AN14823

Dual Quadrature Decoder Implementation Using SmartDMA on the MCX A344

Rev. 1.0 — 23 September 2025 Application note

Document information

Information	Content
Keywords	AN14823, MCX A344, dual quadrature decoder (QDC), SmartDMA
Abstract	This application note demonstrates the MCX A344 SmartDMA dual QDC demo, showcasing high-performance dual-channel quadrature decoding using the SmartDMA coprocessor.

1 Introduction

This application note demonstrates the MCX A344 SmartDMA dual quadrature decoder (QDC) demo, showcasing high-performance dual-channel quadrature decoding using the SmartDMA coprocessor. The demo features concurrent 2.1 MHz dual-axis QDC signal handling with autonomous operation and real-time monitoring capabilities.

The demonstration uses the following two FRDM development boards:

- FRDM-MCXN947: Generates programmable dual-channel QDC waveforms.
- FRDM-MCXA344: Performs real-time signal acquisition and processing.

The following is the performance of the dual QDC implementation by SmartDMA:

- Concurrent processing: 2.1 MHz dual-axis QDC signal handling.
- Autonomous operation: Independent waveform parsing with automatic updates.
- Optimized performance: Reduced CPU loading through SmartDMA coprocessor.
- Real-time monitoring: Continuous counter updates via universal asynchronous receiver/transmitter (UART).

2 The MCX A344 functions in motor control

The MCX A344 microcontroller delivers comprehensive motor control capabilities through the:

- 180 MHz Arm Cortex-M33 CPU with 738 CoreMark performance
- MAU math acceleration unit for motor algorithms
- SmartDMA coprocessor for task offloading

The dual 16-bit analog-to-digital converters (ADCs) sample at 3 Msps across 39 channels. The pulse width modulation (PWM) synchronization supports precise current/voltage/temperature sensing, while integrated operational amplifiers prepare analog signals for optimal ADC performance.

Multiple 32-bit timers and 2x FlexPWM modules (each with four submodules) generate 16 complementary PWM outputs with hardware deadtime and fault protection. The 2x AND/OR/Invert (AOI) modules support up to four output triggers for advanced control logic.

For position feedback, the 2x hardware eQDC (Quadrature Encoder/Decoder) modules deliver zero-CPU quadrature decoding with ±1 pulse accuracy. The SmartDMA can simulate extra virtual QDCs using GPIO/timers/ADC resources to achieve 4-axis control capability. It enables applications like CNC machines (X/Y/Z + spindle) or multi-joint robotics with mixed precision allocation. In mixed precision, the critical axes use hardware eQDCs for maximum accuracy while the auxiliary axes use virtual QDCs for cost-effective control. It provides a single-chip solution that reduces system complexity and development costs.

Table 1. Specifications and applications of function modules

Function module	Specifications	Application
Processor	180 MHz Cortex-M33 738 coreMark performance MAU math acceleration SmartDMA coprocessor	High-performance computing Motor algorithm acceleration Task offloading
ADC system	2x 16-bit ADCs 3 Msps sampling rate 39 channels PWM synchronization	Current/Voltage/Temperature sensing Precise sampling timing
PWM system	2x FlexPWM modules 16 complementary PWM outputs	Motor drive control Shoot-through protection

Table 1. Specifications and applications of function modules*...continued*

Function module	Specifications	Application
	Hardware deadtime Fault protection	
Encoder	2x hardware eQDC modules	Position Feedback Speed Detection
Op-Amps	Integrated operational amplifiers	Analog signal conditioning

3 QDC

The QDC is a specialized hardware module that is designed to process quadrature encoder signals. It is widely used in motor control systems for position and speed detection. The QDC automatically decodes Phase-A and Phase-B signals from incremental encoders, providing precise position counting and direction detection capabilities.

3.1 Core functional features

Table 2 shows the core functional features of signal decoding and position counting modules.

Table 2. Functional features

Function module	Technical specifications	Application description
Signal decoding	Quadrature signal logic decoding Phase-A/Phase-B signal processing Automatic direction recognition	Incremental encoder interface Rotation direction detection Position change tracking
Position counting	32-bit position counter Modulo counting capability Software/external event initialization	Absolute position recording Multi-turn counting Position reference setting

3.2 Direction detection

The hardware automatically recognizes the direction of rotation of phases without any software intervention. The following two are the direction of rotation:

- Positive direction: The Phase-A signal leads the Phase-B signal.
- Negative direction: The Phase-A signal trails the Phase-B signal.

3.3 Typical applications

The following are the typical applications:

- Servo motor control: Precise position feedback and speed detection.
- Stepper motor systems: Position verification and step loss detection.
- Robotic joints: Multi-axis position control and motion planning.
- CNC machine tools: High-precision positioning and speed control.
- Automation equipment: Conveyor belt position tracking and synchronization control.

The QDC module implements complex encoder signal processing in hardware, significantly simplifying software development complexity while improving system real-time performance and reliability. It serves as an essential component in modern motor control systems.

3.4 Phase relationship and direction detection

Table 3 shows the relationship between Phase-A and Phase-B and direction of rotation of phases.

Table 3. Phase relationship and direction detection

Rotation direction	Phase relationship	Signal timing	Detection logic
Forward rotation	Phase-A leads Phase-B by 90°	Phase-B is LOW on Phase-A rising edge Phase-B is HIGH on Phase-A falling edge	Forward count (+1)
Reverse rotation	Phase-A lags Phase-B by 90°	Phase-B is HIGH on Phase-A rising edge Phase-B is LOW on Phase-A falling edge	Reverse count (-1)

4 SmartDMA dual QDC implementation

SmartDMA is a coprocessor unit within the MCX A344 MCU that can execute a reduced instruction set. It provides precise timing control and can access the general-purpose input/output (GPIO) in a single cycle that makes it ideal for implementing timing-critical protocols like QDC.

4.1 QDC implementation principles

Table 4 shows the conditions for the QDC counter increment:

Table 4. Conditions for QDC counter increment

Trigger signal	Edge type	Other phase state	Logic description
Phase-A	Rising Edge ↑	Phase-B = 0	Phase-A rises when Phase-B is low
Phase-B	Rising Edge ↑	Phase-A = 1	Phase-B rises when Phase-A is high
Phase-B	Falling Edge ↓	Phase-A = 0	Phase-B falls when Phase-A is low
Phase-A	Falling Edge ↓	Phase-B = 1	Phase-A falls when Phase-B is high

Table 5 shows the conditions for the QDC counter decrement:

Table 5. Conditions for QDC counter decrement

Trigger signal	Edge type	Other phase state	Logic description
Phase-A	Falling Edge ↓	Phase-B = 0	Phase-A falls when Phase-B is low
Phase-B	Falling Edge ↓	Phase-A = 1	Phase-B falls when Phase-A is high
Phase-B	Rising Edge ↑	Phase-A = 0	Phase-B rises when Phase-A is low
Phase-A	Rising Edge ↑	Phase-B = 1	Phase-A rises when Phase-B is high

4.2 Timing pseudocode

To facilitate better understanding, a SmartDMA-based QDC communication protocol using pseudocode format is presented in this section. The following implementation demonstrates the precise timing sequences for encoder signal processing and counter update operations.

```
BEGIN
1. PhaseA = READ_PIN(QDC_PhaseA_Pin) // Sample current Phase A state
2. PhaseB = READ_PIN(QDC_PhaseB_Pin) // Sample current Phase B state
3. PhaseA_Changed = (PhaseA != PhaseA_Previous) // Detect Phase A transition
```

```
4. PhaseB_Changed = (PhaseB != PhaseB_Previous) // Detect Phase B transition
5. IF (PhaseA_Changed) THEN // Process Phase A edge event
6. IF (PhaseA == PhaseB) THEN
7. QDC_Counter = QDC_Counter - 1 // Decrement for reverse direction
8. ELSE
9. QDC_Counter = QDC_Counter + 1 // Increment for forward direction
10. END IF
11. ELSE IF (PhaseB_Changed) THEN // Process Phase B edge event
12. IF (PhaseA == PhaseB) THEN
13. QDC_Counter = QDC_Counter + 1 // Increment for forward direction
14. ELSE
15. QDC_Counter = QDC_Counter - 1 // Decrement for reverse direction
16. END IF
17. END IF
18. PhaseA_Previous = PhaseA // Store current state for next cycle
19. PhaseB_Previous = PhaseB // Store current state for next cycle
END
```

4.3 Performance

The SmartDMA coprocessor provides the following advantages for QDC applications:

- Reduced CPU loading: Offloads repetitive QDC processing tasks, maintaining less than 1 % CPU utilization overhead.
- High-speed processing: Achieves 2.1 MHz quadrature signal processing capability at 180 MHz system frequency.
- Concurrent operation: Simultaneous dual-channel processing for multiple encoder inputs.
- Autonomous management: Independent counter and direction updates without CPU intervention, preserving system resources for other critical tasks.

Note: *This application only demonstrates support of SmartDMA for a specific QDC mode. By modifying the SmartDMA code implementation, it can also process other modes of waveform combinations for different modes and waveform types.*

5 Demo code

The demo code implements a SmartDMA-driven dual QDC interface, featuring firmware-based initialization, parameter configuration, and real-time position tracking for two independent quadrature encoders with hardware-accelerated decoding.

5.1 SmartDMA initialization

SmartDMA initialization for dual QDC follows the standard initialization pattern, which requires clock enablement and reset state release. The application encapsulates SmartDMA QDC instructions in a firmware array and assigns the array address to the Boot register. This configuration allows SmartDMA to execute QDC decoding operations autonomously. The relevant parameter data and register configurations must be provided before execution as the application manages dual QDC channels.

The following is the initialization code:

```
static void smartdma_init(void)
{
SMARTDMA_InitWithoutFirmware();
smartdmaParam.smartdma_stack = (uint32_t*)g_smartdma_stack;
smartdmaParam.p_qdc_registers = (uint32_t*)&smdma_qdc_reg;
SMARTDMA_InstallFirmware(SMARTDMA_DUAL_QDC_MEM_ADDR,
```

```
s_smartdmaDualQDCFirmware,
s_smartdmaDualQDCFirmwareSize);
SMARTDMA_Boot(kSMARTDMA_dual_qdc, &smartdmaParam, 0x2);
PRINTF("SmartDMA QDC initialized\r\n");
}
```

SMARTDMA_InitWithoutFirmware: Enables the clock of SmartDMA and releases reset.

SMARTDMA_InstallFirmware: Assigns the dual QDC firmware array to SRAMX memory, as SmartDMA code must be executed in SRAMX for optimal performance.

SmartDMA implements dual QDC functionality through the firmware array s_smartdmaDualQDCFirmware, which is defined in the fsl_smartdma_mcxa.c file.

5.2 SmartDMA parameter configuration

The smartdmaParam variable defines the following parameters for dual QDC operation.

Table 6. Parameter configuration

Parameter	Type	Description
p_smartdma_stack	uint32_t*	The SmartDMA stack pointer points to the stack memory area allocated for Smart DMA.
p_qdc_registers	uint32_t*	The QDC register structure pointer points to the dual QDC register configuration and status data.

5.3 QDC register configuration

The QDC register structure contains all necessary configuration and status registers for both the QDC channels:

```
static void qdc_init(void)
{
  memset(&smdma_qdc_reg, 0, sizeof(SMDMA_QDC_TypeDef));
/* Set counter limits */
smdma_qdc_reg.QDC0_MAX_COUNT = QDC_MAX_COUNT;
smdma_qdc_reg.QDC1_MAX_COUNT = QDC_MAX_COUNT;
smdma_qdc_reg.QDC0_MIN_COUNT = QDC_MIN_COUNT;
smdma_qdc_reg.QDC1_MIN_COUNT = QDC_MIN_COUNT;
/* Set preset values */
smdma_qdc_reg.QDC0_PRESET = QDC_PRESET_VALUE;
smdma_qdc_reg.QDC1_PRESET = QDC_PRESET_VALUE;
/* Set initial direction */
smdma_qdc_reg.QDC0_DIRECTION = QDC_DIR_CLOCKWISE;
smdma_qdc_reg.QDC1_DIRECTION = QDC_DIR_CLOCKWISE;
}
```

5.3.1 QDC register structure

The SMDMA_QDC_TypeDef structure contains comprehensive register definitions for dual QDC operation:

```
typedef struct {
uint32_t QDC0_PHASE_A; /**< QDC0 Phase A Input - GPIO1 PDR[6] */
uint32_t QDC1_PHASE_A; /**< QDC1 Phase A Input - GPIO1 PDR[6] */
uint32_t RESERVED_0; /**< Reserved Register (QDC2_PHASE_A) */
uint32_t RESERVED_1; /**< Reserved Register (QDC3_PHASE_A) */
uint32_t QDC0_PHASE_B; /**< QDC0 Phase B Input - GPIO1 PDR[3] */
uint32_t QDC1_PHASE_B; /**< QDC1 Phase B Input - GPIO1 PDR[3] */
```

```
uint32_t RESERVED_2; /**< Reserved Register (QDC2_PHASE_B) */
uint32_t RESERVED_3; /**< Reserved Register (QDC3_PHASE_B) */
uint32_t QDC0_COUNT; /**< QDC0 Counter Value */
uint32_t QDC1_COUNT; /**< QDC1 Counter Value */
uint32_t RESERVED_4; /**< Reserved Register (QDC2_COUNT) */
uint32_t RESERVED_5; /**< Reserved Register (QDC3_COUNT) */
uint32_t QDC0_MIN_COUNT; /**< QDC0 Minimum Counter Value */
uint32_t QDC1_MIN_COUNT; /**< QDC1 Minimum Counter Value */
uint32_t RESERVED_6; /**< Reserved Register (QDC2_MIN_COUNT) */
uint32_t RESERVED_7; /**< Reserved Register (QDC3_MIN_COUNT) */
uint32_t QDC0_MAX_COUNT; /**< QDC0 Maximum Counter Value */
uint32_t QDC1_MAX_COUNT; /**< QDC1 Maximum Counter Value */
uint32_t RESERVED_8; /**< Reserved Register (QDC2_MAX_COUNT) */
uint32_t RESERVED_9; /**< Reserved Register (QDC3_MAX_COUNT) */
uint32_t QDC0_PRESET; /**< QDC0 Preset Value */
uint32_t QDC1_PRESET; /**< QDC1 Preset Value */
uint32_t RESERVED_10; /**< Reserved Register (QDC2_PRESET) */
uint32_t RESERVED_11; /**< Reserved Register (QDC3_PRESET) */
uint32_t QDC0_DIRECTION; /**< QDC0 Direction Register */
uint32_t QDC1_DIRECTION; /**< QDC1 Direction Register */
uint32_t RESERVED_12; /**< Reserved Register (QDC2_DIRECTION) */
uint32_t RESERVED_13; /**< Reserved Register (QDC3_DIRECTION) */
uint32_t QDC0_SIGNAL_STATE; /**< QDC0 Current Signal State Register */
uint32_t QDC1_SIGNAL_STATE; /**< QDC1 Current Signal State Register */
uint32_t RESERVED_14; /**< Reserved Register (QDC2_SIGNAL_STATE) */
uint32_t RESERVED_15; /**< Reserved Register (QDC3_SIGNAL_STATE) */
uint32_t QDC0_PREV_SIGNAL_STATE; /**< QDC0 Previous Signal State Register */
uint32_t QDC1_PREV_SIGNAL_STATE; /**< QDC1 Previous Signal State Register */
uint32_t RESERVED_16; /**< Reserved Register (QDC2_PREV_SIGNAL_STATE) */
uint32_t RESERVED_17; /**< Reserved Register (QDC3_PREV_SIGNAL_STATE) */
union {
uint32_t QDC0_STATUS; /**< QDC0 Status Register */
struct {
uint32_t counter_updated : 1; /**< Bit 0: Counter Updated Flag */
uint32_t reserved : 31; /**< Bits 1-31: Reserved */
} QDC0_STATUS_BITS;
};
union {
uint32_t QDC1_STATUS; /**< QDC1 Status Register */
struct {
uint32_t counter_updated : 1; /**< Bit 0: Counter Updated Flag */
uint32_t reserved : 31; /**< Bits 1-31: Reserved */
} QDC1_STATUS_BITS;
};
uint32_t RESERVED_18; /**< Reserved Register (QDC2_STATUS) */
uint32_t RESERVED_19; /**< Reserved Register (QDC3_STATUS) */
} SMDMA_QDC_TypeDef;
```

Note: *The register structure supports up to four QDC interfaces (QDC0 to QDC3), with reserved registers allocated for future expansion to QDC2 and QDC3 channels. However, supporting more interfaces require a trade-off in waveform frequency parsing capability, as the SmartDMA processing bandwidth is shared among all active channels.*

5.3.2 Key register categories

Table 7 shows the key register categories and their description.

Table 7. Register categories

Register Category	Description
Phase input registers	QDC0/1_PHASE_A/B - Direct GPIO input state for quadrature signals
Counter registers	QDC0/1_COUNT - Current position counter values
Limit registers	QDC0/1_MIN/MAX_COUNT - Counter boundary values for rollover control
Configuration registers	QDC0/1_PRESET - Initial counter values QDC0/1_DIRECTION - Rotation direction
State registers	QDC0/1_SIGNAL_STATE - Current quadrature state, QDC0/1

5.3.3 Dual QDC waveforms generation

This function implements a dual-channel quadrature encoder signal generator designed to simulate rotary encoder outputs. It simultaneously generates two independent quadrature signal channels (Q0 and Q1), each containing A, and B outputs with a 90° phase relationship. The function uses a state lookup table to cycle through the standard quadrature encoding sequence (1,0 → 1,1 → 0,1 → 0,0). In this application, a 2 MHz quadrature encoder waveform is used as an example. It operates at 2 MHz, with each state lasting 125 ns.

The Q1 channel features configurable phase-offset capability, using a delay counter to control the phase difference relative to the Q0 channel. This generator is primarily used for motor control system testing, encoder interface validation, and position feedback system calibration. It provides developers with a software-based solution to simulate encoder behavior without requiring physical rotating hardware.

The following code can be ported to the FRDM-MCXN947 software development kit (SDK) project. Ensure that you initialize the GPIO pins before using them.

```
void generateDualQuadratureWaveforms(uint32_t numCycles)
{
uint32_t halfPeriodNs = 100; // 2MHz signal, 500ns period, 125ns per state
uint32_t cycles = 0;
uint8_t stateQ0 = 0, stateQ1 = 0;
uint32_t q1DelayCounter = 0;
while (cycles < numCycles)
{
// Update Q0 output
switch (stateQ0)
{
case 0:
GPIO_PortSet(BOARD_INITPINS_Q0_A_GPIO, BOARD_INITPINS_Q0_A_GPIO_PIN_MASK);
GPIO_PortClear(BOARD_INITPINS_Q0_B_GPIO, BOARD_INITPINS_Q0_B_GPIO_PIN_MASK);
break;
case 1:
GPIO_PortSet(BOARD_INITPINS_Q0_B_GPIO, BOARD_INITPINS_Q0_B_GPIO_PIN_MASK);
break;
case 2:
GPIO_PortClear(BOARD_INITPINS_Q0_A_GPIO, BOARD_INITPINS_Q0_A_GPIO_PIN_MASK);
break;
case 3:
GPIO_PortClear(BOARD_INITPINS_Q0_B_GPIO, BOARD_INITPINS_Q0_B_GPIO_PIN_MASK);
break;
}
stateQ0 = (stateQ0 + 1) % 4;
// Update Q1 output (with phase shift)
if (q1DelayCounter >= PHASE_SHIFT_NS / halfPeriodNs)
{
switch (stateQ1)
```

```
{
case 0:
GPIO_PortSet(BOARD_INITPINS_Q1_A_GPIO, BOARD_INITPINS_Q1_A_GPIO_PIN_MASK);
GPIO_PortClear(BOARD_INITPINS_Q1_B_GPIO, BOARD_INITPINS_Q1_B_GPIO_PIN_MASK);
break;
case 1:
GPIO_PortSet(BOARD_INITPINS_Q1_B_GPIO, BOARD_INITPINS_Q1_B_GPIO_PIN_MASK);
break;
case 2:
GPIO_PortClear(BOARD_INITPINS_Q1_A_GPIO, BOARD_INITPINS_Q1_A_GPIO_PIN_MASK);
break;
case 3:
GPIO_PortClear(BOARD_INITPINS_Q1_B_GPIO, BOARD_INITPINS_Q1_B_GPIO_PIN_MASK);
break;
}
stateQ1 = (stateQ1 + 1) % 4;
}
else
{
q1DelayCounter++;
}
delayNs(halfPeriodNs); // 125ns delay (calibrated)
cycles++;
}
}
```

6 Demo application

To demonstrate the MCX A344 SmartDMA dual QDC functionality, this application implements a comprehensive quadrature encoder decoding system. The demo uses the following two development boards working in tandem:

- FRDM-MCXN947 generates programmable dual-channel QDC waveforms.
- FRDM-MCXA344 performs real-time signal acquisition and processing using a SmartDMA coprocessor for optimized performance.

6.1 Hardware setup

The demo application requires two NXP development boards connected via specific GPIO pins to establish the QDC signal communication interface.

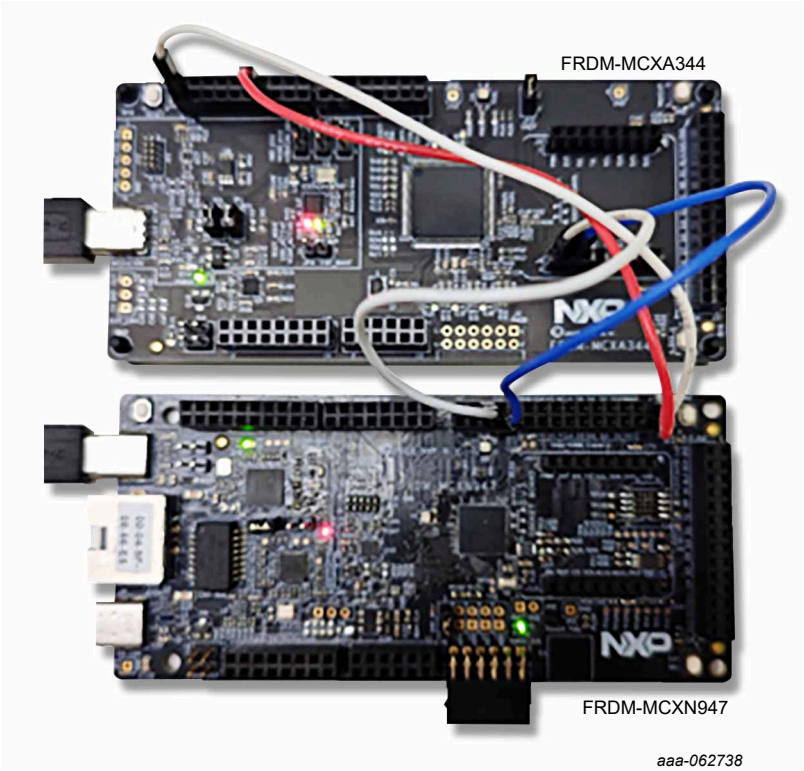

Figure 1. Hardware setup

6.2 Development board configuration

Table 8 shows the configuration for the FRDM-MCXN947 and FRDM-MCXA344 boards.

Table 8. Development board configuration

Board	Function	Key features
FRDM-MCXN947	Signal Generator	Generates programmable dual-channel QDC waveforms
FRDM-MCXA344	Signal Processor	Real-time signal acquisition and processing with SmartDMA

6.2.1 Pin connection mapping

Table 9 shows the exact pin connections between the FRDM-MCXN947 and FRDM-MCXA344 boards:

Table 9. Pin-mapping

FRDM-MCXN947	FRDM-MCXA344	Signal	Function
J9 pin 4 (P1_16)	J2 Pin 7 (P1_4)	Q1_A	QDC1 Phase-A signal
J9 pin 3 (P1_17)	J2 Pin 19 (P1_5)	Q1_B	QDC1 Phase-B signal
J9 pin 28 (P1_12)	J6 pin 1 (P1_6)	Q0_A	QDC0 Phase-A signal
J9 pin 23 (P1_23)	J6 pin 2 (P1_7)	Q0_B	QDC0 Phase-B signal
GND	GND	-	Common ground

6.3 Software setup and demo run

The following sections specify the operation steps for running the SmartDMA dual QDC demonstration.

6.3.1 Development environment setup

To set up the development environment, perform the following steps:

1. Open the MCUXpresso IDE.
2. Import the project from the application GitHub.
3. Find and open the project **an-mcxa344-dual-quadrature-decoder-by-smartdma**.
4. Compile the project and download it to the FRDM-MCXA344 board.

6.3.2 FRDM-MCXN947 signal generation setup

Note: *The MCXN947 application generates the dual-channel quadrature encoder signals. Port the QDC signal generation code from the provided examples into the FRDM-MCXN947 SDK hello_world routine. Download it to the board to generate the required waveforms.*

1. Start with the FRDM-MCXN947 SDK hello_world example.
2. Add the dual QDC signal generation code to the main application.
3. Configure GPIO pins according to Table 9:
   - P1_16 (J9 pin 4) → Q1_A output
   - P1_17 (J9 pin 3) → Q1_B output
   - P1_12 (J9 pin 28) → Q0_A output
   - P1_23 (J9 pin 23) → Q0_B output
4. Configure programmable frequency, phase relationships, and signal patterns.
5. Build the modified hello_world project and download it to the FRDM-MCXN947 board.

6.3.3 Demo execution

To execute the demo, perform the following steps:

1. Connect the FRDM-MCXN947 and FRDM-MCXA344 boards according to Table 9.
2. Connect both the boards to the computer via USB cables.
3. Open the serial port assistant and set the baud rate to 115200.
4. Connect the serial port to the FRDM-MCXA344 debug UART port.
5. To begin operation, press the reset button on the FRDM-MCXA344 board.
6. Ensure that the FRDM-MCXN947 board is generating the QDC waveforms.
7. Observe real-time counter values via UART.

6.3.4 Expected output results

The serial port output displays continuous monitoring of both the QDC channels:

```
MCXA344 Dual QDC Demo
SmartDMA initialized
QDC0: 0, QDC1: 0
QDC0: 0, QDC1: 0
QDC0: 0, QDC1: 0
```

7 Abbreviations

Table 10 lists the acronyms used in this document.

Table 10. Acronym and abbreviations

Acronym	Description
ADC	Analog-to-digital converter
AOI	AND/OR/Invert
GPIO	General-purpose input/output
IDE	Integrated development environment
MCU	Microcontroller unit
PWM	Pulse width modulation
QDC	Quadrature decoder
SDK	Software development kit
UART	Universal asynchronous receiver/transmitter
USB	Universal serial bus

8 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.

9 Revision history

Table 11. Revision history

Document ID	Release date	Description
AN14823 v.1.0	23 September 2025	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.

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.

Contents