AN15161

BiSS-C Interface Implemented by FlexIO on MCX A366

Rev. 1.0 — 18 September 2026 **Application note**

Document information

Information	Content
Keywords	AN15161, MCX A366, FlexIO, BiSS-C, Encoder, Motor control
Abstract	This application note describes a FlexIO-based BiSS-C controller implemented on the FRDM-MCXA366.

1 Introduction

This application note describes a FlexIO-based BiSS-C controller implemented on the FRDM-MCXA366. The project generates the MA clock, waits for the encoder STR transition, captures the SL response, decodes multi-turn and single-turn position, and validates the received CRC6.

The current MCX A366 source tree implements the documented profile. FLEXIO_BISSC_Init() derives the receive window from MT, ST, and the configured trailing sample: this profile has 43 valid frame bits and a 44-bit capture window.

The current implementation keeps the protocol profile centralized in board/app.h while exposing enough runtime state to diagnose physical timing, parser alignment, and sustained-transfer behavior.

1.1 Key features

- FlexIO0 generates MA on D18 and samples SL on D17 through a two-shifter receive chain.
- The ADK-4408-MA16/17B1BLP-BH40 profile uses MT=16, ST=17, ERR, WARN, and CRC6.
- The serial bit rate is configured for 10 MHz from the 180 MHz FlexIO source clock.
- MA starts from a software trigger and continues while the controller waits for the encoder STR rising edge. The receive window is derived from the active profile rather than from a fixed ACK-clock count.
- The STR edge starts a fixed 44-bit receive window. One trailing sample is removed before frame parsing.
- Interrupt completion reads the capture chain and reports MT, ST, signed combined position, CDS, START, ERR, WARN, and Cyclic Redundancy Check (CRC) status. The driver preserves capture ownership and maintains cumulative diagnostics/statistics.

1.2 Scope

This note documents the current MCX A366 project and its ADK-4408 encoder profile. It is not a substitute for the encoder or electrical-interface data sheets. Supply voltage, line-driver compatibility, termination, cable length, and signal margins must be checked in the final system.

2 Fundamentals

This section summarizes the signal roles, transaction boundary, response layout, and CRC6 rule used by the configured BiSS-C profile.

2.1 BiSS-C physical interface

The MCU side uses two single-ended signals: MA is driven from MCX A366 to the encoder interface, and SL is returned from the encoder interface to MCX A366. An optional FLEXIO0_D30 route can expose the TIMER2 gate for bench observation without changing the normal two-wire interface.

Table 1. BiSS-C signal directions

Signal	Function	Direction
MA	BiSS-C clock	MCXA366 to encoder interface
SL	Encoder response	Encoder interface to MCXA366
GND	Logic reference	Common ground

2.2 Position transaction

A write to the trigger shifter starts the MA timer. MA keeps clocking while the controller waits for the encoder STR rising edge. STR starts the receive timer, the fixed RX window captures the response, and the RX timer disable event stops MA through the gate timer.

2.3 Frame and CRC6

The valid response consists of START, CDS, MT, ST, ERR, WARN, and CRC6: 43 bits for this profile. The driver derives a 44-sample capture from this frame length plus one trailing sample, which FLEXIO_BISSC_ReadFrame() removes before parsing.

Table 2. BiSS-C response frame fields

Field	Width	Parser treatment
START	1 bit	STR rising edge starts the receive window
CDS	1 bit	Captured in the frame; not decoded by this project
MT	16 bits	Extracted from normalized data after ST
ST	17 bits	Extracted first after status bits
ERR/WARN/CRC6	1 + 1 + 6 bits	ERR/WARN are raw status; CRC6 is compared with calculated CRC6

CRC6 parameters

CRC6: polynomial 0x43, MSB-first feed, six-bit mask 0x3F, final XOR 0x3F. CRC covers MT + ST + ERR + WARN.

2.4 Role of FlexIO

FlexIO supplies the software trigger shifter, two chained receive shifters, the STR-aligned receive timer, the MA clock timer, and the MA gate timer. The Cortex-M33 starts a transaction, waits for completion, parses the captured words, and publishes diagnostics.

2.5 Profile-specific notes

Do not treat ACK timing or timer counts from another BiSS-C encoder as universal constants. This implementation does not compile MA as a fixed ACK-dependent burst.

- The active project starts the RX window on STR and samples on the positive RX timer edge.
- A 44-bit capture window and the one-bit trailing-sample removal form one alignment rule. Changing only one of them can cause a CRC mismatch.
- CRC6 uses polynomial 0x43, MSB-first feed, and final XOR 0x3F.
- ERR and WARN are raw encoder bits. Their application meaning must come from the encoder data sheet.
- The current project includes timeout recovery so that MA is stopped and FlexIO is reinitialized when STR or RX completion is missing.

3 Hardware pinout and wiring

Figure 1 shows the signal path and the conceptual test setup. Connect by signal name and verify the electrical interface schematic before applying encoder power.

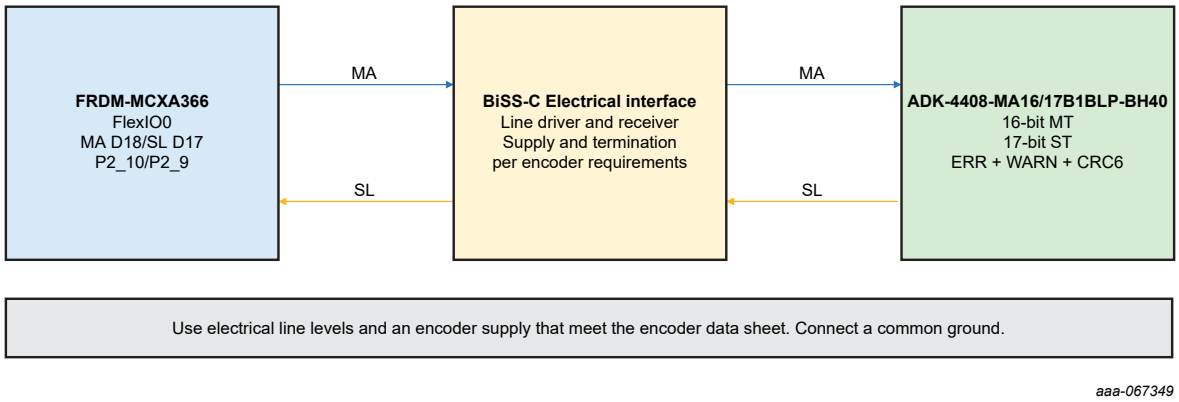


Figure 1. FRDM-MCXA366, BiSS-C electrical interface, and ADK4408 signal mapping

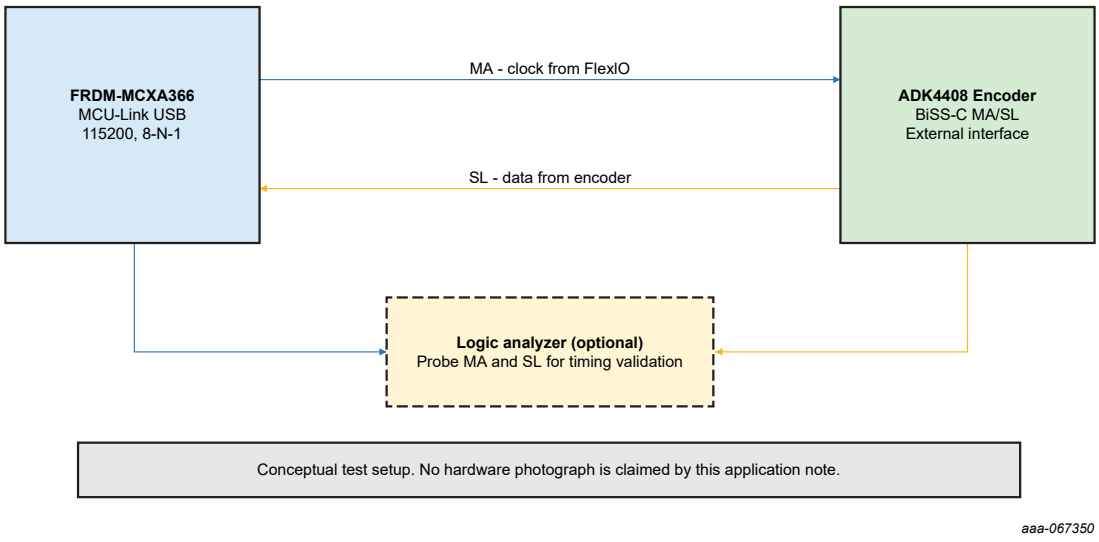


Figure 2. Conceptual hardware and logic-analyzer test setup

3.1 MCU-side signal mapping

Table 3 describes the pin map that identifies the MCX A366 pads used by the project and the required signal direction for each connection.

Table 3. MCX A366 BiSS-C pin mapping

Project signal	FlexIO signal	MCX A366 pad	Direction	Purpose
MA	FLEXIO0_D18	P2_10	Output	BiSS-C clock
SL	FLEXIO0_D17	P2_9	Input	Encoder response/STR
GND	-	GND	Reference	Common logic ground
MA gate (optional)	FLEXIO0_D30	P2_22	Output when enabled	TIMER2 gate probe; set DEMO_BISSC_GATE_PIN_OUTPUT=1

3.2 Electrical interface and SL state

MA is an output on FLEXIO0_D18 and SL is an input on FLEXIO0_D17. The current board configuration disables the MCX A366 internal pullup and pulldown on SL; P2_22 can additionally expose the TIMER2 gate on D30 when DEMO_BISSC_GATE_PIN_OUTPUT is enabled.

- The electrical interface must hold and release SL according to the selected encoder timing. Do not leave SL floating during bring-up.
- The controller treats the encoder STR rising edge as the receive synchronization point.
- A missing STR prevents the RX timer from starting and produces the documented frame-timeout recovery path.
- Probe the configured D17 SL pin, not the adjacent D19 pin, when troubleshooting the receive path.

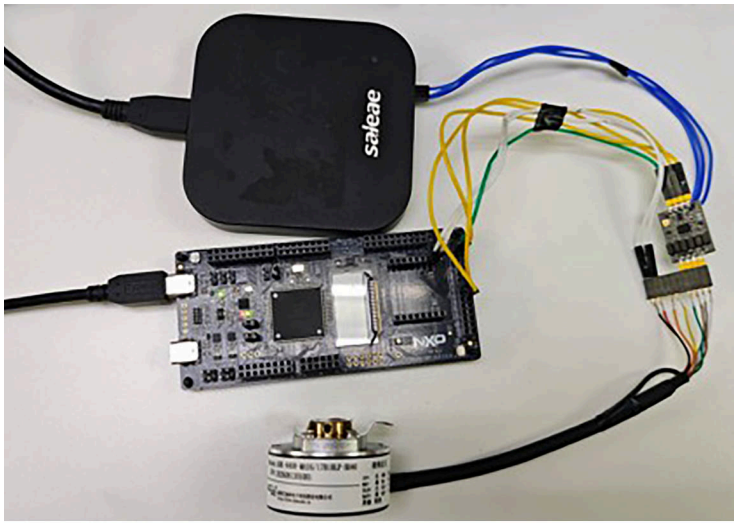
Pre-power electrical-interface check

Pre-Power Check: Confirm MA is connected to D18, SL is connected to D17, ground is common, and the electrical interface meets the encoder voltage and termination requirements.

The normal interface uses only MA, SL, and the common reference. The optional D30 gate-probe signal is for measurement only and is not required for protocol operation.

3.3 Power, ground, and wiring

Figure 3 shows the physical connection used for the MCX A366, BiSS-C electrical interface, encoder, and logic analyzer.



aaa-067351

Figure 3. Hardware connection of FRDM-MCXA366, BiSS-C electrical interface, ADK4408 encoder, and Saleae logic analyzer

- Connect MA from P2_10/FLEXIO0_D18 to the MA input of the encoder electrical interface.
- Connect the SL output of the electrical interface to P2_9/FLEXIO0_D17.
- Connect FRDM-MCXA366 ground to the electrical-interface logic ground.
- Use an encoder-rated supply and verify logic-side voltage compatibility before connecting the board.
- Apply termination, bias, and cable practices required by the encoder and electrical interface.

3.4 Encoder profile and source traceability

Table 4 records the selected encoder profile and the project symbols that must be reviewed when the hardware configuration changes.

Table 4. Encoder profile and source traceability

Item	Current record	Source/required check
Encoder	ADK-4088-MA16/17B1BLP-BH40	board/app.h; verify encoder nameplate
Bit rate	10,000,000 bit/s	DEMO_ENCODER_BIT_RATE
FlexIO clock	180,000,000 Hz	Runtime CLOCK_GetFlexioClkFreq() output
Multi-turn field	16 bits	DEMO_ENCODER_MT_LEN
Single-turn field	17 bits	DEMO_ENCODER_ST_LEN
RX window	44 bits: 43 valid bits + 1 trailing sample	FLEXIO_BISSC_FRAME_BITS() + DEMO_BISSC_RX_TRAILING_BITS; derived by FLEXIO_BISSC_Init()
CRC	CRC6, 0x43/0x3F	FLEXIO_BISSC_CRC6()
Wiring	External electrical interface	Verify data sheets before power
RX shifters	2: ceil(44/32)	base->rxShifterNum; resource checked against FLEXIO_SHIFTBUF_COUNT
MA gate probe	FLEXIO0_D30/P2_22; disabled by default	DEMO_BISSC_GATE_PIN_OUTPUT and DEMO_BISSC_GATE_PIN
Frame wait	100 µs; 5 µs polling step	DEMO_BISSC_WAIT_TIMEOUT_US and FLEXIO_BISSC_POLL_STEP_US

4 Software architecture

Figure 4 and Figure 5 summarize the software-triggered implementation in source/flexio_bissc_adk4408_interrupt_transfer.c, source/flexio_bissc.c, and board/app.h.

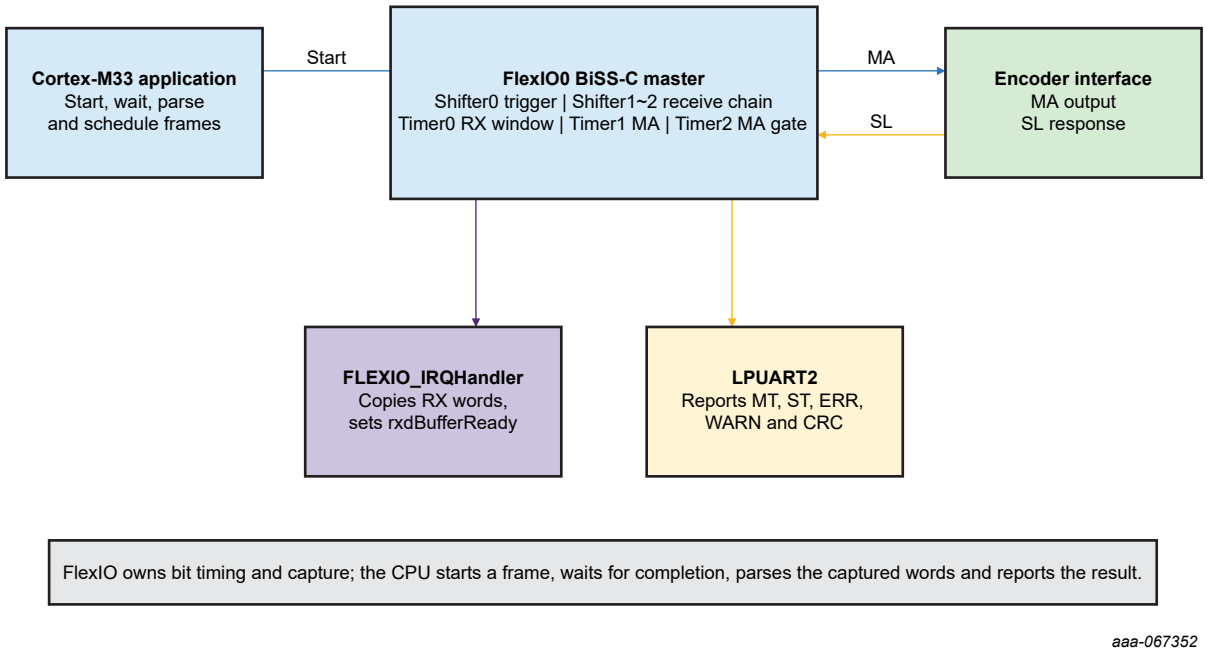


Figure 4. MCX A366 FlexIO BiSS-C controller architecture

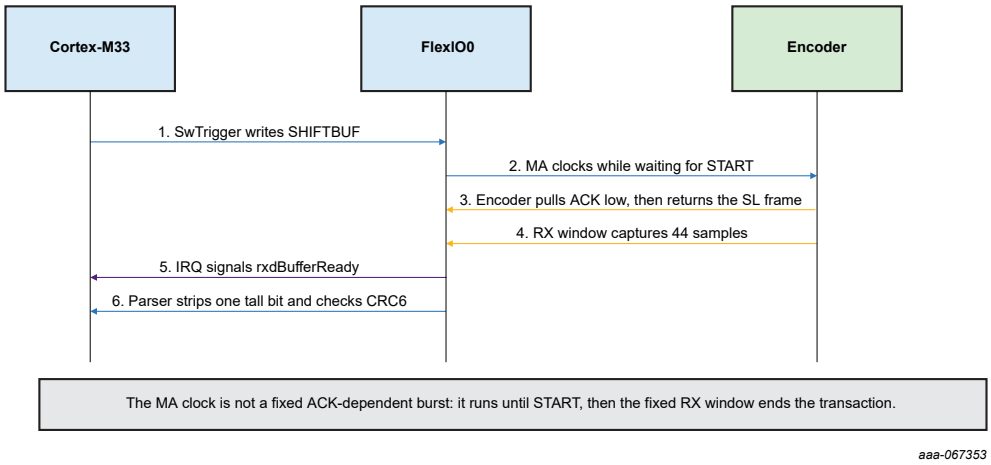


Figure 5. Software trigger, STR synchronization, receive, and parse sequence

4.1 Software modules

Table 5 lists the source modules and their responsibilities in the application flow.

Table 5. Software module responsibilities

File/module	Responsibility
source/flexio_bissc_adk4408_ interrupt_transfer.c	Starts frames, keeps buffer ownership through parsing, prints position/CDS/status, and reports diagnostics/statistics
source/flexio_bissc.c	Derives window/shifter count, configures hardware, captures frames, parses CRC6, maintains diagnostics/statistics, and services the Interrupt Service Routine (ISR)
source/flexio_bissc.h	Profile-dependent macros, handle state, diagnostics, statistics, and FLEXIO_BISSC APIs

Table 5. Software module responsibilities*...continued*

File/module	Responsibility
board/app.h	Pin assignment, MT/ST, trailing sample, optional D30 probe, timeout, soak mode, and parser self-test controls
board/pin_mux.c	P2_9/D17 SL input, P2_10/D18 MA output, and P2_22/D30 optional gate-probe muxing
board/clock_config.*	Board clock tree
drivers/fsl_flexio.*	SDK register-level FlexIO helpers
drivers/fsl_debug_console.*	115,200 bit/s diagnostic output

4.2 Initialization and runtime flow

1. BOARD_InitHardware() initializes the board clocks, pins, debug console, and peripheral resources.
2. DEMO_InitBissc() masks the IRQ, calls FLEXIO_BISSC_Init(), enables the RX interrupt, clears the pending IRQ, and then enables FLEXIO_IRQn.
3. FLEXIO_BISSC_Init() validates the profile, derives rxWindowBitLen and rxShifterNum, configures the shifter chain and three timers, and records the active handle. The current 44-bit capture uses two receive shifters.
4. FLEXIO_BISSC_SwTrigger() clears stale flags and writes the trigger shifter to start MA.
5. The application waits for rxdBufferReady, retains buffer ownership through FLEXIO_BISSC_DataParser(), prints MT, ST, signed position, CDS, START, ERR, WARN, and CRC status, then releases the buffer.
6. On timeout, the application reinitializes FlexIO to stop MA before retrying after the configured frame interval.

4.3 CPU and FlexIO resource allocation

The design assigns all bit-critical work to FlexIO0. The Cortex-M33 defines transaction boundaries and processes a frame only after FlexIO has captured the response.

Table 6. CPU and FlexIO resource allocation

Execution owner	Assigned resource/phase	Responsibility	CPU loading implication
FlexIO0	Shifter 0 + Timer 1	Software-triggered MA generation	No CPU work per MA edge
FlexIO0	Shifters 1-2 + Timer 0	STR-aligned SL capture	The CPU is outside the sample loop
FlexIO0	Timer 2	MA gate, receive qualification, and optional D30 probe output	Hardware timing/measurement support
Cortex-M33	Main loop	Start frame, retain ownership through parse, report CDS/position/status/statistics	Control cost per frame
Cortex-M33	FLEXIO_IRQHandler	Read the full receive chain; publish only released buffers or count an overrun	Short completion path
Cortex-M33 + LPUART2	PRINTF	Report decoded diagnostics	Variable serial-output cost

CPU utilization is not instrumented in this example. The configured frame interval is 200 ms, and serial output is a variable software cost that must be included in any measured CPU-load result.

5 Key implementation details

This section connects the configured FlexIO resources and timing values to the BiSS-C capture and parsing behavior.

5.1 FlexIO clock and 10 MHz bit rate

The FlexIO dual-8-bit baud/bit timer uses the low compare byte as a half-period divider. For a 180 MHz source and 10 MHz target, baudDiv is 8.

10 MHz baud-divider calculation

baudDiv = ((180,000,000 / 10,000,000) >> 1) - 1 = 8; bit rate = 180,000,000 / (2 x (8 + 1)) = 10,000,000 bit/s

Table 7. Timer compare values

Timer/field	High compare byte	Low compare byte	Result
RX timer (TIMER0)	0x57	0x08	44-bit STR-aligned receive window
MA timer (TIMER1)	0xFF	0x08	MA runs until the RX window stops it
MA gate timer (TIMER2)	0xFF	0xFF	Qualifies STR and active transaction

5.1.1 Theoretical transaction waveform

Figure 6 shows MA activity, the STR timing reference, the SL response, and the fixed capture window.

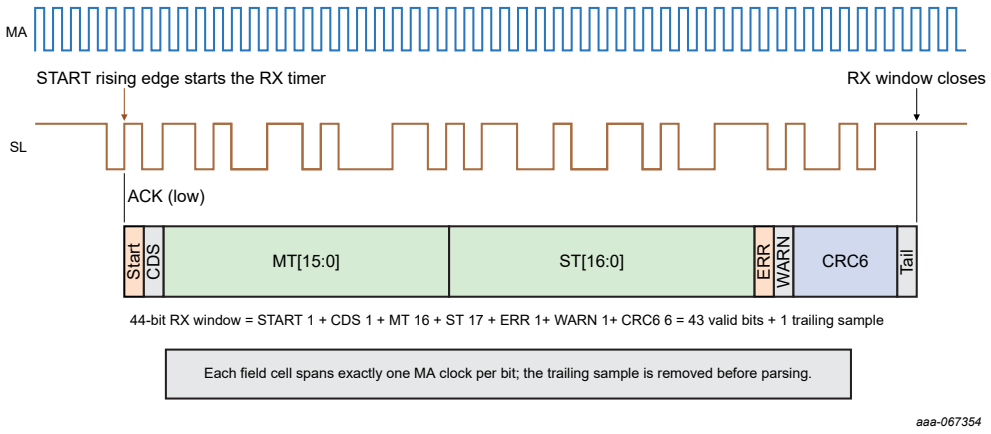


Figure 6. Conceptual MA, STR, SL response, and fixed receive window

5.1.2 FlexIO Register configuration

The following tables record the register images for the active FlexIO allocation. They were taken from the supplied BiSS-C register tables and cross-checked against source/flexio_bissc.c; the receive window is now derived from the profile rather than maintained as a separate constant.

SHIFTER0 register image: SHIFTCTL0=0x01000002 and SHIFTCFG0=0x00000000. A CPU write to SHIFTBUF0 clears SHIFTSTAT0 and activates the active-low TIMER1 trigger.

Table 8. Software trigger SHIFTER0 register configuration

Register	Field	Bits	Value	Description
SHIFTCTL0	TIMSEL	[26:24]	1	Selects TIMER1/MA_TIMER as the shifter clock

Table 8. Software trigger SHIFTER0 register configuration*...continued*

Register	Field	Bits	Value	Description
	TIMPOL	[23]	0	Shifts on the configured timer positive edge
	PINCFG	[17:16]	0	Shifter pin output disabled; no external I/O is driven
	PINSEL	[12:8]	0	Pin selection is unused because PINCFG is disabled
	PINPOL	[7]	0	Inactive because no shifter pin is driven
	SMOD	[2:0]	2	Transmit mode for the software trigger
SHIFTCFG0	PWIDTH	[20:16]	0	The parallel width field is zero
	INSRC	[8]	0	Default input-source configuration
	SSTOP	[5:4]	0	Automatic stop-bit handling disabled
	SSTART	[1:0]	0	Automatic start-bit handling disabled; data loads on enable

SHIFTER1 register image: SHIFTCTL1=0x00001101 and SHIFTCFG1=0x00000100. It receives the earlier portion of the chained data from SHIFTER2 rather than sampling SL directly.

Table 9. Receive-chain SHIFTER1 register configuration

Register	Field	Bits	Value	Description
SHIFTCTL1	TIMSEL	[26:24]	0	Selects TIMER0/RX_TIMER as the shifter clock
	TIMPOL	[23]	0	Samples on the RX timer positive edge
	PINCFG	[17:16]	0	Shifter pin output disabled
	PINSEL	[12:8]	17	D17 is selected but not sampled directly when INSRC=1
	PINPOL	[7]	0	Input is not inverted
	SMOD	[2:0]	1	Receive mode
SHIFTCFG1	PWIDTH	[20:16]	0	The parallel width field is zero
	INSRC	[8]	1	Input comes from the next shifter, SHIFTER2
	SSTOP	[5:4]	0	Stop-bit checking disabled
	SSTART	[1:0]	0	Start-bit checking disabled; data loads on enable

SHIFTER2 register image: SHIFTCTL2=0x00001101 and SHIFTCFG2=0x00000000. It samples D17/SL directly and is the final stage of the chained receive path.

Table 10. SL input SHIFTER2 register configuration

Register	Field	Bits	Value	Description
SHIFTCTL2	TIMSEL	[26:24]	0	Selects TIMER0/RX_TIMER as the shifter clock
	TIMPOL	[23]	0	Samples on the RX timer positive edge
	PINCFG	[17:16]	0	External pin drive disabled; this shifter is used as an input
	PINSEL	[12:8]	17	Selects FLEXIO0_D17/SL
	PINPOL	[7]	0	SL is active high and not inverted
	SMOD	[2:0]	1	Receive mode
SHIFTCFG2	PWIDTH	[20:16]	0	The parallel width field is zero

Table 10. SL input SHIFTER2 register configuration*...continued*

Register	Field	Bits	Value	Description
	INSRC	[8]	0	Input comes directly from the selected SL pin
	SSTOP	[5:4]	0	Stop-bit checking disabled
	SSTART	[1:0]	0	Start-bit checking disabled; data loads on enable

Timer summary: TIMER0 receives the STR-aligned 44-bit window; TIMER1 produces MA; and TIMER2 keeps the receive gate active only while MA is active.

Table 11. FlexIO timer allocation and register-image summary

Timer	Register image	Operation	Pin/trigger	Purpose
TIMER0	0x0B401101/0x01002500/0x00005708	Dual 8-bit; compare disable	D17 rising edge; TIMER2 gate	44-bit STR-aligned RX window
TIMER1	0x01C31281/0x01001200/0x0000FF08	Dual 8-bit; pre-timer disable	SHIFTER0 active-low; D18	MA generation
TIMER2	0x07401203/0x00001100/0x0000FFFF	Single 16-bit; pre-timer enable/disable	Follows TIMER1; optionally routed to D30	MA gate/frame qualification and optional high-during-burst probe

TIMER0 register image: TIMCTL0=0x0B401101, TIMCFG0=0x01002500, and TIMCMP0=0x00005708. It starts on an SL rising edge while the TIMER2 gate is high, then disables on compare.

Table 12. STR-aligned RX TIMER0 register configuration

Register	Field	Bits	Value	Description
TIMCTL0	TRGSEL	[29:24]	11	Internal trigger selects TIMER2 output
	TRGPOL	[23]	0	Trigger is active high
	TRGSRC	[22]	1	Uses the internal FlexIO trigger
	PINCFG	[17:16]	0	Timer does not drive an external pin
	PINSEL	[12:8]	17	Monitors D17/SL
	PINPOL	[7]	0	SL high is active
	TIMOD	[2:0]	1	Dual 8-bit baud/bit mode
TIMCFG0	TIMOUT	[25:24]	1	Internal output starts low when enabled
	TIMDEC	[22:20]	0	Decrements from the FlexIO functional clock
	TIMRST	[18:16]	0	The timer does not reset automatically
	TIMDIS	[14:12]	2	Disables on timer compare
	TIMENA	[10:8]	5	Enables on SL rising edge while trigger is high
	TSTOP	[5:4]	0	Timer stop bit disabled
	TSTART	[1]	0	Timer start bit disabled
TIMCMP0	Baud Div	[7:0]	0x08	Half period is nine FlexIO clocks
	Bit Count	[15:8]	0x57	44-bit receive window

TIMER1 register image: TIMCTL1=0x01C31281, TIMCFG1=0x01001200, and TIMCMP1=0x0000FF08. It drives active-low MA on D18 and stops when TIMER0 disables.

Table 13. MA generation TIMER1 register configuration

Register	Field	Bits	Value	Description
TIMCTL1	TRGSEL	[29:24]	1	Internal trigger selects SHIFTER0 SHIFTSTAT
	TRGPOL	[23]	1	Trigger is active low
	TRGSRC	[22]	1	Uses the internal FlexIO trigger
	PINCFG	[17:16]	3	Timer output is connected to the external MA pin
	PINSEL	[12:8]	18	Selects D18/MA
	PINPOL	[7]	1	MA output is inverted and active low
	TIMOD	[2:0]	1	Dual 8-bit baud/bit mode
TIMCFG1	TIMOUT	[25:24]	1	Internal output starts low when enabled
	TIMDEC	[22:20]	0	Decrements from the FlexIO functional clock
	TIMRST	[18:16]	0	The timer does not reset automatically
	TIMDIS	[14:12]	1	Disables when preceding TIMER0 disables
	TIMENA	[10:8]	2	Enables while the active-low trigger is valid
	TSTOP	[5:4]	0	Timer stop bit disabled
	TSTART	[1]	0	Timer start bit disabled
TIMCMP1	Baud Div	[7:0]	0x08	Half period is nine FlexIO clocks
	Bit Count	[15:8]	0xFF	128-bit internal count capacity; TIMER0 ends the MA transaction

TIMER2 register image: TIMCTL2=0x07401203, TIMCFG2=0x00001100, and TIMCMP2=0x0000FFFF. It follows the active MA transaction, rejects SL edges outside that transaction, and can be routed to D30/P2_22 as a gate probe when DEMO_BISSC_GATE_PIN_OUTPUT is enabled.

Table 14. MA gate TIMER2 register configuration

Register	Field	Bits	Value	Description
TIMCTL2	TRGSEL	[29:24]	7	Internal trigger selects TIMER1 output
	TRGPOL	[23]	0	Trigger is active high
	TRGSRC	[22]	1	Uses the internal FlexIO trigger
	PINCFG	[17:16]	0	Timer does not drive an external pin
	PINSEL	[12:8]	18	D18 selector value; not used as an external output
	PINPOL	[7]	0	High polarity
	TIMOD	[2:0]	3	Single 16-bit timer mode
TIMCFG2	TIMOUT	[25:24]	0	Internal output starts high when enabled
	TIMDEC	[22:20]	0	Decrements from the FlexIO functional clock
	TIMRST	[18:16]	0	The timer does not reset automatically
	TIMDIS	[14:12]	1	Disables when preceding TIMER1 disables
	TIMENA	[10:8]	1	Enables when preceding TIMER1 enables
	TSTOP	[5:4]	0	Timer stop bit disabled
	TSTART	[1]	0	Timer start bit disabled

Table 14. MA gate TIMER2 register configuration*...continued*

Register	Field	Bits	Value	Description
TIMCMP2	Compare	[15:0]	0xFFFF	Maximum compare; TIMER1 disable is the normal stop event

5.2 STR Synchronization

The SL rising edge enables TIMER0 while the MA gate is active. This makes the encoder STR transition the receive timing reference.

The controller keeps MA running while it waits for STR because the delay before STR is encoder-dependent. TIMER0 completion then propagates a stop condition to the MA timer.

5.3 Receive window and shifter capture

Two receive shifters capture the 44-bit derived window. The driver reads SHIFTBUFBIS in descending chain order and removes the one trailing sample. The driver publishes the capture only after the main loop has released the previous buffer.

Receive-shifter count calculation

rxShifterNum = ceil(RX window / 32) = ceil(44 / 32) = 2 receive shifters.

5.4 Frame parsing and CRC6

FLEXIO_BISSC_DataParser removes CRC6 and status fields, extracts ST, MT, CDS, and START, and sets frameValid only when the CRC matches and START is asserted.

Table 15. Frame parsing and CRC6 processing

Parser step	Input	Output/failure mode
Remove tail	44-bit raw capture	Right shift by one trailing sample
Read CRC6	Low six normalized bits	Received CRC6 value
CRC calculation	WARN + ERR + ST + MT	Calculated CRC6 using 0x43/0x3F
Field extraction	Normalized shifted data	WARN, ERR, 17-bit ST, 16-bit MT
Field extraction	MT, ST, ERR, WARN	MT and ST are read from the aligned frame; ERR and WARN are preserved
Extract CDS/START	Leading aligned bits	CDS is reported; START=1 is required for a valid frame
Final status	CRC result and START	frameValid = CRC match AND START asserted

5.5 Timeout and diagnostic output

The interrupt path waits for rxdBufferReady with a real 100 µs timeout, sampled in 5 µs steps. A timeout prints raw FlexIO diagnostics, identifies likely MA/STR/RX/IRQ/gate-wrap stages, reinitializes FlexIO to stop MA and clear state, then retries after the configured interval.

Table 16. Timeout and diagnostic interpretation

Console result	Meaning	Primary checks
CRC=OK	CRC matched; inspect START separately	Verify CDS and signed combined position

Table 16. Timeout and diagnostic interpretation*...continued*

Console result	Meaning	Primary checks
CRC=FAIL	Frame captured but CRC mismatch	Check alignment, wiring, sample edge, and frame profile
Timeout	Raw FlexIO diagnostics classify the waiting stage	Check MA absent, STR absent, RX/IRQ reporting, or gate wrap
START missing	CRC can match but STR alignment was not confirmed	Check STR polarity/edge and TIMER0 start configuration
STATS	Cumulative attempts, frames, CRC, START, timeout, and overrun counts	Use normal or soak mode to report and characterize sustained operation.

6 Main function reference

Table 17 summarizes the application entry points and driver APIs referenced by the example.

Table 17. Main function reference

Function	File	Description
main()	source/flexio_bissc_adk4408_interrupt_transfer.c	Initializes the board, triggers frames, parses results, and prints status
DEMO_InitBissc()	source/flexio_bissc_adk4408_interrupt_transfer.c	Initializes FlexIO and enables the interrupt path
FLEXIO_BISSC_Init()	source/flexio_bissc.c	Derives the capture window and shifter count, validates resources, and configures FlexIO
FLEXIO_BISSC_Sw Trigger()	source/flexio_bissc.c	Clears flags and starts MA through SHIFTER0
FLEXIO_BISSC_Data Parser()	source/flexio_bissc.c	Checks CRC6, extracts CDS/ START/MT/ST/status fields, and sets frameValid
FLEXIO_BISSC_EnableRx Interrupt()	source/flexio_bissc.c	Controls first RX shifter interrupt
FLEXIO_BISSC_ IRQHandler()	source/flexio_bissc.c	Publishes a released buffer or drops an owned one and increments overrun statistics
FLEXIO_IRQHandler()	source/flexio_bissc_adk4408_interrupt_transfer.c	Interrupt the wrapper that passes g_bisscHandle to the driver
FLEXIO_BISSC_IsRx Complete()	source/flexio_bissc.c	Tests the first receive-shifter completion status
FLEXIO_BISSC_Read Frame()	source/flexio_bissc.c	Reads the shifter chain in descending order and strips the trailing sample
FLEXIO_BISSC_Run ParserSelfTest()	source/flexio_bissc.c	Exercises synthetic MT, ST, ERR, WARN, CDS, START, and CRC parsing

Table 17. Main function reference*...continued*

Function	File	Description
FLEXIO_BISSC_GetDiag()	source/flexio_bissc.c	Returns raw FlexIO status and decoded transfer-stage diagnostics
FLEXIO_BISSC_Reset Stats()	source/flexio_bissc.c	Clears cumulative counters after the optional parser self-test

The public handle keeps the active pins, profile, derived window/shifter count, capture ownership, parsed frame fields, diagnostics, and cumulative statistics together. Update board/app.h only after verifying that the derived window fits the available resources.

7 Build, flash, and run

This section gives the build, download, and serial-output steps for the documented MCX A366 demonstration. Find and download the project an-mcxa366-bissc-interface-using-flexio from the NXP Application Code Hub.

7.1 Development environment

- MCUXpresso IDE 25.6.136.
- Project: flexio_bissc_adk4408_interrupt_transfer.
- Board: FRDM-MCXA366 with MCU-Link debug and serial connection.
- Encoder: ADK-4088-MA16/17B1BLP-BH40 (IDENCODER, https://www.idencoder.cn) through a BiSS-C electrical interface and encoder power supply.

7.2 Build

To build the MCX A366 BiSS-C project in MCUXpresso IDE, perform the following steps:

1. Import or select the MCXA366 BiSS-C project in MCUXpresso IDE.
2. Select the debug configuration and run build Project.
3. Confirm that Debug/frdmmcxa366_flexio_bissc_master.axf is generated without build errors.

7.3 Download and run

To run the MCX A366 BiSS-C application and verify its startup parameters, perform the following steps:

1. Connect the FRDM-MCXA366 MCU-Link USB port and open a 115200, 8-N-1 serial terminal with no flow control.
2. Verify MA, SL, ground, encoder supply, and electrical-interface wiring before starting the target.
3. Start a debug session or download the axf, then reset and run the board.
4. Confirm that the startup banner reports 180000000 Hz, 10000000 bit/s, MT=16, and ST=17. Confirm that the startup banner also reports the 44-bit derived RX window, one trailing sample, two receive shifters, and the 100 µs frame-wait timeout.

7.4 Expected serial output

The following record shows the expected startup and decoded-frame fields from the documented serial output.

```
MCUX SDK version: 2026.06.00

FlexIO BiSS-C ADK-4088-MA16/17B1BLP-BH40 demo on MCXA366
```

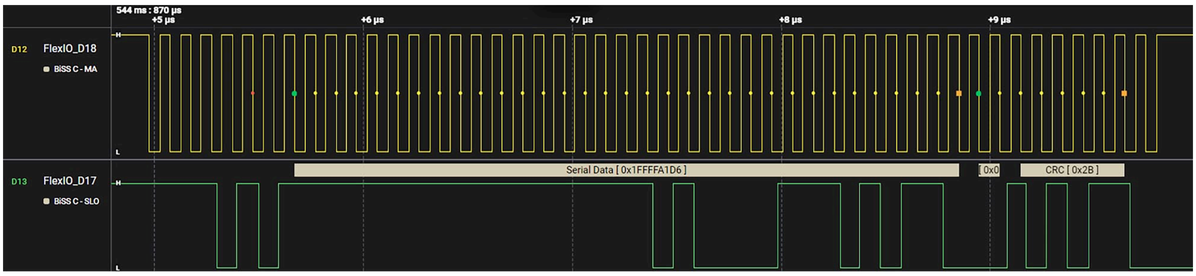
```
FlexIO clock: 180000000 Hz
Bit rate: 10000000 bps, MT: 16 bits, ST: 17 bits
MA: FLEXIO0_D18, SL: FLEXIO0_D17
MA gating: continuous until STR, then stopped by the RX window; SL pull: off
RX start: STR rising edge, RX edge: rising, RX window: 44 bits (1 trailing), 2
 shifters
Frame wait timeout: 100 us, frame interval: 200000 us
Statistics every 20 attempts

Parser self-test: MT=4660, ST=109226, POS=610904746, CDS=1, START=1, ERR=1,
 WARN=0, CRC=OK, RESULT=PASS

MT=65535, ST=118134, POS=-12938, CDS=0, START=1, ERR=1, WARN=0, CRC=FAIL
MT=65535, ST=105197, POS=-25875, CDS=0, START=1, ERR=0, WARN=0, CRC=OK
MT=65535, ST=105197, POS=-25875, CDS=0, START=1, ERR=0, WARN=0, CRC=OK
MT=65535, ST=105197, POS=-25875, CDS=0, START=1, ERR=0, WARN=0, CRC=OK
MT=65535, ST=105197, POS=-25875, CDS=0, START=1, ERR=0, WARN=0, CRC=OK
MT=65535, ST=105197, POS=-25875, CDS=0, START=1, ERR=0, WARN=0, CRC=OK
MT=65535, ST=105197, POS=-25875, CDS=0, START=1, ERR=0, WARN=0, CRC=OK
MT=65535, ST=105197, POS=-25875, CDS=0, START=1, ERR=0, WARN=0, CRC=OK
MT=65535, ST=105197, POS=-25875, CDS=0, START=1, ERR=0, WARN=0, CRC=OK
CDS byte=0x0
MT=65535, ST=105197, POS=-25875, CDS=0, START=1, ERR=0, WARN=0, CRC=OK
```

8 Results and observations

The supplied board README records CRC=OK frames from ADK-4088-MA16/17B1BLP-BH40. Figure 7 preserves that historic UART record.

The ERR and WARN fields are printed as raw encoder status bits. The current parser also reports CDS and START, and frameValid requires both a matching CRC and START=1.

The source evidence is serial output, not a logic-analyzer screenshot. Before qualifying hardware, capture at least 100 stationary frames and 100 rotating frames with CRC=OK and START=1. Then use the MA/SL waveform plus an optional gate probe to confirm the STR-aligned timing.



aaa-067356

Figure 7. Logic-analyzer timing capture of BiSS-C MA (FLEXIO0_D18) and SL (FLEXIO0_D17)

The logic-analyzer trace provides a separate measurement view of the MA and SL relationship. It is retained as captured bench evidence rather than generated output from the current firmware.

9 Conclusion

The MCX A366 project implements a BiSS-C position-acquisition path with FlexIO resources and a small Cortex-M33 parser. FlexIO generates MA, derives the STR-aligned receive window, captures the response, and reports START-qualified CRC and diagnostics.

The key alignment rule is to keep the source-derived receive window coupled with the one-bit trailing-sample removal. The normalized parser extracts START, CDS, MT=16, ST=17, ERR, WARN, and CRC6, and sets frameValid only when CRC and START agree.

When changing the encoder, electrical interface, cable, clock rate, or frame profile, update the board/app.h. Let FLEXIO_BISSC_Init() recompute the capture window, verify the parser layout, and capture MA/SL timing before accepting the new profile.

10 Reference implementation

This application note references the FlexIO BiSS-C SDK examples on i.MX RT1186. It uses the project source as the authority for FlexIO resource allocation, STR-based receive synchronization, frame handling, CRC6 processing, timeout recovery, and serial diagnostics.

11 Acronyms

Table 18 describes the acronyms used in this document.

Table 18. Acronyms

Acronym	Description
CRC	Cyclic Redundancy Check
FlexIO	Flexible Input/Output
FRDM	Freedom Development Platform
IDE	Integrated Development Environment
IR	Interrupt Request
ISR	Interrupt Service Routine
MSB	Most Significant Bit
MT	Multi-Turn
SDK	Software Development Kit

12 Note about the source code in the document

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

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

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

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED

TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

13 Revision history

Table 19 summarizes revisions to this document.

Table 19. Revision history

Document ID	Release date	Description
AN15161 v.1.0	18 September 2026	Initial public release

Legal information

Definitions

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

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

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

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

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

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