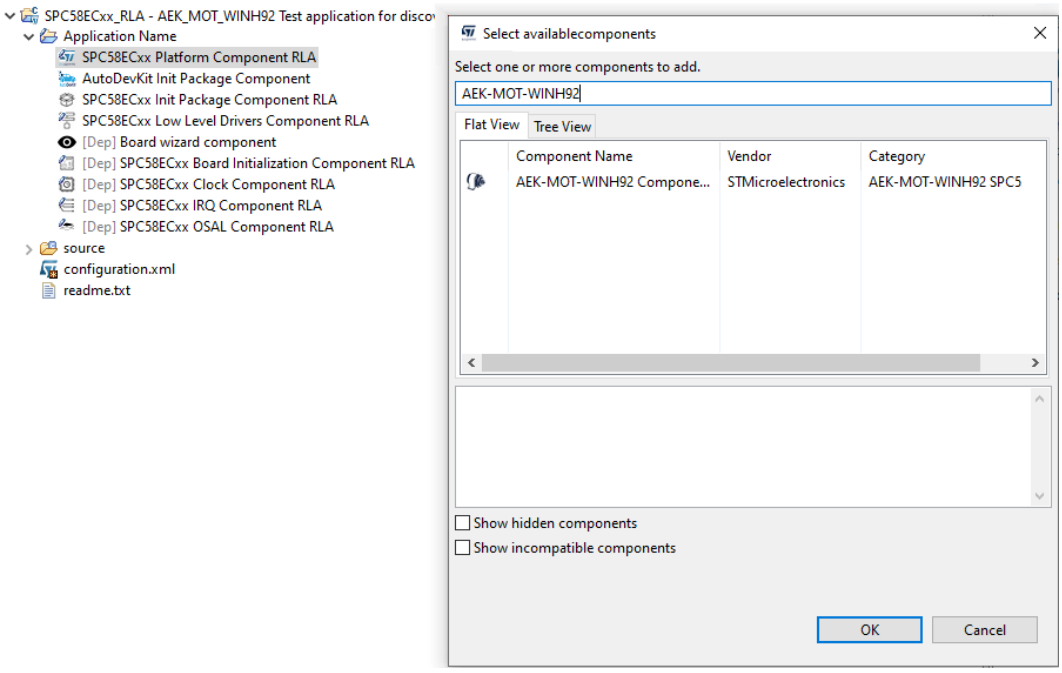
Figure 20. Adding components



Step 2. Add the following additional components:

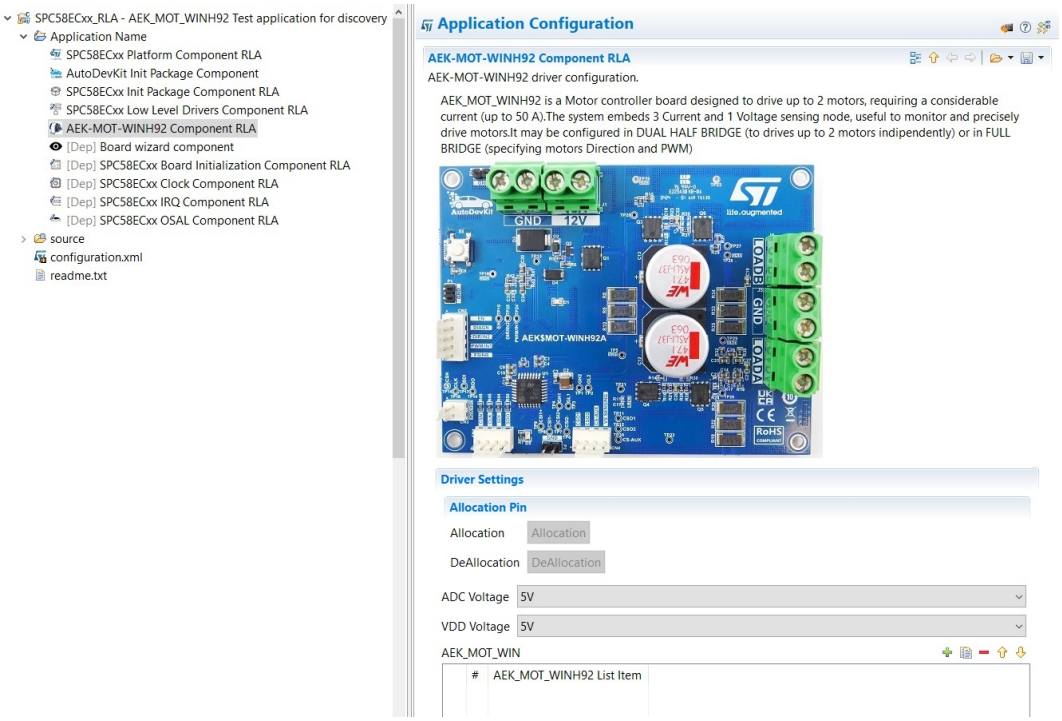
– AEK-MOT-WINH92 Component RLA

Figure 21. Adding AEK-MOT-WINH92 Component RLA



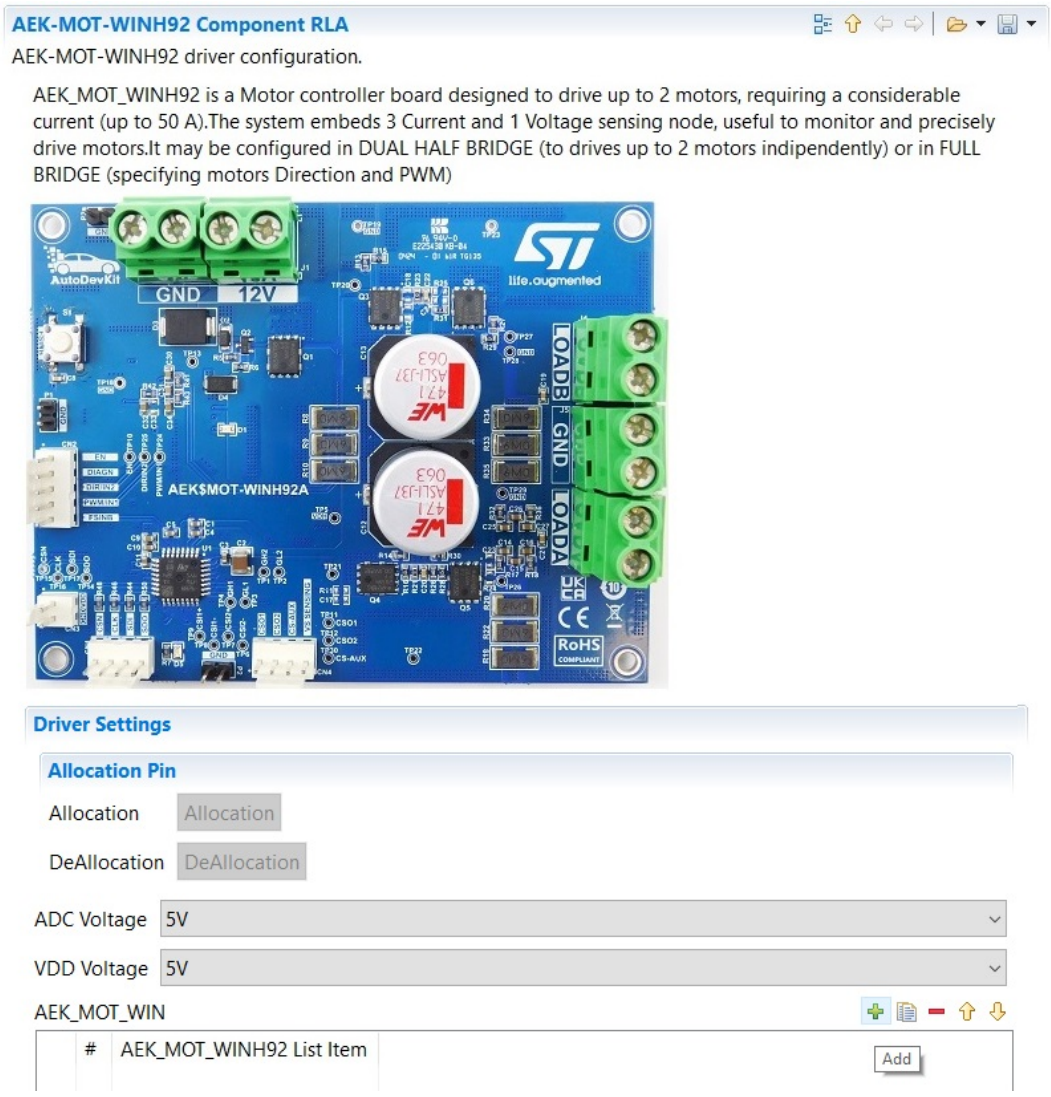
Step 3. Select [AEK-MOT-WINH92 Component RLA] to open the [Application Configuration] window.

Figure 22. Selecting AEK-MOT-WINH92 Component RLA



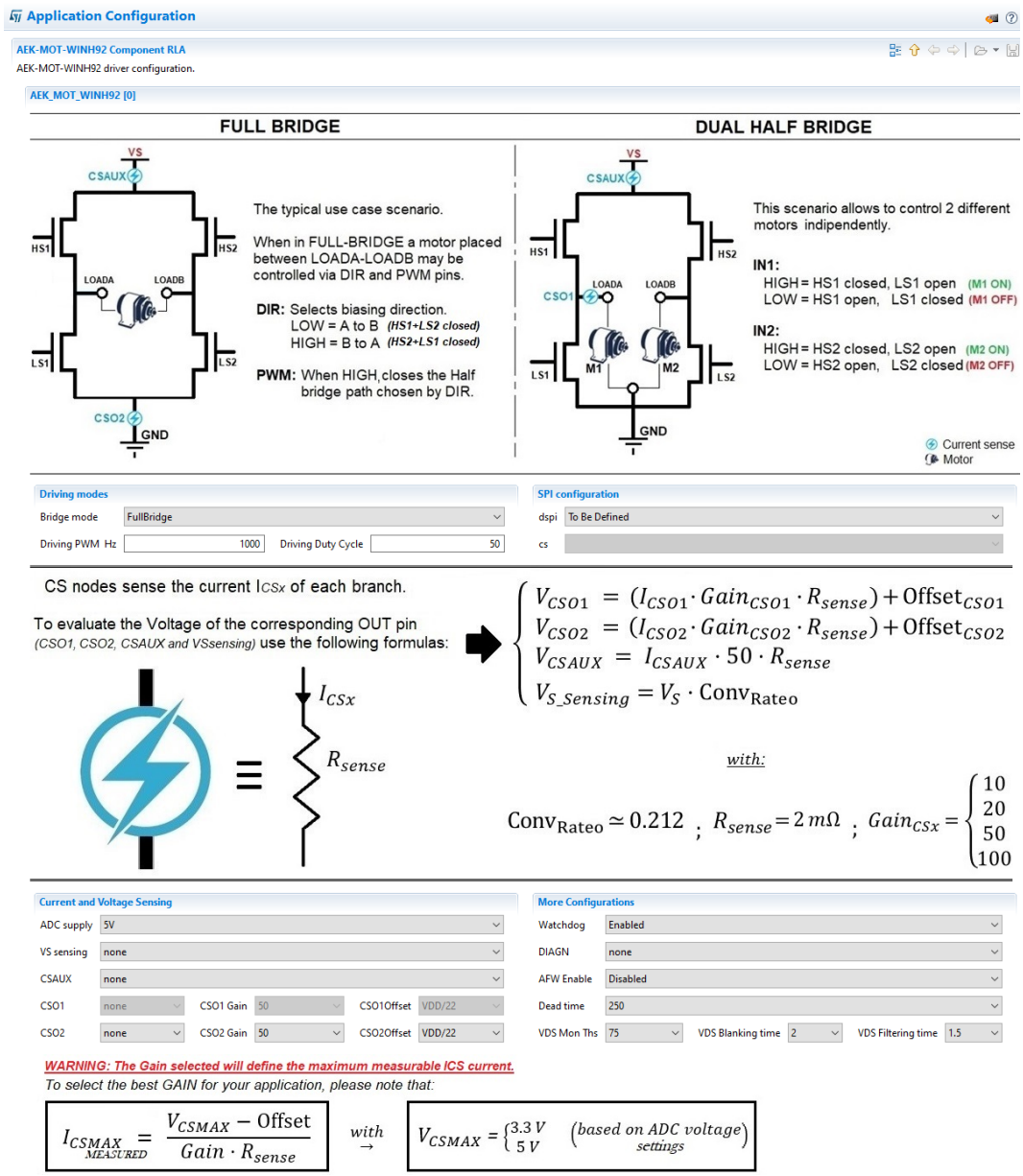
Step 4. Click on [+] to add a new element to the board list.

Figure 23. Adding a new element



Step 5. Double-click on the newly added element to configure the board.

Figure 24. AEK-MOT-WINH92 configuration



Step 6.

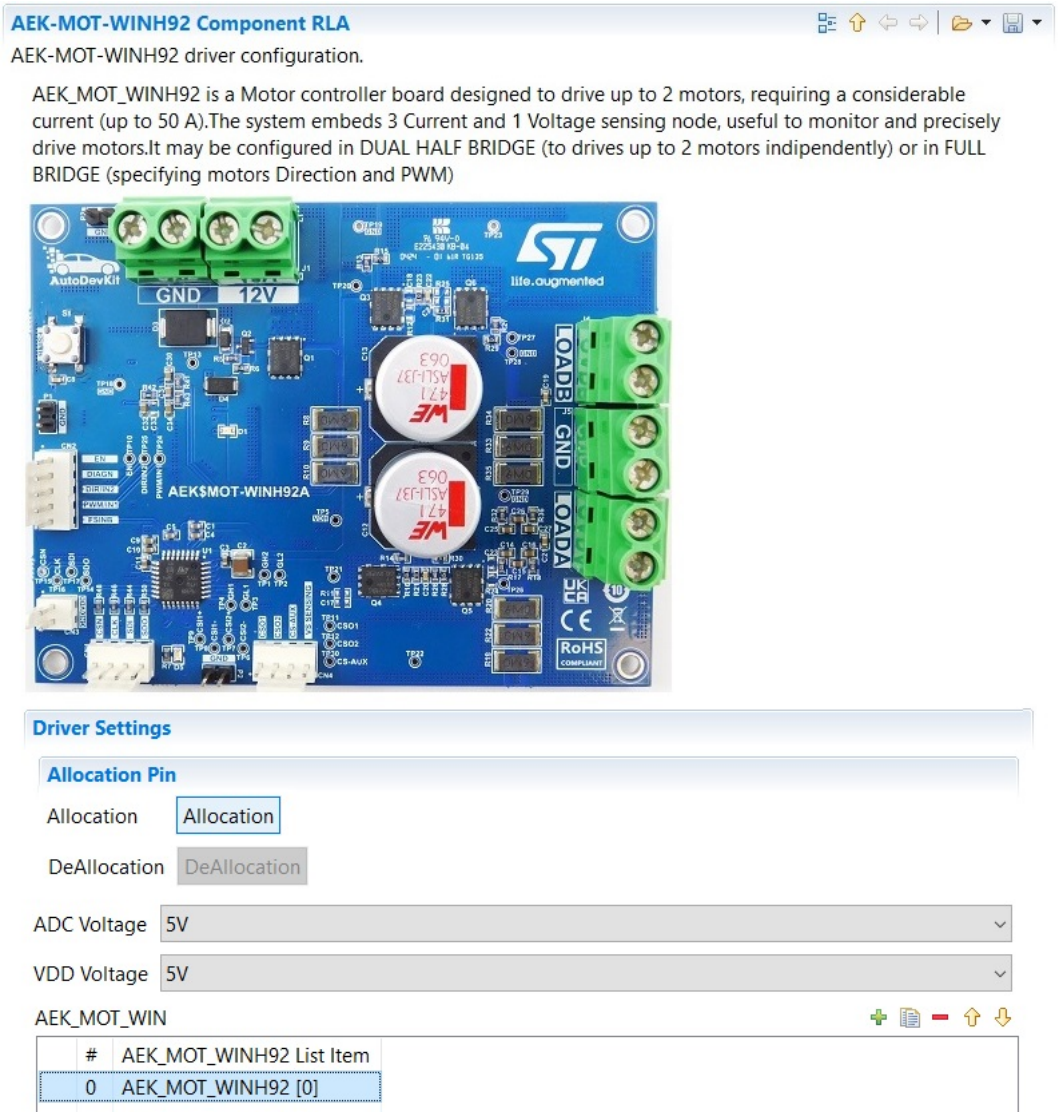
- Select Bridge Mode
- Select Driving PWM and Duty Cycle
- Select DSPI and CS
- Allocate/do not allocate pins for sensing (VSsense, CSAUX, CSO1, and CSO2)
- Enable/Disable ‘’Watchdog’’
- Allocate/don’t allocate ‘’DIAGN’’
- Enable/Disable ‘’AFW’’
- Select ‘’Dead time’’
- Select ‘VDS Monitoring threshold’’
- Select ‘VDS Blanking time’’
- Select ‘VDS Filtering time’’

Whenever a value different from “none“ is selected for the choosen pin, the allocation procedure will allocate a corresponding pin of the microcontroller board.

Step 7. Click on the  icon on the top right corner of the screen to go back to the allocation screen. Then:

- Select power supply used by ADCs(5V or 3.3V)
- Select VDD voltage (5V or 3.3V)
- Click on the “Allocation” button to allocate the AEK-MOT-WINH92 component

Figure 25. Component allocation



Step 8. Click on “Board View” to view the hardware connection between the AEK-MCU-C4MLIT1 board and the AEK-MOT-WINH92.

Figure 26. Editors for Application Name

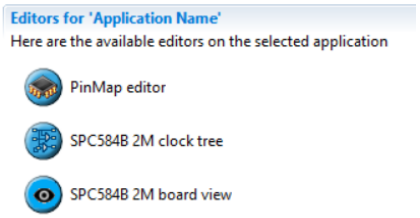
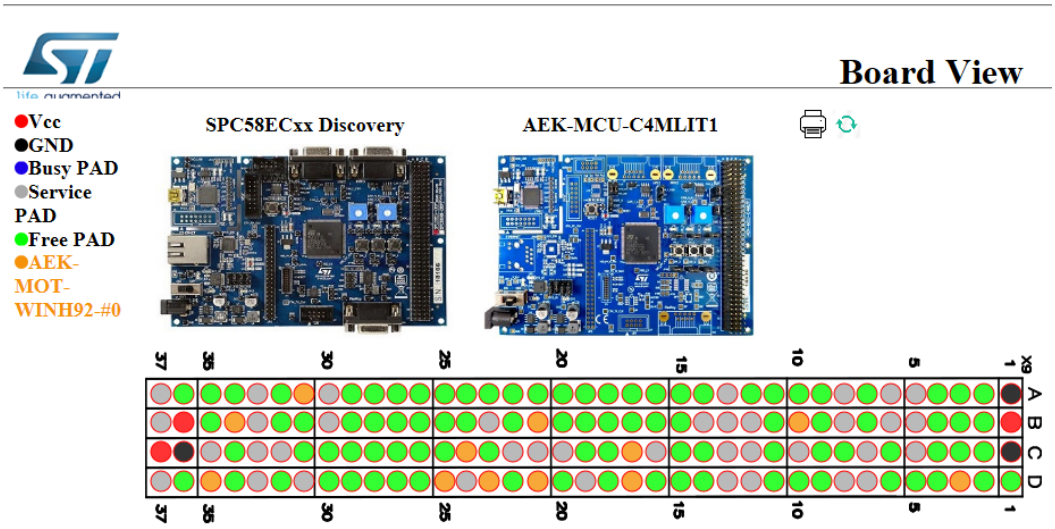
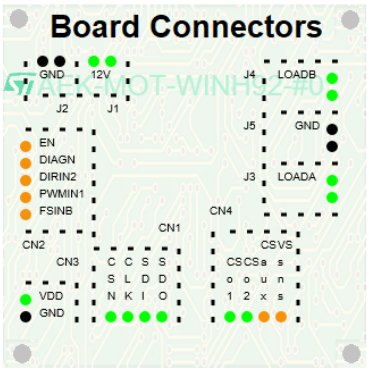


Figure 27. Board view



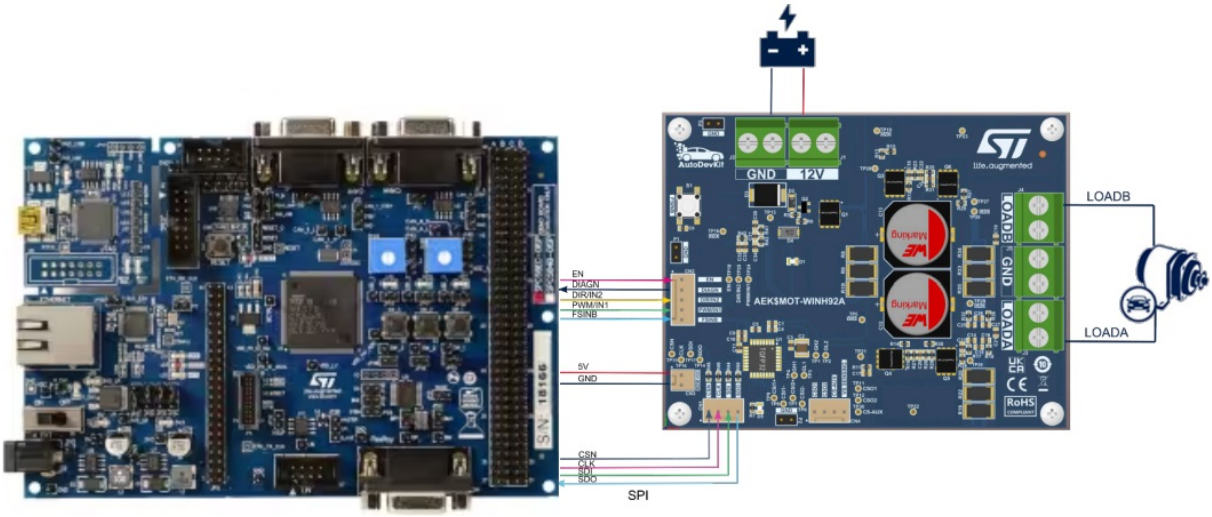
AEK-MOT-WINH92-#0Board

Connector Name	Pin Name	4x37 Connector
CN1	CS0	D21
CN1	MISO	B34
CN1	MOSI	B10
CN1	SCK	D35
CN2	DIAGN	C17
CN2	DIRIN2	A31
CN2	EN	B21
CN2	FSINB	D17
CN2	PWMIN1	D25
CN4	VSsense	D3
CN4	CSAUX	D23



Step 9. The system built should look like this:

Figure 28. System setup



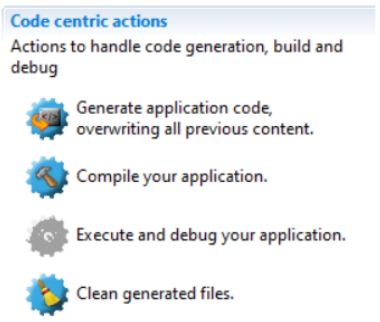
Create your main application example as follows:

```
#include "components.h"
 #include "AEK_MOT_WINH92_dev_mng.h"
 /*
  * Application entry point.
  */
 intmain(void) {
  componentsInit();
  irqIsrEnable();
  AEK_MOT_WINH92_dev_init();
  AEK_MOT_WINH92_dev_settingConf(AEK_MOT_WINH92_DEV0);
  AEK_MOT_WINH92_Set_BridgeMode(AEK_MOT_WINH92_DEV0, AEK_MOT_WINH92_FULL_BRIDGE);
  for ( ; ; ) {
  AEK_MOT_WINH92_EN_DIS_BridgeOut(AEK_MOT_WINH92_DEV0, AEK_MOT_WINH92_ENABLE);
  AEK_MOT_WINH92_FullBridge_Driver(AEK_MOT_WINH92_DEV0, AEK_MOT_WINH92_MotorCW, 100)
;//100% duty
  osalThreadDelayMilliseconds(2000);
  AEK_MOT_WINH92_FullBridge_Driver(AEK_MOT_WINH92_DEV0, AEK_MOT_WINH92_MotorCW, 50);
 //50% duty
  osalThreadDelayMilliseconds(2000);
  AEK_MOT_WINH92_FullBridge_Driver(AEK_MOT_WINH92_DEV0, AEK_MOT_WINH92_MotorStop, 0)
;//0% duty
  AEK_MOT_WINH92_EN_DIS_BridgeOut(AEK_MOT_WINH92_DEV0, AEK_MOT_WINH92_DISABLE);
  osalThreadDelayMilliseconds(3000);
  AEK_MOT_WINH92_EN_DIS_BridgeOut(AEK_MOT_WINH92_DEV0, AEK_MOT_WINH92_ENABLE);
  AEK_MOT_WINH92_FullBridge_Driver(AEK_MOT_WINH92_DEV0, AEK_MOT_WINH92_MotorCCW, 100
);//100% duty
  osalThreadDelayMilliseconds(2000);
  AEK_MOT_WINH92_FullBridge_Driver(AEK_MOT_WINH92_DEV0, AEK_MOT_WINH92_MotorCCW, 50)
;//50% duty
  osalThreadDelayMilliseconds(2000);
  AEK_MOT_WINH92_FullBridge_Driver(AEK_MOT_WINH92_DEV0, AEK_MOT_WINH92_MotorStop, 0)
;//0% duty
  AEK_MOT_WINH92_EN_DIS_BridgeOut(AEK_MOT_WINH92_DEV0, AEK_MOT_WINH92_DISABLE);
  osalThreadDelayMilliseconds(3000);
  }
 }
```

Note: This application configures the AEK-MOT-WINH92 via SPI, enabling VSsense and CAUX current sensing. The board is set in full bridge mode and a PWM is supplied to the PWMIN1 pin, in order to control bridge outputs. The motor placed between LOADA/LOADB Is going to be actived. The direction of the rotation is defined by DIRIN2 polarity.

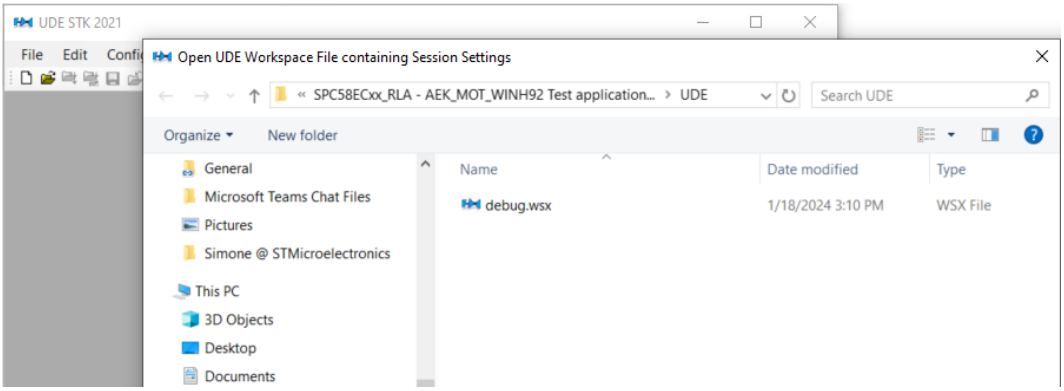
Step 10. Generate and compile your application.

Figure 29. Code generation and compilation



Step 11. Open “UDE Starterkit” and import the “.wsx” file from the workspace to flash your application.

Figure 30. Importing debug.wsx file



Step 12. Switch on your AEK-MCU-C4MLIT1 and run your AEK-MOT-WINH92 application.

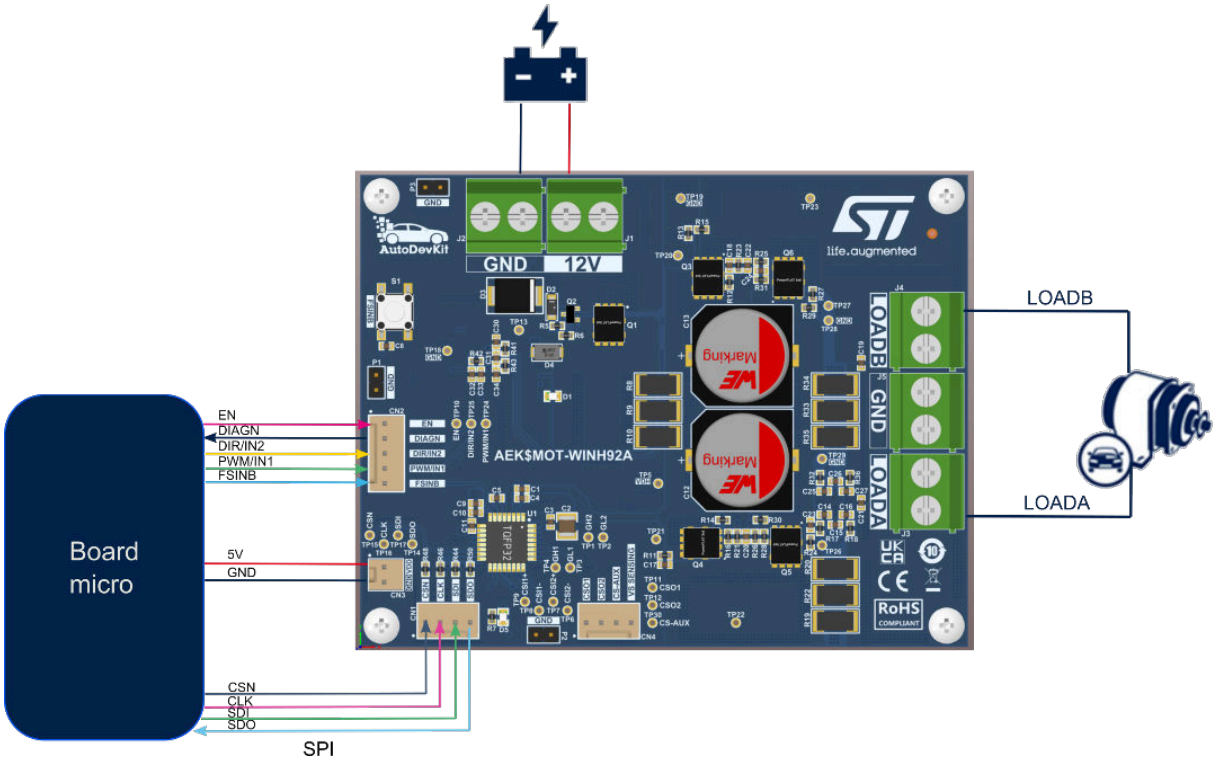
3 Available demos for AEK-MOT-WINH92 and connection schemes

In the AutoDevKit release 2.2.0 (or higher), the following demos are available for the AEK-MOT-WINH92:

- Full bridge mode demos
  - ***SPC582Bxx_RLA - AEK_MOT_WINH92 full bridge test application***, which is a demo application for the AEK-MCU-C4MLIT1 to configure an AEK-MOT-WINH92 in order to drive a motor, varying direction and PWM duty cycle, in full bridge mode;
  - ***SPC584Bxx_RLA - AEK_MOT_WINH92 full bridge test application***, which is a demo application for the *SPC584B-DIS* to configure an AEK-MOT-WINH92 in order to drive a motor, varying direction and PWM duty cycle, in full bridge mode;
  - ***SPC58ECxx_RLA - AEK_MOT_WINH92 full bridge test application***, which is a demo application for the AEK-MCU-C4MLIT1 to configure an AEK-MOT-WINH92 in order to drive a motor, varying direction and PWM duty cycle, in full bridge mode;
  - ***SPC58xNxx_RLA - AEK_MOT_WINH92 full bridge test application***, which is a demo available in AutoDevKit Studio for reference, in order to drive a motor, varying direction and PWM duty cycle, in full bridge mode.

  The following block diagram shows the connection scheme (with no current sensing implemented) for using the board with the four demos above.

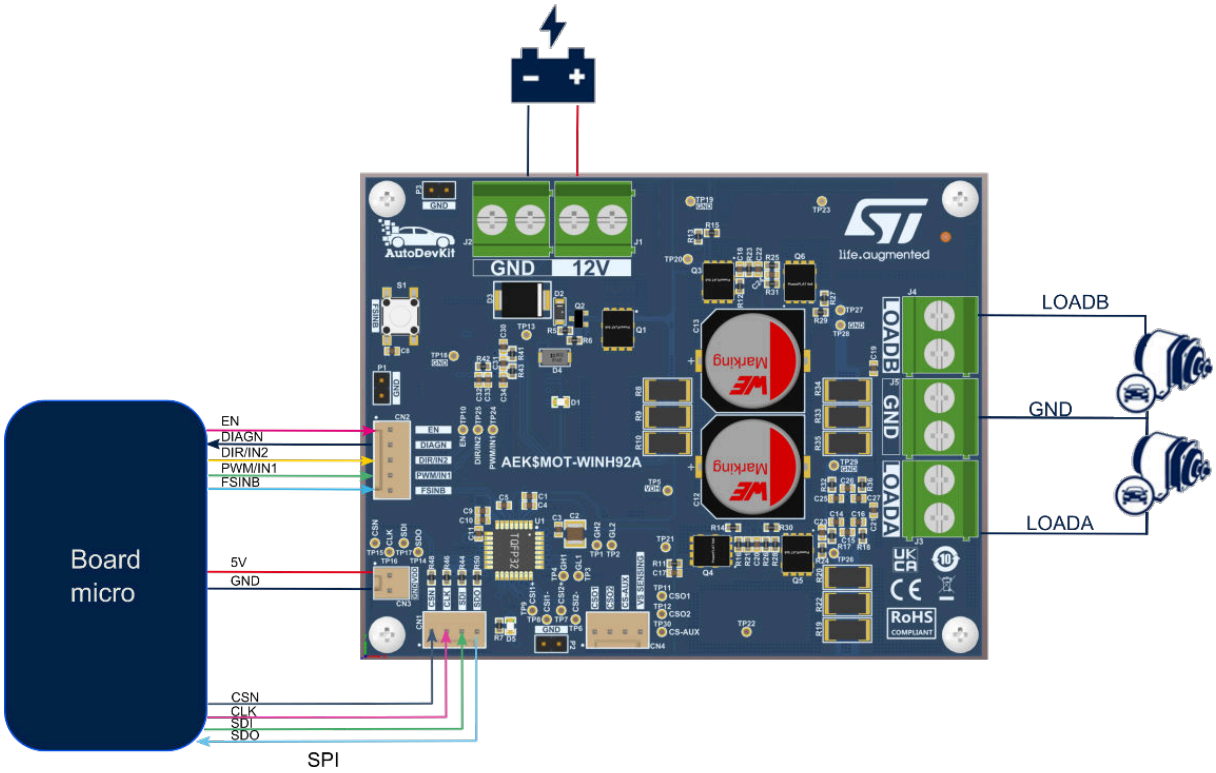
Figure 31. Full-bridge functional block diagram



- Dual half-bridge mode demo:
  - ***SPC58ECxx_RLA - AEK_MOT_WINH92 dual half-bridge test application***, which is a demo application for the AEK-MCU-C4MLIT1 to configure an AEK-MOT-WINH92 in order to drive in Dual Half bridge mode 2 DC motors varying PWM signals duty cycle.

The following block diagram shows the connection scheme (with no current sensing implemented) for using the board with the demo above.

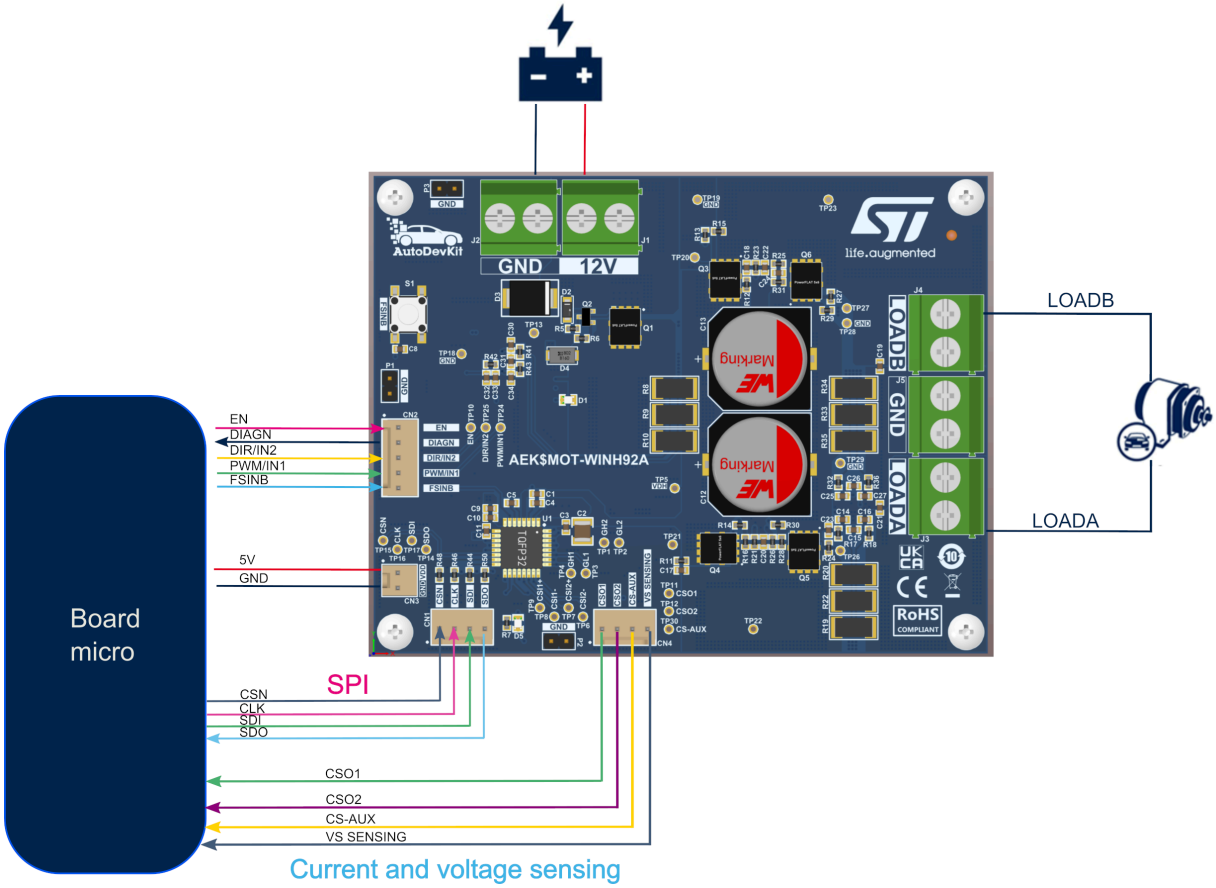
Figure 32. Dual half-bridge functional block diagram



- Full bridge mode demo for window lift application:
  - ***SPC58ECxx_RLA - AEK_MOT_WINH92 Window lift test application***, to be downloaded on the SPC58EC hosted on the AEK-MCU-C4MLIT1 MCU board. This demo makes a DC motor rotate in both directions by pressing two different buttons on the MCU board. If you press SW_User1, the motor rotates in clockwise direction, whereas, if you press SW_User3, the motor rotates in counterclockwise direction. Press the related button once again to stop the motor. Each time the motor rotates, in both directions, the system acquires the current value and stops the rotation in case the current is higher than a certain threshold fixed via software.

The following block diagram shows the connection scheme for using the board with the demo above.

Figure 33. Full bridge functional block diagram with current sensing



4 Available APIs

The APIs listed in the following table are declared in the "*AEK_MOT_WINH92_dev_mng.h*" file. These routines are used to initialize and configure the board.

Table 1. Device Manager APIs for the AEK-MOT-WINH92

API name	Description
AEK_MOT_WINH92_dev_init()	Initializes each AEK-MOT-WINH92 of the stack
AEK_MOT_WINH92_dev_deInit()	De-initializes each AEK-MOT-WINH92 of the stack
AEK_MOT_WINH92_dev_settingConf()	Sets each UI defined property for the selected AEK-MOT-WINH92 board

The APIs listed in the following table are declared in the "*AEK_MOT_WINH92_lld.h*" file. These high-level routines are necessary to configure the board properly.

Table 2. APIs for the AEK-MOT-WINH92

API name	Description
AEK_MOT_WINH92_WD_Refresh	Triggers a watchdog refresh message
AEK_MOT_WINH92_enterActive	Enters active mode
AEK_MOT_WINH92_enterStandby	Enters standby mode
AEK_MOT_WINH92_enterFailSafe	Enters FailSafe mode
AEK_MOT_WINH92_WD_Disable	Disables the watchdog
AEK_MOT_WINH92_EN_DIS_BridgeOut	Enables the OUTE bit, which starts the motors
AEK_MOT_WINH92_Set_BridgeMode	Sets bridge mode (full-bridge or dual-half bridge)
AEK_MOT_WINH92_EN_DIS_HalfBridgeX	Enables/disables half bridges specified by sel
AEK_MOT_WINH92_HalfBridge1_Driver	Drives pin IN1 to control Half Bridge1
AEK_MOT_WINH92_HalfBridge2_Driver	Drives pin IN2 to control Half Bridge2
AEK_MOT_WINH92_DualHalfBridge_Driver	Drives pin IN1 and IN2 to control both half bridges, simulating FULL BRIDGE functionalities.
AEK_MOT_WINH92_FullBridge_Driver	Drives PWM and DIR pins in order to control both bridges in FULL BRIDGE mode
AEK_MOT_WINH92_EN_DIS_AFW	Enables/disables active free wheeling (AFW) for full bridge
AEK_MOT_WINH92_Set_FreeWheeling_Side	Sets free wheeling side
AEK_MOT_WINH92_Set_SwitchOnSlew	Sets slew rate percentage for gate switching-on speed
AEK_MOT_WINH92_Set_SwitchOffSlew	Sets slew rate percentage for gate switching-off speed
AEK_MOT_WINH92_Set_DeadTime	Sets CrossCurrent protection time between HS-LS switching of the same leg
AEK_MOT_WINH92_EN_DIS_VDS_Monitoring	Enables/disables VDS monitoring
AEK_MOT_WINH92_Set_VDS_Mon_Thr	Sets VDS monitoring threshold
AEK_MOT_WINH92_Set_VDS_Mon_BlankingTime	Set VDS monitoring blanking time
AEK_MOT_WINH92_Set_VDS_Mon_FilteringTime	Set VDS monitoring filtering time
AEK_MOT_WINH92_EN_DIS_CurrentSensingX	Enables/disables current sensing (CSO1/CSO2)
AEK_MOT_WINH92_Set_CurrentSensingX_Gain	Sets current sensing gain value (CSO1/CSO2)
AEK_MOT_WINH92_Set_CurrentSensingX_OFFSET	Sets current sensing offset (CSO1/CSO2)
AEK_MOT_WINH92_EN_DIS_ChargePumpDithering	Enables/disables charge pump frequency dithering

API name	Description
AEK_MOT_WINH92_EN_DIS_ChargePump_UV_autorecover	Enables/disables charge pump fault autorecover
AEK_MOT_WINH92_FullBridge_OffState_Monitoring	Checks whether, during motor OFFState, a hardare fault occurs
AEK_MOT_WINH92_Set_VSVDH_OverVoltage_Ths	Sets VS/VDH overvoltage threshold

The APIs listed below are declared in the "*AEK_MOT_WINH92_reg_mng.h*" file. These routines are used to read registers.

Table 3. Register Manager APIs for the AEK-MOT-WINH92

API name	Description
AEK_MOT_WINH92_reg_readAllStatusReg	Reads all status registers
AEK_MOT_WINH92_reg_readAllControlReg	Reads all control registers
AEK_MOT_WINH92_reg_readAllReg	Reads all registers except the information registers
AEK_MOT_WINH92_reg_clearAllFlagStatusReg	Reads and clears DSR1
AEK_MOT_WINH92_reg_readAllInfoReg	Reads all information registers

The APIs listed in the following table are declared in the "*AEK_MOT_WINH92_spi_hld.h*" file. These routines are used to drive SPI peripheral.

Table 4. SPI high level driver APIs for the AEK-MOT-WINH92

API name	Description
AEK_MOT_WINH92_spi_init	Initializes DSPI
AEK_MOT_WINH92_spi_deInit	De-initializes DSPI
AEK_MOT_WINH92_spi_writeReg	Writes a register via SPI
AEK_MOT_WINH92_spi_readReg	Reads a register via SPI
AEK_MOT_WINH92_spi_readInfo	Reads an information register via SPI
AEK_MOT_WINH92_spi_readClearReg	Reads and clears a register via SPI
AEK_MOT_WINH92_parity_calc	Evaluates parity for the AEK-MOT-WINH92
AEK_MOT_WINH92_get_spi_de	Gets DE Error
AEK_MOT_WINH92_get_spi_fe	Gets FE Error
AEK_MOT_WINH92_get_spi_fs	Gets FS Error
AEK_MOT_WINH92_get_spi_gsbn	Gets GSBN Error
AEK_MOT_WINH92_get_spi_gw	Gets GW Error
AEK_MOT_WINH92_get_spi_ple	Gets PLE Error
AEK_MOT_WINH92_get_spi_rstb	Gets RSTB Error
AEK_MOT_WINH92_get_spi_spie	Gets SPIE Error

The APIs listed in the following table are declared in the "*AEK_MOT_WINH92_adc_hld.h*" file. These routines are used to drive ADC peripheral.

Table 5. ADC high level APIs for the AEK-MOT-WINH92

API name	Description
AEK_MOT_WINH92_adc_init	Initializes each allocated ADC pin

API name	Description
AEK_MOT_WINH92_adc_deInit	De-initializes ADC pins
AEK_MOT_WINH92_get_ADC_meas	Gets the selected ADC voltage value (VS/CSAUX/CSO1/CSO2)
AEK_MOT_WINH92_get_converted_ADC_meas	Gets the selected ADC converted value (VS/CSAUX/CSO1/CSO2)
AEK_MOT_WINH92_GetModule	Gets ADC data structure

The following function is the ADC callback and is defined in “AEK_MOT_WINH92_adc_cfg.c”.

Table 6. ADC callback for the AEK-MOT-WINH92

API name	Description
AEKMOTWINH92Config_adcAEKMOTWINH92Callback	Evaluates current and voltages from ADC measurement

The APIs listed in the following table are declared in the “*AEK_MOT_WINH92_pwm_hld.h*” file. These routines are used to drive PWM peripheral.

Table 7. PWM high level APIs for the AEK-MOT-WINH92

API name	Description
AEK_MOT_WINH92_pwm_init	Initializes PWM pins
AEK_MOT_WINH92_pwm_deInit	De-initializes PWM pin
AEK_MOT_WINH92_pwm_Set_Duty	Sets PWM duty cycle
AEK_MOT_WINH92_pwm_Set_Freq	Sets PWM frequency
AEK_MOT_WINH92_setpad_PWM	Treats an Emios channel as a GPIO, setting its value to HIGH
AEK_MOT_WINH92_clearpad_PWM	Treats an Emios channel as a GPIO, setting its value to LOW
AEK_MOT_WINH92_pwm_dev_settingConf	Sets PWM configuration based on UI parameters

The APIs listed in the following table are declared in the “*AEK_MOT_WINH92_stm_hld.h*” file. These routines are used to initialize and configure the system timer module (STM) peripheral. The STM is a timer that can be used to generate periodic interrupts, count external events or generate clock signals for other peripherals of the MCU.

Table 8. STM high level APIs for the AEK-MOT-WINH92

API name	Description
AEK_MOT_WINH92_stm_init	Initializes the STM peripheral
AEK_MOT_WINH92_stm_deInit	De-initializes the STM peripheral
AEK_MOT_WINH92_stm_setFreq	Configures manually the STM timer

STM periodically triggers **AEK_MOT_WINH92_WDSTM** callback, defined in “AEK_MOT_WINH92_STM_cfg.c”.

Table 9. STM callback

API name	Description
AEK_MOT_WINH92_WDSTM	This callback is periodically executed by the STM

The following APIs are used when the DIAG pin passes from low to high, signaling that an issue has occurred.

Table 10. **DIAG APIs**

API name	Description
AEK_MOT_WINH92_set_VDHOVE_diag	Sets VDH overvoltage enable (VDHOVE) diagnostic error
AEK_MOT_WINH92_set_VDHUVE_diag	Sets VDH undervoltage enable (VDHUVE) diagnostic error
AEK_MOT_WINH92_set_VSOVWE_diag	Sets VS overvoltage warning enable (VSOVWE) diagnostic error
AEK_MOT_WINH92_set_VDDOVE_diag	Sets VDD overvoltage enable (VDDOVE) diagnostic error
AEK_MOT_WINH92_set_TWE_diag	Sets thermal warning enable (TWE) diagnostic error
AEK_MOT_WINH92_set_TSDE_diag	Sets thermal shutdown enable (TSDE) diagnostic error
AEK_MOT_WINH92_set_FSINE_diag	Sets failsafe inputnot enable (FSINE) diagnostic error
AEK_MOT_WINH92_set_NRDYE_diag	Sets not ready enable (NRDYE) diagnostic error
AEK_MOT_WINH92_set_CPLOWE_diag	Sets charge pump low enable (CPLOWE) diagnostic error
AEK_MOT_WINH92_set_WDGFE_diag	Sets window watchdog flag enable (WDGFE) diagnostic error
AEK_MOT_WINH92_set_DIOOVE_diag	Sets digital input/output overvoltage enable (DIOOVE) diagnostic error
AEK_MOT_WINH92_set_DS1E_diag	Sets drain source leg1 enable (DS1E) diagnostic error
AEK_MOT_WINH92_set_SPIEE_diag	Sets SPI error enable (SPIEE) diagnostic error
AEK_MOT_WINH92_set_DS2E_diag	Sets drain source leg2 enable (DS2E) diagnostic error
AEK_MOT_WINH92_get_VDHOV_diag	Gets VDHOV diagnostic error
AEK_MOT_WINH92_get_VDHUV_diag	Gets VDHUV diagnostic error
AEK_MOT_WINH92_get_VSOVW_diag	Gets VSOVW diagnostic error
AEK_MOT_WINH92_get_VDDOV_diag	Gets VDDOV diagnostic error
AEK_MOT_WINH92_get_TW_diag	Gets TW diagnostic error
AEK_MOT_WINH92_get_TSD_diag	Gets TSD diagnostic error
AEK_MOT_WINH92_get_FSIN_diag	Gets FSIN diagnostic error
AEK_MOT_WINH92_get_NRDY_diag	Gets NRDY diagnostic error
AEK_MOT_WINH92_get_CPLOW_diag	Gets CPLOW diagnostic error
AEK_MOT_WINH92_get_WDGF_diag	Gets WDGF diagnostic error
AEK_MOT_WINH92_get_DIOOV_diag	Gets DIOOV diagnostic error
AEK_MOT_WINH92_get_DSHS1_diag	Gets drain source highside 1(DSHS1) diagnostic error
AEK_MOT_WINH92_get_DSLS1_diag	Gets drain source lowside 1 (DSLS1) diagnostic error
AEK_MOT_WINH92_get_DSHS2_diag	Gets drain source highside 2 (DSHS2) diagnostic error
AEK_MOT_WINH92_get_DSLS2_diag	Gets Gets drain source lowside 2 (DSLS2) diagnostic error
AEK_MOT_WINH92_get_FSINLL_diag	Gets fail safe input not logic level (FSINLL) diagnostic error
AEK_MOT_WINH92_get_FSINHS1_diag	Gets fail safe input not highside 1 (FSINHS1) diagnostic error
AEK_MOT_WINH92_get_FSINLS1_diag	Gets fail safe input not lowside 1 (FSINLS1) diagnostic error
AEK_MOT_WINH92_get_FSINHS2_diag	Gets fail safe input not highside 2 (FSINHS2) diagnostic error
AEK_MOT_WINH92_get_FSINLS2_diag	Gets fail safe input not lowside 2 (FSINLS2) diagnostic error
AEK_MOT_WINH92_get_O1DS_diag	Gets output 1 digital status (O1DS) diagnostic error
AEK_MOT_WINH92_get_O2DS_diag	Gets output 2 digital status (O2DS) diagnostic error
AEK_MOT_WINH92_get_diagn	Gets diagnostic diagn pin status
AEK_MOT_WINH92_get_DiagSts	Gets diagnostic diagn status

5 Schematic diagrams

Figure 34. **AEK-MOT-WINH92 circuit schematic (1 of 4)**

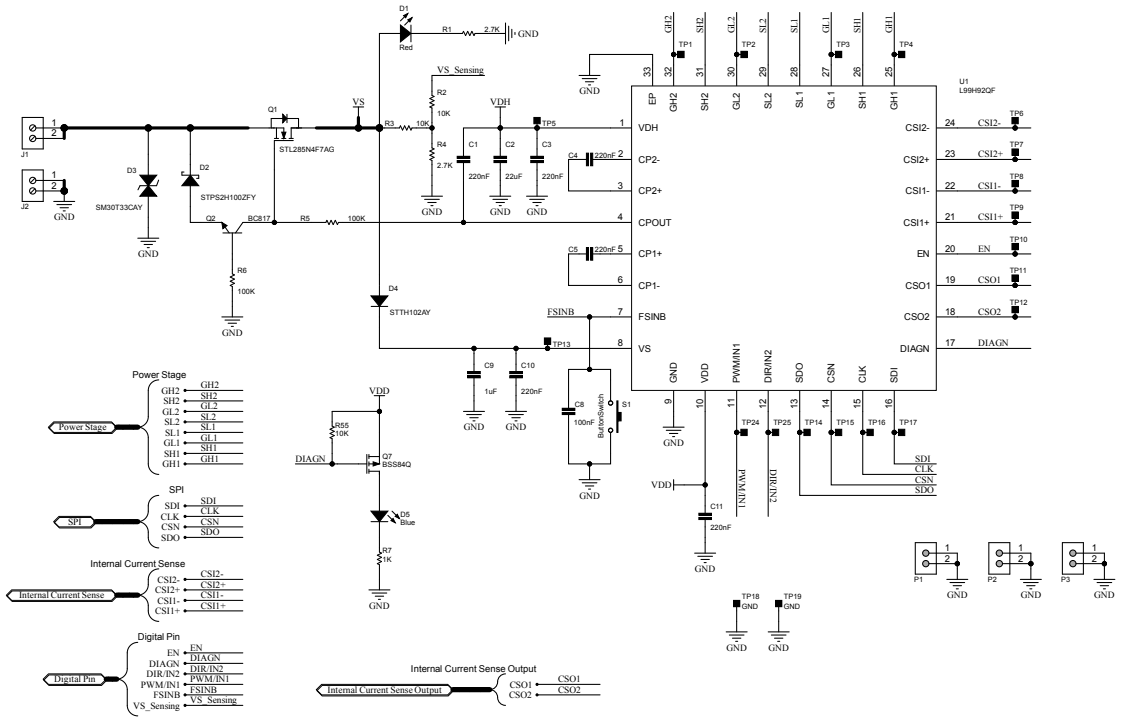


Figure 35. **AEK-MOT-WINH92 circuit schematic (2 of 4)**

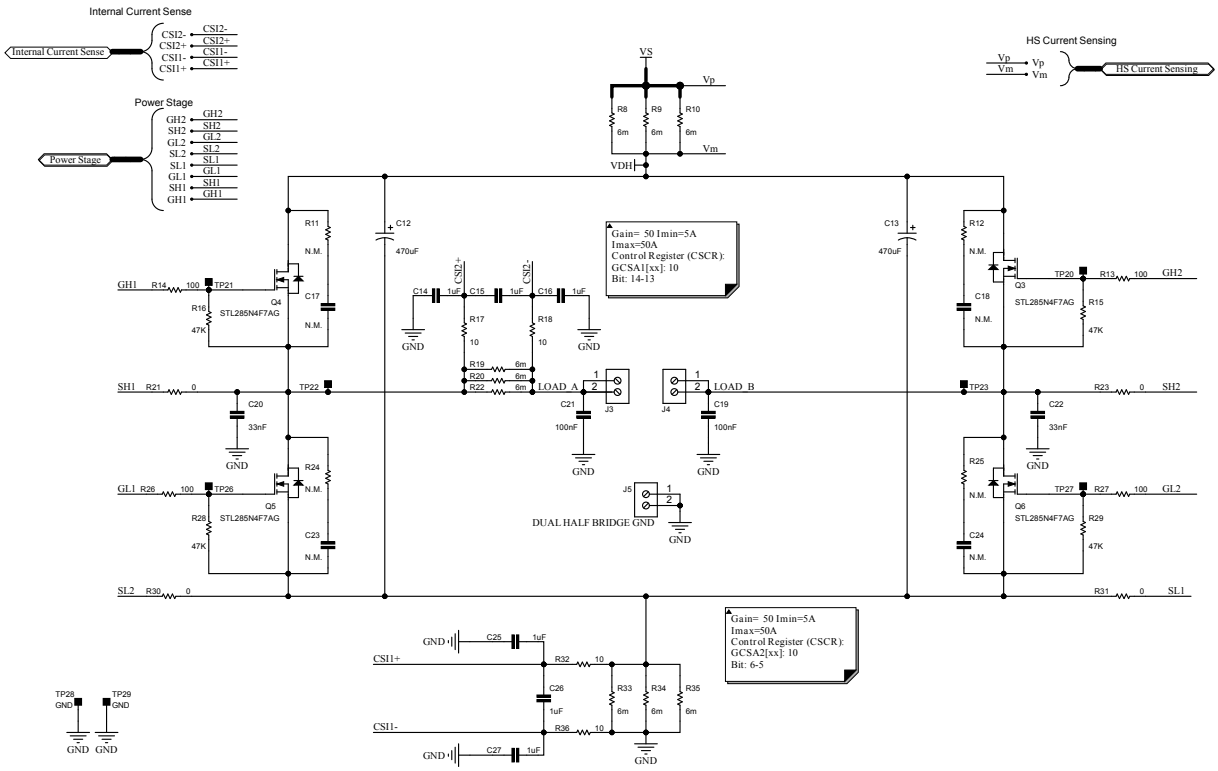


Figure 36. **AEK-MOT-WINH92 circuit schematic (3 of 4)**

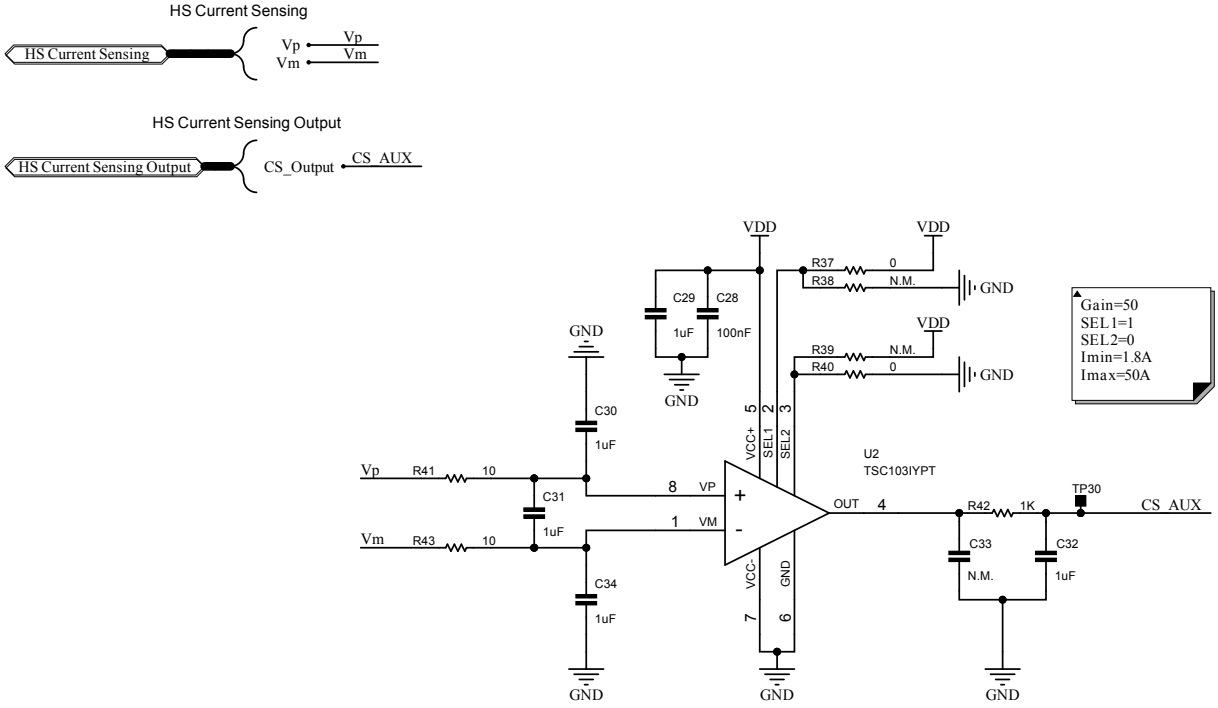
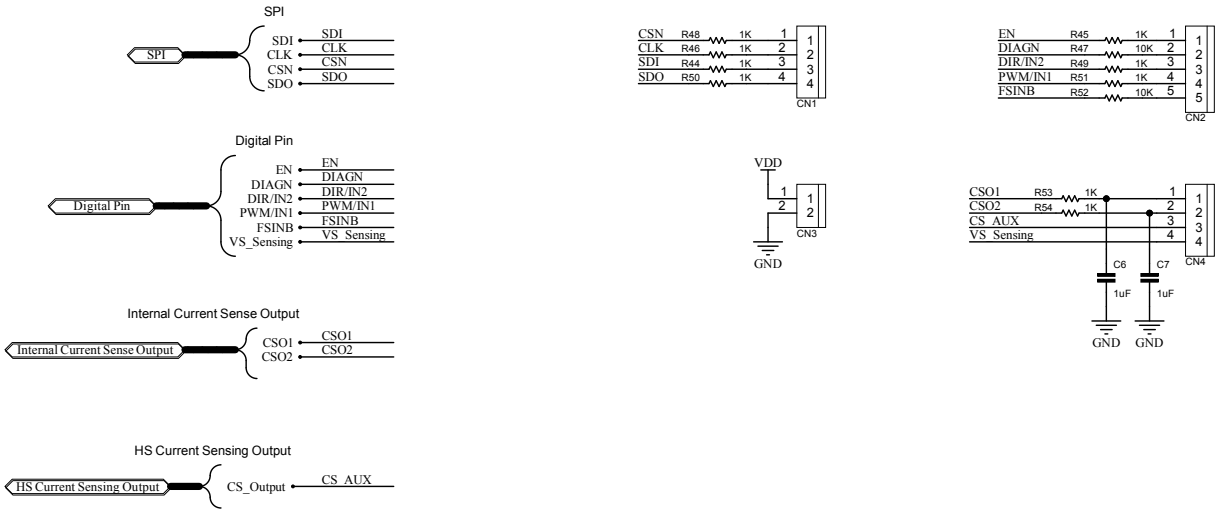


Figure 37. **AEK-MOT-WINH92 circuit schematic (4 of 4)**



6 Bill of materials

Table 11. **AEK-MOT-WINH92 bill of materials**

Item	Q.ty	Ref.	Part/value	Description	Manufacturer	Order code
1	6	C1, C3, C4, C5, C10, C11	220nF	0603 - 50V - X7R Class II	WE	885012206125
2	1	C2	22uF	1210 - 25V - X5R Class II	WE	885012109014
3	14	C6, C7, C9, C14, C15, C16, C25, C26, C27, C29, C30, C31, C32, C34	1uF	0603 - 50V - X7R Class II	WE	885012206126
4	4	C8, C19, C21, C28	100nF	0603 - 50V - X7R Class II	WE	885012206095
5	2	C12, C13	470uF	WCAP-ASLI Aluminum Electrolytic Capacitors - 63V - 16x17	WE	865080763016
6	5	C17, C18, C23, C24, C33	N.M.	0603	N.A.	N.A.
7	2	C20, C22	33nF	0603 - 50V - X7R Class II	WE	885012206092
8	2	CN1, CN4	Con 4P	2.54mm - 1 row - KK254 - Male	WE	61900411121
9	1	CN2	Con 5P	2.54mm - 1 row - KK254 - Male	WE	61900511121
10	1	CN3	Con 2P	2.54mm - 1 row - KK254 - Male	WE	61900211121
11	1	D1	Red	0805 - Led Red - 2V	WE	150080RS75000
12	1	D2	STPS2H100ZFY, SOD123Flat	Automotive 100 V - 2 A power Schottky diode	ST	STPS2H100ZFY
13	1	D3	SM30T33CAY, SMC	SMC TVS -28VDC - Bidirectional	ST	SM30T33CAY
14	1	D4	STTH102AY, SMA	Automotive high efficiency ultrafast diode	ST	STTH102AY
15	1	D5	Blue	0805 - Led Blue - 3.2V	WE	150080BS75000
16	5	J1, J2, J3, J4, J5	Con 2p 6.35_green	6.35mm - WR-TBL Series 2506 Horizontal Entry Modular w. Rising Cage Clamp	WE	691250610002
17	3	P1, P2, P3	Header 2p 2.54	2.54mm - WR-PHD Pin Header, THT, pitch 2.54mm, Single Row, Vertical, 2p	WE	61300211121

Item	Q.ty	Ref.	Part/value	Description	Manufacturer	Order code
18	5	Q1, Q3, Q4, Q5, Q6	STL285N4F7AG, PowerFLAT 5x6 WF	Automotive-grade N-channel 40 V, 0.9 mΩ typ., 120 A STripFET™ F7 Power MOSFET in a PowerFLAT 5x6 package	ST	STL285N4F7AG
19	1	Q2	BC817	NPN Bipolar Transistor 45V 500mA	Nexperia	BC817
20	1	Q7	BSS84Q	P-Channel Enhancement Mosfet	DIODES	BSS84Q-7-F
21	2	R1, R4	2.7K	0603 - ±1% - 0.125W	Vishay	MCT06030C2701FP500
22	5	R2, R3, R47, R52, R55	10K	0603 - ±1% - 0.2W	Panasonic	ERJP03F1002V
23	2	R5, R6	100K	0603 - ±1% - 0.2W	Panasonic	ERJP03F1003V
24	11	R7, R42, R44, R45, R46, R48, R49, R50, R51, R53, R54	1K	0603 - ±1% - 0.25W	Panasonic	ERJPA3F1001V
25	9	R8, R9, R10, R19, R20, R22, R33, R34, R35	6m	2512 - ±1% -3W	Panasonic	ERJ-MS4HF6M0U
26	6	R11, R12, R24, R25, R38, R39	N.M.	0603	N.A.	N.A.
27	4	R13, R14, R26, R27	100	0603 - ±1% - 0.125W	Panasonic	ERJH3EF1000V
28	4	R15, R16, R28, R29	47K	0603 - ±1% - 0.25W	Panasonic	ERJPA3F4702V
29	6	R17, R18, R32, R36, R41, R43	10	0603 - ±1% - 0.25W	Panasonic	ERJPA3F10R0V
30	6	R21, R23, R30, R31, R37, R40	0	0603 - ±1% - 0.1W	Panasonic	ERJ3GEY0R00V
31	1	S1	ButtonSwitch	Switch	WE	430152043826
32	1	U1	L99H92QF, TQFP 32 7x7x1.0	Half-bridge pre-driver for automotive applications	ST	L99H92QF-TR
33	1	U2	TSC103IYPT, TSSOP-8L	High-voltage, high-side current sense amplifier	ST	TSC103IYPT
34	4	for blister	970150365	WA-SPAII Plastic Spacer Stud, metric, internal/ internal	WE	970150365
35	4	for blister	97790403211	WA-SCRW Pan Head Screw w. cross slot M3	WE	97790403211

Item	Q.ty	Ref.	Part/value	Description	Manufacturer	Order code
36	1	for blister	61900511621	WR-WTB 2.54 mm Female Terminal Housing 5p	WE	61900511621
37	2	for blister	61900411621	WR-WTB 2.54 mm Female Terminal Housing 4p	WE	61900411621
38	1	for blister	61900211621	WR-WTB 2.54 mm Female Terminal Housing 2p	WE	61900211621
39	15	for blister	61900113722	WR-WTB 2.54 mm Female Crimp Contact	WE	61900113722

7 Board versions

Table 12. **AEK-MOT-WINH92 versions**

Finished good	Schematic diagrams	Bill of materials
AEK$MOT-WINH92A (1)	AEK$MOT-WINH92A schematic diagrams	AEK$MOT-WINH92A bill of materials

1. *This code identifies the AEK-MOT-WINH92 evaluation board first version.*

8 Regulatory compliance information

Notice for US Federal Communication Commission (FCC)

For evaluation only; not FCC approved for resale

FCC NOTICE - This kit is designed to allow:

(1) Product developers to evaluate electronic components, circuitry, or software associated with the kit to determine

whether to incorporate such items in a finished product and

(2) Software developers to write software applications for use with the end product.

This kit is not a finished product and when assembled may not be resold or otherwise marketed unless all required FCC equipment authorizations are first obtained. Operation is subject to the condition that this product not cause harmful interference to licensed radio stations and that this product accept harmful interference. Unless the assembled kit is designed to operate under part 15, part 18 or part 95 of this chapter, the operator of the kit must operate under the authority of an FCC license holder or must secure an experimental authorization under part 5 of this chapter 3.1.2.

Notice for Innovation, Science and Economic Development Canada (ISED)

For evaluation purposes only. This kit generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to Industry Canada (IC) rules.

À des fins d'évaluation uniquement. Ce kit génère, utilise et peut émettre de l'énergie radiofréquence et n'a pas été testé pour sa conformité aux limites des appareils informatiques conformément aux règles d'Industrie Canada (IC).

Notice for the European Union

This device is in conformity with the essential requirements of the Directive 2014/30/EU (EMC) and of the Directive 2015/863/EU (RoHS).

Notice for the United Kingdom

This device is in compliance with the UK Electromagnetic Compatibility Regulations 2016 (UK S.I. 2016 No. 1091) and with the Restriction of the Use of Certain Hazardous Substances in Electrical and Electronic Equipment Regulations 2012 (UK S.I. 2012 No. 3032).

Revision history

Table 13. **Document revision history**

Date	Revision	Changes
11-Mar-2024	1	Initial release.

Contents

List of tables

List of figures