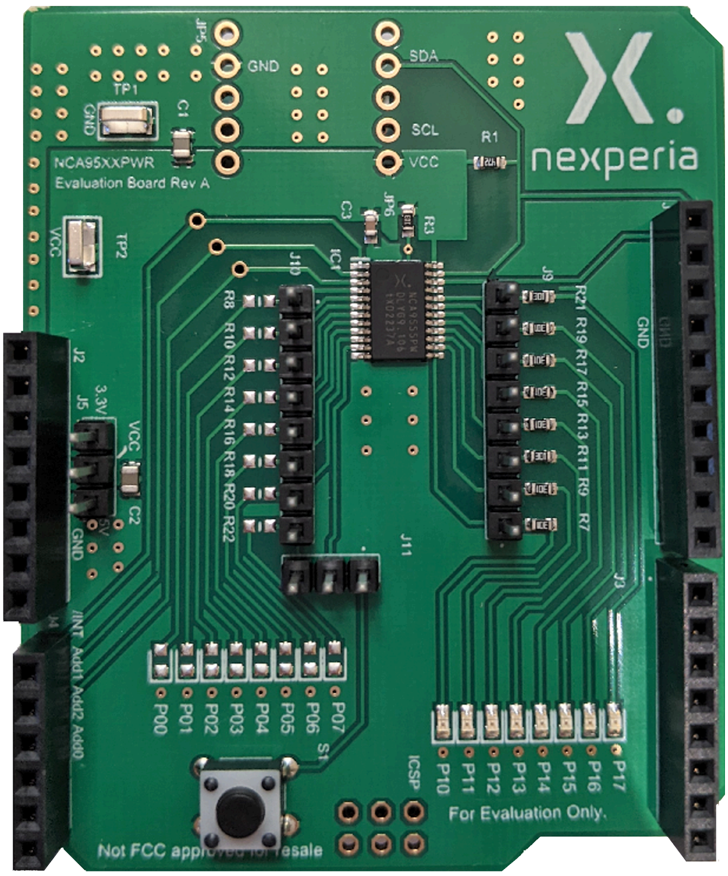
UM90016

Rev. 2.0 — 19 January 2024 **user manual**

NCA95xx GPIO expander EVB (Arduino™ shield compatible)



Abstract: The NCA95xx EVB (Evaluation Board) is an Arduino™ compatible shield form factor PCB designed for the Nexperia I²C controlled GPIO expander family. The board provides convenient test points for GND, I²C-Bus signals SCL and SDA as well as an interrupt open drain output. The EVB can be used for the GPIO expander derivatives NCA9555, NCA9535, NCA9539 and NCA9595.

Keywords: General Purpose Input/Output (GPIO) Expander, Evaluation Board (EVB), Arduino™ shield

1. Introduction

The NCA95xx EVB (Evaluation Board) is a PCB designed for the Nexperia I²C controlled GPIO expander family. The EVB arrives enclosed in an antistatic ESD bag with labeling. The board provides convenient test points for GND, I²C-Bus signals SCL and SDA as well as an interrupt open drain output. The EVB can be used for the GPIO expander derivatives NCA9555, NCA9535, NCA9539 and NCA9595.

The EVB is designed in an Arduino™ compatible shield form factor that allows easy development with Nexperia's line of available I²C GPIO I/O expanders. There is also supplementary firmware for the Arduino™ to aid in the development of applications with the NCA95xx family of devices. An image of the NCA95XX EVB is shown in Fig. 1

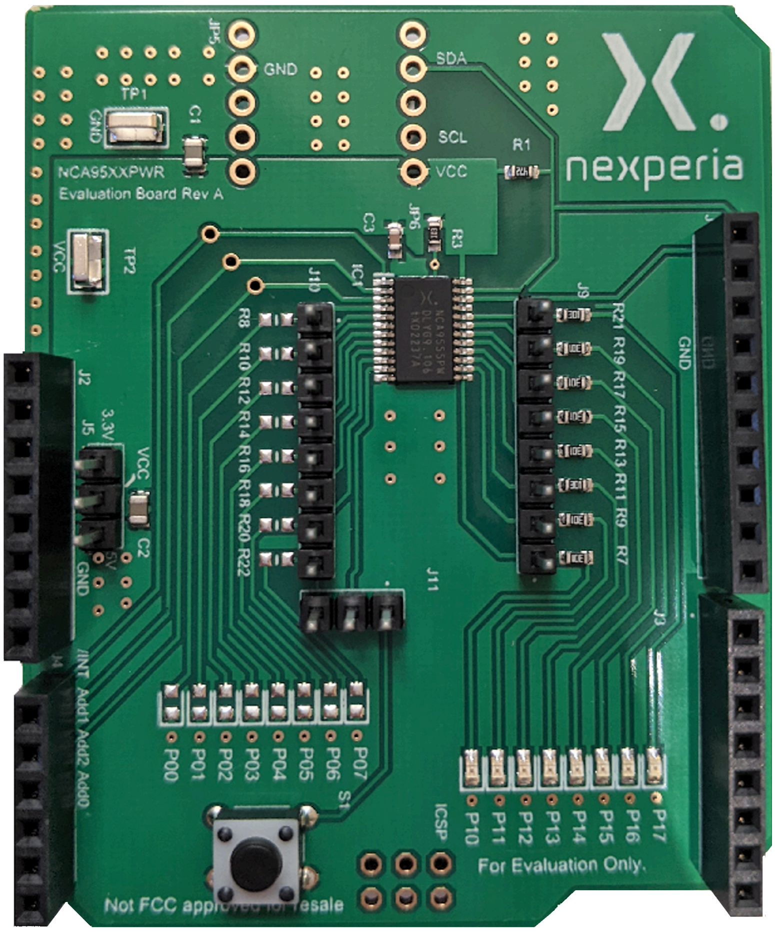


Fig. 1. NCA95xx (Arduino™ shield compatible) Evaluation Board

1.1. Features

- EVB name = NCA95xx EVB
- Device = NCA95xx
- Input voltage = 1.6 V to 5.5 V
- I²C to parallel port expansion
- Number of GPIOs = 2 ports with 8 bits
- Current drive capability = ± 25 mA
- I²C interface, Fast-Mode 400 kHz
- Open-drain active-low interrupt output
- Configurable slave address via 3 selection pins (2 pins for NCA9539)
- Low-active reset input (NCA9539)
- Polarity inversion register (for read operation)

Further details about the specification and parameters of NCA9555, NCA9535, NCA9539 and NCA9595 can be found in their respective data sheets, but a high-level summary is provided in Table 1 below:

Table 1. NCA95xx family features

Device	Package	Feature
NCA9555PW	TSSOP24	• integrated 100 kΩ pull-up resistors for GPIO pins
NCA9555BY	HWQFN24	
NCA9535PW	TSSOP24	• no integrated termination of GPIO pins • advantage: no extra current for active outputs in low state
NCA9595PW		• 2 additional I²C registers for switchable pull-up resistors
NCA9539PW		• additional Reset pin (low active) A2 address selection pin is sacrificed (4 address options) • no internal termination like NCA9535

2. Hardware set up

The proceeding section describes the EVB jumper and header description, installation of firmware on the Arduino™ UNO board or compatible alternative and installation of the software (GUI) on PC.

2.1. Headers J1/J2/J3/J4 - Arduino™ Uno interface

Headers J1, J2, J3, and J4 serve as the physical interface between the Arduino board and the EVB. The included software supports fast development and evaluation with the NCA95XX devices. If using other microcontrollers, or wiring to an application, these headers also allow quick prototyping by either directly connecting to a breadboard or using wires to directly connect nets to customer applications.

A detailed description of the header pinouts are listed in the tables presented below:

Table 2. J1 headers

J1 net name	Pin number	Description
SCL	1	SCL I²C-Bus Signal Line
SDA	2	SDA I²C-Bus Signal Line
GND	4	GND connection

Table 3. J2 headers

J2 net name	Pin number	Description
3.3 V	4	3.3 V Arduino Supply Voltage
5 V	5	5V Arduino Supply Voltage
GND	6	GND connection
GND	7	GND connection

J3 header has no connections from NCA95XX device. Please reference Arduino pin table.

Table 4. J4 headers

J4 net name	Pin number	Description
/INT	1	/INT pin (9555)
Add1	2	Addr 1 pin
Add2	3	Addr 2 pin
Add0	4	Addr 0 pin

2.2. Header J5 - selectable VCC

J5 allows a shunt to connect VCC to either 3.3 V or 5 V core voltages. As the Arduino UNO is native to 5 V, the default option is for the shunt to connect to 5 V supply.

Table 5. J5 header - selectable VCC supply

J5 net name	Pin number	Description
3.3 V	1	3.3 V from Arduino
VCC	2	Device supply
5 V	3	5 V from Arduino

Note: If using 3.3 V with Arduino UNO, please verify that address connections to Arduino microcontroller are set to inputs. If setting the Address pins low the digital pin can then be reconfigured as output and set to LOW. All other I/Os on the NCA95XX are 5 tolerant.

2.3. JP5 and JP6 headers

These headers serve two purposes:

- Additional connection points to VCC, SCL, SDA, GND and other NCA95XX nets
- The EVB can also be evaluated with the Adafruit Trinket M0.

Table 6 and Table 7 display the header pinout with Net connections.

Table 6. JP5 header net names

JP5 net name	Pin number	Description
GND	2	GND connection
/INT	3	/INT pin (9555)
Add1	4	Addr1 net

Table 7. JP6 header net names

JP6 net name	Pin number	Description
VCC	1	VCC Supply voltage
SCL	2	SCL I²C-BUS signal line
Add2	3	Addr2 net

2.4. Headers J9/J10/J11 - NCA95XX GPIO

These headers provide direct connection to the GPIO pins of the populated NCA95XX device. GPIO0 through GPIO7 refer to PORT0, and GPIO10 to GPIO17 refer to PORT1. In the EVB, PORT1 I/Os are connected to a 300 Ω series resistor and LED. Configuring PORT1 as outputs through software will allow the user to turn on/off their respective I/O LED.

These components are DNI on PORT0. However, the footprints are included in the case the user wishes to manually populate. J11 includes a selectable interface between these footprints and a push button. The push button can be connected to GPIO7 of PORT0.

Table 8. J9 pinout

J9 net name	Pin number	Description
GPIO0	1	GPIO, DNI Resistor/LED
GPIO1	2	GPIO, DNI Resistor/LED
GPIO2	3	GPIO, DNI Resistor/LED
GPIO3	4	GPIO, DNI Resistor/LED
GPIO4	5	GPIO, DNI Resistor/LED
GPIO5	6	GPIO, DNI Resistor/LED
GPIO6	7	GPIO, DNI Resistor/LED
J11- pin 3	8	GPIO, DNI Resistor/LED

Table 9. J10 pinout

J10 net name	Pin number	Description
GPIO10	1	GPIO, Populated Resistor/LED
GPIO11	2	GPIO, Populated Resistor/LED
GPIO12	3	GPIO, Populated Resistor/LED
GPIO13	4	GPIO, Populated Resistor/LED
GPIO14	5	GPIO, Populated Resistor/LED
GPIO15	6	GPIO, Populated Resistor/LED
GPIO16	7	GPIO, Populated Resistor/LED
GPIO17	8	GPIO, Populated Resistor/LED

Table 10. J11 pinout

J11 net name	Pin number	Description
Push-button with pull-up	1	Connecting shunt to 1-2 of J11 will connect push button
GPIO07	2	GPIO, DNI resistor/LED
Resistor and LED footprint	3	Connecting shunt to 2-3 of J11 will connect I/O to resistor and LED footprint

3. Software Setup

3.1. Installing the Arduino™ software

Download the software files from the EVM product page. The directory should resemble the following with title “NCA95XX_GUI_Installation_Files”:

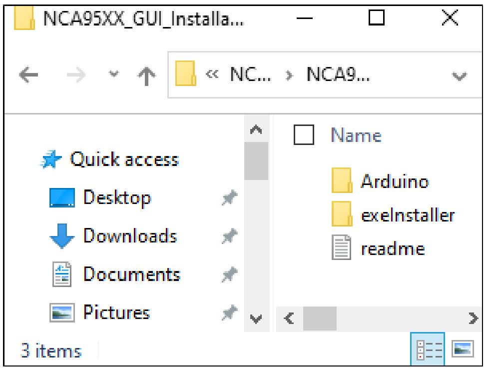


Fig. 2. GUI Installation Files

Navigate to the Arduino folder and run “Arduino-1.8.19-windows” or latest Arduino installer executable. This will download necessary drivers. Once complete, verify properly installed by verifying “Arduino” program in Start Menu, this is shown in Fig. 3 below.

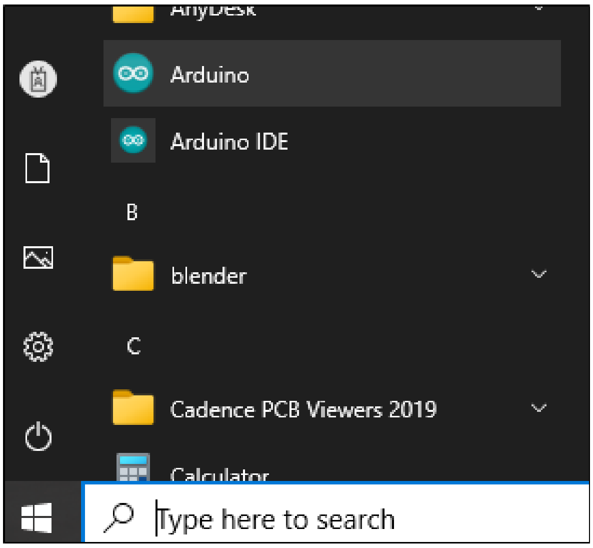


Fig. 3. Arduino program at startup menu

After the Arduino software is installed, connect the NCA95XX EVB to the Arduino UNO board and connect the USB cable to PC. Please verify correct VCC is set for the desired operation.

Now navigate in the folder to “NCA95XX_GUI_Installation_Files>Arduino>nca95xx_main” and click on the “nca95XX_main” file. Once the Arduino IDE is up, select the correct board and port by clicking into “Tools>Board” and “Tools>Port”, shown in Fig. 4. If the Arduino UNO is connected to the PC via USB, the PORT information can be found by navigating to the “Device Manager” tool and verifying port information under “Ports (COM&LPT)”.

NOTE: Upload of the nca95XX_main program is required for PC GUI "NCA95XX EVB GUI" evaluation.

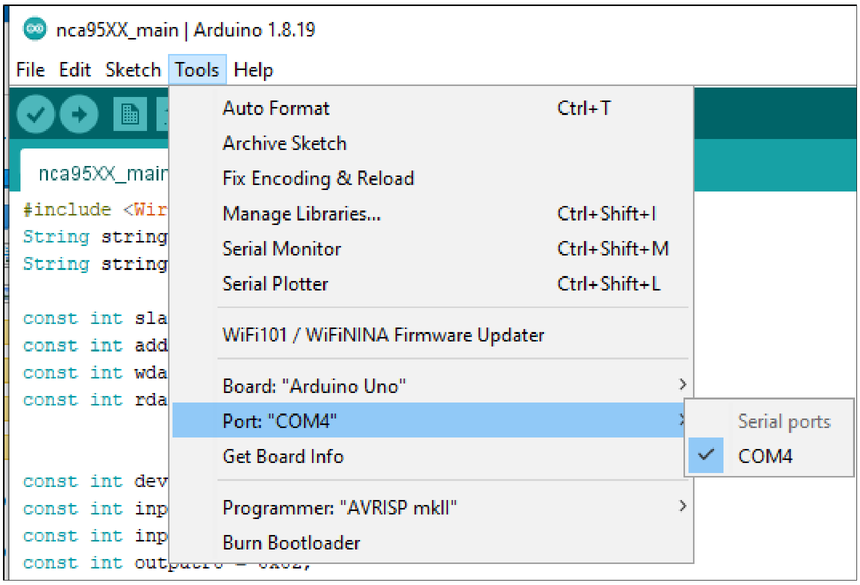


Fig. 4. Set Port COM Number

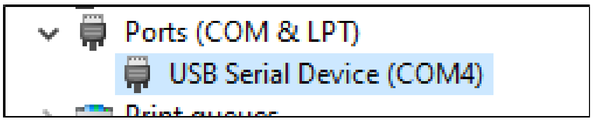


Fig. 5. Device Manager to Find COM port

Next navigate to “Sketch>Upload” to flash the program to the board, this is shown in Fig. 6.

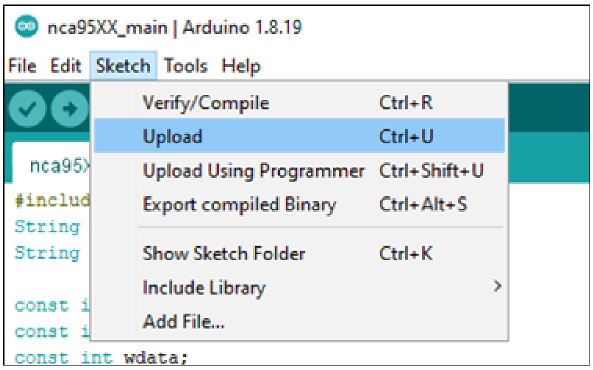


Fig. 6. Uploading the Arduino sketch

If successfully programed, the bottom of the IDE will display “Done uploading”. Now exit the Arduino IDE.

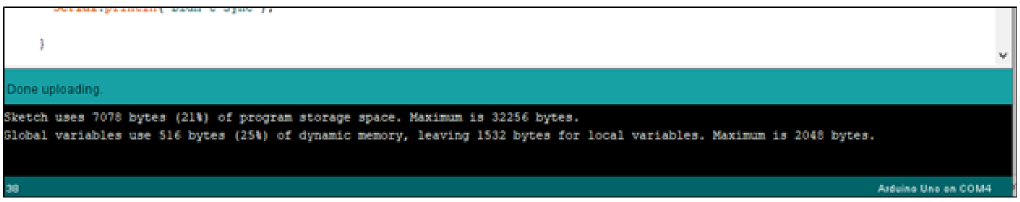


Fig. 7. Arduino sketch upload complete

3.2. GUI software setup

The next step is to navigate to the “exeInstaller” folder and double-click the “NCA95XX” setup file. The application is shown below, in Fig. 8, for reference.

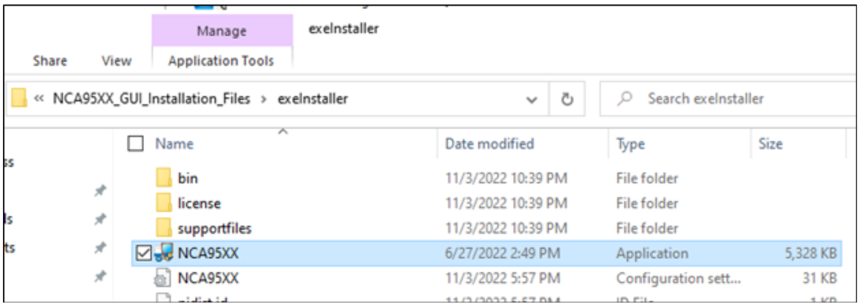


Fig. 8. Setup executable for NCA95XX EVB GUI

This will invoke the Installer, which is shown in Fig. 9. Please click next and accept the license agreements to install the NCA95XX EVM GUI.

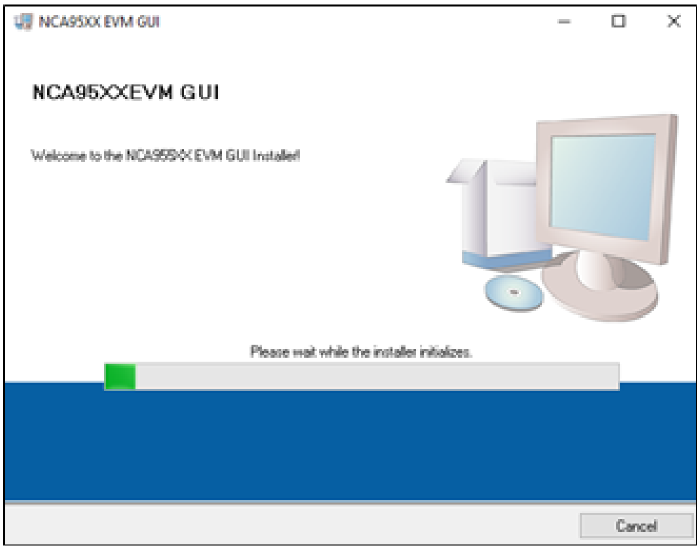


Fig. 9. NCA95XX EVB GUI installer screen

After everything installs correctly, you should see the following at your startup menu:

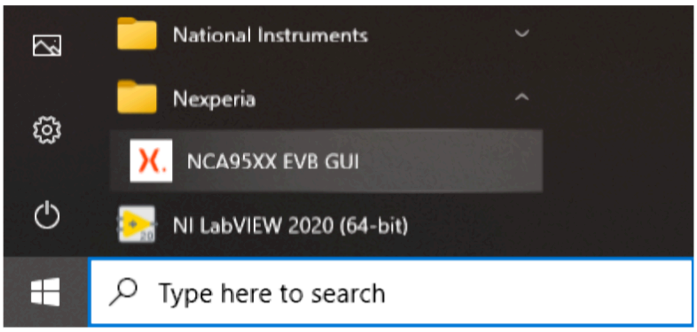


Fig. 10. NCA95XXEVB GUI screen on startup

Clicking on the “NCA95XX EVM GUI” icon will invoke the GUI displayed in Fig. 13.

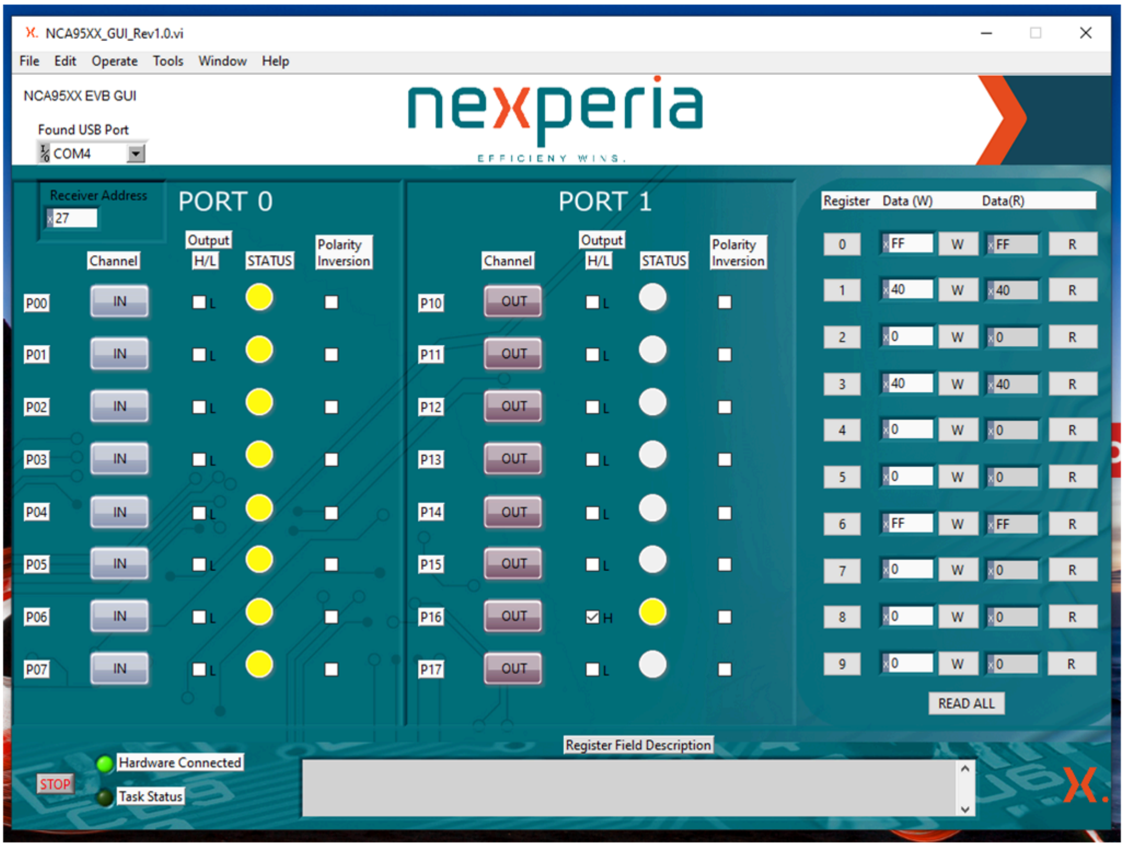


Fig. 11. NCA95XX GUI program

At startup, the GUI will automatically search for the correct COM PORT that the NCA95XX EVB and Arduino is connected to – this will take a few seconds. Once connected successfully, the “Hardware Connected” LED will turn on, indicating that the boards are found. Another LED, “Task Status” will activate, this LED means that a task is being completed, and will deactivate once the task is complete. This indicator is similar to the Windows “hourglass” state.

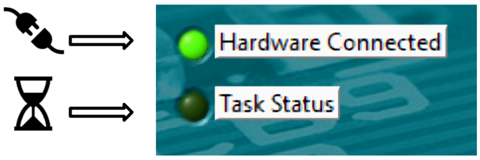


Fig. 12. Hardware Connected and Task Status

After the GUI finds the hardware, and initial routine is executed to find the current state of the NCA95XX device, the controls and indicators will automatically update and the “Task Status” light will turn off. After this initial task is performed, the user can now interact with the GUI through writes/reads or button presses. Additionally, when pressing buttons the “task status” light will turn on when an operation/task is performed, please wait for this light to turn off before proceeding with other button presses.

The GUI is split into two different levels of Interaction, shown in Fig. 13. The first Interactive region, shown in the “Yellow Box” are bit representations of the addresses that control GPIO configuration, output state, GPIO Read Status and Polarity Inversion. This is meant to serve as a high level of interaction with the device to manually select GPIO channel features.

The “Orange Box” represents a Lower Level of Interaction by directly writing hexadecimal values to the Registers. Reads on the registers can also be performed and are updated on their respective status indicator.

As the two, colored boxes are not directly tied to one another, to properly synchronize the two the user must press "Read All" to synchronize both blocks if going to and from the other box during interactive evaluation. Additionally, input port registers 0/1, which represents the GPIO STATUS LEDs are updated only when the "READ ALL" Button is pressed

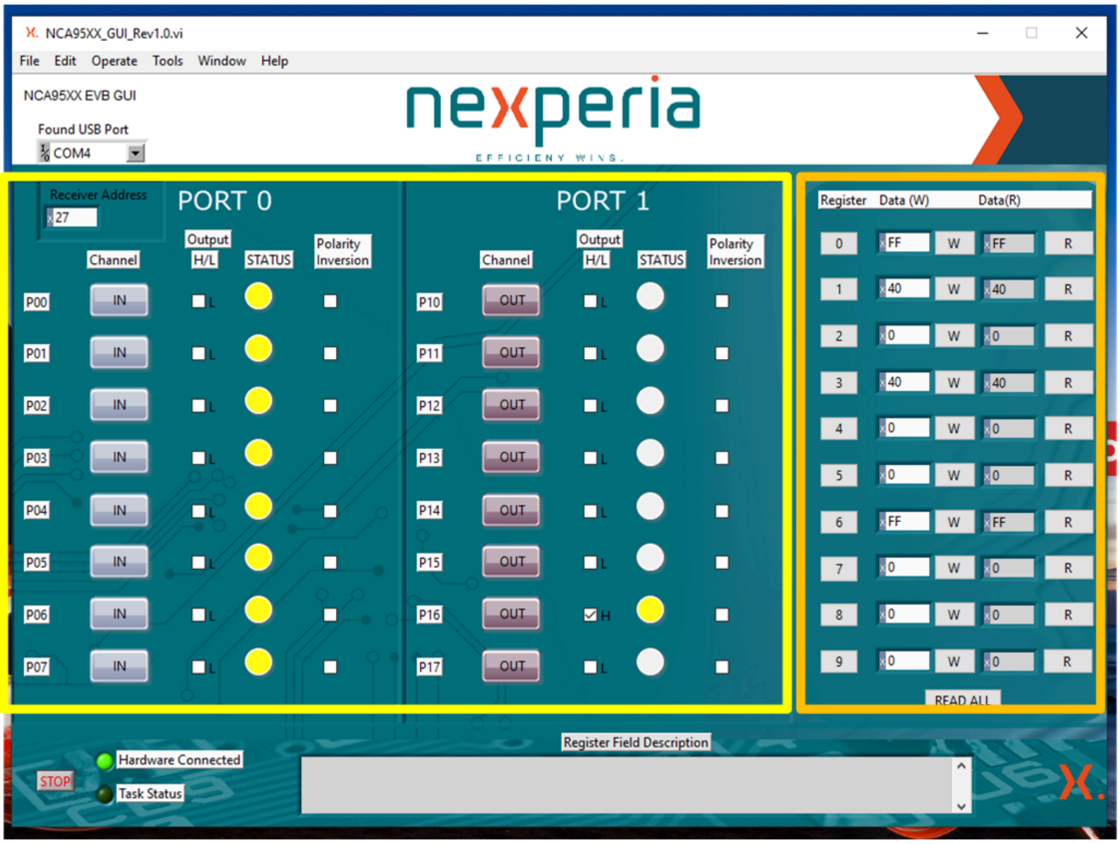


Fig. 13. NCA95XX GUI with interactive regions

Pressing the Register number button in the "Orange Box" will update the "Register Field Description" text shown in the bottom of the GUI. This text is updated with the Register field description.

GPIO Output example

An example for configuring “P00” as an output and writing a HIGH value is shown in Fig. 14. To turn the output on change Channel to “OUT” and check “ON/OFF” state, this will turn output on (high). To force output low, uncheck box for “OFF”.

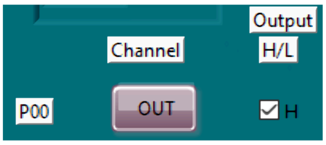


Fig. 14. GPIO Output example

GPIO Input example

An example for configuring P00 as an input and reading the state of the input the “STATUS” LED is shown in Fig. 15. Select Channel as “IN” and press “READ ALL” to update “STATUS” LED.

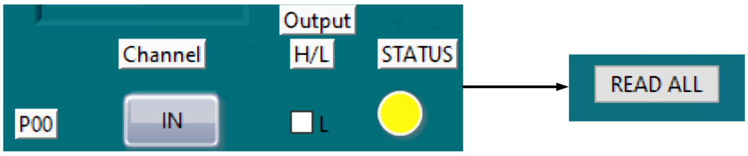


Fig. 15. GPIO Input example

To exit from the GUI, press the “X” icon on the top right of the GUI. To reinvoke the GUI at a later time navigate back to the startup menu and select the application.

4. Schematic diagram

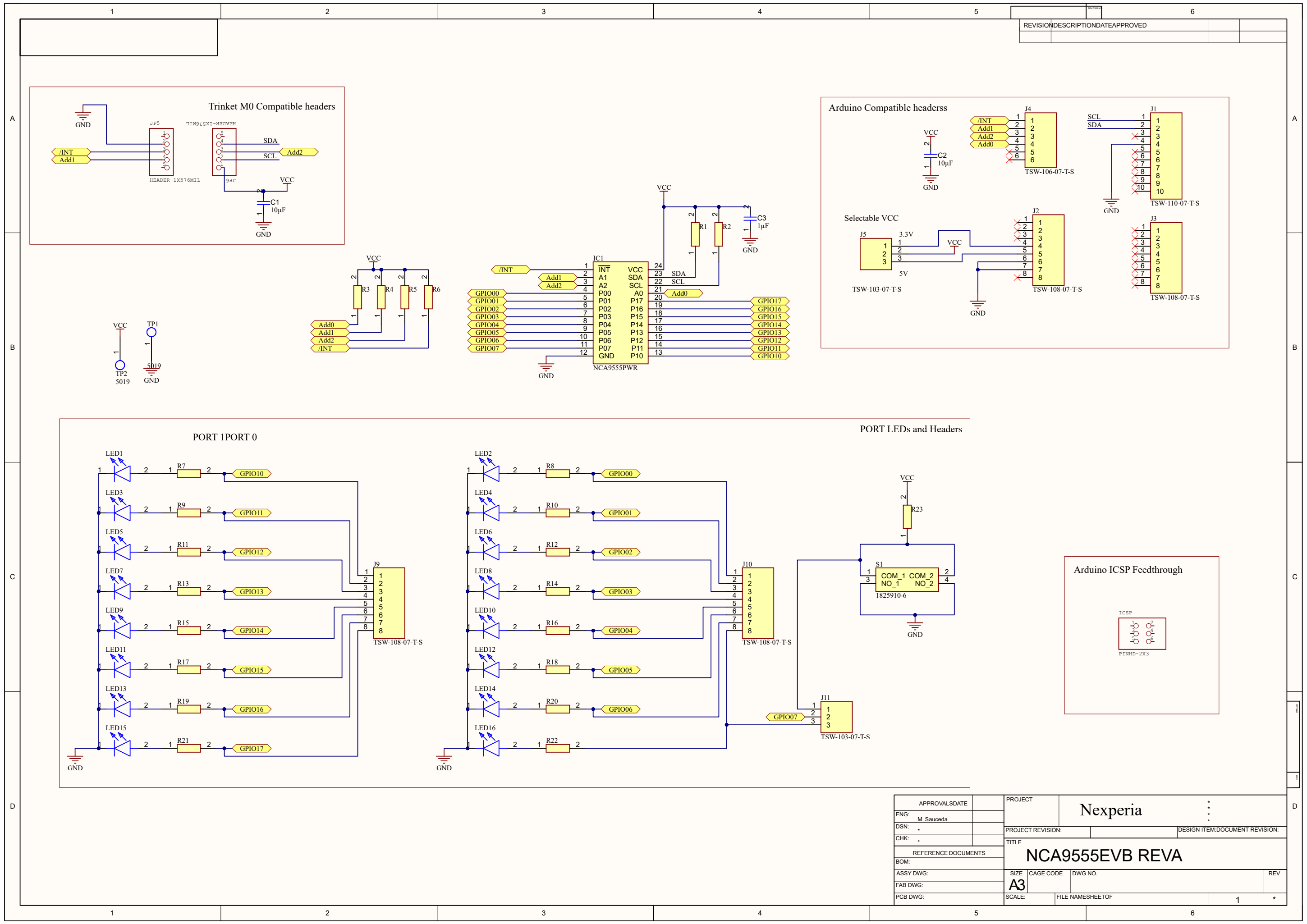


Fig. 16. NCA95xx EVB schematic diagram

All information provided in this document is subject to legal disclaimers.

5. PCB layout

Fig. 17 and Fig. 18 depicts the PCB layout of the NCA95xx GPIO expander EVB (Arduino™ shield compatible). The PCB has two layers, the top layer is shown in red, the bottom layer (ground plane) is shown in green.

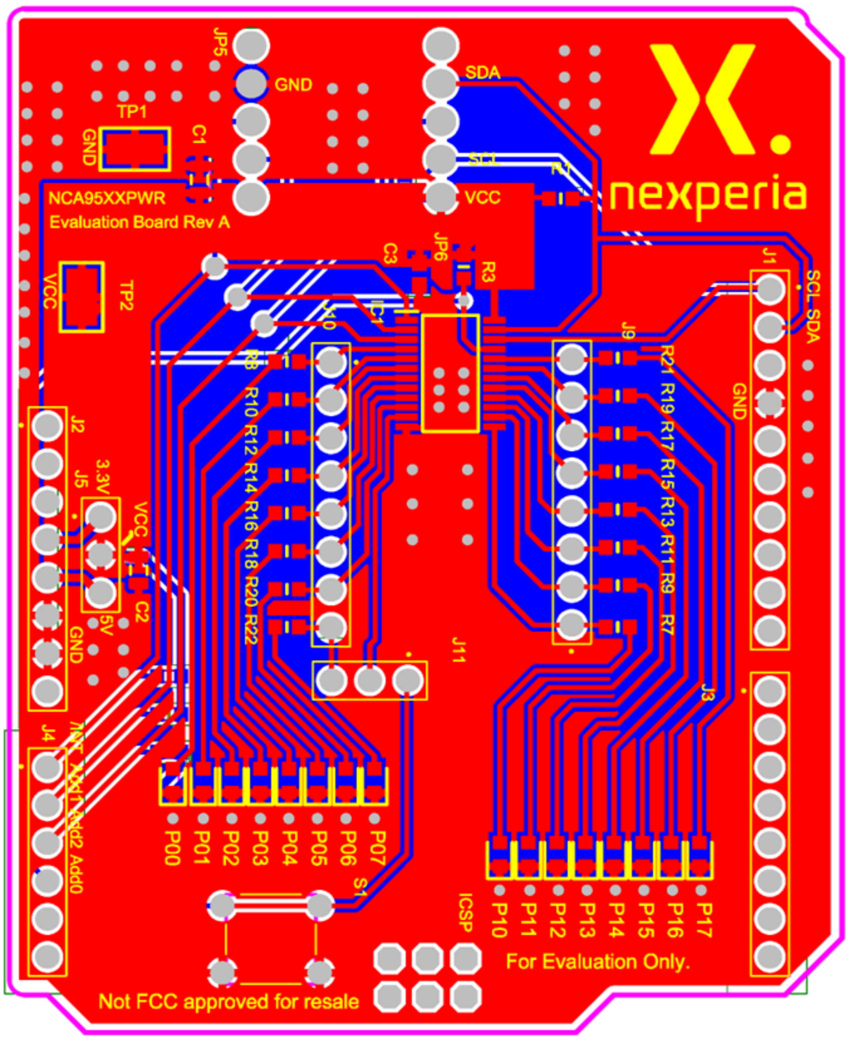


Fig. 17. NCA95xx GPIO expander EVB (Arduino™ shield compatible) top layer PCB layout

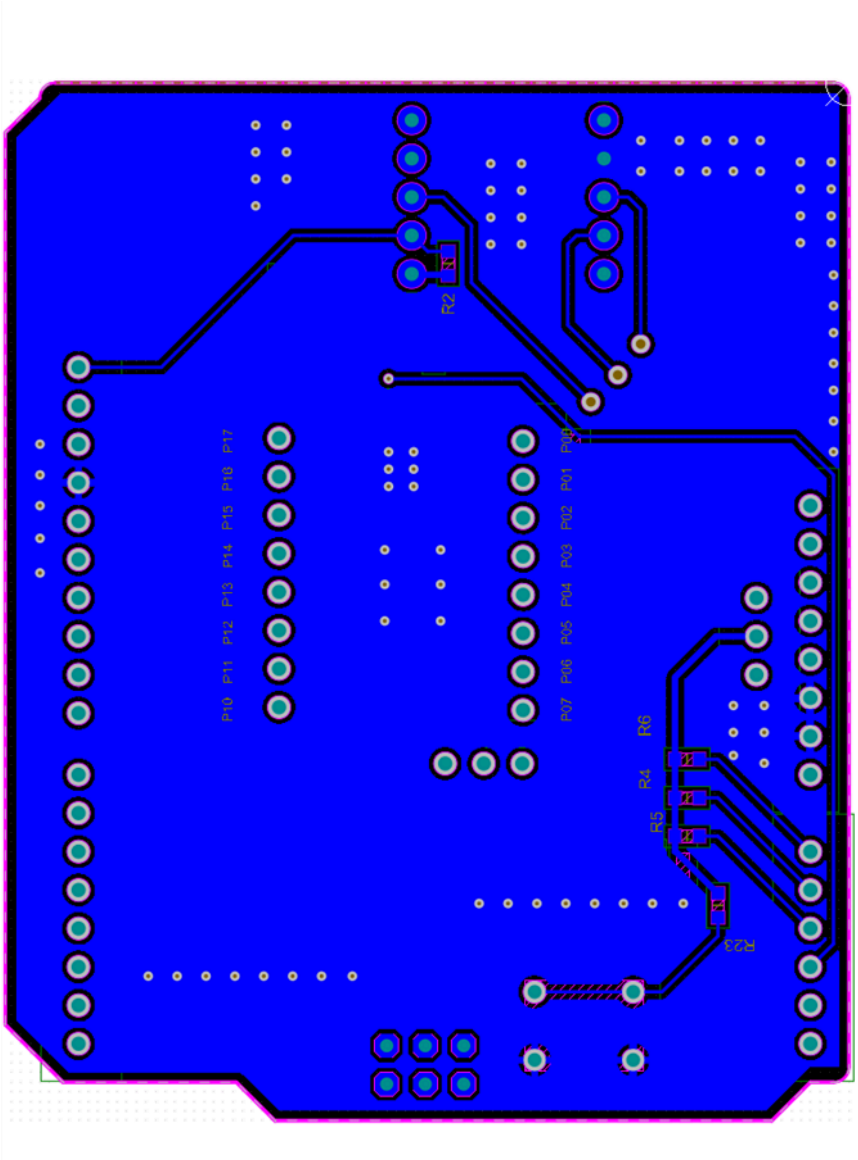


Fig. 18. NCA95xx GPIO expander EVB (Arduino™ shield compatible) bottom layer PCB layout

6. Bill of Materials (BOM)

Table 11. Bill of Materials

Component	Description	Designator	Quantity
CC0805KPX5R6BB106	CC0805KPX5R6BB106	C1, C2	2
CC0603KRX7R7BB105	CC0603KRX7R7BB105	C3	1
NCA9555PWR	Integrated Circuit	IC1	1
PRT-11417	Connector	J1,J2,J3,J4	1
TSW-108-07-T-S	Connector	J9, J10	2
TSW-103-07-T-S	Connector	J5, J11	2
LTST-C190KRKT	LED	P10,P11,P12,P13,P14,P15,P16,P17	8
RC0603JR-074K7L	RC0603JR-074K7L	R1, R2	2
CRCW060310K0JNEAC	CRCW060310K0JNEAC	R3, R4, R5, R6, R23	5
RC0603JR-07300RL	RC0603JR-07300RL	R7, R9, R11, R13, R15, R17, R19, R21	8
1825910-6	Switch	S1	1
5019	5019	TP1, TP2	2

7. Register Information

7.1. Command byte

In the I²C communication to the GPIO expanders first a command byte must be sent which points to 8 registers. The values 0x00 to 0x07 are used and supported only. If an illegal value is transmitted this message is not acknowledged by the NCA95xx.

Table 12. Command byte

Pointer Register bits – NCA95xx								Register	Protocol	Power-up default
B7	B6	B5	B4	B3	B2	B1	B0			
0	0	0	0	0	0	0	0	Input Port 0	Read byte	XXXX XXXX
0	0	0	0	0	0	0	1	Input Port 1	Read byte	XXXX XXXX
0	0	0	0	0	0	1	0	Output Port 0	R/W byte	1111 1111
0	0	0	0	0	0	1	1	Output Port 1	R/W byte	1111 1111
0	0	0	0	0	1	0	0	Polarity Inv 0	R/W byte	0000 0000
0	0	0	0	0	1	0	1	Polarity Inv 1	R/W byte	0000 0000
0	0	0	0	0	1	1	0	Config Port 0	R/W byte	1111 1111
0	0	0	0	0	1	1	1	Config Port 1	R/W byte	1111 1111
0	0	0	0	1	0	0	0	Pull up Res 0*	R/W byte	0000 0000
0	0	0	0	1	0	0	1	Pull up Res 1*	R/W byte	0000 0000

Table 12 shows a list of valid command byte values. The registers in the NCA95xx are provided as pairs for port 0 and port 1 always. Registers 0x00 and 0x01 are read-only registers for the logic states of the two ports. The next two registers 0x02 and 0x03 define the logic state of an output. Via the registers 0x04 and 0x05 a mask for polarity inversion of input bits can be defined. Registers 0x06 and 0x07 configure the bits of each port to an input or output. Registers 0x02 to 0x07 can be written as well as read.

7.2. Input port register pair (0x00, 0x01)

The input port registers return the logic state of the related port pins regardless if a pin is configured as an input or output. With the readback of output pins, failures in a hardware can be detected like a short circuit at an output programmed to high level.

Table 13. Input port 0 register (address 0x00)

Bit	7	6	5	4	3	2	1	0
Symbol	I0.7	I0.6	I0.5	I0.4	I0.3	I0.2	I0.1	I0.0
Default	X	X	X	X	X	X	X	X

The input port read registers cannot be written. Table 13 and Table 14 show the bit assignment in the two read registers. The default status X means that there is no power on state. If a read operation is done after power on without further register changes, the actual state of all GPIOs pins which are inputs as default state can be derived.

Before starting a read operation the command byte had to be programmed to 0x00. After this the input port bytes can be read back.

Table 14. Input port 1 register (address 0x01)

Bit	7	6	5	4	3	2	1	0
Symbol	I1.7	I1.6	I1.5	I1.4	I1.3	I1.2	I1.1	I1.0
Default	X	X	X	X	X	X	X	X

7.3. Configuration register pair (0x06, 0x07)

With the configuration registers it can be defined if a GPIO pin is an input or an output. As power-on and default state the ports are defined as inputs. For the NCA9555 all the port pins start with a logic high state supported from the integrated pull-up resistors unless they are forced to low state externally. Programming a bit to 0 activates the push-pull stage at the GPIO and the port pin becomes an output. If a control bit is set to 1, the related GPIO pin works as an input

Table 15. Configuration port 0 register (address 0x06)

Bit	7	6	5	4	3	2	1	0
Symbol	C0.7	C0.6	C0.5	C0.4	C0.3	C0.2	C0.1	C0.0
Default	1	1	1	1	1	1	1	1

Table 16. Configuration port 1 register (address 0x07)

Bit	7	6	5	4	3	2	1	0
Symbol	C1.7	C1.6	C1.5	C1.4	C1.3	C1.2	C1.1	C1.0
Default	1	1	1	1	1	1	1	1

7.4. Ouput port register pair (0x00, 0x01)

The Output registers define the logic state of an output. The default state is high level at the output.

Table 17. Output port 0 register (address 0x02)

Bit	7	6	5	4	3	2	1	0
Symbol	O0.7	O0.6	O0.5	O0.4	O0.3	O0.2	O0.1	O0.0
Default	1	1	1	1	1	1	1	1

Table 18. Output port 1 register (address 0x03)

Bit	7	6	5	4	3	2	1	0
Symbol	O1.7	O1.6	O1.5	O1.4	O1.3	O1.2	O1.1	O1.0
Default	1	1	1	1	1	1	1	1

7.5. Polarity Inversion register pair (0x04, 0x05)

By means of the Polarity Inversion registers, the logic level of those pins that are defined as inputs via the Configuration registers can be inverted. The inversion does not effect pins programmed as outputs. They appear as programmed in non-inverted polarity.

Table 19. Polarity Inversion Port 0 register (address 0x04

Bit	7	6	5	4	3	2	1	0
Symbol	N0.7	N0.6	N0.5	N0.4	N0.3	N0.2	N0.1	N0.0
Default	0	0	0	0	0	0	0	0

Table 20. Polarity Inversion Port 1 register (address 0x05)

Bit	7	6	5	4	3	2	1	0
Symbol	N1.7	N1.6	N1.5	N1.4	N1.3	N1.2	N1.1	N1.0
Default	0	0	0	0	0	0	0	0

7.6. Pull Up Resistor register pair (0x08, 0x09)

The Pull Up Resistor registers allow to turn on integrated pull-up resistors in the NCA9595 derivative in the NCA95xx GPIO expander family. In default condition the input pins not teminated internally, so the device behaves like an xCA9535 from several competitors.

If a GPIO pin is configure as an output, connected pull-up resistors create unnessary power consumption in logic low state. If a bit in a Pull Up Resistor register is set to 1, the integrated 100 kΩ pull up resistor gets connected to the related port pin.

Table 21. Pull up resistors Port 0 register (address 0x08)

Bit	7	6	5	4	3	2	1	0
Symbol	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
Default	0	0	0	0	0	0	0	0

Table 22. Pull up resistors Port 1 register (address 0x09)

Bit	7	6	5	4	3	2	1	0
Symbol	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
Default	0	0	0	0	0	0	0	0

8. Example Arduino Firmware

Additional example firmware is provided in the software folder. Program “nca95xx_demo.ino” writes to PORT 1 to flash LEDs back and forth. PORT 0 is continuously read back and byte value is updated through Arduino IDE. Serial Monitor is used to monitor PORT 0. Pressing the pushbutton will change the speed of flashing.

9. Revision history

Table 23. Revision history

Revision number	Date	Description
2.0	2024-01-19	Fig. 16 updated.
1.1	2023-01-23	Minor text formatting change in Section 3.1
1.0	2023-01-05	Initial version.

10. Legal information

Definitions

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

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, Nexperia 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. Nexperia takes no responsibility for the content in this document if provided by an information source outside of Nexperia.

In no event shall Nexperia 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, Nexperia's 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 Nexperia.

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

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

Applications — Applications that are described herein for any of these products are for illustrative purposes only. Nexperia 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 Nexperia products, and Nexperia accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the Nexperia 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.

Nexperia 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 Nexperia 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). Nexperia does not accept any liability in this respect.

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.

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

Trademarks

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

List of Tables

List of Figures

Contents