UM12257

TJA14xx evaluation boards

Rev. 1.0 — 3 June 2025

User manual

Document information

Information	Content
Keywords	TJA1445A-EVB, TJA1446B-EVB, TJA1465A-EVB, TJA1465B-EVB, TJA1466B-EVB, TJA1466C-EVB, high-speed CAN transceiver, partial networking, FlexGUI software interface, Arduino UNO
Abstract	This document describes the operation of the TJA14xx-EVB evaluation boards and accompanying FlexGUI software package.

1 Introduction

This document is the user guide for the TJA14xx family of evaluation boards (referred to throughout this document as TJA14xx-EVB). It is intended for engineers involved in the evaluation, design, implementation, and validation of the TJA1445/46 high-speed CAN and TJA1465/66 CAN SIC transceivers for partial networking. This guide discusses power supply requirements, the microcontroller (MCU) and CAN bus interfaces and describes how to connect the board into an ECU/CAN network. It also describes how to install and use the accompanying FlexGUI software.

The TJA14xx-EVB boards were designed to facilitate the testing and evaluation of TJA14xx product features in a variety of microcontroller IO interface environments. All MCU interface signals can be accessed in two ways: they are available at a header row on the top side of the board and also on connectors on the bottom side that can be plugged directly into many NXP MCU evaluation boards. All TJA14xx evaluation boards are designed to be compatible with the S32K148 evaluation board from NXP and to support the use of standard software development tools and drivers. The S32K148-Q176 MCU evaluation board is available at nxp.com/S32K148-Q176.

2 Finding kit resources and information on the NXP website

NXP Semiconductors provides online resources for their evaluation boards and supported device(s) on nxp.com.

The information pages for the TJA14xx-EVB evaluation boards can be found at http://www.nxp.com/TJA1445A-EVB, http://www.nxp.com/TJA1446B-EVB, http://www.nxp.com/TJA1465A-EVB and http://www.nxp.com/TJA1466-EVB. The information pages provide overview information, documentation, software and tools, parametrics, ordering information and a **Getting Started** guide. The Getting Started guide provides quick-reference information applicable to using the TJA14xx-EVB evaluation board, including the downloadable assets referenced in this document.

3 Getting ready

The supplied kit contents, additional hardware and a Windows PC workstation with installed software are needed to work with the TJA14xx-EVB.

3.1 Kit contents

- Assembled and tested TJA14xx-EVB evaluation board in an antistatic bag

3.2 Additional hardware

The following components are not included:

- A 12 V power supply
- A Windows PC and a USB cable to run the FlexGUI application
- S32K148-Q176 MCU evaluation board, available at nxp.com. This board provides the 5 V V_{CC} and 5 V or 3.3 V V_{IO} supplies for the plugged-on TJA14xx-EVB.

3.3 Software

When the TJA14xx-EVB is coupled with the S32K148-Q176 MCU evaluation board, the microcontroller board can be used as a USB/SPI interface between the TJA14xx-EVB and a PC. The FlexGUI application must be installed on the PC and the FlexGUI firmware loaded onto the microcontroller board (S32K148EVB). Details are provided in Section 5.3. The FlexGUI software package can be downloaded from nxp.com/FlexGUI.

4 Getting to know the hardware

The TJA14xx-EVB family consists of the following evaluation boards:

- TJA1445A-EVB
- TJA1446B-EVB
- TJA1465A-EVB
- TJA1465B-EVB
- TJA1466B-EVB
- TJA1466C-EVB

Board dimensions are 42 mm x 78 mm. When a TJA1445B device is being evaluated, take a TJA1465B-EVB and replace the TJA1465B with a TJA1445B. When a TJA1446C device is being evaluated, take a TJA1466C-EVB and replace the TJA1466C with a TJA1446C. The TJA1446A and TJA1466A would need a 1.8 V V_{IO} supply, which is not supported by the S32K148EVB.

4.1 Board image

A top view of the The TJA1446B-EVB board is shown in Figure 1. Board dimensions are 42 mm x 78 mm. Only components needed to support basic functionality are included. The other evaluation boards have a similar layout.

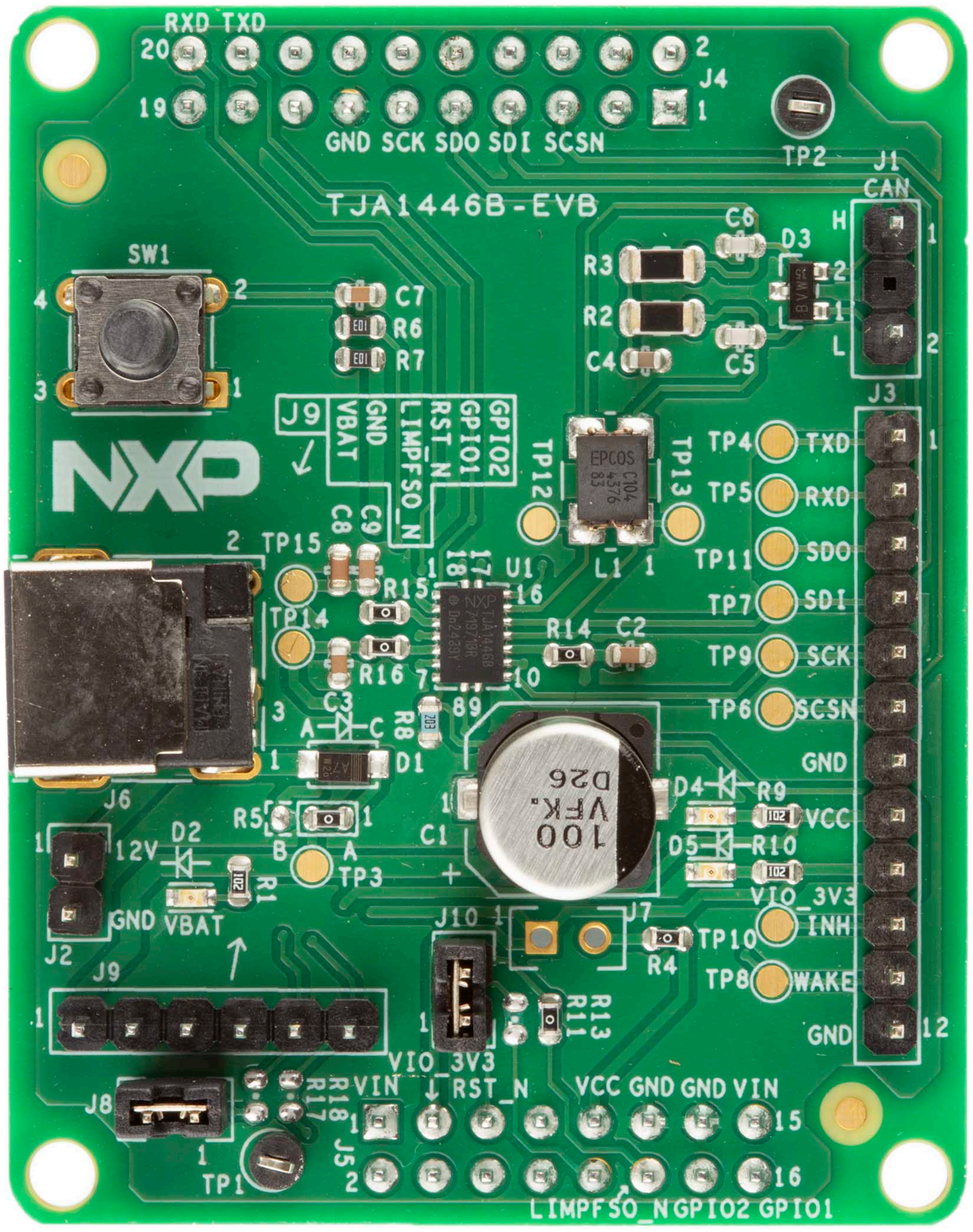


Figure 1. TJA1446B-EVB

4.2 Board overview

Top and bottom views of he TJA1466B-EVB are shown are in Figure 2, with the main components highlighted and labeled.

The board contains CAN bus filter, termination and protection circuitry along with power supply and wake-up circuitry and LEDs indicating when pins VBAT, VCC and VIO are supplied. Header rows (2.54 mm pitch) are provided for connecting the MCU interface and application signals. All ground pins are connected to the ground plane.

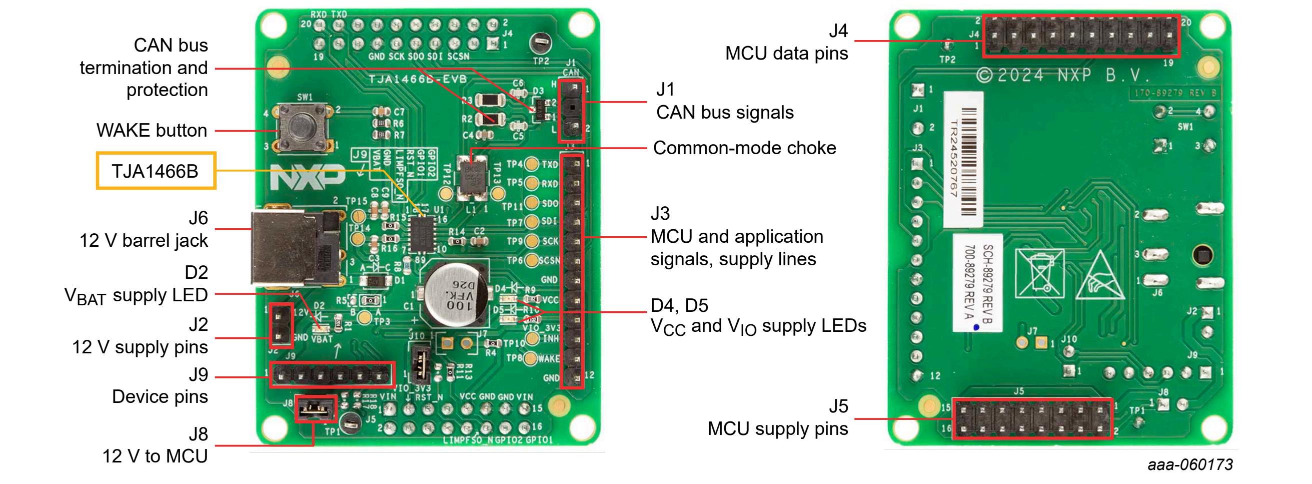


Figure 2. TJA1466B-EVB top (left) and bottom (right) views

4.2.1 Ground connections

All ground pins are connected to the ground plane.

4.2.2 Power supply

4.2.2.1 Battery connections

An external 12 V power supply must be connected to either power jack J6 or 2-pin connector J2. Green LED D2 lights up once the 12 V supply has been connected.

By default, the TJA14xx-EVB board battery supply is routed to the MCU board via pin VIN on Arduino connector J5. This allows the supply to the entire module to be managed via the TJA14xx-EVB board. This feature can be disabled by removing jumper J8, disconnecting the battery supply from pin VIN.

4.2.2.2 V_{CC}/V_{IO} connections

A 5 V V_{CC} supply is needed to operate the CAN transmitter and receiver in Normal and Standby modes. A V_{IO} supply is needed to supply the digital IOs and MCU interface (e.g. SPI pins). The V_{IO} voltage must be aligned with the MCU interface supply voltage. The V_{CC} and V_{IO} voltages are not needed in Sleep mode. Detailed information on the functionality and operation of the TJA14xx devices can be found in the data sheets and application notes.

The V_{CC} and V_{IO} supplies can be connected to either J3 or J5 (pins VCC and VIO). J3 is on the top of the TJA14xx-EVB board and J5 is mounted on the bottom. The pin arrangement on J5 follows the Arduino Uno pinout order, allowing the TJA14xx-EVB to be connected directly to a variety of NXP MCU evaluation boards. LED D4 lights up when V_{CC} is present. LED D5 lights up when V_{IO} is present.

The TJA1446B-EVB and TJA1466B-EVB need a 3.3 V V_{IO} supply. The other TJA14xx-EVBs will work with a 3.3 V or 5 V supply. The jumper setting of the MCU board may need to be adjusted before plugging on the TJA14xx-EVB (see Section 5.2.2.1).

4.2.3 CAN communication circuitry

The TJA14xx-EVB evaluation board contains typical CAN filter, termination and protection circuitry. The CANH and CANL bus signals are available on connector J1.

Equipped with termination resistors R2 and R3, the TJA14xx-EVB board is pre-prepared to be used as a termination node in a CAN network. If the CAN network is already terminated at both ends, it is recommended to remove R2 and R3 or replace them with higher value resistors to ensure that the impedance on the bus meets the CAN bus load specification, typically 60 Ω.

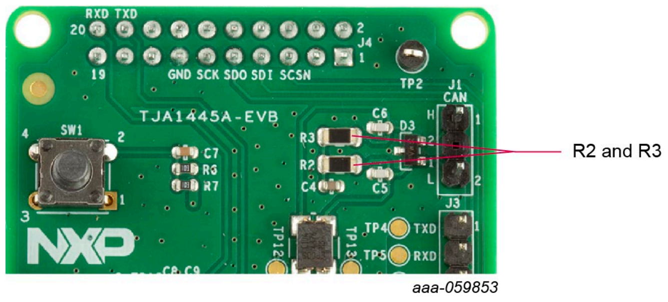


Figure 3. Termination resistors R2 and R3

4.2.4 Wake-up and INH functionality

The TJA14xx supports a Sleep mode for use in energy-sensitive applications. Once the device has entered Sleep mode, it will remain in this low-power mode until a wake-up request is received. A wake-up event can be triggered remotely via a standard wake-up pattern or a dedicated wake-up frame on the CAN bus, or locally via the WAKE pin (details of wake-up functionality can be found in the TJA14xx data sheets and application notes).

The TJA14xx-EVB evaluation board features local wake-up test circuitry. The WAKE pin is pulled HIGH by default via 10 kΩ resistors R6 and R7. When switch SW1 is pressed, the WAKE pin is pulled LOW. To make use of this feature, falling-edge detection on the WAKE pin must be enabled in the TJA14xx register map (as described in the data sheet).

Pin INH is typically used to control the supply to the MCU and peripherals. In Normal and Standby modes, the level on this pin is equivalent to the voltage on pin VBAT. Pin INH is pulled LOW via resistor R8 when the TJA14xx switches to Sleep mode. The WAKE and INH signals are not routed to the Arduino connectors. They can be accessed via connector J3 on the top of the board.

4.2.5 MCU interface

The digital interface signals are available on top connector J3 (J3-01 to J3-06), as well as on bottom connector J4 (J4-18, 20, 9, 7, 11, 5). Two of these pins, TXD and RXD, are used for CAN data communication with the MCU. The remaining four pins are used for SPI communication with the MCU.

4.2.6 TJA14xx GPIO interface

J9 provides access to selected TJA14xx pins, e.g. for attaching oscilloscope probes or for connecting application-specific hardware. J9 is a 2-pin connector on the TJA1445A-EVB and TJA1465A-EVB and a 6-pin conector on the other boards, as detailed in Table 1.

Table 1. J9 pinning

Pin	TJA14x5A	TJA14x5B	TJA1466B
1	VBAT		
2	GND		
3	n.a.	GPIO3	LIMPFSO_N
4	n.a.	TXEN_N	RST_N
5	n.a.	GPIO1	GPIO1
6	n.a.	GPIO2	GPIO2

4.3 Integrating the TJA14xx-EVB into an existing network

To use the evaluation board with an existing CAN network, the CANL and CANH signals on connector J1 need to be connected to the CAN bus lines and the following connections need to be made between the MCU and the evaluation board:

Table 2. MCU to TJA14xx-EVB connections

MCU	TJA14xx-EVB
MISO	SDO
MOSI	SDI
SCK	SCK
CS	SCSN
CAN TXD	TXD
CAN RXD	RXD
GND	GND
µC supply	VIO
5 V	VCC
Reset input	RST_N (TJA1446/66 only)

The INH signal should be connected to the control input for the MCU supply, if applicable. When a dedicated wake-up signal is connected to the WAKE pin on the transceiver, check if R6 needs to be removed to disconnect the onboard wake circuit.

These connections (except INH and WAKE) are established automatically when using the TJA14xx-EVB with a S32K148-Q176 board running the FlexGUI firmware. See Section 5 for details.

4.4 Schematic diagrams

Full size PDF schematic diagrams can be downloaded from www.nxp.com.

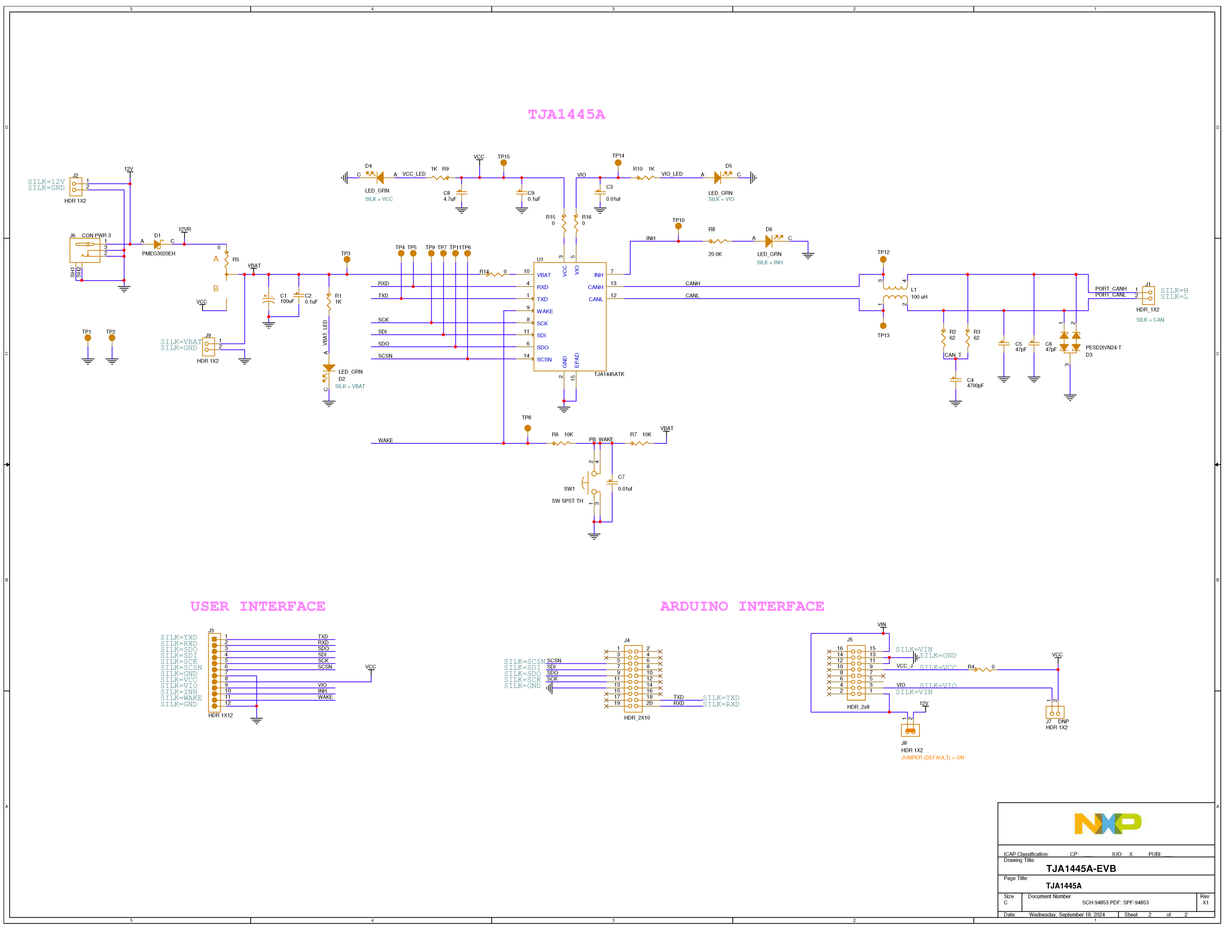


Figure 4. TJA1445A-EVB schematic diagram

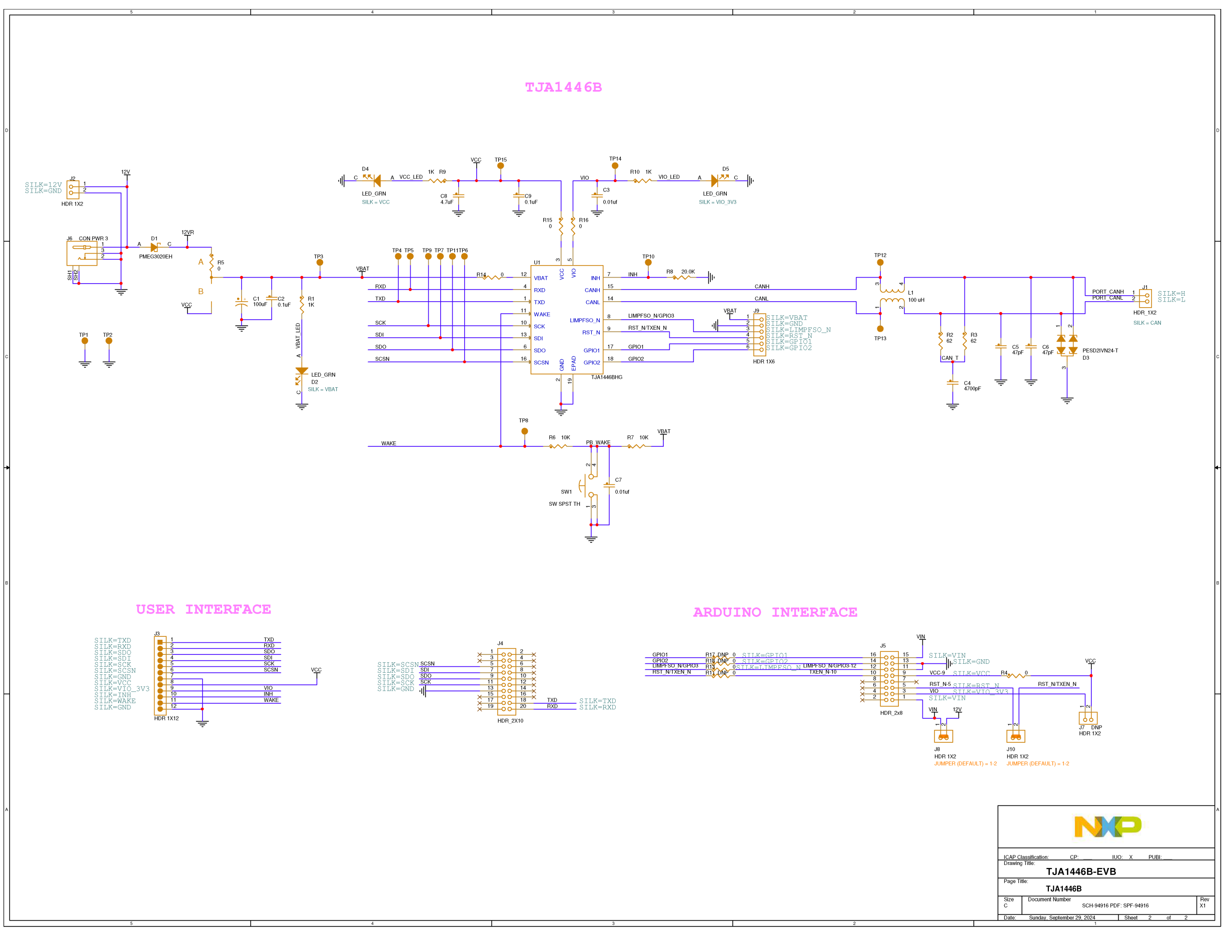


Figure 5. TJA1446B-EVB schematic diagram

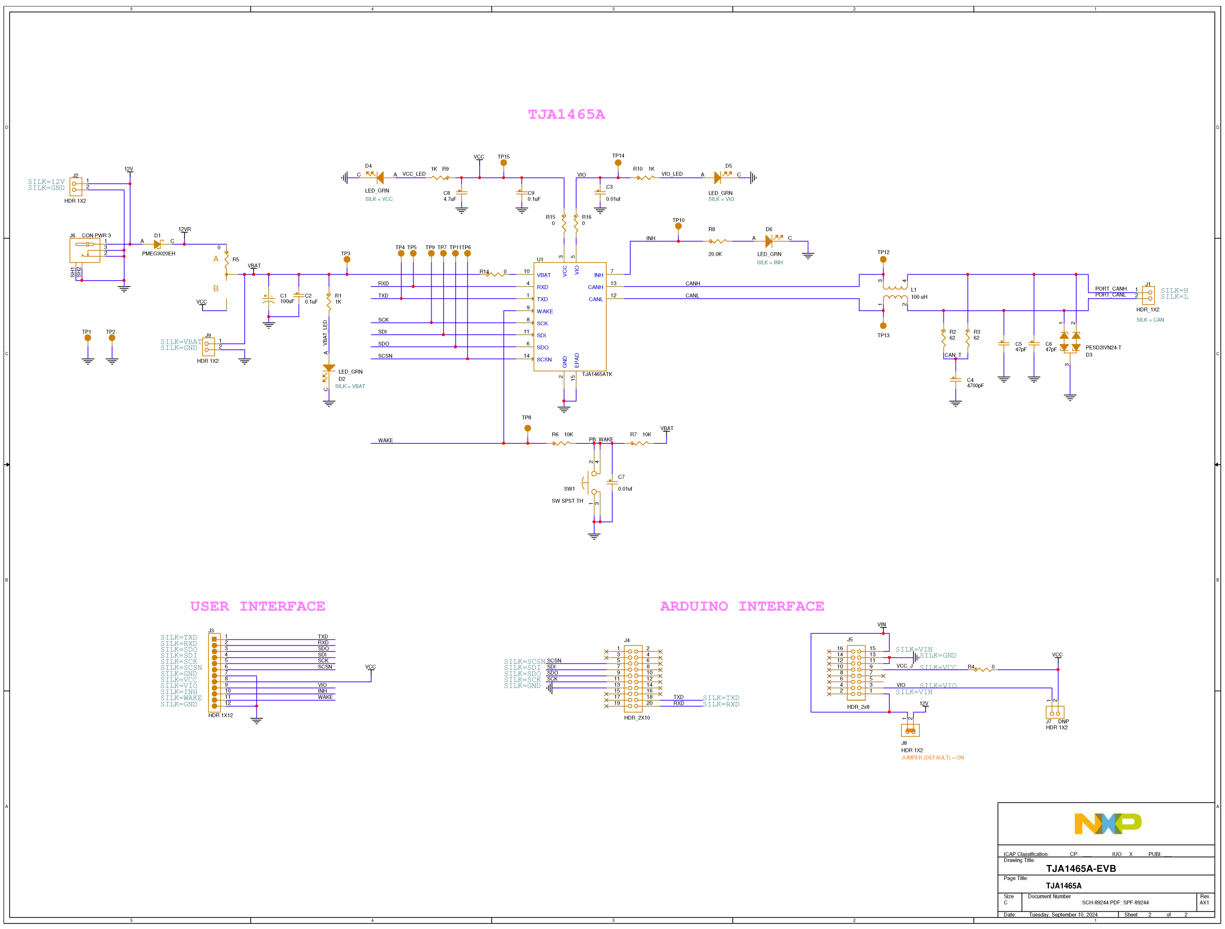


Figure 6. TJA1465A-EVB schematic diagram

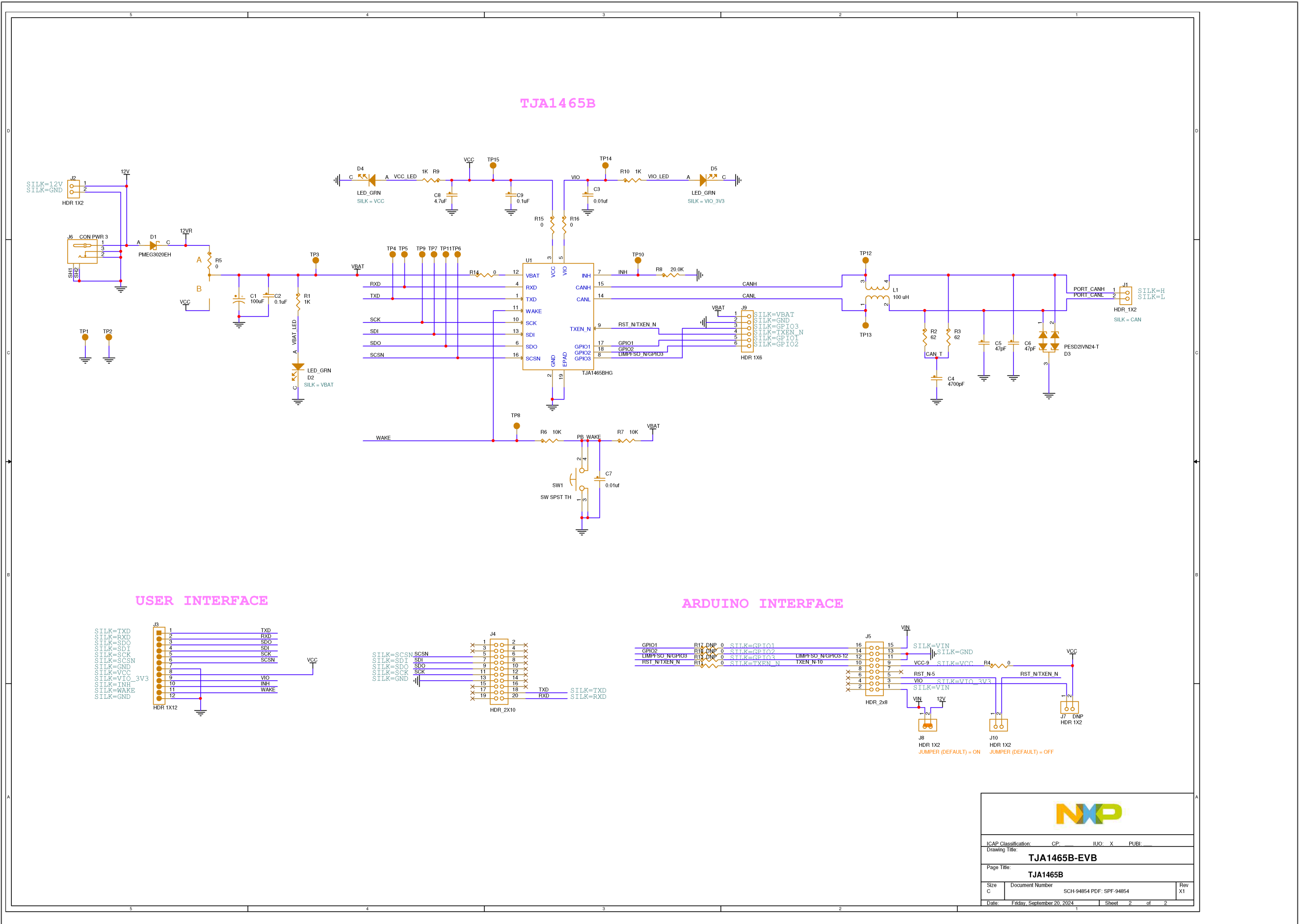


Figure 7. TJA1465B-EVB schematic diagram

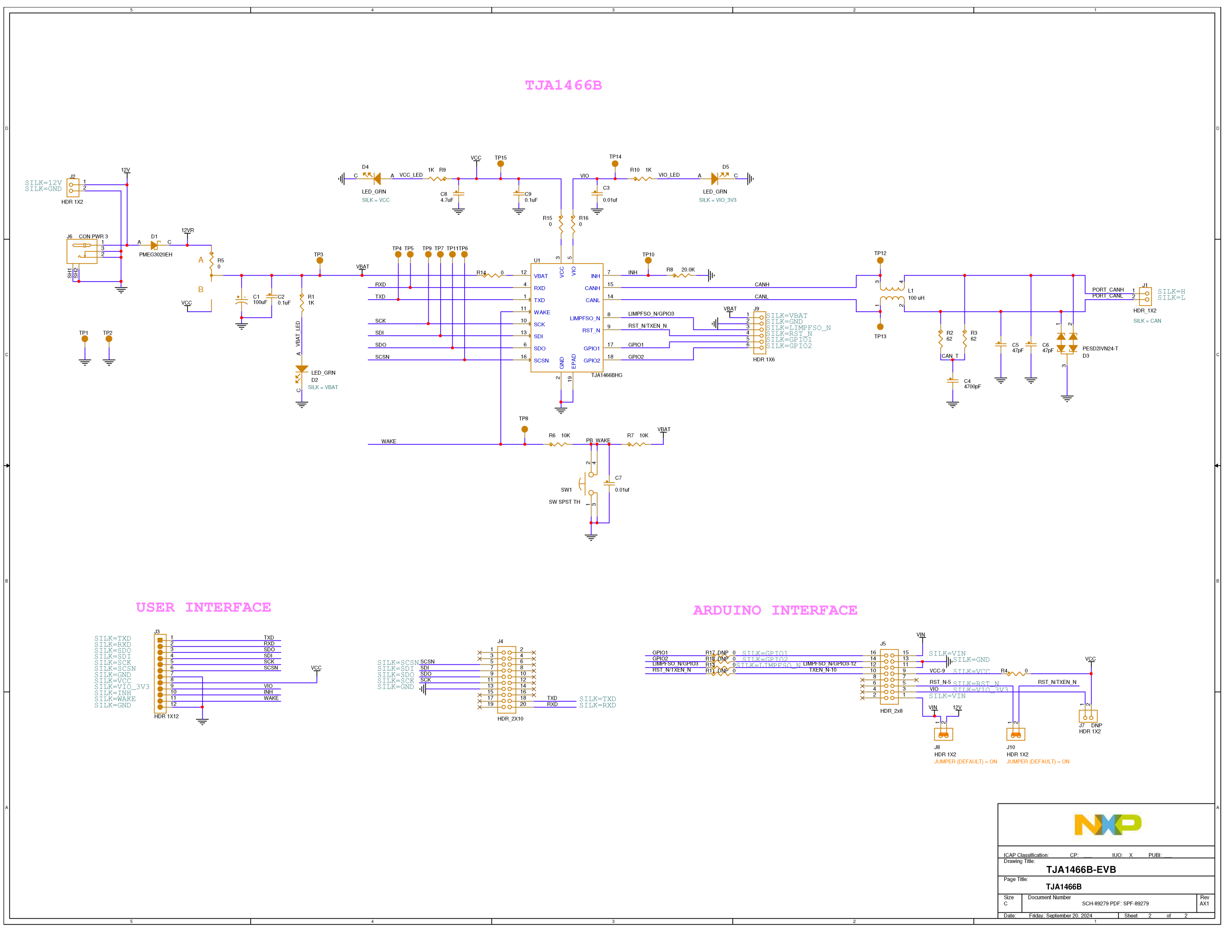


Figure 8. TJA1466B_EVB schematic diagram

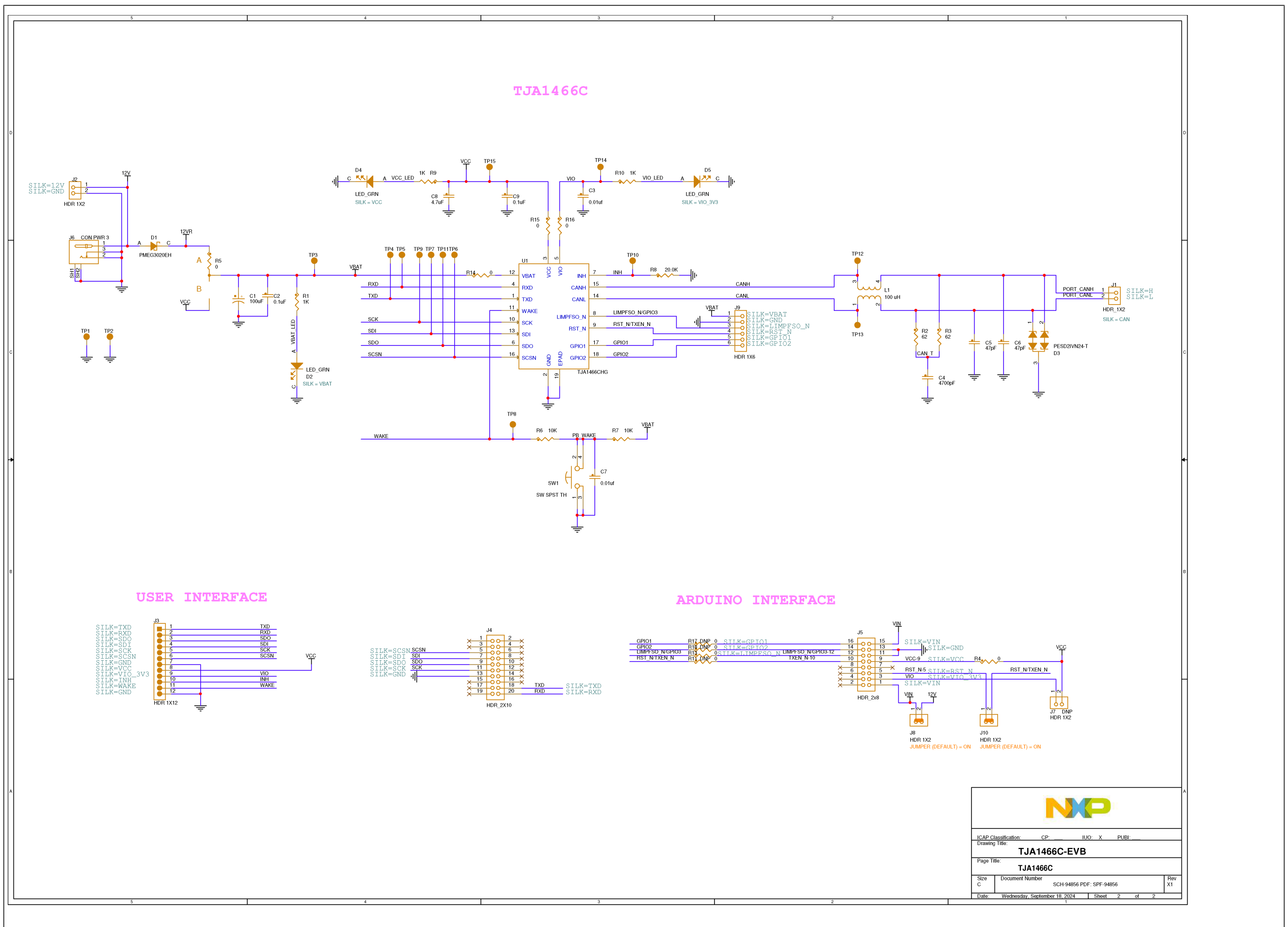


Figure 9. TJA1466C_EVB schematic diagram

5 FlexGUI: interactive register control via USB

When the TJA14xx-EVB is mounted on the microcontroller evaluation board, the microcontroller board can be used as a USB/SPI interface between the TJA14xx-EVB and a PC. After installing the FlexGUI application on a Windows PC (see Section 5.3), the contents of the TJA1445/46/65/66 registers can be viewed and/or changed interactively.

The FlexGUI software for the TJA14xx-EVB currently supports the S32K148-Q176 board. See nxp.com/S32K148-Q176 to learn more about the S32K148-Q176 board.

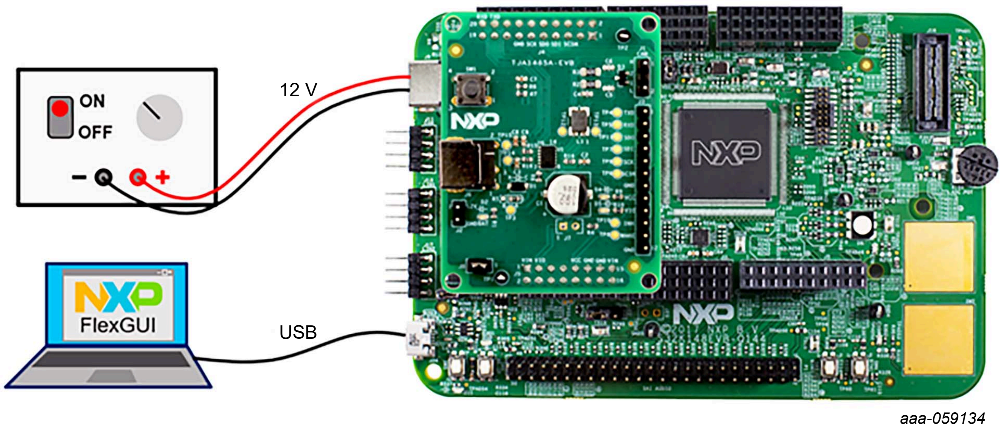


Figure 10. Using FlexGUI with the S32K148EVB as a USB/SPI interface

5.1 FlexGUI software package overview

The FlexGUI SW package for the TJA14xx-EVB can be downloaded from www.nxp.com/FlexGUI. It includes:

- the FlexGUI PC installer (see also Section 5.3)
- FlexGUI firmware for the microcontroller board (see also Section 5.2.1)

5.2 Preparations for using the S32K148EVB as a USB interface

The FlexGUI firmware must be loaded onto the S32K148EVB before connecting the TJA14xx-EVB.

5.2.1 FlexGUI firmware installation on S32K148EVB

Firmware programming in the S32K148EVB is straightforward:

1. Connect a 12 V power supply to the barrel jack.
2. Connect the board to the PC with a USB cable
3. Wait until the mass storage device S32K148EVB appears on the PC.
4. Copy the firmware file to that drive

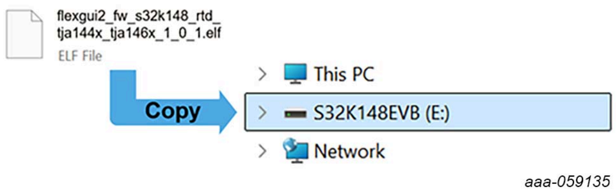


Figure 11. Copying the firmware file to the EVB-S32K144 drive

5.2.2 Hardware setup for FlexGUI operation

Once the FlexGUI firmware has been installed on the microcontroller board, the following steps need to be taken to ensure correct operation and to prevent damage to either of the evaluation boards.

5.2.2.1 Voltage selection

Before mounting the TJA14xx-EVB on the microcontroller board, it is recommended to set the V_{IO} and V_{CC} supply jumpers on the S32K148EVB as indicated in Table 3. Further information about the jumpers and available voltage levels can be found in the S32K148EVB Getting Started guide.

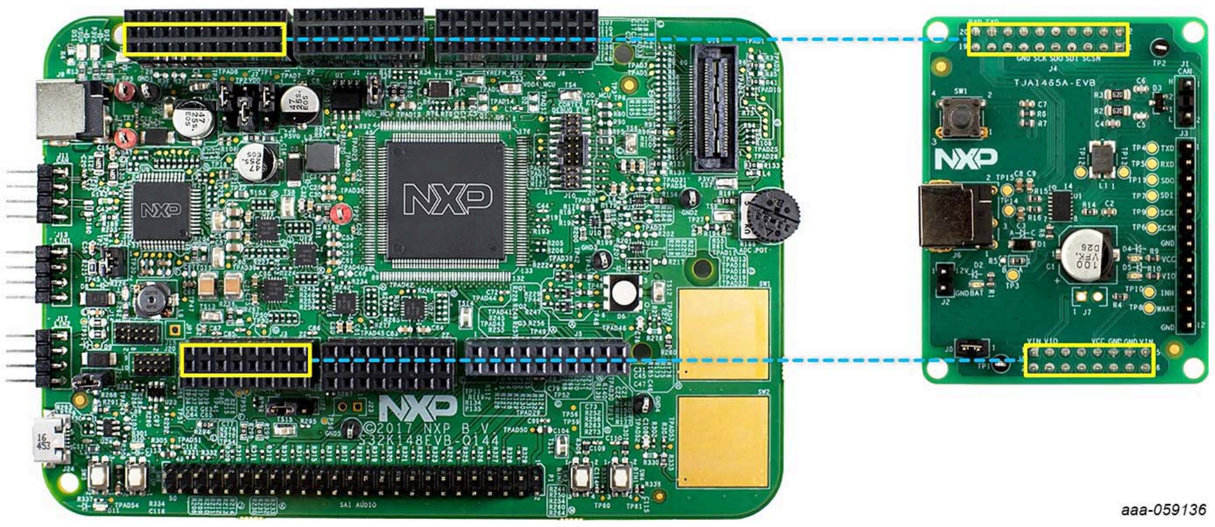
Table 3. Jumper setting on S32K148EVB

	J7 (MCU V_{DD})	J8 (5 V V_{CC})	J18 (3.3 V source)
TJA14x6A (1.8 V V_{IO})	not supported[1]		
TJA14x6B (3.3 V V_{IO})	1-2 (3.3 V)	1-2	1-2 (V_{BAT})
TJA14x6C (3.3 V - 5 V V_{IO})	1-2 (3.3 V)[2]	1-2	1-2 (V_{BAT})[2]
TJA14x5A/B	1-2 (3.3 V)	1-2	1-2 (V_{BAT})

[1] 1.8 V V_{IO} is not supported by the S32K148EVB.
[2] Although the TJA14x6C recommended V_{IO} level is 5 V, this FlexGUI setup is more reliable with this 3.3 V configuration.

5.2.2.2 Mounting the TJA14xx-EVB on the S32K148EVB board

Once the S32K148EVB board has been configured, the TJA14xx-EVB needs to be connected. The correct placement is shown in Figure 12.



Valid for all TJA14xx evaluation boards.

Figure 12. Connecting the TJA1465A-EVB and S32K144EVB for FlexGUI operation

5.2.2.3 Disconnecting the reset signal (TJA14x6-EVB only)

To prevent the MCU being reset by the TJA14x6, for example while the TJA14x6 is in Sleep mode, it is recommended to remove J10 on the TJA14x6-EVB. Removing this jumper cuts the reset connection between the TJA14x6 device and the MCU.

Avoiding unnecessary MCU resets ensures that the FlexGUI always remains responsive while the TJA14x6-EVB is being evaluated. The TJA14x6 ignores all SPI access during a reset or in Sleep mode.

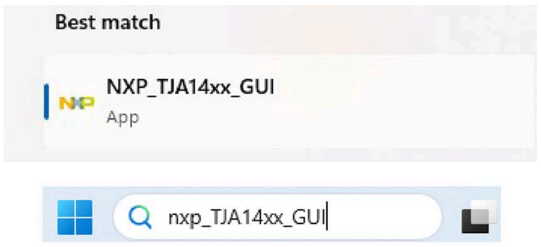
5.3 Installing the FlexGUI on a PC

Executing **NXP_TJA14xx_GUI-1.1.0.msi** will start the installation wizard. All options can be left at their default settings. Remember which installation folder path was selected. The default folder path may require access privileges. If this is a problem, choose a different path.

5.4 Using the FlexGUI

5.4.1 Starting the FlexGUI application

FlexGUI can be started by typing **NXP_TJA14xx_GUI** in the Windows Search bar (Figure 13a) or with the shortcut symbol on the desktop (Figure 13b).



aaa-059137

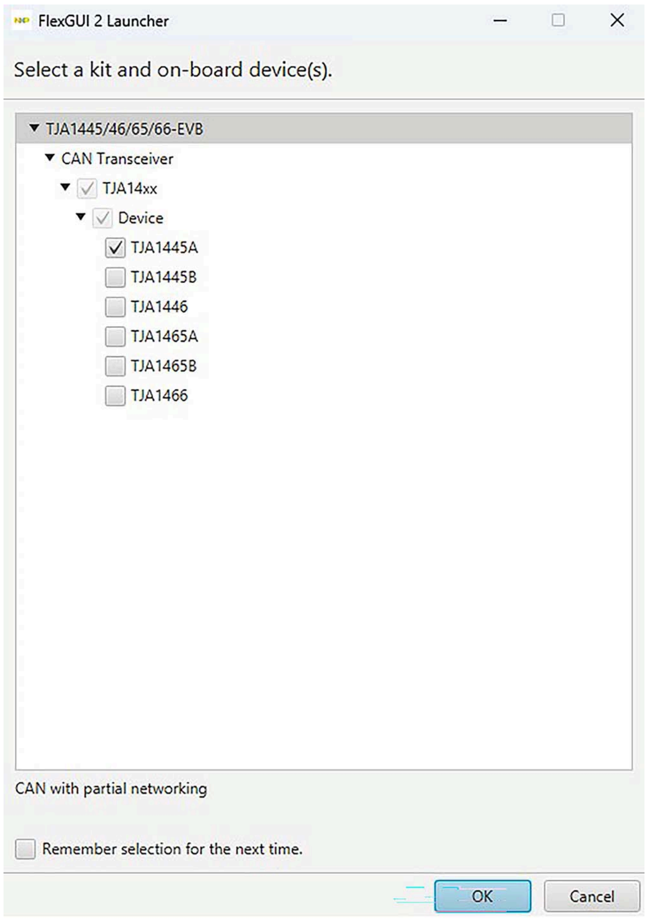
a. Windows search

aaa-059138

b. Desktop shortcut

Figure 13. FlexGUI start options

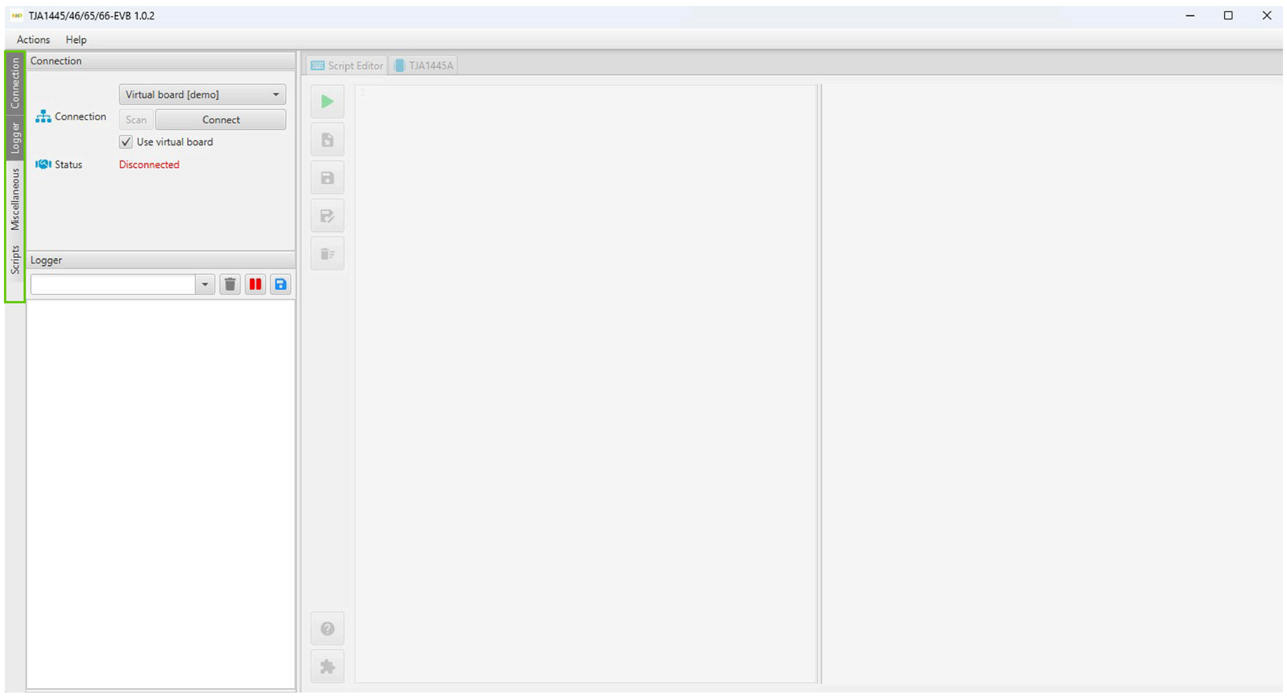
It may take a few seconds for the FlexGUI to load and display the launch window (Figure 14). The short-form type numbers of all supported devices are displayed in the launch window. Select the appropriate device and click '**OK**'.



aaa-059140

Figure 14. Launch window - EVB selection

Once loading is complete, the FlexGUI startup window is displayed (Figure 15). The connection status is ‘Disconnected’, indicating that the application has not yet established a logical connection to the board. Section 5.4.2 explains how to establish a connection.



aaa-059382

Figure 15. FlexGUI startup window; boards not yet connected

The FlexGUI startup window contains tabs for the Script Editor (Section 5.4.6) and the selected device. The device tab displays the register map (Section 5.4.5). A number of windows can be selected from the left side panel:

- The **Communication** window, selected by default, is used to detect and connect to a S32K148EVB board.
- The **Logger** window is used to display error messages.
- The **Miscellaneous** window shows the version numbers of the GUI, firmware and register map.
- The **Scripts** window provides the list of already loaded Python scripts..

5.4.2 Establishing a connection between the FlexGUI and the hardware

To establish a connection between the FlexGUI and the hardware, the microcontroller board needs to be connected to the PC with a USB cable (see Figure 10). When the board is connected for the first time, the PC installs the communication driver (virtual COM port) automatically. This may take a few seconds.

Once the USB link is ready, a FlexGUI session can be started:

- Click the **Scan** button in the upper left corner of the FlexGUI window (Figure 15) to detect all available serial connections.
- Identify and select the COM port on the board. It is usually the last item on the list if no other USB cables have been connected to the PC since the board was plugged in.
- Click **Connect** to enable the connection.

If the board was connected before starting the FlexGUI, it may already be selected. In this case, you can skip the first two steps and immediately press **Connect**.

The Status should change from Disconnected to Connected.

5.4.3 Watchdog (TJA14x6) and MCU reaction timeout (TJA14x5) handling

The FlexGUI firmware takes care of handling the watchdog to prevent a reset. Watchdog answers are sent at intervals of 70 % of the watchdog period configured in the TJA14x6 device. The default values are 200 ms (watchdog period) and 140 ms (handling interval).

At power-on, the FlexGUI firmware prevents an MCU reaction timeout event being triggered on the TJA14x5 device by reading the device ID. However, after the device wakes up from Sleep mode, the user must send a valid SPI command within $t_{to(MCU)}$. Otherwise the transceiver will return to Sleep mode and the user will then need to change the device mode via an SPI command..

5.4.4 Using the FlexGUI software without hardware

The FlexGUI can be used without hardware. Clicking on **Use virtual board** (see Figure 15) selects a board called 'Virtual board [demo]'. After clicking **Connect**, FlexGUI register operations can be performed as if physical hardware was connected. Random data is displayed when reading registers.

5.4.5 Register map

When a connection has been established with the connected evaluation board (or the 'virtual' board), the **Script Editor** tab in the startup window is selected by default. Select the tab labeled with the selected CAN transceiver to display the register map of the selected device. Device registers can be read or written to interactively via this window.

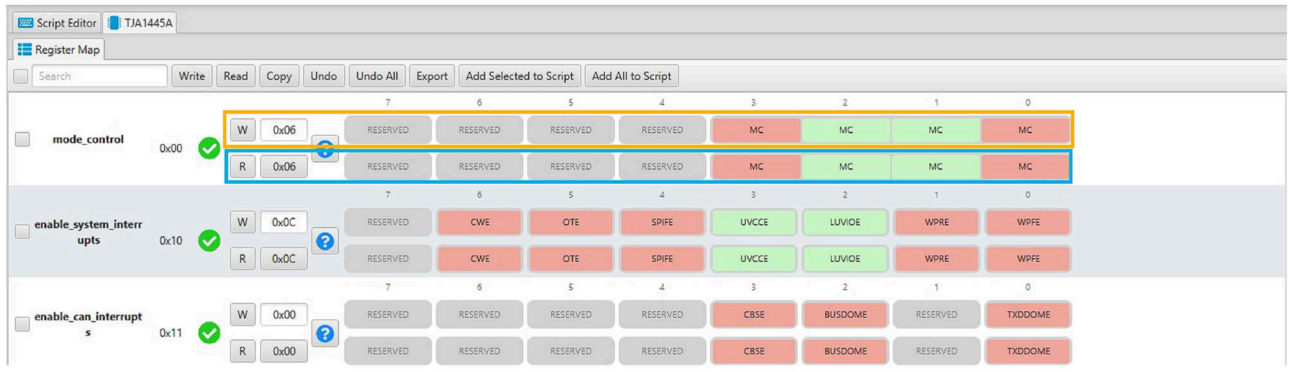


Figure 16. Register map tab

Register data can be edited in the top row (outlined in orange) in preparation for writing to the register.

Actual register contents as received from a previous read access is shown in the lower row (outlined in blue).

Register data is displayed in three formats:

- As a single hexadecimal value for the entire register
- As individual hexadecimal values for each bit-group, when clicking the "**?**" button
- A color-coded button for each bit:
  - red = 0 (cleared)
  - green = 1 (set)

For each register, individual write and read operations can be triggered using the 'W' and 'R' buttons. Clicking the hexadecimal value next to the 'R' button overwrites the value in the upper row with the read value in the lower row.

If no values have been read or written previously, the default values are shown. This behavior can be modified (see Section 5.4.7).

Multiple registers can be selected using the checkboxes to the left of the register names. The selected registers will be included in later multi-register operations. The following associated buttons are provided:

- **Write** and **Read** trigger write and read operations, respectively.
- **Copy** can be used to copy data from the 'read' row(s) to the selected 'write' row(s). This does not work if the read row contains unwritten changes.
- **Undo** undoes changes made to the 'write' row(s) since the most recently executed write action(s) on the associated register(s). If a register has not been written to previously, the selected rows are reinitialized with the default values.
- **Export** writes a table of register names and data to a file or to the clipboard.
- **Add to Script** adds write commands for all selected registers to the Script Editor.
- **Add all to Script** adds write commands for all registers, selected or not, to the Script Editor.

For each register, an 'OK' (✓), 'ERROR' (X) or 'pencil' (✎) symbol is displayed to the left of the W/R buttons. The ✓ symbol indicates that the last operation was successful (read or write). The X symbol indicates that the last operation failed, most likely due to a lost connection or operation on a reserved field. A ✎ symbol indicates that the data in the editable text field differs from the data previously written to the register (or from the default values).

5.4.6 Script editor

Selecting the **Script editor** tab opens a tool for creating, executing, loading and saving Python command sequences ('scripts'). These are used for reading from or writing to registers as well as sending and receiving CAN messages.

CAN messages are sent with a default bit rate of 500 kbit/s for the arbitration phase or non-FD frames, and 5 Mbit/s for the data phase when using the CAN FD bit rate switch.

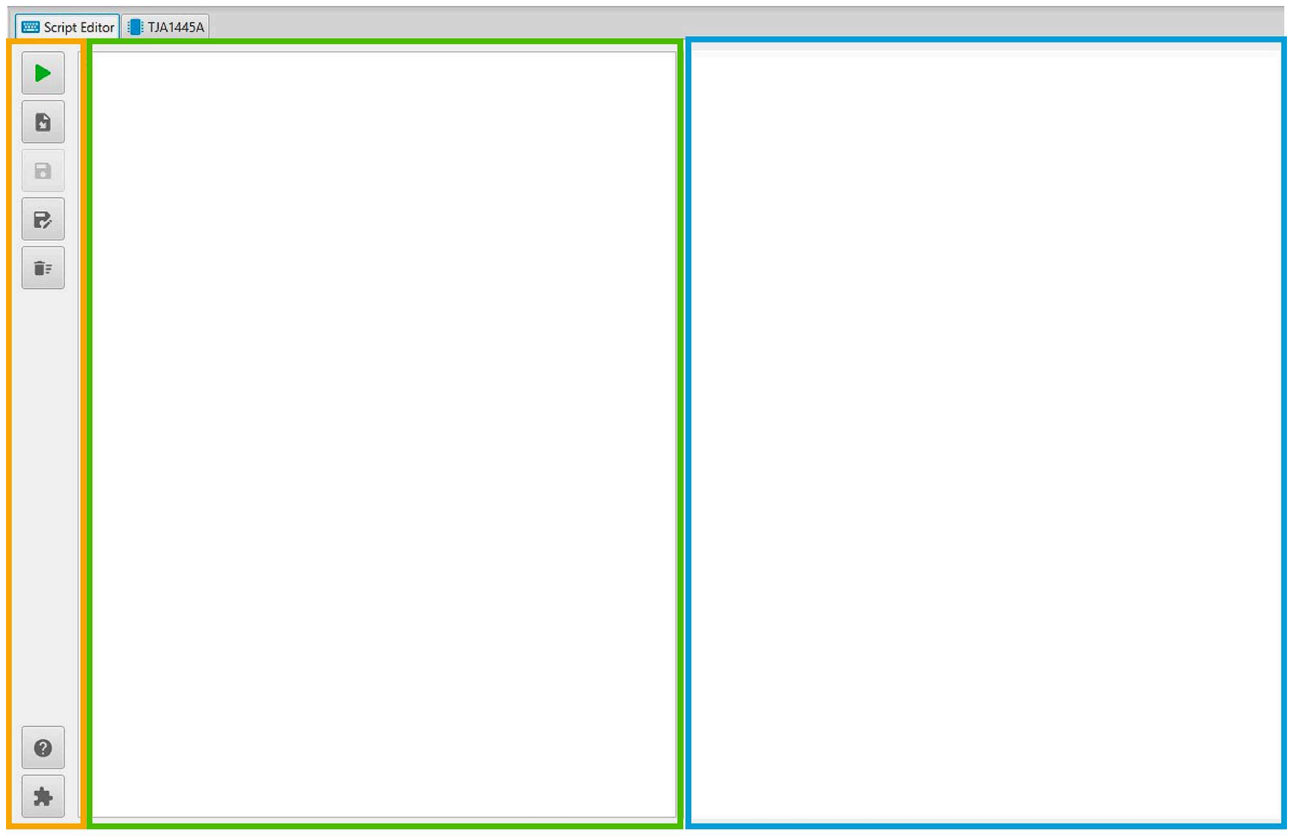


Figure 17. Script editor window

The script editor contains three main panels:

The action panel (outlined in orange)

Using the buttons at the top of the panel, the script can be:

- executed (triangle symbol)
- loaded from a file (file symbol)
- saved, if already associated with a file (floppy disk symbol)
- stored in a new file (floppy disk with pen symbol)
- wiped from the code editor (trashcan symbol)

The buttons at the bottom of the panel can be used to:

- open the script help window (? symbol)
- open the command generator (puzzle piece symbol)

The code editor (outlined in green)

The script code is displayed and can be edited here.

For information on how to write scripts, see the *FlexGUI Script Editor Manual* and example scripts, both included with the FlexGUI package.

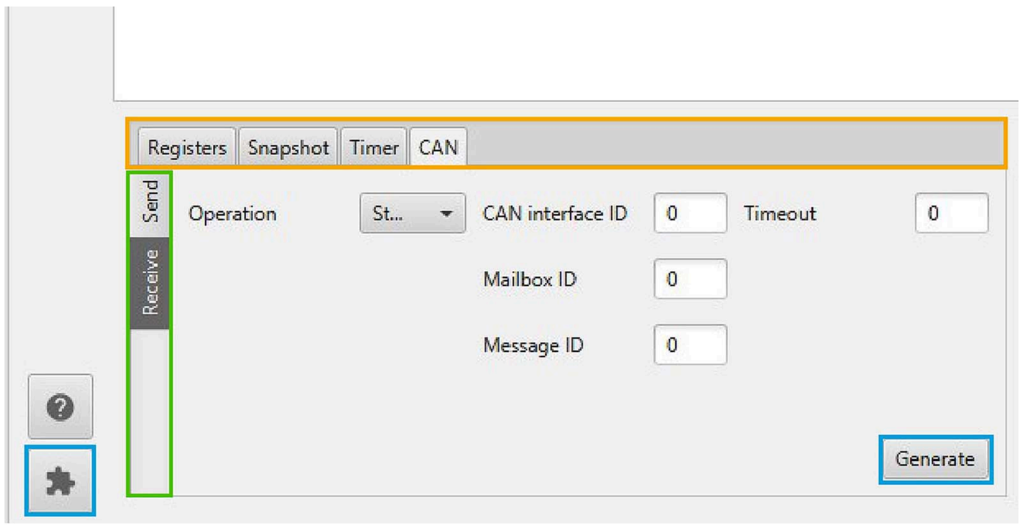
Scripts can also be created using the Command Generator without writing code. The Command Generator is explained in Section 5.4.6.1.

The output panel (outlined in blue)

The script output is shown here. This includes printed strings and errors. Be aware that errors not generated by the script itself may not be shown here but in the **Logger** section of the main window (Figure 15).

5.4.6.1 Command generator

The command generator (selected via the puzzle piece symbol) generates script commands from the input provided. The command category is selected using the tabs in the top panel. Multiple commands are available for some categories, selected from the left panel. Click Generate to add the generated command(s) at the current cursor position.



aaa-059144

Figure 18. Command generator

The **Registers** tab can be used to generate read or write commands for any register. The appropriate group/set must be selected before a register can be selected. For the register map version 1.0, only one group exists.

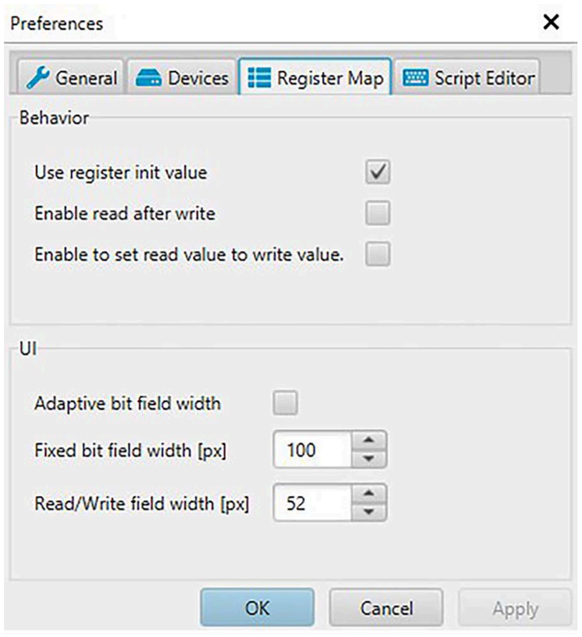
The **Snapshot** tab is used to quickly generate write commands for every register. This function can be used to restore the current state of the transceiver at a later date. If the Append mode checkbox is ticked, the commands will be inserted at the current cursor position; otherwise the contents of the code editor will be overwritten.

The **Timer** tab can be used to insert a **Delay** of a given number of microseconds or a **Pause** until a user acknowledgement has been received.

The **CAN** tab can be used to **Send** or **Receive** CAN messages - either via the transceiver on the TJA14xx-EVB (interface ID = 0) or via the transceiver (SBC UJA1132) on the S32K148EVB (interface ID = 1). CANH/CANL on header J1 of the TJA14xx-EVB must be connected to CANH/CANL on header J11 of the S32K148EVB to enable communication between these two interfaces.

5.4.7 Preferences

The Preferences window is opened by clicking on Actions in the top-left corner of the FlexGUI start-up window (Figure 15) and selecting 'Edit Preferences'.



Behavior

Use register init value [✓]

Enable read after write [ ]

Enable to set read value to write value. [ ]

UI

Adaptive bit field width [ ]

Fixed bit field width [px] 100

Read/Write field width [px] 52

aaa-059145

Figure 19. Preferences

The **General** tab is used to modify logging and polling settings. In most cases, this will not be necessary.

The **Devices** tab provides settings that are not applicable to this single-device setup.

Register Map preferences are split into two groups, **Behavior** and **UI**.

- The most important Behavior options are:
  - **Use register init value** sets default write values in the register map; otherwise, it will contain all zeros.
  - When **Enable read after write** is selected, a read operation is automatically performed after every write operation. This can be used to ensure that the register has been updated correctly.
- Available UI settings are:
  - **Fixed bit field width** sets the display width of the bit fields in the register map (if 'Adaptive bit field width' is not selected)
  - **Read/write field width** sets the display width of the hexadecimal value fields next to the R and W buttons

The **Script Editor** tab displays the 'Generate register name' checkbox. Ticking this box causes commands (generated in the script generator or via the 'Add to Script' buttons in the register map) to address registers by name as a string and not using hexadecimal values.

6 Revision history

Table 4. Revision history

Document ID	Release date	Description
UM12257 v.1.0	3 June 2025	• Initial version

Legal information

Definitions

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

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

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

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

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

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

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

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

Suitability for use in automotive applications (functional safety) — This NXP product has been qualified for use in automotive applications. It has been developed in accordance with ISO 26262, and has been ASIL classified accordingly. If this product is used by customer in the development of, or for incorporation into, products or services (a) used in safety critical applications or (b) in which failure could lead to death, personal injury, or severe physical or environmental damage (such products and services hereinafter referred to as “Critical Applications”), then customer makes the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, safety, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. As such, customer assumes all risk related to use of any products in Critical Applications and NXP and its suppliers shall not be liable for any such use by customer. Accordingly, customer will indemnify and hold NXP harmless from any claims, liabilities, damages and associated costs and expenses (including attorneys' fees) that NXP may incur related to customer's incorporation of any product in a Critical Application.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

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

NXP — wordmark and logo are trademarks of NXP B.V.

Contents

Date of release: 3 June 2025
Document identifier: UM12257