AN14470

How to Use FlexIO State Mode to Generate Center-Aligned PWM

Rev. 1.0 — 6 February 2025 Application note

Document information

Information	Content
Keywords	AN14470, MCXN, FlexIO, PWM
Abstract	This application note describes how to use the FlexIO state mode to generate a center-aligned PWM waveform on the MCXN series MCUs.

1 Introduction

This application note describes how to use the FlexIO state mode to generate a center-aligned PWM waveform on the MCXN series MCUs.

The MCXN series MCUs have eight FlexIO shifters, which can implement up to eight states. In every state, a different level logic can be output on up to eight FlexIO data pins. The transitions between states can be controlled by up to eight FlexIO timers.

Controlling state transitions using external input pins is a common demonstration method for FlexIO. However, there are many instances where users simply want to use a state machine controlled by a timer sequence without considering external input states. This application note is based on such a starting point, using timers to control the state machine composed of shifters, to generate center-aligned PWM waveforms.

The configuration tool provided by NXP can quickly generate functional code for FlexIO. One purpose of this application is to demonstrate how to use the configuration tool to implement the state machine function of FlexIO, making the implementation simple and fast.

2 FlexIO module state mode

The FlexIO module state mode enables you to create a hardware state machine with up to eight states. Each state uses three selectable inputs and eight dedicated outputs. Such a state machine offloads the CPU rapidly when compared to the software approach.

2.1 FlexIO state I/O assignments

Figure 1 shows the I/O assignments for a given state in general.

- The three inputs to the state can be represented by a set of three FlexIO pins (starting from the `SHIFTCTLn[PINSEL]` value) or by three LSB bits of the next (`n+1`) shifter. Select this using the input source selection field INSRC in the `SHIFTCFGn[INSRC]` shifter-configuration register.
- Configure the inputs' PINPOL polarity in the `SHIFTCTLn[PINPOL]` shifter-control register when the FlexIO pins are selected as inputs.
- The eight outputs from the state are fixed and assigned to FlexIO pins `FXIO_D0-FXIO_D7` (all states share the same outputs). Disable the `FXIO_D0-FXIO_D7` outputs partially by setting the adequate bits in the `PWIDTH[3:0]`, `SSTOP[1:0]`, and `SSTART[1:0]` fields in the `SHIFTCFGn` shifter-configuration register.

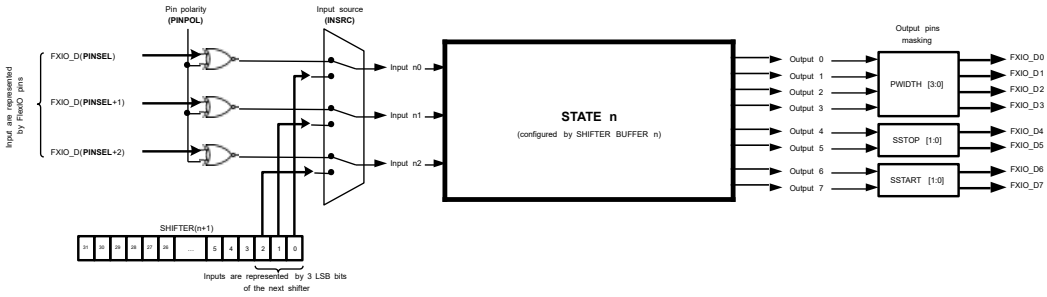

Figure 1. FlexIO state I/O assignments

2.2 FlexIO state configuration

The state mode is selected when the SMOD field in the `SHIFTCTLn[SMOD]` shifter-control register is set to `0x6`.

Table 1. State mode

Field	Function
2-0 SMOD	Shifter mode Configure the mode of the shifter 000b - Disable 001b - Receive mode; capture the current shifter content into SHIFTBUF on expiration of the timer 010b - Transmit mode; load SHIFTBUF contents into the shifter on expiration of the timer 011b - Reserved 100b - Match store mode; shifter data is compared to SHI FTBUF content on expiration of the timer 101b - Match continuous mode; shifter data is continuously compared to SHIFTBUF contents 110b - State mode; SHIFTBUF contents store programmable state attributes 111b - Logic mode; SHIFTBUF contents implement programmable logic lookup table

The `SHIFTSTATE` shifter-state register defines the current state when you select the state mode and enable the FlexIO module. It is set to `0x0` by default (after reset). Initialize it if shifter 0 is not configured to the state mode to avoid incorrect state machine startup.

Table 2. Shifter state

Field	Function
2-0 STATE	Current state pointer. Maintains a pointer to track the current shifter (configured for the State mode) enabled to drive outputs and compute the next state. Reading this register when the state pointer is updating can result in the return of an incorrect state. The value that you write to this field overrides the current state.

When started correctly, `SHIFTSTATE` points to the current state, which is defined by the shifter buffer `n` register `SHIFTERBUFFn` value. Its 32-bit value contains the current-state outputs' configuration (`SHIFTBUFF[31:24]`) and the next state selection (`SHIFTBUFF[23:0]`).

The 24 LSBs of the shifter-buffer register value represent eight groups of three bits. The three bits in each group define the value of the next state. The group (next state) is selected according to the combination of inputs. If the input combination is `000`, then the value defined by the `SHIFTBUFF[2:0]` bits represents the next state. If the input combination is `011`, then the value defined by the `SHIFTBUFF[11:9]` bits represents the next state.

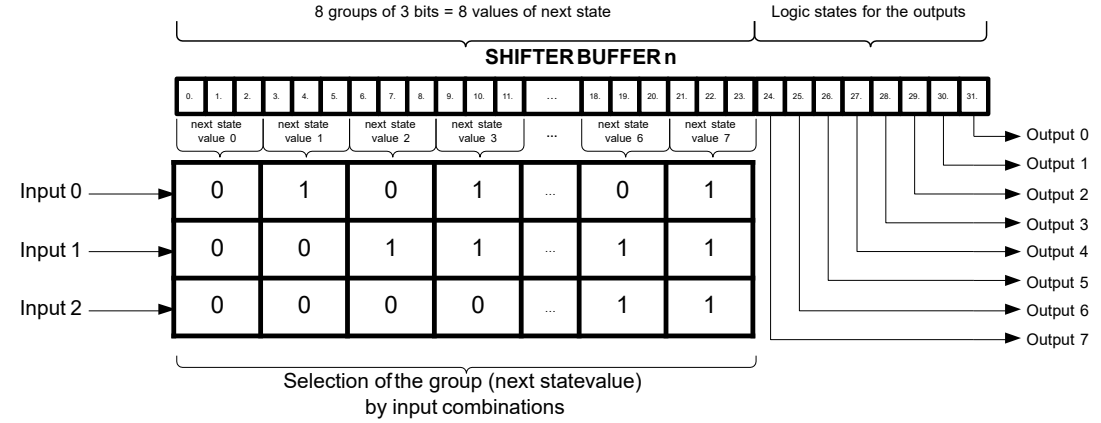

Figure 2. Input combination

Table 3. Shifter buffer

Field	Function
31-0 SHIFTBUF	Shifter buffer Contains the data to be matched with the shifter contents and is used for various other functions, depending on the setting. of SHIFTCTLn[SMOD]: • If SHIFTCTL0[SMOD] is 1b (Receive mode), the shifter data is transferred into SHIFTBUF at the expiration of the timer. You must read this register only when the corresponding SHIFTSTAT[SSF] flag is set, indicating that new shifter data is available. • If SHIFTCTL0[SMOD] is 10b (Transmit mode), SHIFTBUF data is transferred into the shifter before the timer begins. • If SHIFTCTL0[SMOD] is 100b (Match Store mode), SHIFTBUF[31:16] contains the data to be matched with the shifter contents and SHIFTBUF[15:0] can be used to mask the match result (1 = mask, 0 = no mask). The match is checked when the timer expires. Shifter data [31:16] is written to SHIFTBUF[31:16] whenever a match event occurs. You must read this register only when the corresponding shifter status flag is set, indicating that new shifter data is available. • If SHIFTCTL0[SMOD] is 101b (Match Continuous mode), SHIFTBUF[31:16] contains the data to be matched with the shifter contents, and SHIFTBUF[15:0] can be used to mask the match result (1 = mask, 0 = no mask). • If SHIFTCTL0[SMOD] is 111b (Logic mode), SHIFTBUF[31:0] implements a 5-input, 32-bit programmable logic lookup table. • If SHIFTCTL0[SMOD] is 110b (State mode), use SHIFTBUF[31:24] to drive the output value when this shifter is selected by the current state pointer and use SHIFTBUF[23:0] to configure the value of the next state transition.

2.3 FlexIO state transitions

The next state transition is generated when the timer selected by the current state expires. It depends on the `TIMSEL` timer select value in the shifter control register `SHIFTCTLn[TIMSEL]`.

An appropriate shifter status flag is set when the current state pointer `STATE` in the `SHIFTSTATE` shifter state register points to the adequate shifter.

Use the status flag to trigger the interrupt/DMA. Switch to a particular state to trigger the CPU/DMA, or even force a timer to assert the FlexIO output trigger (trigger the start of the ADC conversion). The flag clears when you select a different state.

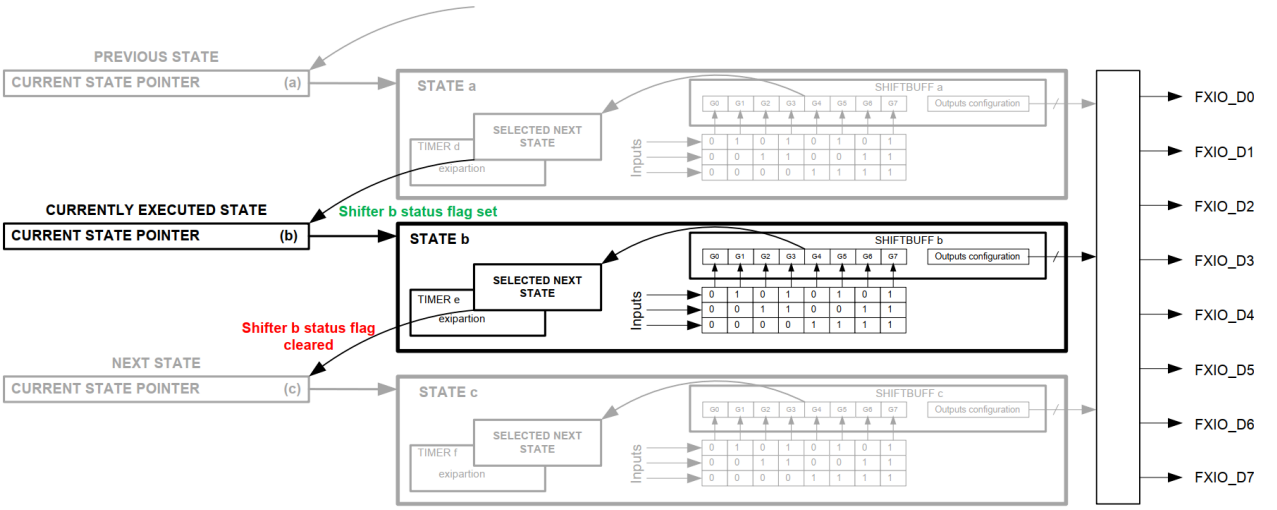

Figure 3. FlexIO state transitions

3 How to implement center-aligned PWM in state mode

Regarding the upper and lower bridge arms of motor control as a pair of complementary PWM waves, this pair is center-aligned with dead time. The signal of one cycle of the complementary PWM waveform can be divided into five states: State 0, State 1, State 2, State 3, and State 4.

FlexIO has a maximum of eight states, with each state capable of controlling up to eight IO outputs. This is sufficient for generating a pair of complementary PWM signals.

3.1 PWM with state machine

Figure 4 defines the corresponding states for each change in the PWM signal, with the duration of each state controlled by the FlexIO timer.

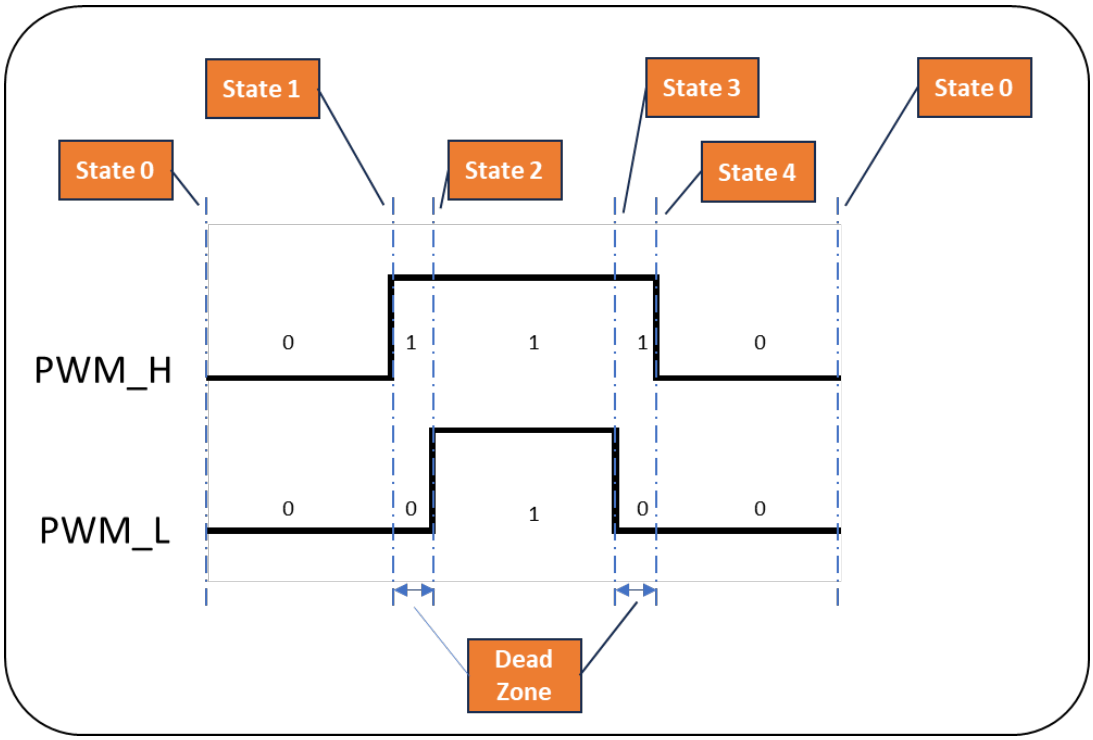

Figure 4. PWM with state machine

3.2 GPIO configuration on state machine

The `P0_8` pin is used to output the waveform of the `PWM_H` signal, while the `P0_9` pin is selected to output the waveform of the `PWM_L` signal.

Table 4. GPIO configuration on state machine

State	PWM_H(P0_8) value	PWM_L(P0_9) value
State 0	0	0
State 1	1	0
State 2	1	1
State 3	1	0
State 4	0	0

3.3 Output observation of internal timer waveform

The internal timers (0/1/2/3/4) are used as the timing base for state transitions. To better describe the operation of the state machine, the actual waveform of the internal timer is output in this application.

Table 5. Output observation of internal timer waveform

Timer	GPIO pin
Timer 0	P4_16
Timer 1	P4_17
Timer 2	P4_18

Table 5. Output observation of internal timer waveform*...continued*

Timer	GPIO pin
Timer 3	P4_19
Timer 4	P4_20

4 FlexIO shifter configuration

This application configures the FlexIO shifter in state machine mode, where different state machines control the output levels of different FlexIO data lines.

4.1 FlexIO state mode used in this application

In this application, the inputs are not routed to the pin pads of the chip. Therefore, all three inputs can be considered as a logic low level, and state group 0 is selected in all the shifters.

Each shifter has only one status group (`Group0`) available for use. Five shifters are utilized, hence there are five states.

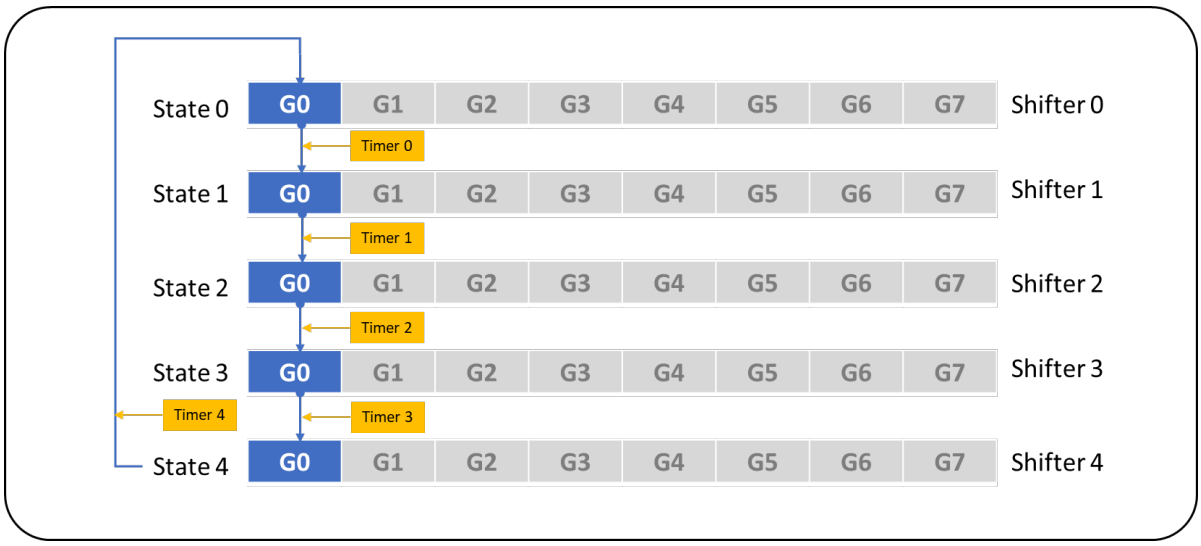

Figure 5. Five shifters

4.2 FlexIO state mode configuration

When the shifter is in the state mode, the main considerations are as follows:

1. Which timer to choose for triggering the state transition.
2. The polarity selection of the timer trigger level.
3. Using three pins as external inputs to determine which state to select.
4. Whether to enable the output and the output polarity.
5. The selection of output pins and whether to mask the pins.
6. The selection of the next state.

Table 6 is the related configuration of this application.

Table 6. FlexIO state mode configuration

-	Shifter 0	Shifter 1	Shifter 2	Shifter 3	Shifter 4
Mode	State mode	State mode	State mode	State mode	State mode
Timer Select	Timer 0	Timer 1	Timer 2	Timer 3	Timer 4
Timer Polarity	Positive	Positive	Positive	Positive	Positive
Pin Select	FlexIO_D16	FlexIO_D16	FlexIO_D16	FlexIO_D16	FlexIO_D16
Pin output	Enable	Enable	Enable	Enable	Enable
Pin polarity	Active High	Active High	Active High	Active High	Active High
Output pins	D0_LOW, D1_ HIGH.	D0_LOW, D1_LOW.	D0_HIGH, D1_LOW.	D0_LOW, D1_LOW.	D0_LOW, D1_ HIGH.
Mask pins	D2~D7	D2~D7	D2~D7	D2~D7	D2~D7
Next state shifter	Shifter 1	Shifter 2	Shifter 3	Shifter 4	Shifter 0

5 FlexIO timer configuration

The FlexIO timer has the following modes:

1. Timer 8-bit Baud Counter mode: The lower eight bits are used to configure the baud rate of the shift clock and the upper eight bits are used to configure the number of shift clock edges in the transfer.
   Note: *In some motor applications, 8-bit precision may not be sufficient, so it is necessary to use 16 bits to generate the baud rate of the shift clock.*
2. Timer 8-bit High PWM mode: The lower eight bits are used to configure the timer output high period and the upper eight bits are used to configure the timer output low period.
3. Timer 16-bit Counter mode: Use the 16-bit counter to configure the baud rate of the shift clock.
   Note: *This application usees the Timer 16-bit Counter mode. Due to the limited number of timers, the number of PWMs is no longer controlled by additional timers, but by adopting a method that enables or disables the timer under specific conditions.*
4. The remaining timer modes are also unsuitable. See the reference manual for details.

5.1 Timer configuration in the application

When the timer is used as a time trigger signal for the shifter state machine, consider the following configurations:

1. Select the trigger source and polarity for the timer.
2. To observe the waveform inside the timer, select the output pin and polarity.
3. Choose the conditions under which to enable/disable/reset the timer.
4. Set the initial output level and whether it is controlled by a reset.
5. Configure the compare register and set the baud rate.

Table 7 is the configuration table for timers 0/1/2/3/4.

Table 7. Timer configuration

-	Timer 0	Timer 1	Timer 2	Timer 3	Timer 4
Mode	**Single 16-bit counter mode**				
Trigger source	Timer 4 trigger	FlexIO_D1 pin	Timer 4 trigger	FlexIO_D0 pin	-
Trigger polarity	Active high	Active low	Active low	Active low	-
Pin select	FlexIO_D24	FlexIO_D25	FlexIO_D26	FlexIO_D27	FlexIO_D28

Table 7. Timer configuration...*continued*

-	Timer 0	Timer 1	Timer 2	Timer 3	Timer 4
Mode	**Single 16-bit counter mode**				
Pin output	Enable	Enable	Enable	Enable	Enable
Pin polarity	Active high	Active high	Active high	Active high	Active high
Enable	Trigger rising edge	Trigger rising edge	Trigger rising edge	Trigger rising edge	Always enable
Disable	Trigger falling edge	Trigger falling edge	Trigger falling edge	Trigger falling edge	Never disable
Reset	Never reset	Never reset	Never reset	Never reset	Never reset
Initial output	Zero	Zero	Zero	Zero	One
CMP register value	49	4	149	4	199
Baud rate	1.5 Mbit/s	15 Mbit/s	500 kbit/s	15 Mbit/s	375 kbps

5.2 Timer trigger and states transition timing

If only one timer is used to generate the trigger waveform for a shifter, the number of PWM waves produced by the timer cannot be controlled. In this application, the method of enabling and disabling the timer is used to allow the timer to generate PWM waveforms within a certain time domain. Then, based on the principle that the trigger source is effective only under specific conditions, it ensures accurate triggering time and that the extra PWM wave cycles generated do not affect the state.

Figure 6 shows all the time points for state transitions and the corresponding timer waveforms.

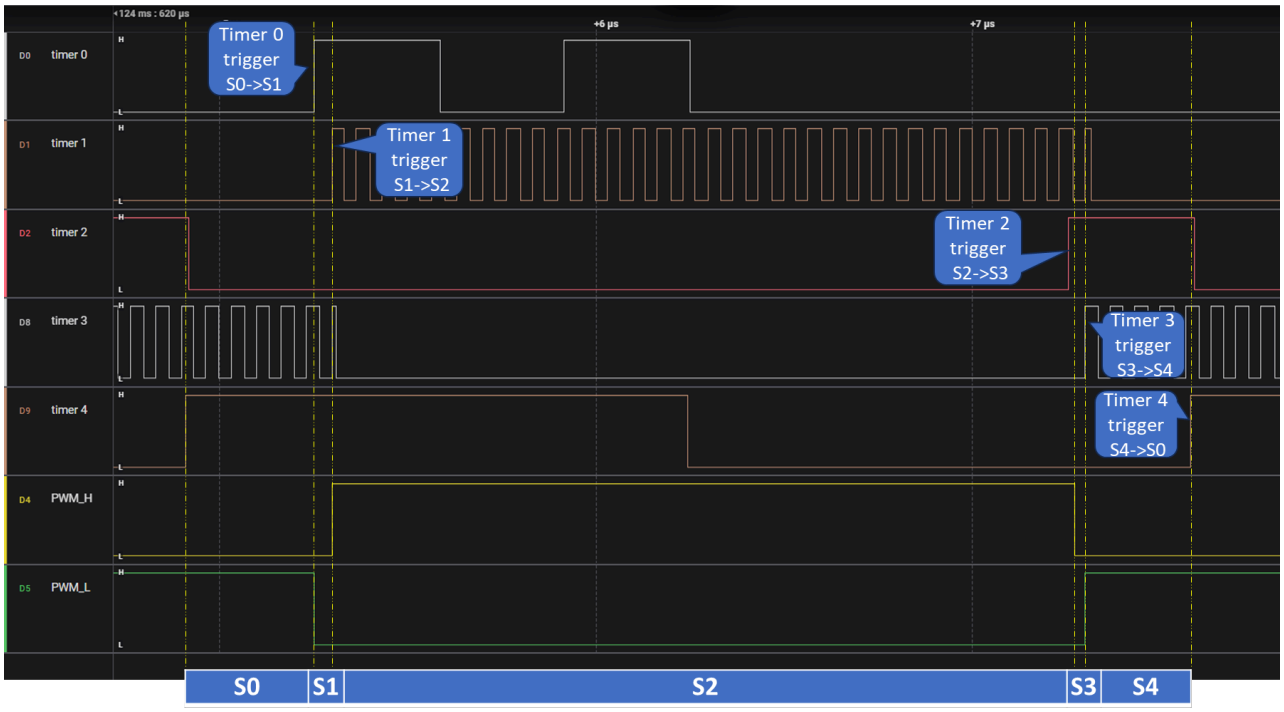

Figure 6. Time points for state transitions

1. Figure 7 shows the process of state transition from state 0 to state 1, triggered by timer 0. The PWM waveform generated by timer 0 is controlled to be enabled and disabled by timer 4. The first rising edge of

timer 0 triggers a state transition. When state 1 is entered, later rising edges of timer 0 do not trigger state transitions. However, to avoid creating more interference, timer 0 is disabled on the rising edge of timer 4.

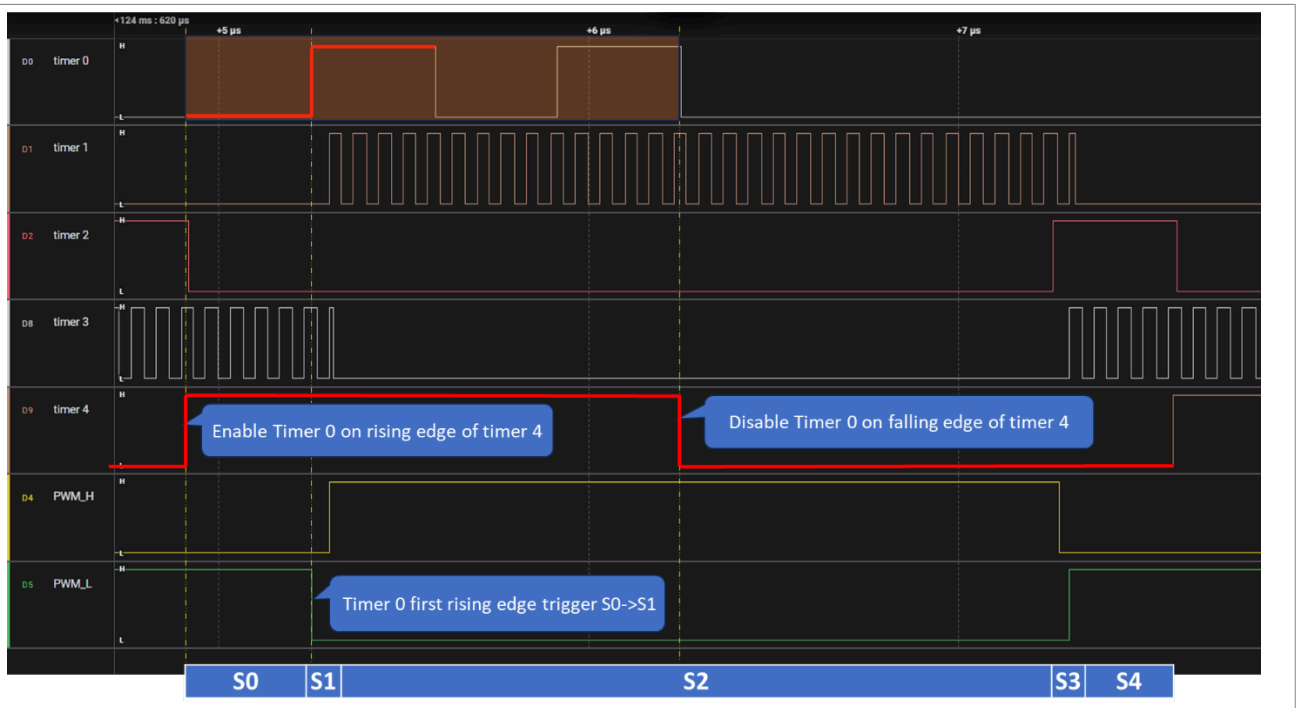

Figure 7. Transition from state 0 to state 1

2. Figure 8 shows the waveform for timer 1 and the state trigger transition. Enable timer 1 on the falling edge of `PWM_L` and disable timer 1 on the rising edge. The first rising edge of timer 1 triggers the transition from state 1 to state 2. Subsequent rising edges do not have an effect.

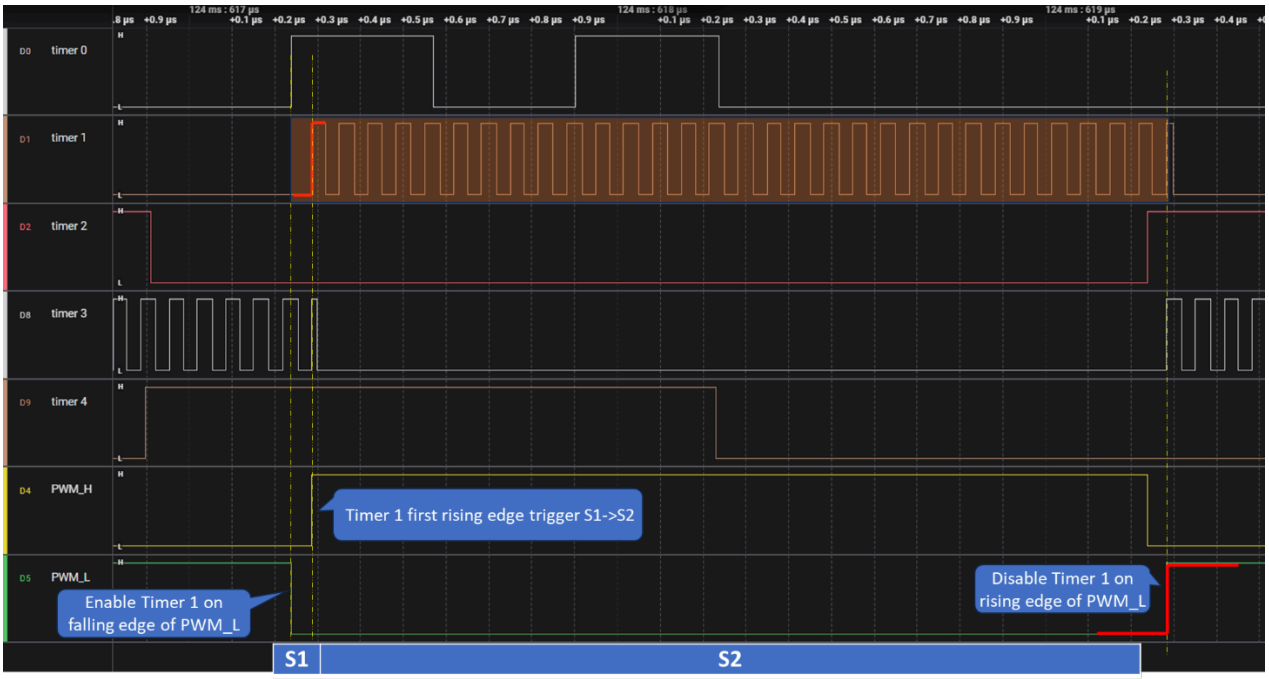

Figure 8. Timer 1 and state trigger transition

3. Figure 9 shows the waveform for timer 2 and the transition from state 2 to state 3. The period of timer 2 is less than half the period of timer 4. The falling edge of timer 4 enables timer 2, while the rising edge of timer 4 disables timer 2, ensuring that timer 2's PWM is within the lower half of timer 4's cycle. Timer 2 does not complete a full cycle, as its lower half is partially "cut off" by timer 4. Timer 2 only generates a single rising edge, which triggers the transition from state 2 to state 3.

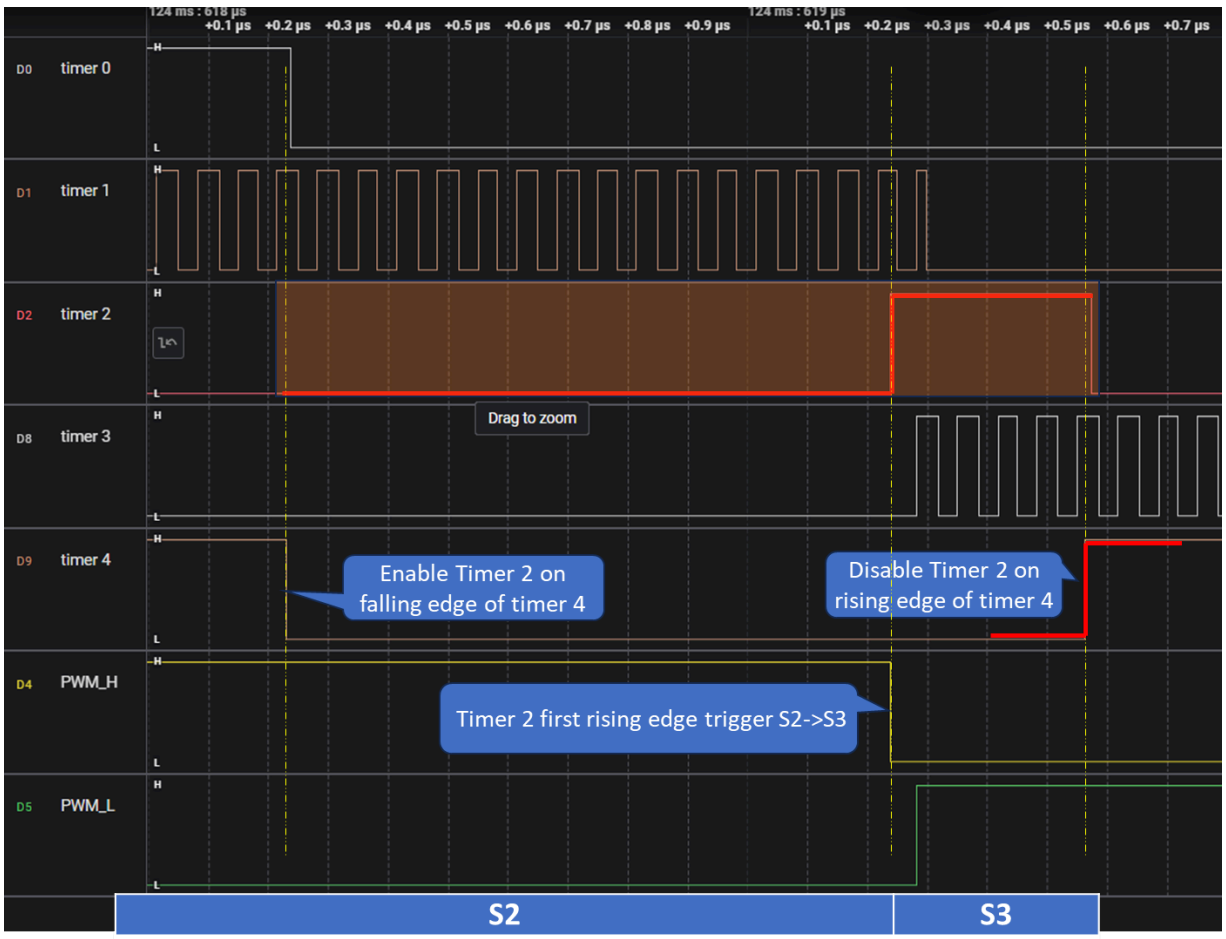

Figure 9. Timer 2 and transition from state 2 to state 3

4. Figure 10 shows the waveforms for timer 3 and the transition from state 3 to state 4. The falling edge of PWM_H enables timer 3, while the rising edge disables it. The first rising edge of timer 3 triggers the transition from state 3 to state 4. Subsequent rising edges do not have an effect.

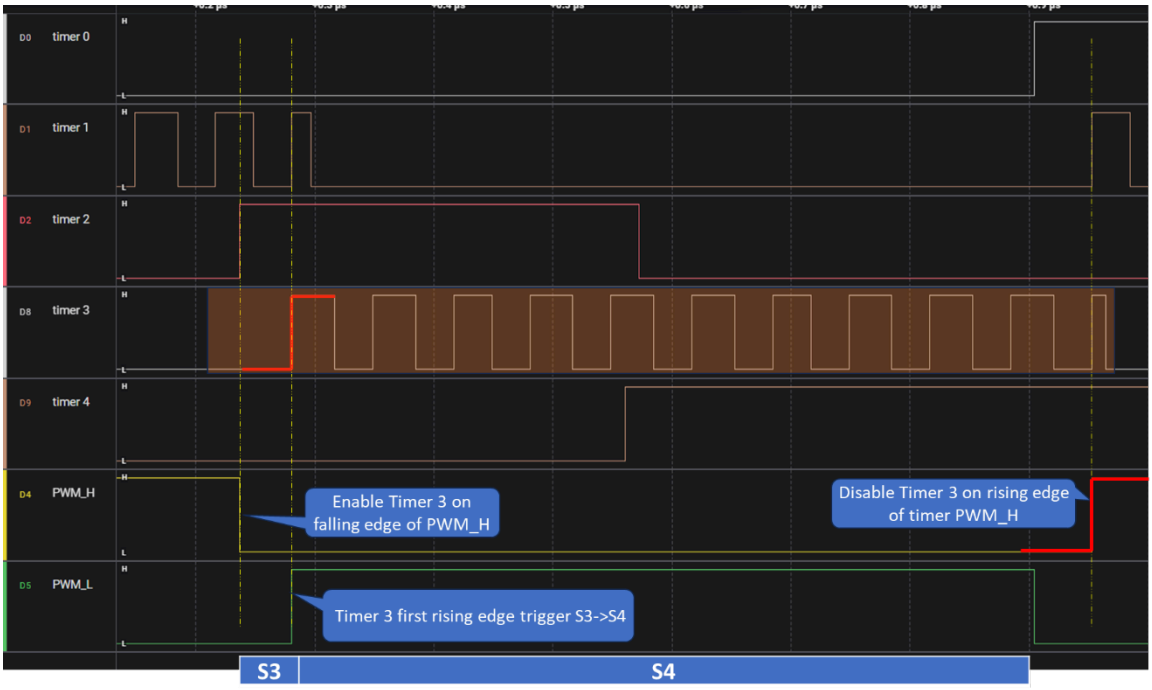

Figure 10. Timer 3 and transition from state 3 to state 4

5. Figure 11 shows the waveforms for timer 4 and the transition from state 4 to state 0. The falling edge of timer 4 triggers the transition from state 4 to state 0. The period of timer 4 is synchronized with the PWM period.

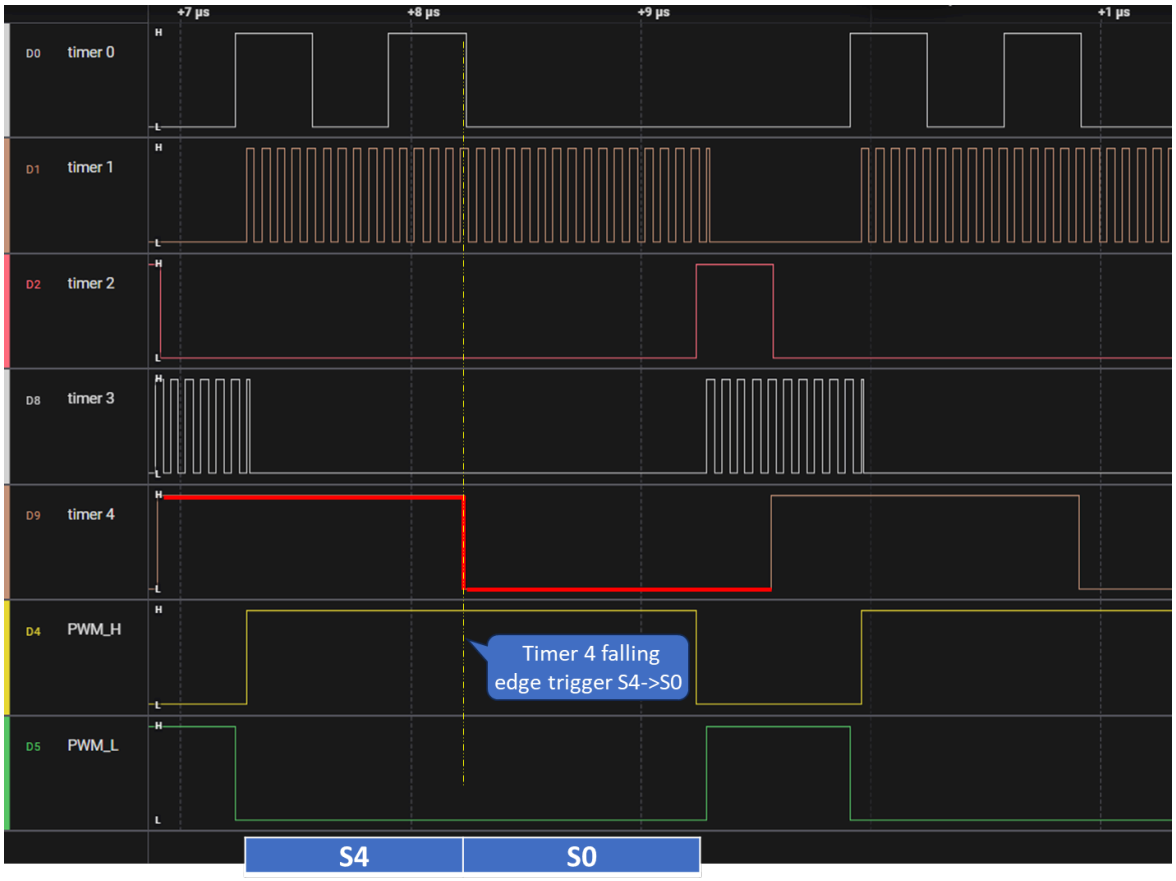

Figure 11. Timer 4 and transition from state 4 to state 0

In summary, the waveforms generated by each timer are segments of PWM waveforms. Their "life cycles" are controlled by the waveforms of timer 4, `PWM_H`, and `PWM_L`. State transitions are primarily triggered by the first rising edge of each timer. Subsequent rising edges do not have a significant effect.

6 MCUXpresso Config Tools in the application

The MCUXpresso Config Tools set is a suite of evaluation and configuration tools that help you from the initial evaluation to the production software development. With the MCUXpresso Config Tools, you can configure NXP Cortex-M processors and generate the initialization of SDK drivers. The MCUXpresso Config Tools is intended for general use and aims to help hardware designers, software engineers, embedded engineers, and Field Application Engineers (FAEs).

The tool is distributed free of charge. The installer for Windows, Linux, or Mac can be downloaded from http://www.nxp.com.

To compile the generated code, the MCUXpresso SDK package is required. You can download SDK packages from http://mcuxpresso.nxp.com. The SDK package contains many example projects that can help you to get started.

6.1 FlexIO state machine

The FlexIO state machine is a complex combination system. Using the configuration tool is very simple and convenient. The generated code is based on the register configuration, which is more intuitive and efficient.

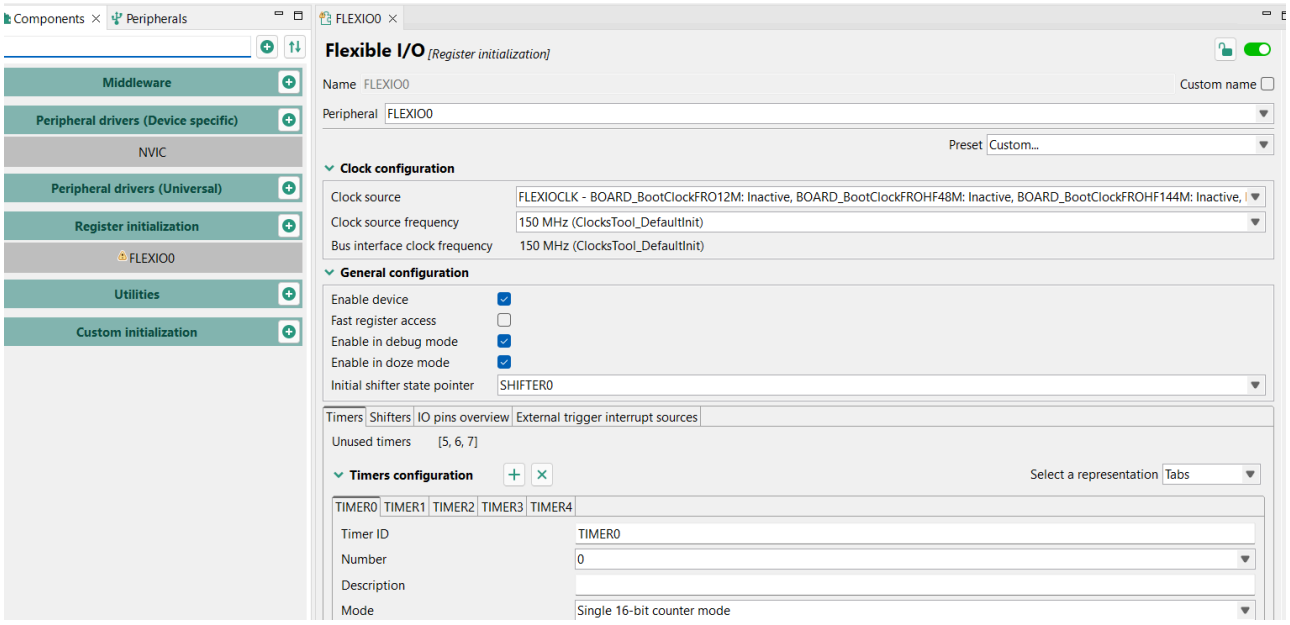

Figure 12. FlexIO state machine

```
/* Pin output register initialization */
#ifdef FLEXIO0_PINOUTD_INIT
  FLEXIO0->PINOUTD = FLEXIO0_PINOUTD_INIT;
#else
  FLEXIO0->PINOUTD = 0x0U;
#endif /* FLEXIO0_PINOUTD_INIT */
#endif /* FLEXIO_PINOUTD_OUTD_MASK */

#ifdef FLEXIO0_CTRL_INIT
  FLEXIO0->CTRL = (FLEXIO0_CTRL_INIT & ~FLEXIO_CTRL_SWRST_MASK);
#else
  FLEXIO0->CTRL = 0x0U;
#endif /* FLEXIO0_CTRL_INIT */
}

/***************************
 * Initialization functions
 ***************************
static void BOARD_InitPeripherals_CommonPreInit(void)
{
  /* Enable clock gate of the FLEXIO0 peripheral. */
  SYSCON0->AHBCLKCTRL2 |= SYSCON_AHBCLKCTRL2_FLEXIO_MASK;
}

void BOARD_InitPeripherals(void)
{
  /* Common pre-initialization */
  BOARD_InitPeripherals_CommonPreInit();
  /* Initialize components */
  FLEXIO0_init();
}

/***********************************
 * BOARD_InitBootPeripherals function
 ***********************************/
```

```
void BOARD_InitBootPeripherals(void)
{
  BOARD_InitPeripherals();
}
```

6.2 Timer setting in MCUXpresso Config Tools

The settings for a timer mainly include when to enable it, when to disable it, and when to reset it. They also include which pin the output comes from, what the polarity of the output is, what the trigger source is, and how the timer's baud rate is set.

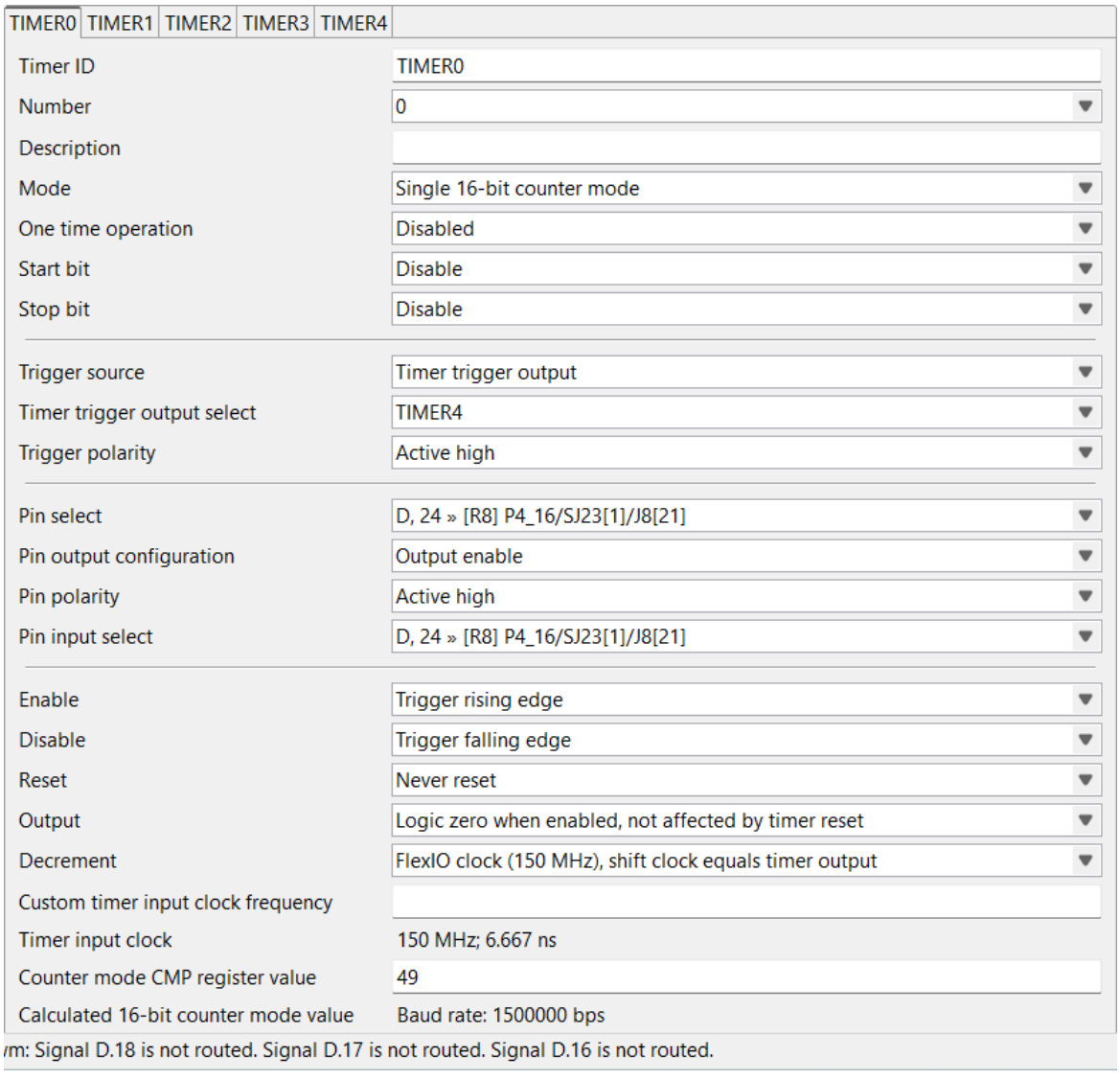

Figure 13. Timer setting in MCUXpresso Config Tools

6.3 Timer setting in MCUXpresso Config Tools

The settings for a shifter primarily include selecting which timer serves as the trigger source, choosing the IO output waveform for the current state, and setting which three pins are used for the state selection.

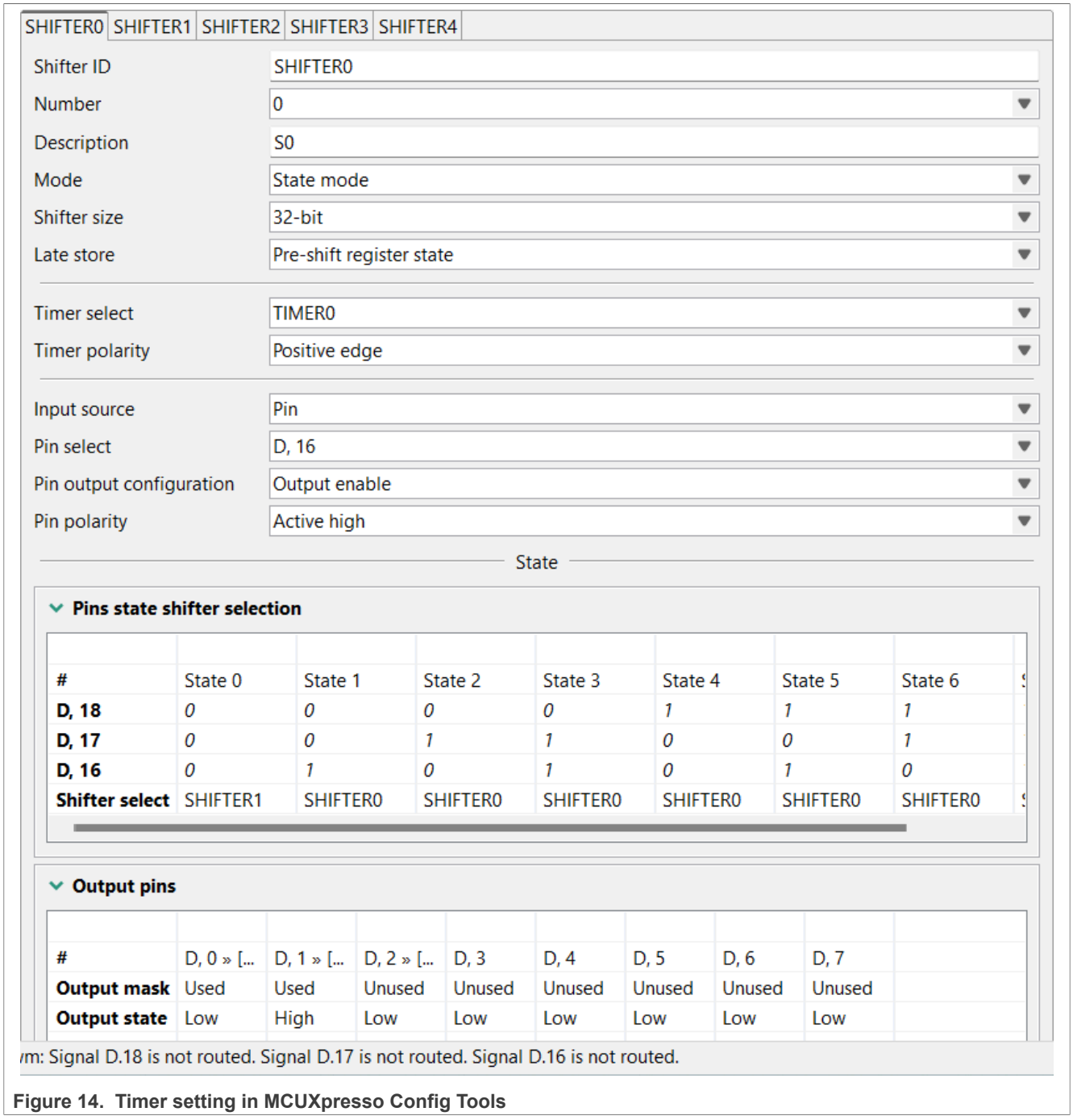

Figure 14. Timer setting in MCUXpresso Config Tools

Summary: Configuring state machines using the configuration tools is indeed quite intuitive and simple. During use, always compare them with the register descriptions, which help to better understand their configuration content.

7 Demo

This application comes with an accompanying program where you can see the relevant configurations of the configuration tools in the reference project.

7.1 Hardware preparation

In the application, the demo is designed and developed based on the FRDM-MCXN947 board. Table 8 is the pin assignment table.

Table 8. Pin assignment table

Function	IO pin	Position on FRDM-MCXN947
PWM_H	P0_8	J8-11
PWM_L	P0_9	J8-10
Timer 0	P4_16	J8-21
Timer 1	P4_17	J8-22
Timer 2	P4_18	J8-23
Timer 3	P4_19	J8-24
Timer 4	P4_20	J8-25

Figure 15 is the hardware connection diagram.

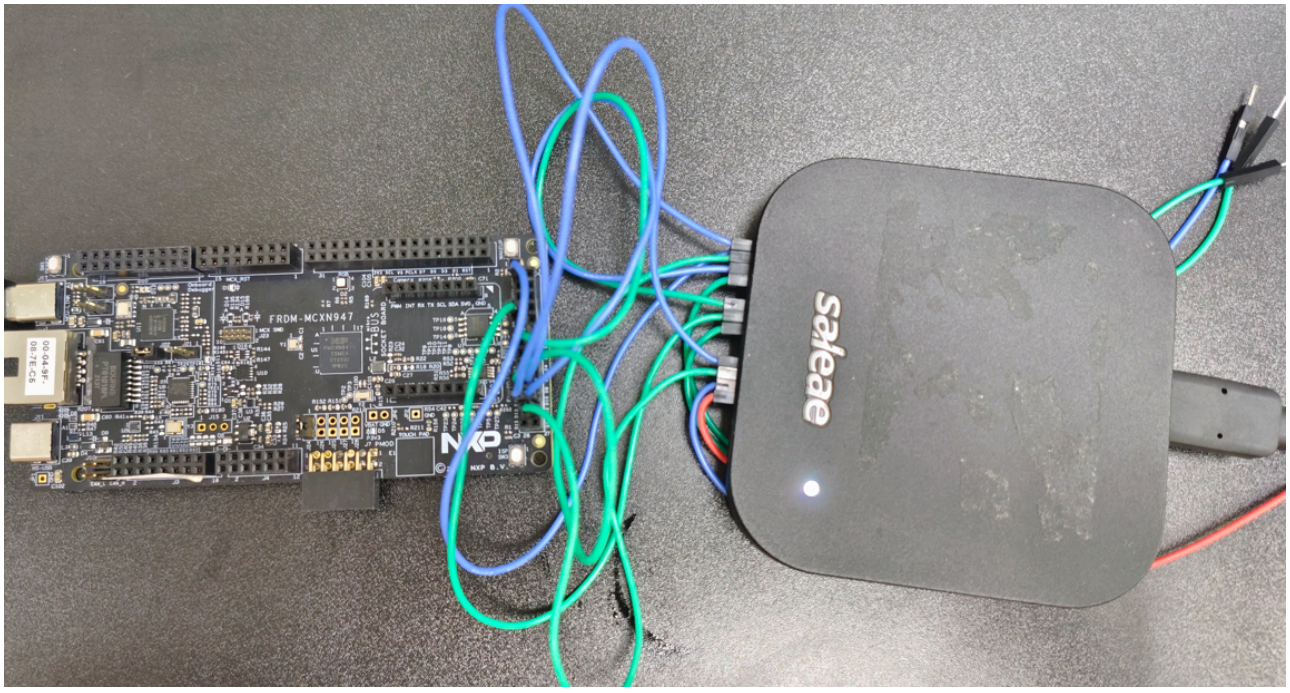

Figure 15. Hardware connection diagram

Board: FRDM-MCXN947 rev. B

Logic device: Saleae logic pro 16

1. Connect the logic device to a PC with a USB cable.
2. Connect FRDM-MCXN947 to a PC with a USB-Type C cable.
3. Connect the capture channel to the header signals with the DuPont wire.

7.2 Software preparation

Unzip the attached software project and open it with the MCUXpresso IDE.

1. Import the project into the IDE.
2. Build the project code.
3. Download the firmware.

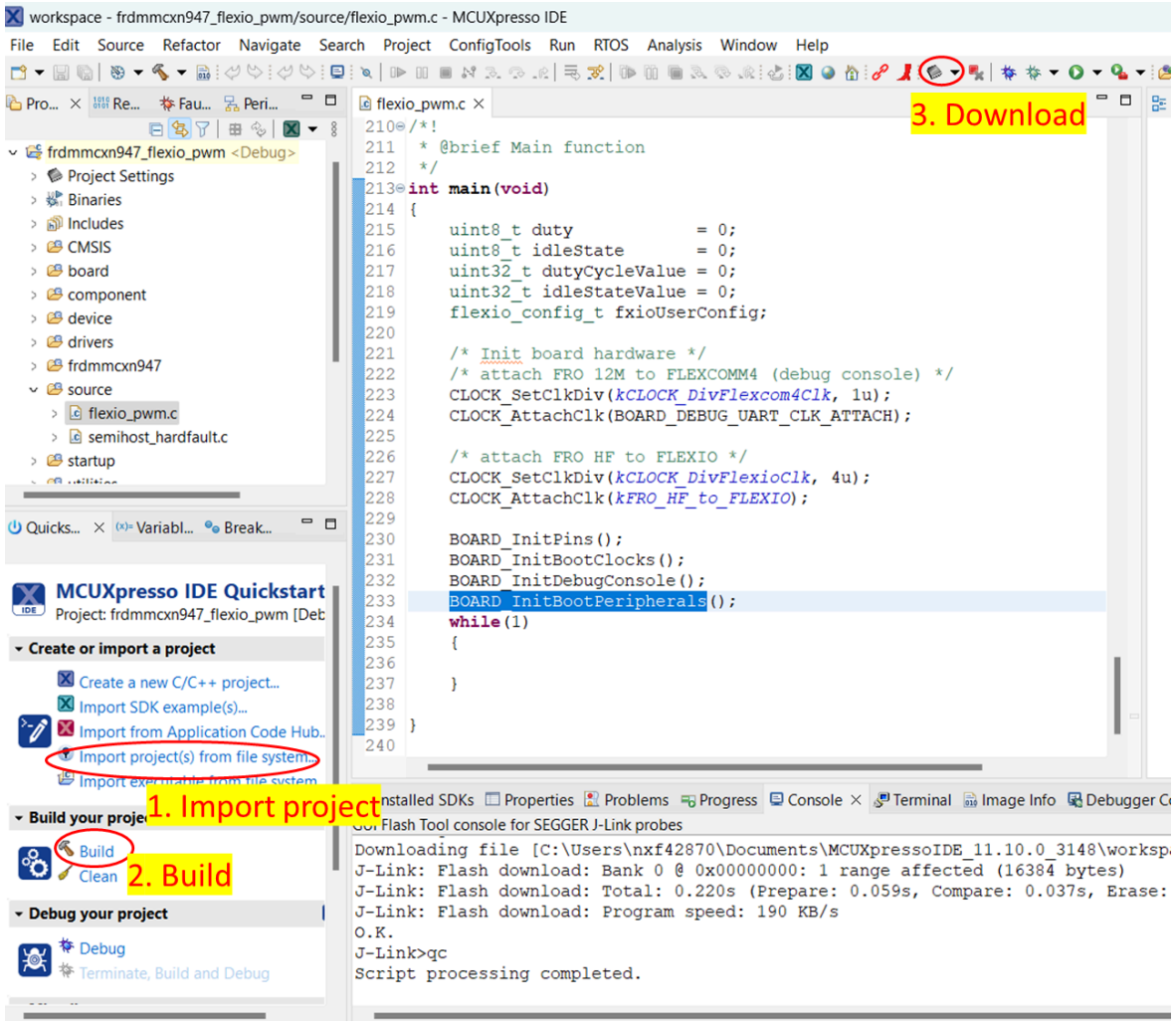

Figure 16. Software preparation

7.3 Result

Reset the board, open the logic application on a PC, and capture the timing, as shown in Figure 17.

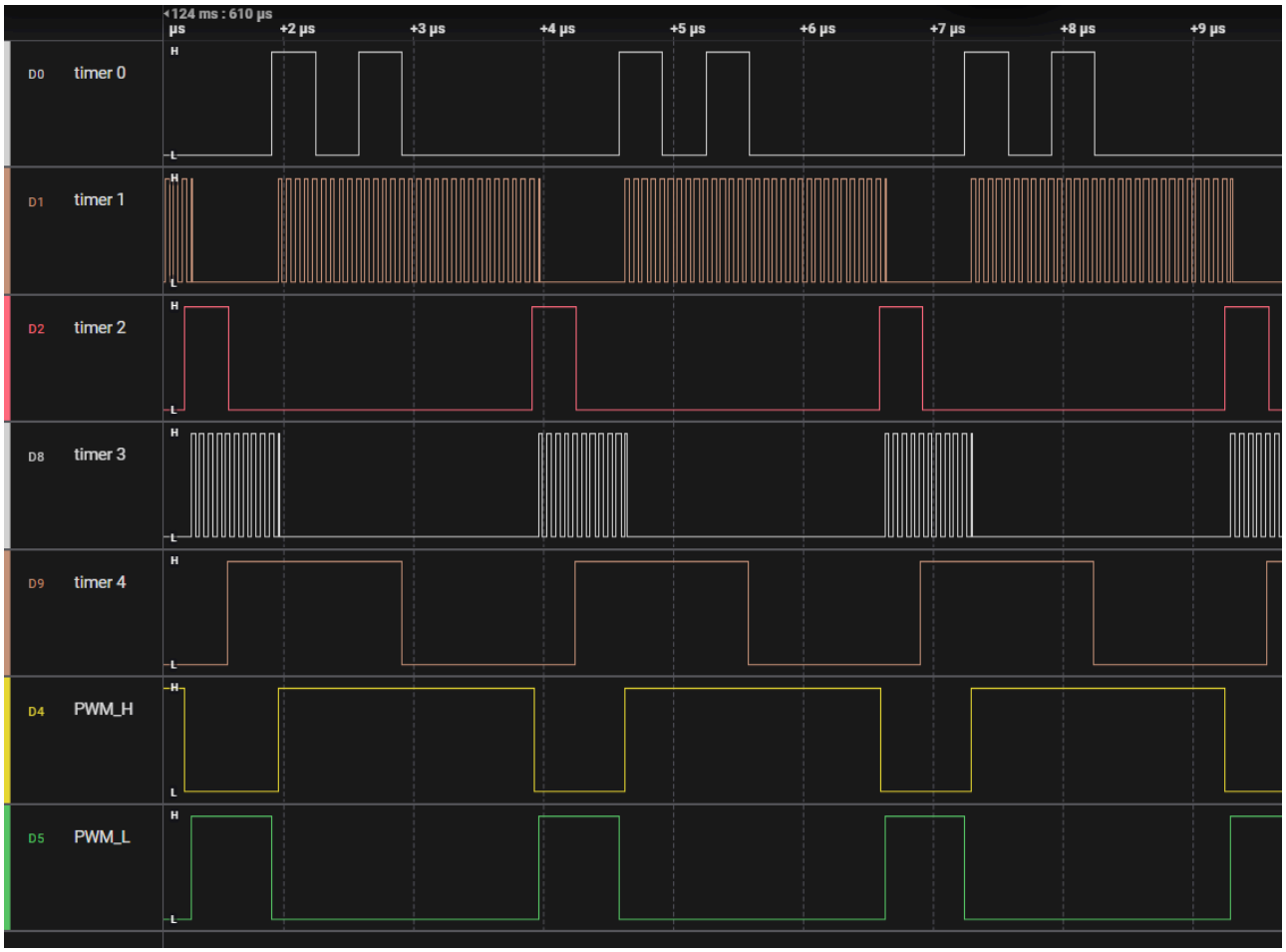

Figure 17. Result

8 References

- *Emulating Hardware State Machine Using FlexIO Module* (document AN5239)

9 Note about the source code in the document

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

Copyright 2025 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.

10 Revision history

Table 9. Revision history

Document ID	Release date	Description
AN14470 v.1.0	06 February 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.

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.

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.

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.

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

Contents