AN15179

Nikon A-format Protocol Implementation on MCX A366 using FlexIO

Rev. 1.0 — 23 November 2026 Application note

Document information

Information	Content
Keywords	AN15179, MCX A366, FRDM-MCXA366, Nikon A-format protocol, M50AHN00 encoder, FlexIO, RS-485, logic analyzer
Abstract	This document describes the controller-side implementation of the Nikon A-format protocol on the NXP MCX A366 microcontroller using a FlexIO peripheral.

1 Introduction

This document describes the controller-side implementation of the Nikon A-format version 2.0 protocol on the NXP MCX A366 microcontroller using a FlexIO peripheral. It explains how the demo software allocates FlexIO resources, generates CDF/MDF frames, captures and parses encoder response fields, and validates CRCs. It also covers how the software verifies encoder status, controls the RS-485 driver direction, and reports decoded position and status information through an application interface. The document also describes the demo test flow used with the Nikon M50AHN00 encoder.

The application software performs the following major steps:

1. Control a half-duplex RS-485 communication link
2. Transmit command and memory data frames
3. Receive and process encoder response frames
4. Perform communication integrity checks
5. Present decoded position and status information through an application interface

The following topics are out of the scope of this document:

- External RS-485 transceiver
- Termination and biasing network
- Cable assembly
- Encoder power supply
- Sensor configuration
- Encoder resolution
- Installation requirements
- Safety certification
- EMC compliance
- Performance characterization

These items are product-specific hardware and system-level qualification responsibilities and must be addressed before a production-ready interface is released.

2 Features

Some key features of the application software are given below:

- FlexIO-based transmit, receive, and direction-control timing for an RS-485 interface
- Support for 18-bit command and memory frames with CRC3 protection, and 18-bit response fields with CRC8 protection
- Single-device and multi-device transmission command groups with encoder address and ES validation
- APIs for readout, status monitoring, EEPROM access, temperature measurement, device identification, clear, zero-preset, and address management operations
- Support for both blocking and callback-based receive-completion mechanisms

3 A-format protocol fundamentals

This section introduces the fundamental communication principles of the Nikon A-format protocol as implemented by the MCX A366 driver and supporting protocol layer. It describes the physical link characteristics, controller-to-encoder frame structures, encoder response formats, command classifications, and CRC mechanisms.

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

© 2026 NXP B.V. All rights reserved.

3.1 Link characteristics

Nikon A-format uses asynchronous, Non-Return-to-Zero (NRZ), half-duplex serial communication over an RS-485-compatible physical link. A controller transmits command data, switches its driver/receiver state, and then receives encoder data over the same differential pair. In a multidrop (bus-capable) configuration, up to eight encoders can be connected to a single bus.

3.2 Connection modes

The Nikon A-format protocol supports the following connection modes:

- **Individual connection:** In this mode, the encoder whose assigned EAX value matches the command EA returns data.
- **Multidrop connection:** In this mode, bus encoders with EAX ≤ EA return data in address order using the prescribed offsets.

One-to-one encoders and bus encoders use different connection methods and are not interchangeable.

The MCX A366 driver transmits controller-side CDF/MDF frames and receives IF/DF response fields. Detailed information is provided in the subsequent sections.

3.3 Controller frame format

The MCX A366 driver communicates with the encoder using two controller-side frame types:

- Command Data Frame (CDF) for command transmission
- Memory Data Frame (MDF) for EEPROM access and identification-data transfers

Both frame types are 18 bits long and contain:

- Start bit
- Sync code
- Payload fields
- Frame Code (FC)
- 3-bit CRC
- Stop bit

Data bits are transmitted with LSB first.

3.3.1 Command data frame and memory data frame

A Command Data Frame (CDF) uses Frame Code (FC) = 00 and contains:

- Encoder address (EA[2:0])
- Command code (CC[4:0])

A Memory Data Frame (MDF) carries data bits (DA[7:0]) and uses the following frame codes:

- FC = 01 (1) for MDF0
- FC = 10 (2) for MDF1
- FC = 11 (3) for MDF2

For both frame types:

- Start bit = 0
- Sync code = 010 (2, fixed value)
- Stop bit = 1

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

© 2026 NXP B.V. All rights reserved.

Figure 1 shows the CDF and MDF bit layouts for the Nikon A-format protocol. Both frames are 18 bits long and consist of a start bit, sync code, frame code, data field (EA+CC or DA), 3-bit CRC, and a stop bit. The CDF is used to issue commands, while the MDF carries memory data. These layouts serve as the primary frame-format reference for command builder, CRC routines, and response parser.

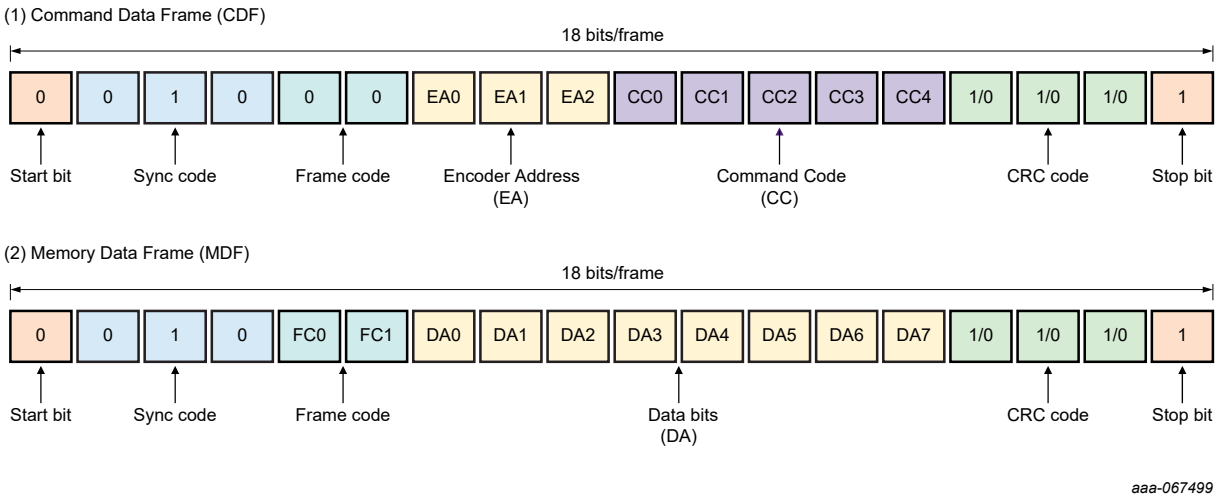


Figure 1. A-format CDF and MDF bit layouts

3.4 Command code groups

Table 1 lists the implemented command code groups and explains the expected behavior of the controller and encoder for each command group.

Table 1. A-format command groups relevant to implementation

Command category	Commands	Operation / response
Individual data	CDF0 (ABS full), CDF1 (lower), CDF2 (upper), CDF3 (status)	Only the encoder whose EA matches the specified address returns an IF frame followed by the corresponding DF response.
Multiple data	CDF4–CDF7, CDF22, CDF28, CDF30	Encoders that satisfy the condition EAX ≤ EA respond sequentially.
Clear / set up	CDF8 (status clear), CDF9 (multiturn clear), CDF10 (clear both), CDF11 (address I), CDF12 (zero preset)	Transmit the same CDF command eight consecutive times.
EEPROM	CDF13 (read), CDF14 (write)	Read operations use CDF + MDF2 and require retransmission after the EEPROM access time. Write operations use CDF + MDF0-MDF2.
Temperature	CDF15	Returns an IF frame followed by two DF fields. DB[9:0] represents the temperature value.
Identification	CDF16/CDF17 (read), CDF18/CDF19 (write), CDF20 (address II)	Identification operations use 24-bit data transmitted with LSB first and require the associated MDF frames.
Combined data	CDF27/CDF28 (ABS lower + status), CDF29/ CDF30 (ABS lower + temperature)	Operates in either individual or multiple-response mode, depending on the command code.

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

© 2026 NXP B.V. All rights reserved.

The source code provides APIs for all command groups listed in Table 1. Because several commands modify persistent encoder settings or internal state, state-changing commands must be executed only within a controlled test procedure. These commands can affect:

- Encoder addresses
- EEPROM contents
- Zero-position settings
- Latched diagnostic status information

This precaution helps prevent unintended configuration changes and ensures repeatable test results.

3.5 Encoder response format

Encoder responses contain an Information Field (IF) followed by one to three Data Fields (DFs), resulting in a response of two to four 18-bit fields.

The IF provides command and status information, whereas the DFs carry command-specific payload data.

Figure 2 shows the encoder response field configuration.

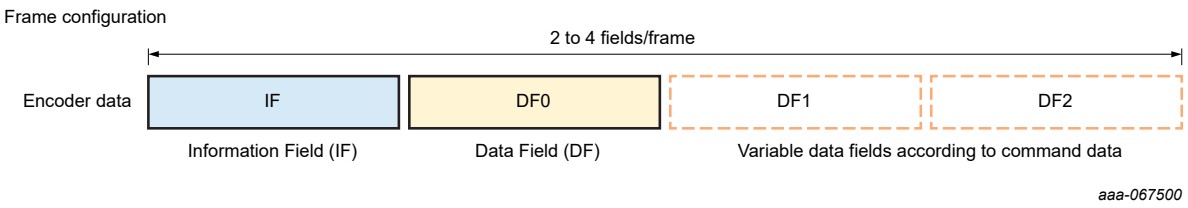


Figure 2. Encoder response field configuration

3.5.1 Information field

The Information Field (IF) contains:

- Sync information
- Encoder address (EAX[2:0])
- Echo of the received command code (CC[4:0])
- Encoder status (ES[3:0])

The IF allows the controller to verify that the response corresponds to the transmitted command and to evaluate encoder status conditions.

Figure 3 shows information field layouts for command-code groups and encoder status.

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

© 2026 NXP B.V. All rights reserved.

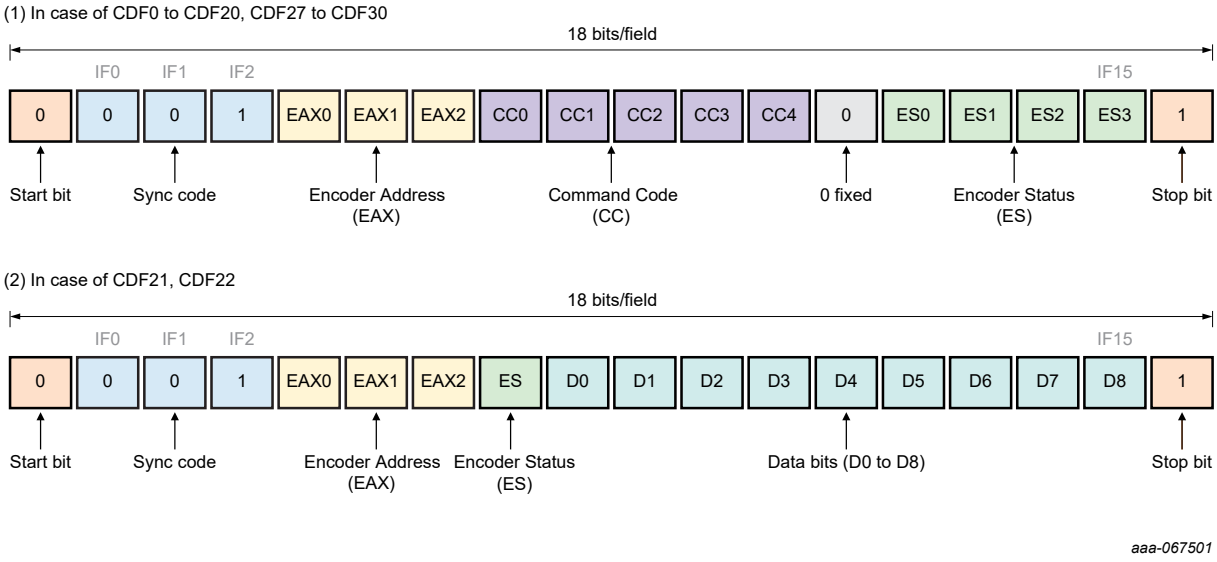


Figure 3. IF layouts

3.5.2 Data fields

Data fields (DF0–DF2) carry command-specific payload data, transmitted LSB first.

Depending on the command, one to three data fields are returned. The last data field includes an 8-bit CRC in its upper byte.

The response parser validates the protocol and CRC before publishing:

- Absolute position (ABS)
- Alarm status (ALM)
- Temperature data
- EEPROM data
- Identification data

Figure 4 shows the bit order and payload representation for a Nikon A-format data field.

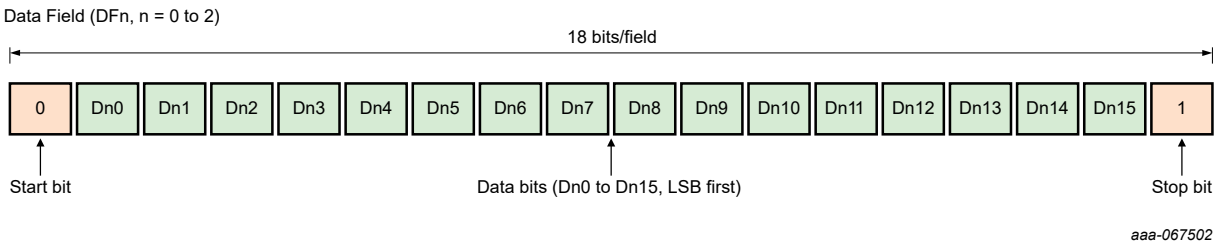


Figure 4. DF bit order and payload representation

3.6 CRC rules

The controller uses the CRC polynomial $P(x) = x^3 + x + 1$. The CRC3 value is calculated over all command bits except the Start, Stop, and Sync bits.

The encoder uses the CRC polynomial $P(x) = x^8 + x^4 + x^3 + x^2 + 1$. The CRC8 value is calculated over all response data bits, excluding the Start and Stop bits of each received field.

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

© 2026 NXP B.V. All rights reserved.

3.7 Parser responsibility

The source protocol layer validates the response address, latches the ES bit, verifies the expected command information and CRC8, and then extracts the payload. A response address mismatch is treated as a frame error. If a transfer terminates before usable data is received, the parser marks the corresponding response slot as invalid.

3.8 Protocol traceability

Table 2 summarizes how Nikon A-format protocol rules are implemented in the MCX A366 demo software. Nikon product-specific parameters, such as encoder resolution, battery topology, and interface circuitry, are not included in this traceability matrix.

Table 2. Protocol concept-to-implementation traceability

Nikon A-format rule	Implementation usage
CRC3: P(x) = x3 + x + 1, excluding the Start, Stop, and Sync bits	The command builder calculates and inserts CRC3 before transmitting CDF or MDF commands.
CRC8: P(x) = x8 + x4 + x3 + x2 + 1	The response parser verifies CRC8 in the final data field before accepting response data.
IF ES signals encoder status	The most recent ES value is latched and exposed through the `A_Format_GetLastES()` API.
Encoder response has two to four fields, depending on the command	The readout API uses the parser and res2 (DF0–DF1) or res3 (DF0–DF2) response buffers, depending on the command

4 Hardware setup

Figure 5 shows the hardware setup diagram used for the demo described in this document. The setup consists of the following components:

- FRDM-MCXA366 board, used as the controller
- RS-485 transceiver
- Nikon M50AHN00 A-format encoder

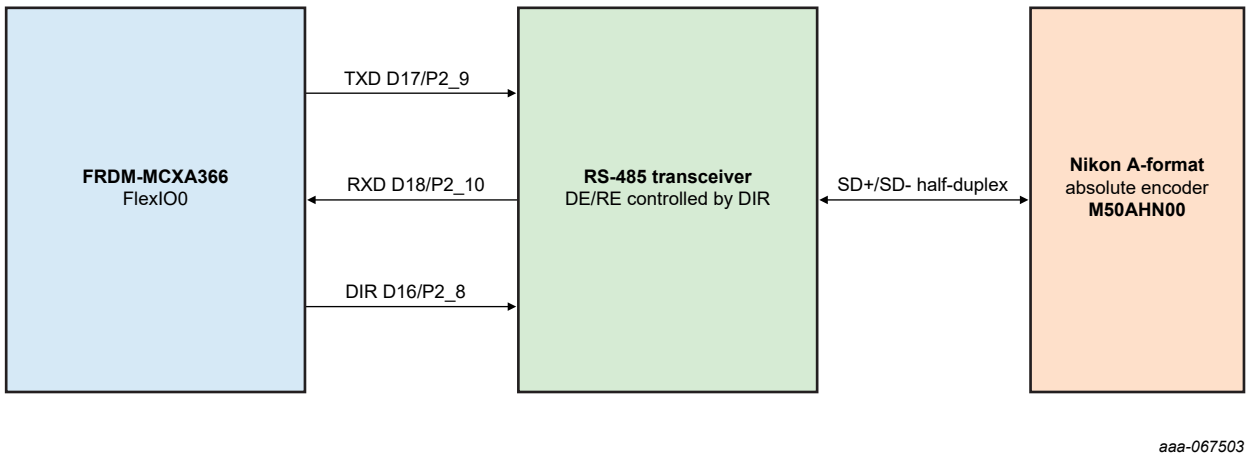


Figure 5. Demo setup diagram

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

© 2026 NXP B.V. All rights reserved.

Nikon M50AHN00 is the recommended A-format absolute encoder for running the A-format protocol examples presented in this document. When preparing the test setup, verify all power supply, cabling, and configuration requirements against the applicable encoder documentation.

Figure 6 shows the hardware setup used to capture the console output examples presented in this document. In addition to the controller, RS-485 transceiver, and Nikon A-format encoder, the setup includes a Saleae logic analyzer for signal monitoring and verification. The image is provided for reference only and may not exactly match your implementation. Use an appropriately qualified RS-485 transceiver and implement application-specific wiring, grounding, isolation, and safety measures as required.

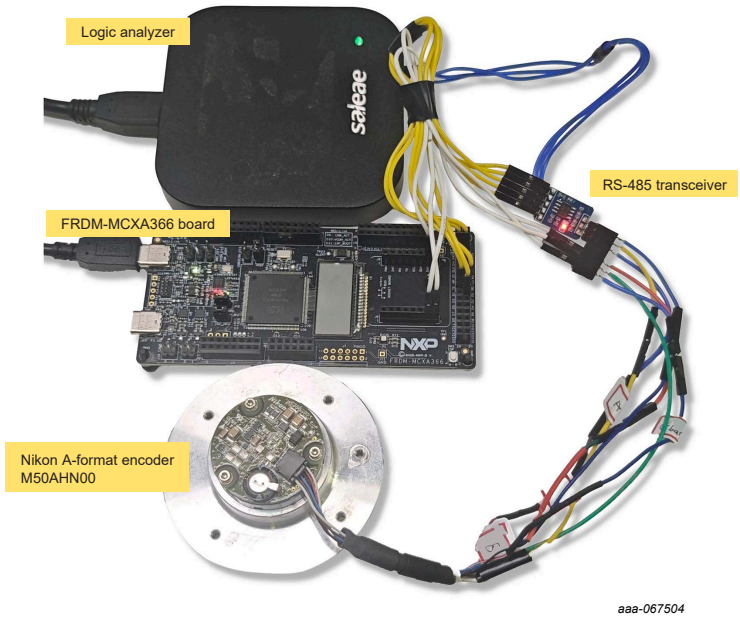


Figure 6. Demo setup

4.1 Hardware configuration

The reference project configures only the MCU-side interface pins. An external RS-485 transceiver is required to implement the differential SD+/SD− communication link. Its Driver Enable (DE) / Receiver Enable (RE) pin characteristics, fail-safe output behavior, supply-voltage compatibility, and turnaround delay must be verified against the FRDM-MCXA366 board schematics.

4.2 MCU-side pin mapping

Table 3 lists the controller-side pin assignments used by the MCX A366 reference project.

Table 3. MCX A366 controller-side pin mapping

Project signal	FlexIO signal	MCX A366 pad	Direction	Function
DIR	FLEXIO0_D16	P2_8	Output	RS-485 Driver Enable (DE) / Receiver Enable (RE) control
TXD	FLEXIO0_D17	P2_9	Output	Controller CDF/MDF data
RXD	FLEXIO0_D18	P2_10	Input	Encoder IF/DF data
GND	—	GND	Reference	Logic-side common ground

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

© 2026 NXP B.V. All rights reserved.

4.3 Termination, bias, and grounding

The termination and fail-safe bias values in the Nikon specification are reference circuit values. Select termination and bias resistor values based on the characteristics of the target cable, RS-485 transceiver, and network topology. Verify that the receiver maintains a defined idle-state output during driver turnaround and that the differential signal polarity matches the encoder pinout.

4.4 Pre-power checklist

Before applying power to the system, perform the following checks:

- Confirm the encoder supply voltage, polarity, and any required battery backup configuration.
- Verify that the transceiver I/O voltage levels are compatible with MCU pins P2_8 to P2_10.
- Verify SD+ and SD− polarity at both ends of the link and establish an appropriate common-ground strategy.
- Verify that the cable length and transceiver characteristics support the selected communication rate.
- Validate basic read-only operations, such as CDF0, CDF1, and CDF3 commands, before issuing any commands that modify nonvolatile or persistent settings.

The physical interface design is the responsibility of the system designer. This document does not prescribe or validate any specific component selection, circuit implementation, or system-level design.

5 Application execution sequence

Figure 7 shows the demo application execution sequence. The sequence includes A-format protocol command generation, FlexIO data transfer, and encoder response validation. It has six steps:

- Steps 1 and 2 are executed by the FlexIO driver.
- Steps 3 and 4 are completed through Polling mode.
- Steps 5 and 6 are handled by the A-format protocol layer before the position data is passed to the application.

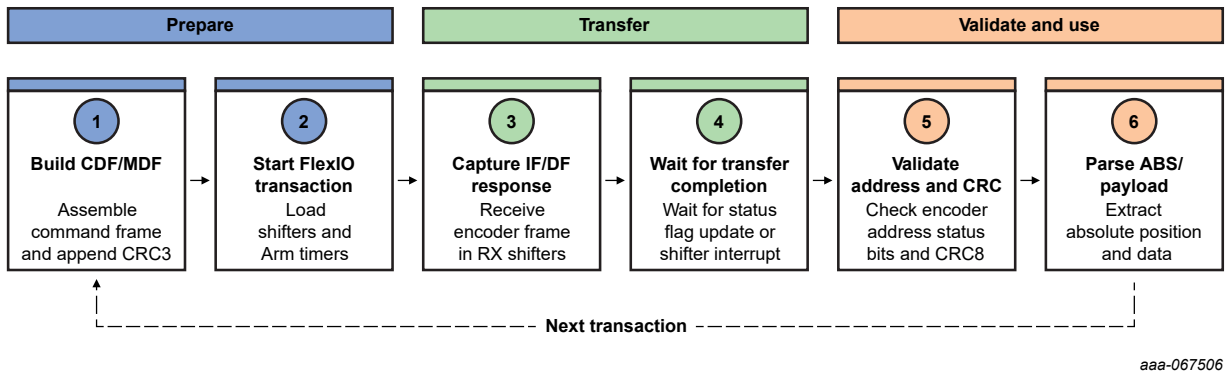


Figure 7. Demo application execution sequence

6 Software architecture

The application software architecture consists of a software stack and a FLEXIO0 peripheral that it configures. Figure 8 shows the different layers of the software stack and illustrates software execution sequence.

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

© 2026 NXP B.V. All rights reserved.

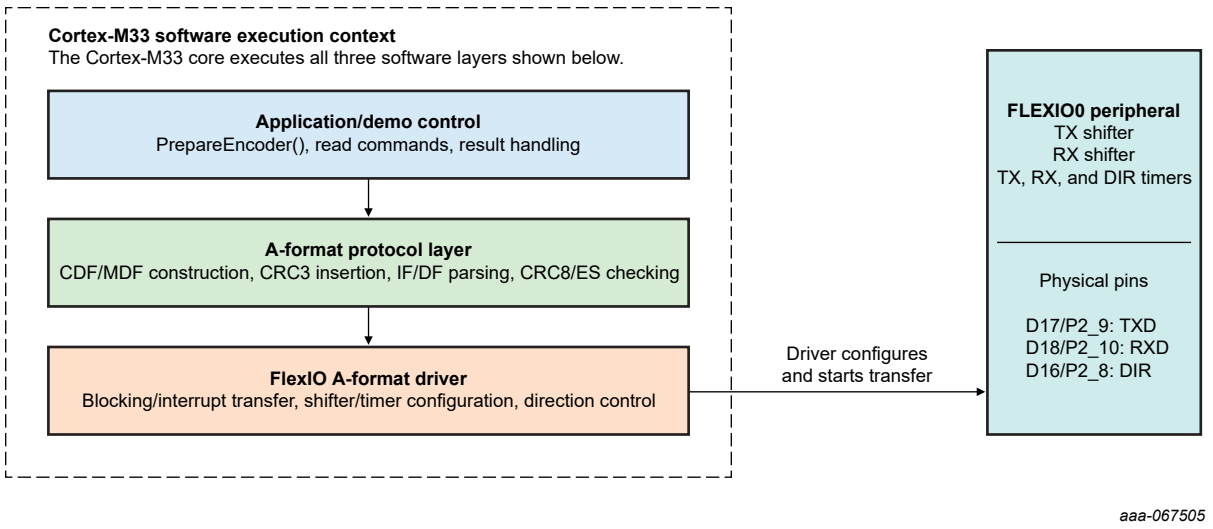


Figure 8. Software stack and execution sequence

As shown in Figure 8, the software stack has three layers:

- Application/demo control
- A-format protocol
- FlexIO A-format driver

All these layers execute in the Arm Cortex-M33 context.

6.1 Software modules

Table 4 lists the application, protocol, and driver modules in the controller software.

Table 4. Software modules

Source module	Role
`a-format_polling_transfer.c`	Demonstration flow control, board initialization, and test-case execution
`a-format.c` / `a-format.h`	CDF/MDF frame construction, CRC calculation, field validation, and data parsing
`fsl_flexio_a-format.c` / `fsl_flexio_a-format.h`	FlexIO peripheral configuration and data-transfer services
`board/app.h`	Instance selection, pin assignments, shifter and timer index definitions, and baud rate configuration
`board/pin_mux.c`	MCX A366 FLEXIO0 pin multiplexing configuration

7 FlexIO configuration

This section describes the FlexIO peripheral resource allocation, timing configuration, data-transfer paths, and register settings used to implement the Nikon A-format protocol.

7.1 Shifter allocation

Table 5 shows the resource allocation used by the FlexIO driver for transmit, receive, and transceiver direction-control timing.

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

© 2026 NXP B.V. All rights reserved.

Table 5. FlexIO resource allocation

Resource	Purpose	Configuration intent
TX shifter	Command/MDF serialization	Outputs data on the TXD line and is clocked by the TX timer.
RX shifter	IF/DF capture	Receives data from the RXD line and is clocked by the RX timer.
TX timer	Transmit bit timing	Provides dual 8-bit baud rate and bit timing.
RX timer	Receive bit timing	Provides receive timing with RX-triggered enable and reset behavior.
DR timer	Transceiver control	Controls the direction signal and limits driver enable to the command-transmission window.

7.2 Baud divider

In normal dual-8-bit mode, the driver derives the baud rate using the formula:

baud rate = sourceClock / (2 x (baudDiv + 1))

Select the source clock frequency and baud divider value such that the resulting baud rate matches the required Nikon A-format data rate within an acceptable tolerance.

7.3 Transmit and receive data paths

The Transmit (TX) path writes the command half-words to the FlexIO buffer and, in Blocking mode, waits for the transmit shifter status to indicate completion. The Receive (RX) path reads captured half-words after the receive shifter reaches the full-status condition. Non-blocking APIs maintain application buffer state and complete the transfer through the FlexIO Interrupt Request (IRQ) service path.

7.4 Direction-control window

The DR timer is configured based on the duration of the transmitted A-format frame sequence. It enables the external transceiver only while the controller owns the bus and then releases the bus to allow the encoder response to be received. When the command sequence consists of multiple CDF/MDF frames instead of a single frame, the configured drive duration must cover the entire command transmission period.

7.5 Error and timeout recovery

When retry support is enabled, a blocking-transfer timeout reinitializes the FlexIO state before returning an error. A timeout alone does not indicate whether a sensor fault or a protocol error caused the failure. To determine the root cause, inspect the TXD, DIR, and RXD signals, and the differential bus. Use the captured signals to identify issues related to clocking, bus turn-around timing, power, or frame integrity.

7.6 Register configuration reference

Table 6, Table 7, Table 8, Table 9, and Table 10 provide the 2.5 Mbit/s A-format configuration used by this implementation. The configuration corresponds to the following pin allocation at the controller side:

- D16/P2_8 = DIR
- D17/P2_9 = TXD
- D18/P2_10 = RXD

The driver calculates TIMCMP0 and TIMCMP1 at run time based on the baud rate. Therefore, these fields are intentionally shown as variable rather than fixed values in the supplied configuration. TIMCMP2 is fixed at 0x1F11 for the specified 2.5 Mbit/s receive timing configuration.

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

© 2026 NXP B.V. All rights reserved.

Table 6. Direction-control timer (TIMER0) register configuration

Register	Field	Bits	Value	Description
TIMCTL0	TRGSEL	[29:24]	7 (0x07)	Internal trigger selects TIMER1 trigger output.
	TRGPOL	[23]	0	Active-high trigger polarity
	TRGSRC	[22]	1	Internal trigger source
	PINCFG	[17:16]	3	Timer output drives the selected pin.
	PINSEL	[12:8]	16	Selects FLEXIO0_D16 (DIR).
	PINPOL	[7]	0	Active-high pin polarity
	TIMOD	[2:0]	3	Single 16-bit counter mode
TIMCFG0	TIMOUT	[25:24]	0	Timer output remains high and is unaffected by reset.
	TIMRST	[18:16]	0	Timer never resets.
	TIMDIS	[14:12]	2	Timer was disabled when a compare operation resulted in a match.
	TIMENA	[10:8]	2	Timer was enabled when the trigger turned high.
	TSTOP	[5:4]	0	Stop bit verification is disabled.
	TSTART	[1]	0	Start bit verification is disabled.

Table 7. TX bit clock timer (TIMER1) register configuration

Register	Field	Bits	Value	Description
TIMCTL1	TRGSEL	[29:24]	1 (0x01)	Internal trigger selects Shifter 0 status flag.
	TRGPOL	[23]	1	Active-low trigger polarity
	TRGSRC	[22]	1	Internal trigger source
	PINCFG	[17:16]	0	Timer output is not connected to a pin.
	PINSEL	[12:8]	17	Selects FLEXIO0_D17 (TXD).
	PINPOL	[7]	0	Active-high pin polarity
	TIMOD	[2:0]	1	Dual 8-bit Baud/Bit mode
TIMCFG1	TIMOUT	[25:24]	0	Timer output remains high and is unaffected by reset.
	TIMRST	[18:16]	0	Timer never resets.
	TIMDIS	[14:12]	2	Timer was disabled when a compare operation resulted in a match.
	TIMENA	[10:8]	2	Timer was enabled when the trigger turned high.
	TSTOP	[5:4]	2	Stop bit verification is enabled.
	TSTART	[1]	1	Start bit verification is enabled.

Table 8. RX baud / start bit timer (TIMER2) register configuration

Register	Field	Bits	Value	Description
TIMCTL2	TRGSEL	[29:24]	36 (0x24)	External trigger selects FLEXIO0_D18.

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

© 2026 NXP B.V. All rights reserved.

Table 8. RX baud / start bit timer (TIMER2) register configuration*...continued*

Register	Field	Bits	Value	Description
	TRGPOL	[23]	0	Active-high trigger polarity
	TRGSRC	[22]	0	External trigger source
	PINCFG	[17:16]	0	Timer output is not connected to a pin.
	PINSEL	[12:8]	18	Selects FLEXIO0_D18 (RXD).
	PINPOL	[7]	1	Active-low polarity for start bit detection
	TIMOD	[2:0]	1	Dual 8-bit Baud/Bit mode
TIMCFG2	TIMOUT	[25:24]	2	Timer output was set high on timer reset.
	TIMRST	[18:16]	4	Timer was reset on pin rising edge.
	TIMDIS	[14:12]	2	Timer was disabled when a compare operation resulted in a match.
	TIMENA	[10:8]	4	Timer was enabled on pin rising edge.
	TSTOP	[5:4]	2	Stop bit verification is enabled.
	TSTART	[1]	1	Start bit verification is enabled.
TIMCMP2	Compare	[15:0]	0x1F11	Configures 2.5 Mbit/s receive timing in dual 8-bit mode.

Table 9. TX data shifter (SHIFTER0) register configuration

Register	Field	Bits	Value	Description
SHIFTCTL0	TIMSEL	[25:24]	1	Selects TIMER1 as the shift clock source.
	TIMPOL	[23]	0	Shift on the positive clock edge
	PINCFG	[17:16]	3	Shifter drives the selected output pin.
	PINSEL	[12:8]	17	Selects FLEXIO0_D17 (TXD).
	PINPOL	[7]	0	Active-high pin polarity
	SMOD	[2:0]	2	Transmit mode
SHIFTCFG0	PWIDTH	[20:16]	0	Single-bit width
	SSIZE	[12]	0	32-bit shift register
	LATST	[9]	0	Load data on pre-shift store.
	INSRC	[8]	0	Input source is the pin.
	SSTOP	[5:4]	3	Inserts a logic '1' stop bit.
	SSTART	[1:0]	2	Inserts a logic '0' start bit.

Table 10. RX data shifter (SHIFTER1) register configuration

Register	Field	Bits	Value	Description
SHIFTCTL1	TIMSEL	[25:24]	2	Selects TIMER2 as the shift clock source.
	TIMPOL	[23]	1	Shift on the negative clock edge
	PINCFG	[17:16]	0	Input-only operation; pin is not driven.

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

© 2026 NXP B.V. All rights reserved.

Table 10. RX data shifter (SHIFTER1) register configuration...*continued*

Register	Field	Bits	Value	Description
	PINSEL	[12:8]	18	Selects FLEXIO0_D18 (RXD).
	PINPOL	[7]	0	Active-high pin polarity
	SMOD	[2:0]	1	Receive mode
SHIFTCFG1	PWIDTH	[20:16]	0	Single-bit width
	SSIZE	[12]	0	32-bit shift register
	LATST	[9]	0	Load data on pre-shift store.
	INSRC	[8]	0	Input source is the pin.
	SSTOP	[5:4]	3	Verifies a logic '1' stop bit.
	SSTART	[1:0]	2	Verifies a logic '0' start bit.

8 Application API reference

This section summarizes the public APIs provided by the application software for encoder readout, configuration, maintenance operations, and transfer management.

8.1 Readout and diagnostic APIs

Table 11 lists APIs that provide read-only access to position, status, and diagnostic information used by the application software.

Table 11. Readout and diagnostic APIs

API	Function
`A_Format_ABS_Readout_Multi_Single()`	Reads both multiturn and single-turn portions of the absolute position value.
`A_Format_ABS_Readout_Single()` / `A_Format_ABS_Readout_Multi()`	Reads a single portion of the absolute position value.
`A_Format_Readout_Encoder_status()`	Reads encoder status and alarm information.
`A_Format_GetLastES()` / `A_Format_PrintfES()`	Retrieves and displays the most recently validated ES field.
`A_Format_Get_Temperature()`	Reads and scales the encoder temperature value.

8.2 Maintenance and identity APIs

Table 12 lists APIs that modify or retrieve persistent encoder identity and maintenance information. Use these APIs only on a controlled unit because some operations can change encoder configuration or nonvolatile data.

Table 12. Maintenance and identity APIs

API	Function
`A_Format_Clear_Stat_Multi()`	Clears status information and performs a multiturn counter clear operation.
`A_Format_Memory_Read()` / `A_Format_Memory_Write()`	Performs EEPROM read and write transfer sequences.
`A_Format_Get_ID()` / `A_Format_Set_ID()`	Reads or writes the 24-bit identification code.

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

© 2026 NXP B.V. All rights reserved.

Table 12. Maintenance and identity APIs*...continued*

API	Function
`A_Format_Set_Encoder_Address_*()`	Sets the encoder address using either a one-to-one assignment method or a matching ID method.
`A_Format_*_CMD()` / `A_Format_*_Parse()`	Separates command generation from response parsing and validation.

The `*_CMD()` and `*_Parse()` APIs share file-scope protocol state. They are not reentrant and must be paired with the corresponding parser before another A-format operation is started.

8.3 Blocking and non-blocking transfer APIs

The `FLEXIO_A_Format_WriteBlocking()` and `FLEXIO_A_Format_ReadBlocking()` APIs are suitable for simple demonstration applications that execute transfers sequentially. The `FLEXIO_A_Format_TransferSendNonBlocking()` and `FLEXIO_A_Format_TransferReceiveNonBlocking()` APIs support interrupt completion and optional ring buffering. Transfer completion status only indicates that buffer processing has finished; transaction-level validity must be verified through response validation checks.

8.4 Validation scope

The recorded validation covers a single MCX A366 controller, one external RS-485 transceiver path, and one Nikon M50AHN00 encoder operating at the selected 2.5 Mbit/s configuration. The following functions are validated:

- One-to-one address assignment
- ID write/read
- Addressed ID access
- EEPROM write/read
- Status and alarm readout
- Temperature readout
- Single-turn position readout
- Multiturn position readout
- Combined readout commands

Do not consider this validation as production qualification of the RS-485 transceiver, cable length, or termination and bias network. Also, do not consider it as a qualification of EMC performance, operating temperature range, timing margins, multidrop bus loading, or multi-encoder multiple-transmission operation. These aspects must be evaluated and verified as part of the final product design.

8.5 Test cases and expected protocol frames

Table 13 lists some test cases that provide a repeatable protocol-level validation matrix for the A-format driver. The "Host transmitted frame sequence (TX)" column lists the controller frame words transmitted during each test. The "Expected response from encoder (RX)" column identifies the field structure of the encoder response. The encoder response is verified after bus ownership returns to the controller from the encoder.

Note: *CDF(n) indicates a command data frame with command code 'n' and frame code 0 (fixed).*

Note: *MDF0, MDF1, and MDF2 indicate memory data frames with frame codes 1, 2, and 3, respectively.*

Note: *If error-state clearing is required, the `PrepareEncoder()` API transmits CDF(10) = 0x2A62 eight times before executing test cases 9 to 15.*

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

© 2026 NXP B.V. All rights reserved.

Table 13. A-format protocol test cases

Test case	Operation	Host transmitted frame sequence (TX)	Expected response from encoder (RX)
1	One-to-one: Set the address to 3; repeat eight times.	CDF(11)=0xAB62 -> CDF(11)=0xAB62 -> CDF(11)=0xAB62 -> CDF(11)=0xAB62 -> CDF(11)=0xAB62 -> DF(11)=0x AB62 -> CDF(11)=0xAB62 -> CDF(11)=0xAB62	For each transmission: IF + DF[0] + DF[1]
2	One-to-one: Write ID 0x03 0201.	CDF(19)=0xF302 -> MDF0=0x202A -> MDF1=0x4052 -> MDF2=0x607A	IF + DF[0] + DF[1], including the ID field
3	One-to-one: Read ID.	CDF(17)=0x5102	IF + DF[0] + DF[1], including the 24-bit ID
4	Set the address to 3 by matching ID 0x030201.	CDF(20)=0xD462 -> MDF0=0x202A -> MDF1=0x4052 -> MDF2=0x607A	IF + DF[0] + DF[1]
5	Write ID 0x030201 to address 3.	CDF(18)=0x9262 -> MDF0=0x202A -> MDF1=0x4052 -> MDF2=0x607A	IF + DF[0] + DF[1]
6	Read ID from address 3.	CDF(16)=0x3062	IF + DF[0] + DF[1], including the ID field
7	Write 0x5A2B to EEPROM address 0x01.	CDF(14)=0xCE62 -> MDF0=0x656A -> MDF1=0x0B52 -> MDF2=0x403A	IF + DF[0] + DF[1]
8	Read EEPROM address 0x01.	CDF(13)=0xED62 -> MDF2=0x403A -> CDF(13)=0xED62 -> MDF2=0x403A	IF + DF[0] + DF[1]
9	Read encoder status and alarm information.	CDF(3)=0xC362	IF + DF[0] + DF[1]
10	Read encoder temperature.	CDF(15)=0x4F62	IF + DF[0] + DF[1]
11	Read multiturn position.	CDF(2)=0x4262	IF + DF[0] + DF[1]
12	Read single-turn position.	CDF(1)=0x6162	IF + DF[0] + DF[1]
13	Read single-turn position and status.	CDF(27)=0x7B62	IF + DF[0] + DF[1] + DF[2]
14	Read single-turn position and temperature.	CDF(29)=0x3D62	IF + DF[0] + DF[1] + DF[2]
15	Read multiturn and single-turn positions together.	CDF(0)=0xE062	IF + DF[0] + DF[1] + DF[2]
16	Multiple transmissions: Read encoder data once for addresses 0 to 3.	CDF(4)=0x0462	Four response groups, each containing IF + DF[0] + DF[1] + DF[2]

8.6 Result handling guidance

Table 14 maps common return conditions to the corresponding application responses.

Table 14. Result handling guidance

Return condition	Interpretation	Next action
Success with `ES=0`	The response frame is valid and the encoder reports normal operation.	Consume and process the payload data.
`EncoderStatErr` or nonzero `ES`	The response frame is valid, but the encoder reports a status or an alarm condition.	Decode the encoder status and alarm fields, and then evaluate the reported encoder condition.

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

© 2026 NXP B.V. All rights reserved.

Table 14. Result handling guidance*...continued*

Return condition	Interpretation	Next action
`FrameErr`	No valid response frame was received, or the received frame does not match the expected format.	Verify address settings, signal polarity, timing configuration, and wiring connections.
Timeout	The transfer did not complete within the expected period.	Verify clock configuration, trigger settings, transceiver direction (DIR) control, and the receive path.
CRC or echo mismatch	Data was received, but protocol validation failed.	Check communication rate, signal noise, frame length, and overall signal integrity.

9 Hands-on example

This section provides a practical workflow for building, programming, validating, and monitoring the Nikon A-format demo application.

9.1 Development environment

Download the `an-mcxa366-nikon-aformat-interface-using-flexio` project from the NXP Application Code Hub (ACH). The project uses MCUXpresso SDK drivers and includes board clock initialization, pin initialization, and a diagnostic serial console for application output and debugging.

9.2 Build project and program AXF file

To build the `an-mcxa366-nikon-aformat-interface-using-flexio` project and program the generated Arm eXecutable Format (AXF) file, follow these steps:

1. Import or open the project in MCUXpresso IDE.
2. Build the project and verify that the build completes without errors.
3. Connect the FRDM-MCXA366 board having an onboard MCU-Link debug probe.
4. Connect the encoder through a validated RS-485 interface.
5. Open the serial console using the communication settings defined by the project.
6. Program the generated AXF file to the target device.
7. Reset the target and verify that the application starts correctly and produces the expected console output.

9.3 Capture MCU data using a logic analyzer

To capture the MCX A366 data using a logic analyzer, follow these steps:

1. Capture the MCU DIR, TXD, and RXD signals using a logic analyzer.
2. Annotate the selected communication rate, CDF/MDF frame count, response field count, and Encoder Address (EA).
3. Compare the measured drive-release time, response-start time, and bus-recovery time against the Nikon A-format timing requirements for the selected connection mode.

Figure 9 shows timing data capture using Saleae logic analyzer for the status / multiturn data clear test.

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

© 2026 NXP B.V. All rights reserved.

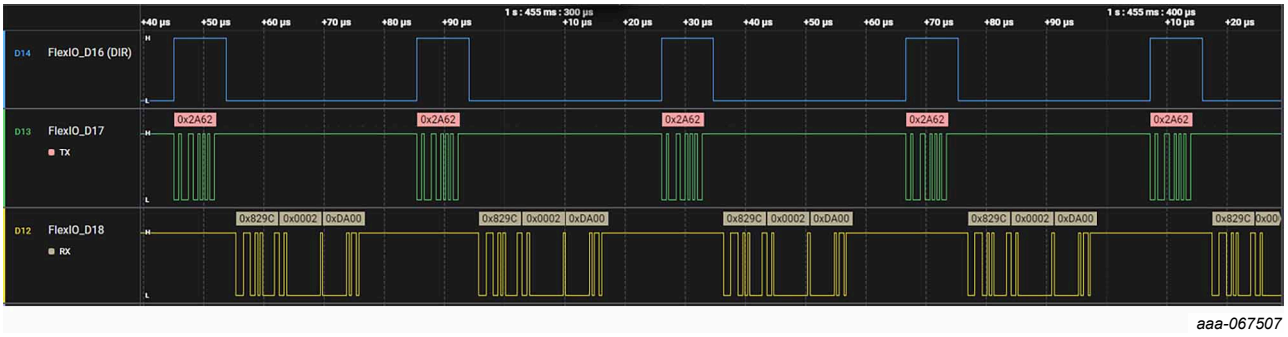


Figure 9. Timing data capture using Saleae logic analyzer for status / multiturn data clear test

9.4 Logs

The console log of running the demo application is shown below:

```
Oct  9 2026 18:05:43
Encoder A-format example
FlexIO Root Clock is 90 MHz

******************** 1 to 1 mode (Only one encoder connects to the board)
 ********************
****************
* Test case  1 *
****************
> Set the encoder address to 3 ==> [encoder ES 0x01] write request acknowledged
 (completion not verified)
****************
* Test case  2 *
****************
> Set the encoder ID to 0x030201 ==> [encoder ES 0x01] write request
 acknowledged (completion not verified)
****************

* Test case  3 *
****************
> Get the encoder ID ==> 0x030201 (successful)

******************** Individual Transmission mode (The board accesses only one
 encoder on the bus) ********************
****************
* Test case  4 *
****************
> The encoder address 3 is set by matching the ID 0x030201 ==> [encoder ES 0x01]
 write request acknowledged (completion not verified)
****************
* Test case  5 *
****************
> Set the encoder ID to 0x030201 ==> [encoder ES 0x01] write request
 acknowledged (completion not verified)
****************
* Test case  6 *
****************
> Get the encoder ID ==> 0x030201 (successful)
****************
* Test case  7 *
```


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

© 2026 NXP B.V. All rights reserved.

```
****************
> Write the data 0x5A2B to the address 0x01 of EEPROM ==> [encoder ES 0x01]
 write request acknowledged (completion not verified)
****************
* Test case  8 *
****************
> Read the data from the address 0x01 of EEPROM ==> 0x5A2B (readback verified)

******************** Preparing the encoder ********************
> Read the status of the encoder 0x03 ==> ES: 0x00, ALM: 0x0000
 ALM: no alarm bit is set
> No CDF10-clearable ALM fault is set, the multi-turn counter is kept as is
****************
* Test case  9 *
****************
> Read the status of the encoder 0x03 ==> ES: 0x00, ALM: 0x0000 (status response
 received)
 ALM: no alarm bit is set
****************
* Test case 10 *
****************
> Get the temperature of the encoder ==> 45.000000 (successful)
****************
* Test case 11 *
****************
> Get the multi-turn data of the encoder ==> ES: 0x00, Multi-turn: 1
 (successful)
****************
* Test case 12 *
****************
> Get the single-turn data of the encoder ==> ES: 0x00, Single-turn: 153479
 (successful)
****************
* Test case 13 *
****************
> Get the single-turn data and status of the encoder ==> ES: 0x00, Single-turn:
 153479, status: 0x0000 (successful)
****************
* Test case 14 *
****************
> Get the single-turn data and temperature of the encoder ==> ES: 0x00, Single-
turn: 153479, temperature: 45.000000 (successful)
****************
* Test case 15 *

****************
> Get the single-turn data and the multi-turn data of the encoder ==> successful
 Multi-turn data: 1, single-turn data: 153479


******************** Multiple Transmission mode (The board can access multiple
 encoders on the bus simultaneously) ********************
****************
* Test case 16 *
****************
> Skipped: needs 4 encoders on the bus (see DEMO_ENABLE_MT_TEST)

******************** Running the loop test ********************
```


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

© 2026 NXP B.V. All rights reserved.

```
[29.158s] Multi-turn data: 1, single-turn data: 152954
 Temperature: 45.000000

[29.292s] Multi-turn data: 1, single-turn data: 148446
 Temperature: 45.000000

[29.425s] Multi-turn data: 1, single-turn data: 147454
```

Before executing test cases 9 to 15, the application issued the read status command (0x03) and reported `ES = 0x00` and `ALM = 0x0000`. Therefore, the console output indicated that no error condition was present and that the multiturn counter remained unchanged.

The execution results confirm successful completion of the one-to-one address and ID operations and addressed ID and EEPROM operations. The results also confirm successful status and alarm readout, temperature readout, and single-turn and multiturn data parsing.

Test case 16 was intentionally omitted because the test setup did not include the four encoders required to demonstrate multiple-transmission operation.

10 Conclusion

The MCX A366 reference project provides a baseline implementation of the Nikon A-format protocol using FlexIO timing resources and an Arm Cortex-M33-based protocol layer. The implementation generates controller command frames, manages the external RS-485 transceiver direction-control window, and captures encoder response fields. It also validates address, ES, and CRC information, and exposes parsed encoder data through an application interface.

Before adopting this design in a production system, the system designer must address the remaining hardware and qualification activities, including:

- RS-485 transceiver selection
- Termination and fail-safe bias design
- Cable and connector selection
- Grounding or isolation strategy
- EMC validation
- Timing-margin verification across operating voltage and temperature ranges
- Multi-encoder bus validation when multiple-transmission operation is required

11 Acronyms

Table 15 lists the acronyms used in this document.

Table 15. Acronyms

Acronym	Description
ABS	Absolute position
ACH	Application Code Hub
ALM	Alarm status
API	Application Programming Interface
AXF	Arm eXecutable Format
BSP	Board Support Package
CC	Command Code

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

© 2026 NXP B.V. All rights reserved.

Table 15. Acronyms...*continued*

Acronym	Description
CDF	Command Data Frame
CMD	Command
CRC	Cyclic Redundancy Check
DA	Data bit
DE	Driver Enable
DF	Data Field
DIR	Direction
EA / EAX	Encoder Address
EEPROM	Electrically Erasable Programmable Read-Only Memory
EMC	Electromagnetic Compatibility
ES	Encoder Status
FC	Frame Code
FlexIO	Flexible Input/Output
GND	Ground
I/O	Input/Output
IDE	Integrated Development Environment
IF	Information Field
IRQ	Interrupt Request
LSB	Least Significant Bit
MCU	Microcontroller Unit
MDF	Memory Data Frame
MT	Multiturn
NRZ	Non-Return-to-Zero
RE	Receiver Enable
RX	Receive
RXD	Receive Data
SDK	Software Development Kit
TX	Transmit
TXD	Transmit Data

12 References

For more information on Nikon A-format protocol implementation, refer to the following resources:

- NXP Application Code Hub (ACH) project `an-mcxa366-nikon-aformat-interface-using-flexio`
- *Nikon A-format Protocol Specification*: Use specification revision that applies to the target encoder.
- MCX A345, A346, A355, A356, A365 and A366 Reference Manual (MCXAP144M240F60RM): Specifically refer to the "Flexible I/O (FlexIO)" chapter and pin muxing information.
- MCUXpresso SDK driver documentation: Use for FlexIO and Board Support Package (BSP) configuration.

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

© 2026 NXP B.V. All rights reserved.

- NXP i.MX RT1186 A-format SDK example: It is used as a protocol-layer reference implementation for the current application software.

13 Revision history

Table 16 summarizes the revisions to this document.

Table 16. Revision history

Document ID	Release date	Description
AN15179 v.1.0	23 November 2026	Initial public release

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

© 2026 NXP B.V. All rights reserved.

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.

A-format — is a registered trademark of Nikon Corporation.

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

© 2026 NXP B.V. All rights reserved.

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.

Contents

1 Introduction 2
2 Features 2
3 A-format protocol fundamentals 2
3.1 Link characteristics 3
3.2 Connection modes 3
3.3 Controller frame format 3
3.3.1 Command data frame and memory data frame 3
3.4 Command code groups 4
3.5 Encoder response format 5
3.5.1 Information field 5
3.5.2 Data fields 6
3.6 CRC rules 6
3.7 Parser responsibility 7
3.8 Protocol traceability 7
4 Hardware setup 7
4.1 Hardware configuration 8
4.2 MCU-side pin mapping 8
4.3 Termination, bias, and grounding 9
4.4 Pre-power checklist 9
5 Application execution sequence 9
6 Software architecture 9
6.1 Software modules 10
7 FlexIO configuration 10
7.1 Shifter allocation 10
7.2 Baud divider 11
7.3 Transmit and receive data paths 11
7.4 Direction-control window 11
7.5 Error and timeout recovery 11
7.6 Register configuration reference 11
8 Application API reference 14
8.1 Readout and diagnostic APIs 14
8.2 Maintenance and identity APIs 14
8.3 Blocking and non-blocking transfer APIs 15
8.4 Validation scope 15
8.5 Test cases and expected protocol frames 15
8.6 Result handling guidance 16
9 Hands-on example 17
9.1 Development environment 17
9.2 Build project and program AXF file 17
9.3 Capture MCU data using a logic analyzer 17
9.4 Logs 18
10 Conclusion 20
11 Acronyms 20
12 References 21
13 Revision history 22
Legal information 23

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

© 2026 NXP B.V. **All rights reserved.**