AN14908

How to Implement an 8K USB Keyboard on MCX N236

Rev. 1.0 — 30 January 2026 Application note

Document information

Information	Content
Keywords	AN14908, MCX N236, USB, Keyboard, WS2812
Abstract	This application note describes how to implement an 8K USB keyboard based on the MCX N236.

1 Introduction

As the gaming and professional input device markets continue to grow, users are demanding increasingly higher keyboard response speeds and input precision. Traditional USB keyboards use a polling rate of 125 Hz (8 ms) or 1000 Hz (1 ms), while high-end gaming keyboards support ultra-high polling rates of 8000 Hz (0.125 ms). This high polling rate can significantly reduce input latency and improve the user experience, especially in competitive games and high-speed input scenarios.

More manufacturers are launching keyboards with 4K or 8K polling rates to meet the low latency needs of professional gamers. USB 2.0 High-Speed (480 Mbit/s) has become the standard interface for high-performance keyboards, supporting higher data transfer rates.

The MCX N23x series MCUs of NXP are microcontrollers designed for high-performance and low-power applications. Based on the Cortex-M33 core, they feature a maximum clock speed of 150 MHz, up to 1 MB of Flash memory, and up to 352 kB of RAM. It also includes key peripherals for the keyboard application, such as the High-speed (HS) USB controller and SmartDMA.

- USB 2.0 High-Speed (HS) controller supports a transfer rate of 480 Mbit/s, meeting the bandwidth requirements of 8K polling rates keyboard.
- SmartDMA is a co-processor. The primary objective of SmartDMA is to offload tasks from the Arm processor, and optimize performance and efficiency. SmartDMA provides significant advantages over Arm processing, specifically in:
  - Fast response: It provides a quick reaction to input and output (I/O), and Boolean events.
  - Efficiency at lower frequencies: It offers improved performances as SmartDMA eliminates interrupt delays.
  - Expanded instruction set: It uses a 32-bit instruction set offering broader functionality to the 16-bit thumb instruction set.
  - Advanced extension: It includes built-in innovative functionalities, such as heartbeat, timer, and supervised execution.
  - Concurrent data access: It implements a dual AHB bus controller, allowing simultaneous data read and code execution.

In applications like key scanning, SmartDMA uses bit-slice operation for key detection. Its register-based I/O handling accelerates key scan response time and enhances sensitivity.

Therefore, the MCX N23x is well-suited for 8K USB keyboard applications, enabling hardware acceleration of key scanning tasks through SmartDMA and data transmission at 8K report rate through HS USB.

This application note describes how to implement an 8K USB keyboard based on the MCX N236.

2 System overview

The system block diagram of the 8K USB keyboard implemented in this application note is as shown in Figure 1.

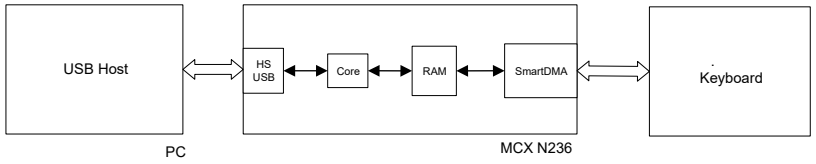

Figure 1. System block diagram of 8K USB keyboard

The following features are supported by this solution:

- Supporting 8K USB Human Interface Device (HID) report rate
- SmartDMA performs keyboard scanning

- Debouncing keys via software
- Using LPSPI to control WS2812 LED array, supporting four lighting effects

As shown in Figure 1, this solution uses SmartDMA for fast I/O read/write operations to control the key matrix to detect key events. Upon detecting a key event, SmartDMA triggers an interrupt to instruct the CPU to handle it. The CPU retrieves data from shared RAM, parses which key was pressed, and then updates the USB HID report. HS USB achieves an 8K reporting rate, meaning that a key event is reported to the USB host within 125 µs, resulting in a low latency and improved user experience in competitive games. Also, this solution uses a Low-Power Serial Peripheral Interface (LPSPI) interface to control the WS2812 LED matrix, implementing four simple lighting effects. For details, see Section 3.

3 Hardware requirements

This section introduces the hardware used in this application note.

3.1 FRDM-MCXN236 board

The FRDM-MCX236 development board is an official NXP board used to evaluate the MCX N236 features, as shown in Figure 2. It features an MCX N236 MCU and supports an HS USB interface.

- It provides a rich set of GPIO interfaces, making it suitable for connecting keyboard.
- Integrated onboard debugger facilitates development and debugging.

In addition, the board has many sensors and a wealth of expansion interfaces. For details, see the *FRDM-MCXN236 User Manual* (document UM12041).

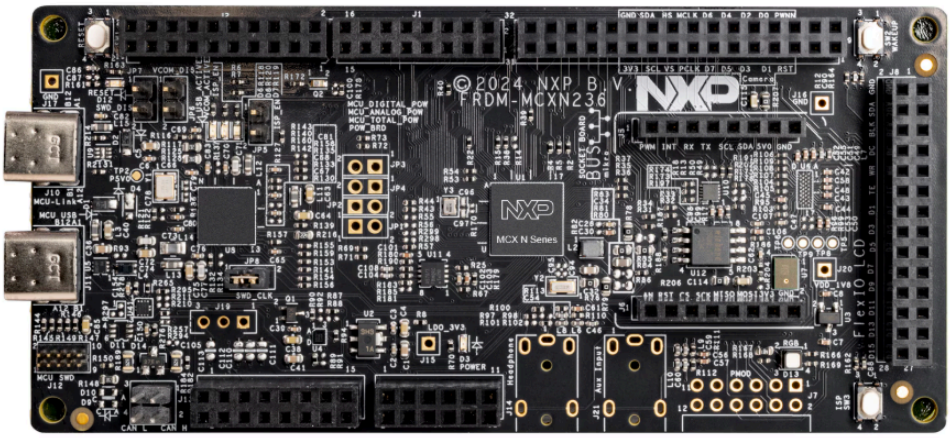

Figure 2. FRDM-MCXN236 board

3.2 ESP-Keyboard

ESP-Keyboard is an open-source custom keyboard based on the ESP32S3. It adopts a standard keyboard matrix layout, supports a 6 × 15 matrix of keys, and provides a Flexible Printed Circuit (FPC) 30-pin connector to bring out the row and column interfaces of the key matrix for easy connection to the MCU main control board. This application note only uses the keyboard and does not use the ESP32-S3 main control board.

Figure 3 shows the FPC30 pin interface on the ESP-Keyboard.

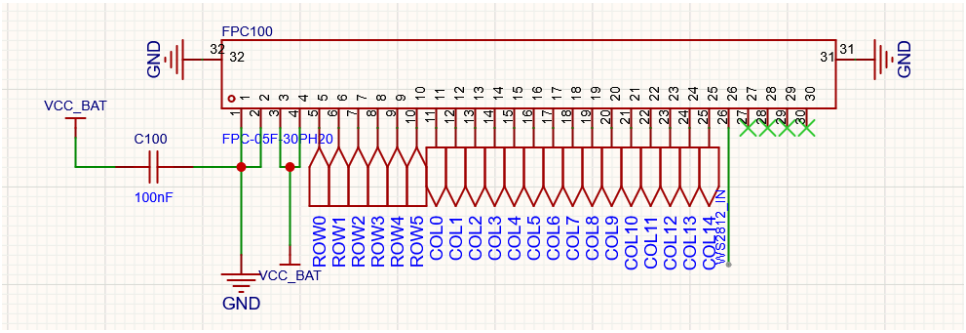

Figure 3. FPC connector of ESP-Keyboard

Figure 4 shows the partial schematic of ESP-Keyboard.

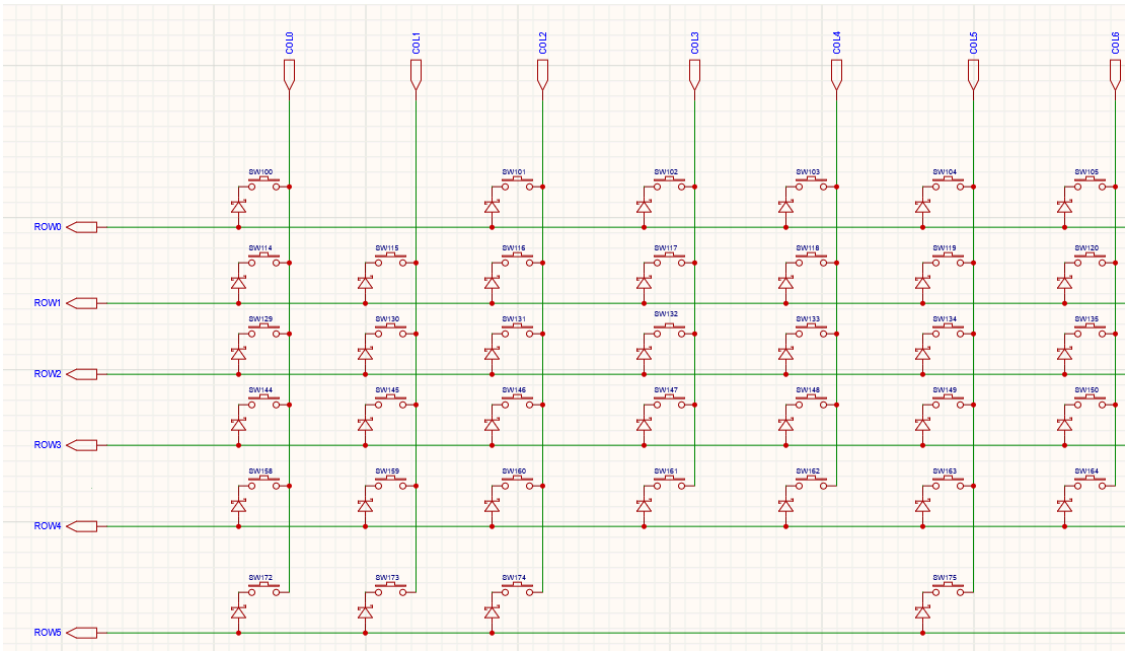

Figure 4. ESP-Keyboard schematic

This solution is open source on GitHub. For more information, see the GitHub

3.3 Keyboard adapter board

Since the FRDM-MCXN236 board does not have an FPC 30-pin connector, an adapter board is needed to connect the FRDM-MCXN236 board and the ESP-Keyboard. Therefore, a keyboard adapter board is designed to connect the FRDM-MCXN236 board and the ESP-Keyboard, as shown in Figure 5. It connects 22 GPIOs from the Arduino interface to the FPC 30-pin connector, including 21 GPIOs (6+15) and one `LPSPI_SDO` pin. The 21 GPIOs are used to control the 15 column signals and 6 row signals of the key matrix. The `LPSPI_SDO` is used to control the WS2812 LED array. Figure 6 shows the schematic diagram of the board. For the schematic diagram and PCB file of the keyboard adapter board, see AN14908SW.

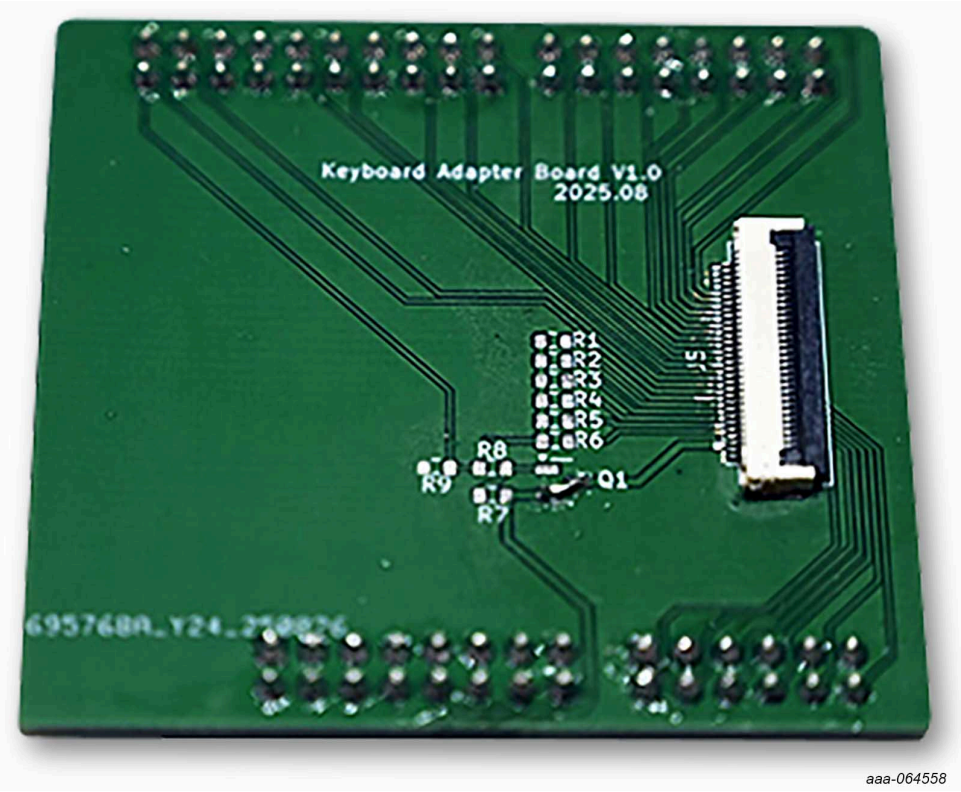

Figure 5. Keyboard adapter board

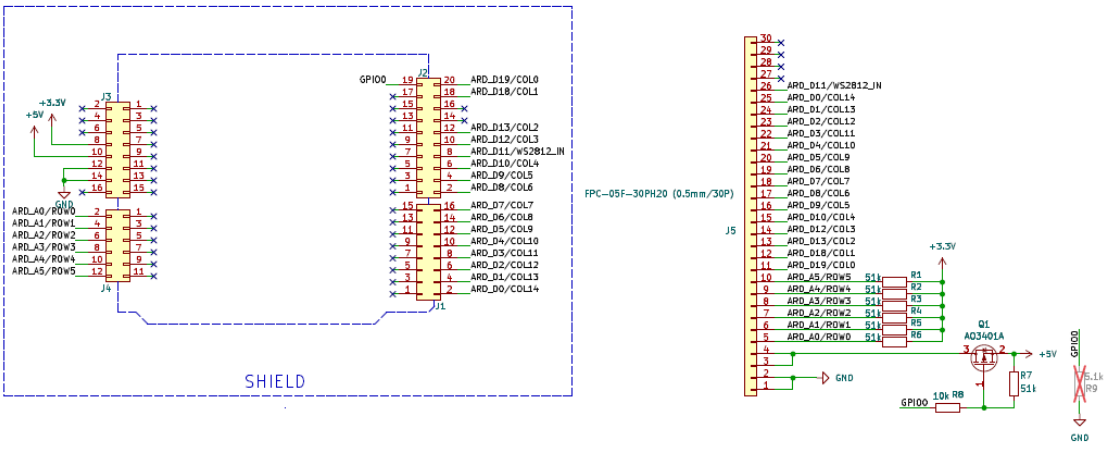

Figure 6. Keyboard adapter board schematic

Figure 7. 8K USB keyboard based on MCX N236

4 Software implementation

This chapter describes the software implementation of the 8K USB Keyboard based on MCX N236, including how to achieve the 8K USB HID report rate, how to perform key scanning using SmartDMA, and how to control the WS2812 LED array using LPSPI.

4.1 Achieving 8K USB HID polling rate

The software in this application note is based on the `dev_hid_generic_bm` example in the FRDM-MCXN236 Software Development Kit (SDK). It is necessary to modify the `HS_HID_GENERIC_INTERRUPT_IN_INTERVAL` macro from 4 to 1 to make the interval of the USB HID packet 125 µs, thereby achieving an 8K USB HID report rate.

```
#define HS_HID_GENERIC_INTERRUPT_IN_INTERVAL (0x01U)
```

The `USB_DeviceHidKeyboardCallback()` function is called in USB interrupt service function and it must be modified to parse key events, perform software key debouncing, and update the HID report, so that we can report the key events to the USB host within 125 µs. For detailed information about this function, see AN14908SW.

4.2 Implementing key scanning using SmartDMA

Traditional keyboard scanning is typically performed by CPU polling, and frequent GPIO operations consume significant processing time. The SmartDMA of the MCX N236, acting as a coprocessor, enables single-cycle I/O access, allowing for rapid keyboard matrix scanning without CPU intervention. The CPU can then be freed up to perform other tasks, such as key mapping, parsing of key combination events, and control of lighting effects. It is important to note that all key-related tasks must be completed within 125 µs, including SmartDMA key detection, key event parsing, key debouncing, and HID report updates.

The method for implementing key matrix scanning using SmartDMA is as follows:

1. Set the GPIO of 15 control column signals as outputs and pull them high by default, and set the GPIO of 6 control row signals as inputs.
2. SmartDMA pulls the first column GPIO low, then reads the status of each of the six rows of GPIOs one by one, and saves the results to SmartDMA's general-purpose registers (R0 - R3). Each bit stores the value of a key.
3. Follow step 2 to complete the scan of column 2 to column 15.
4. After the key matrix scan is completed, the values in the SmartDMA general-purpose registers (R0 - R3) are compared with the values in the shared RAM. If the values of some bits are found to be different, it indicates that the state of the key corresponding to that bit has changed. The key state includes two states: pressed and released. After detecting a key state change, a SmartDMA interrupt is triggered to notify the CPU and the values of SmartDMA registers R0-R3 are also stored to the shared RAM. The CPU reads the data in the shared RAM in the SmartDMA interrupt service function. The CPU parses the key event, updates the HID report in the USB interrupt service function and reports it to the USB host.

Figure 8 shows the SmartDMA configuration code snippet.

```
SMARTDMA_InitWithoutFirmware();
SMARTDMA_InstallFirmware(SMARTDMA_KEYSCAN_MEM_ADDR,s_smartdmaKeyscanFirmware_6x15,
        s_smartdmaKeyscanFirmwareSize_6x15);
SMARTDMA_InstallCallback(SmartDMA_keyscan_callback, NULL);
NVIC_EnableIRQ(SMARTDMA_IRQn);
NVIC_SetPriority(SMARTDMA_IRQn, 3);
smartdmaParam.smartdma_stack     = (uint32_t*)g_samrtdma_stack;
smartdmaParam.p_gpio_reg         = (uint32_t*)g_keyscan_gpio_register;
smartdmaParam.p_keyvalue         = (uint32_t*)g_KeyValueSharedRAM;
smartdmaParam.p_keycan_interval  = (uint32_t*)&g_keyscan_interval;
SMARTDMA_Boot(kSMARTDMA_Keyscan_4x4, &smartdmaParam, 0x2);
```

Figure 8. SmartDMA configuration

4.3 Using LPSPI to control the WS2812 LED array

As described in Section 3.3, the keyboard adapter board also connects the `LPSPI_SDO` signal to the ESP-Keyboard to control the WS2812 LED array. The LPSPI clock rate is 2.4 MHz, and three LPSPI clocks are used to simulate one WS2812 bit. Use the non-blocking mode when transmitting WS2812 data via LPSPI, because the CPU must handle SmartDMA and USB interrupts when using LPSPI for data transmission. It is recommended to use DMA to move WS2812 control data from RAM to the LPSPI FIFO. Figure 9 shows the LPSPI configuration code.

```
/*Master config*/
LPSPI_MasterGetDefaultConfig(&masterConfig);
masterConfig.baudRate = TRANSFER_BAUDRATE;
masterConfig.whichPcs = EXAMPLE_LPSPI_MASTER_PCS_FOR_INIT;
masterConfig.pcsToSckDelayInNanoSec        = 1000000000U / (masterConfig.baudRate * 2U);
masterConfig.lastSckToPcsDelayInNanoSec    = 1000000000U / (masterConfig.baudRate * 2U);
masterConfig.betweenTransferDelayInNanoSec = 1000000000U / (masterConfig.baudRate * 2U);

srcClock_Hz = LPSPI_MASTER_CLK_FREQ;
LPSPI_MasterInit(EXAMPLE_LPSPI_MASTER_BASEADDR, &masterConfig, srcClock_Hz);/* Function successful */

/*Set up lpspi master*/
memset(&(lpspiEdmaMasterRxRegToRxDataHandle), 0, sizeof(lpspiEdmaMasterRxRegToRxDataHandle));
memset(&(lpspiEdmaMasterTxDataToTxRegHandle), 0, sizeof(lpspiEdmaMasterTxDataToTxRegHandle));

EDMA_CreateHandle(&(lpspiEdmaMasterRxRegToRxDataHandle), EXAMPLE_LPSPI_MASTER_DMA_BASE,
                  EXAMPLE_LPSPI_MASTER_DMA_RX_CHANNEL);
EDMA_CreateHandle(&(lpspiEdmaMasterTxDataToTxRegHandle), EXAMPLE_LPSPI_MASTER_DMA_BASE,
                  EXAMPLE_LPSPI_MASTER_DMA_TX_CHANNEL);

EDMA_SetChannelMux(EXAMPLE_LPSPI_MASTER_DMA_BASE, EXAMPLE_LPSPI_MASTER_DMA_TX_CHANNEL,
                   DEMO_LPSPI_TRANSMIT_EDMA_CHANNEL);
EDMA_SetChannelMux(EXAMPLE_LPSPI_MASTER_DMA_BASE, EXAMPLE_LPSPI_MASTER_DMA_RX_CHANNEL,
                   DEMO_LPSPI_RECEIVE_EDMA_CHANNEL);

LPSPI_MasterTransferCreateHandleEDMA(EXAMPLE_LPSPI_MASTER_BASEADDR, &g_m_edma_handle, LPSPI_MasterUserCallback,
                                     NULL, &lpspiEdmaMasterRxRegToRxDataHandle,
                                     &lpspiEdmaMasterTxDataToTxRegHandle);
```

Figure 9. LPSPI configuration

This application note only implements four simple LED lighting effects. Users can implement more custom lighting effects based on AN14908SW.

The method for controlling lighting effects using the keyboard is as follows:

- Press **Fn+Home** to turn the lighting effect on or off.
- Press **Fn+Page Up/Down** to switch between four lighting effects.

5 Test

AN14908SW provides sample code, which is developed based on MCUXpresso Integrated Development Environment (IDE) and FRDM-MCXN236 SDK version 25.06. Before running this example, follow Section 3.2 to build an ESP-Keyboard. In addition, AN14908SW also provides the schematic and PCB files for the keyboard adapter board. Build a keyboard adapter board and then connect the three boards, as shown in Figure 7.

The testing steps are as follows:

1. Import the sample code from the attachment into the MCUXpresso IDE.
2. Compile the project.
3. Use two USB Type-C cables to connect ports J10 and J11 on the FRDM-MCXN236 to the PC's USB ports, respectively. J10 is the onboard debugger interface and J11 is the HS USB interface of the MCX N236.
4. Download the compiled code to the FRDM-MCXN236 board using the onboard debugger.
5. Pressing the reset button (**SW1**) on the FRDM-MCXN236 board to run the code.
6. Press **Fn+Home** and **Fn+Page Up/Down** keys to test the lighting effects.
7. Open any text editor and press the number and letter keys on the keyboard to test the key functions.

In addition, a logic analyzer is used to measure the SmartDMA key matrix scan time, which was 79.8 µs, as shown in Figure 10. The execution time of the USB interrupt service function was measured as 28.5 µs, as shown in Figure 11. The parsing of key events is completed within the USB interrupt service function. The time for executing one key matrix scan and data parsing is less than 125 µs, indicating that the time from the occurrence of a key event to report it to the USB host is less than 125 µs.

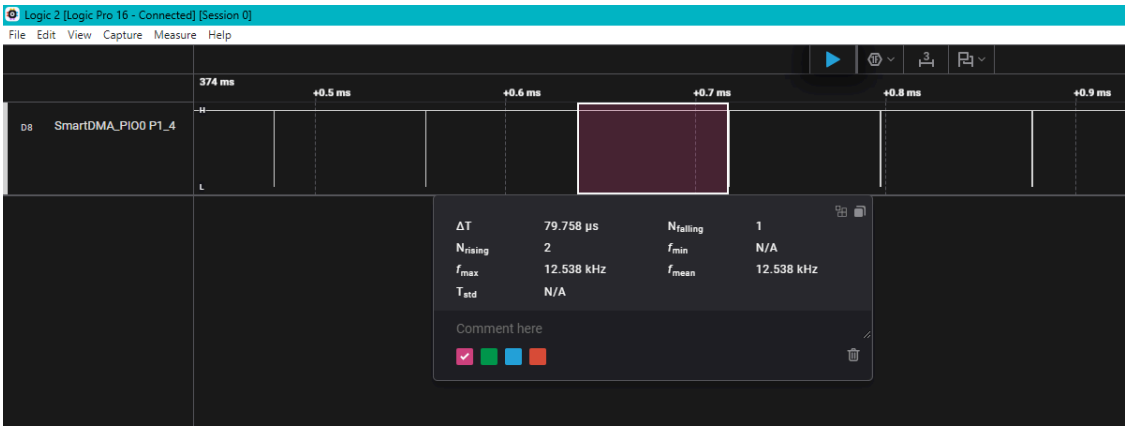

Figure 10. SmartDMA key matrix scan time

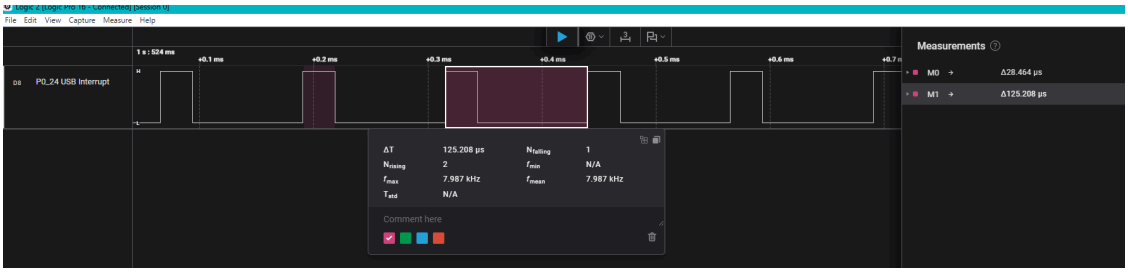

Figure 11. Execution time of the USB interrupt service function

6 Conclusion

This application note demonstrates how to implement an 8K USB keyboard based on the MCX N236 It explains the principle of using SmartDMA to achieve key matrix scanning, and proves through testing that this method of implementing an 8K USB keyboard is feasible. In addition, the application note also provides relevant hardware information and software example code, so that customers can quickly develop their own 8K USB keyboard products based on this application note.

7 Reference

- *MCX N23x Reference Manual* (document MCXN23XRM)
- *SmartDMA Cookbook* (document AN14650)
- *FRDM-MCXN236 Board User Manual* (document UM12041)
- https://github.com/espressif/esp-iot-solution/tree/master/examples/keyboard
- ESP-KeyBoard

8 Note about the source code in the document

The example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2026 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

9 Revision history

Table 1 summarizes the revisions to this document.

Table 1. Revision history

Document ID	Release date	Description
AN14908 v.1.0	30 January 2026	Initial public release

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.

Suitability for use — NXP Semiconductors 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 NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors 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. 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.

Limiting values — Stress above one or more limiting values (as defined in the Absolute Maximum Ratings System of IEC 60134) will cause permanent damage to the device. Limiting values are stress ratings only and (proper) operation of the device at these or any other conditions above those given in the Recommended operating conditions section (if present) or the Characteristics sections of this document is not warranted. Constant or repeated exposure to limiting values will permanently and irreversibly affect the quality and reliability of the device.

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.

No offer to sell or license — Nothing in this document may be interpreted or construed as an offer to sell products that is open for acceptance or the grant, conveyance or implication of any license under any copyrights, patents or other industrial or intellectual property rights.

Quick reference data — The Quick reference data is an extract of the product data given in the Limiting values and Characteristics sections of this document, and as such is not complete, exhaustive or legally binding.

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.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

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.

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.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

Microsoft, Azure, and ThreadX — are trademarks of the Microsoft group of companies.

Contents