AN15236

Magnetic Encoder Interface Design with MCX A344 MCU

Rev. 1.0 — 10 November 2026 Application note

Document information

Information	Content
Keywords	AN15236, FRDM-MCXA344, MCX A344, MEMSIC, MDA2000BT, magnetic encoder, AMR, Hall, SPI, MCUXpresso for VS Code, Python, Web workbench
Abstract	This application note describes a magnetic encoder interface implemented on the NXP MCX A344 MCU using the FRDM-MCXA344 Development Board and a MEMSIC MDA2000BT sensor. It explains the six-wire SPI connection, a 100 Hz bare-metal acquisition task, software data checks and timestamp-based speed estimation. MCUXpresso SDK and VS Code provide the firmware development path; a local Python service and bilingual Web workbench support observation and design exploration. The implementation separates sensor, board and processing responsibilities for reuse. Zero setting, persistent configuration and nonlinear calibration are outside this implementation.

1 Introduction

This application note shows Micro Controller Unit (MCU) application engineers how to implement a digital magnetic encoder interface on the NXP MCX A344 MCU. It uses the MEMSIC MDA2000BT for shaft-angle sensing and the FRDM-MCXA344 Development Board for acquisition and processing. The design addresses sensor status, communication errors, angle wraparound and host connection state, providing a starting point for applications that need valid angle and low-speed rotation data.

The MCU schedules acquisition at 100 Hz and computes signed Revolutions Per Minute (RPM) from angle and MCU time. A local Python service validates and forwards samples to a buildless browser interface. The **Live instruments** view shows the rotor dial, recent traces, and diagnostics; the **How it works** view follows the same measured presentation angle. The **Design & Development** view explains the sensor, MCU, MCU-Link, and host responsibilities. The firmware development and reuse path centers on MCUXpresso Software Development Kit (SDK) and MCUXpresso for VS Code.

The intended operating range is manual rotation and low-speed operation up to 1000 rpm. The upper value is a design range, not a demonstrated physical limit. Manual motion and algorithm fixtures do not establish speed accuracy, angular accuracy, or behavior at 1000 rpm. The first implementation includes acquisition and diagnostics; it does not set zero. Perform Non-Linear Calibration (NLC), write Electrically Erasable Programmable Read-Only Memory (EEPROM), or control a motor.

The accompanying Application Code Hub (ACH) v0.2 candidate contains the firmware, host software, and setup instructions[5]-[7]. Sensor descriptions refer to the supplied preliminary MEMSIC data sheet[1]. MCU capabilities refer to the exact NXP device data sheet[8].

2 System overview and operating principle

The MDA2000BT provides a digital absolute angle over Serial Peripheral Interface (SPI). Its sensing and angle calculation occur inside the sensor. MCX A344 acts as the SPI controller: it starts transactions, checks the returned data, tracks acquisition time, and derives speed. MCU-Link carries the MCU Universal Asynchronous Receiver/Transmitter (UART) to the Personal Computer (PC) as a USB serial port. A single Python process owns that port and serves the interface on the loopback address. No cloud service or remote browser asset is required during the normal use.

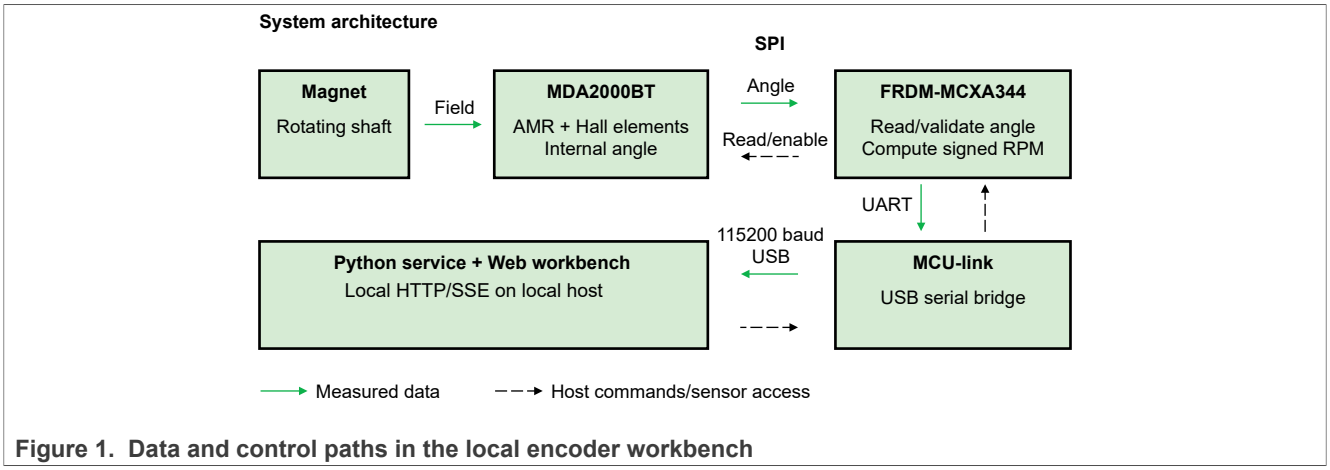


Figure 1. Data and control paths in the local encoder workbench

2.1 MCX A344 resources in the interface

Table 1 lists MCU and development resources used by the interface.

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

© 2026 NXP B.V. All rights reserved.

Table 1. MCU and development resources used by the interface

Resource	Role in this design
LPSPI1	SPI controller for angle and register transactions using the SDK driver
SysTick and MCU software	Timebase, foreground scheduling, software Cyclic Redundancy Check (CRC), and timestamp-based speed estimation
LPUART2 and MCU-Link	UART telemetry through the onboard USB serial bridge; MCU-Link also provides Serial Wire Debug (SWD) debugging
MCUXpresso SDK and VS Code	Peripheral drivers and board configuration, CMake/GCC builds, and LinkServer debugging

2.2 Magnetic angle sensing

The supplied MDA2000B-family data sheet describes Anisotropic Magneto Resistance (AMR) and Hall sensing elements, sensitivity to magnetic-field direction in the X–Y plane, and internal signal processing[1]. Its block diagram places amplification and analog-to-digital conversion on the AMR signal path and comparator paths on the Hall channels. This note does not infer the proprietary combination algorithm. The public MDA2000BT description identifies full-turn sensing and a 21-bit, four-wire SPI angle output[2].

As a shaft-mounted magnet rotates, its in-plane field direction changes. The sensor converts that information to digital angle. The MCU does not sample external sine/cosine channels or calculate the primary sensor angle with its Analog-to-Digital Converter (ADC). Figure 2 is an original conceptual explanation. Pole colors, field geometry and the illustrated zero are teaching aids. They are not measured field strength, a dimensioned magnet arrangement or a verified physical clockwise convention.

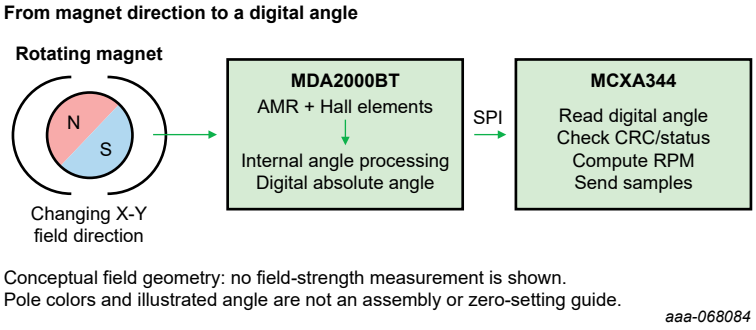


Figure 2. Division of responsibility between magnetic sensing and MCU processing

2.3 Rates and timing domains

Table 2. Implemented rate and timing definitions

Quantity	Definition
Acquisition	100 Hz target; one foreground slot every 10 ms
MCU time	1 ms SysTick count captured at sampling start
Speed window	Approximately 100 ms of valid MCU measurements
SSE delivery	At most 20 updates per second, with sample batches
Angle presentation	Nominal 80 ms buffer of measured positions
Stale indication	No new sample for more than 500 ms, or disconnected

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

© 2026 NXP B.V. All rights reserved.

These values describe different stages. The presentation buffer is not an end-to-end latency specification, and the stale threshold is software logic rather than a measured physical unplug-to-indication bound. Speed uses MCU timestamps, not PC arrival time. Batching or browser refresh therefore does not redefine the MCU speed calculation.

3 Hardware setup

Use the FRDM-MCXA344 Development Board[3], the MEMSIC sensor board configured for 3.3 V, a rotating magnet assembly, and six short connections. The tested sensor board accepts 3.3 V. Do not assume that another MEMSIC board has the same supply configuration. Remove power before wiring. Disconnect both power and signal connections from the original STM32 controller so that the sensor has one host. Do not install a MOSI-to-MISO loopback with the sensor attached.

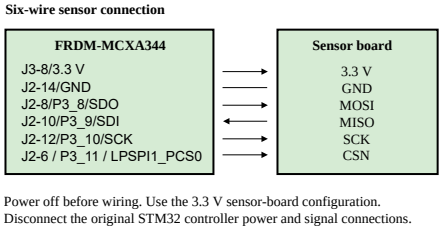


Figure 3. Six-wire connection for the tested 3.3 V sensor-board configuration

The FRDM headers expose LPSPI1 directly: P3_8 is SDO, P3_9 is SDI, P3_10 is SCK, and P3_11 is PCS0. Connect the MCU-Link USB port to the PC for power, programming, debugging and serial communication. Select the enumerated MCU-Link serial port rather than a fixed COM number. The development bench used COM55 and USB serial 0KCOTX11HTXSQ; a different PC or board can enumerate differently.

Keep the magnet-to-sensor spacing and alignment fixed while rotating the shaft, and secure the wire bundle against motion. The demonstration does not define magnet dimensions, permitted air gap, or assembly tolerances. Use the sensor supplier's requirements for a final mechanical design. Before interpreting a transient fault as a software defect, distinguish shaft rotation from changes in magnet distance, supply connection, or signal contact. A constant-speed fixture is required for later upper-speed and NLC work.

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

© 2026 NXP B.V. All rights reserved.

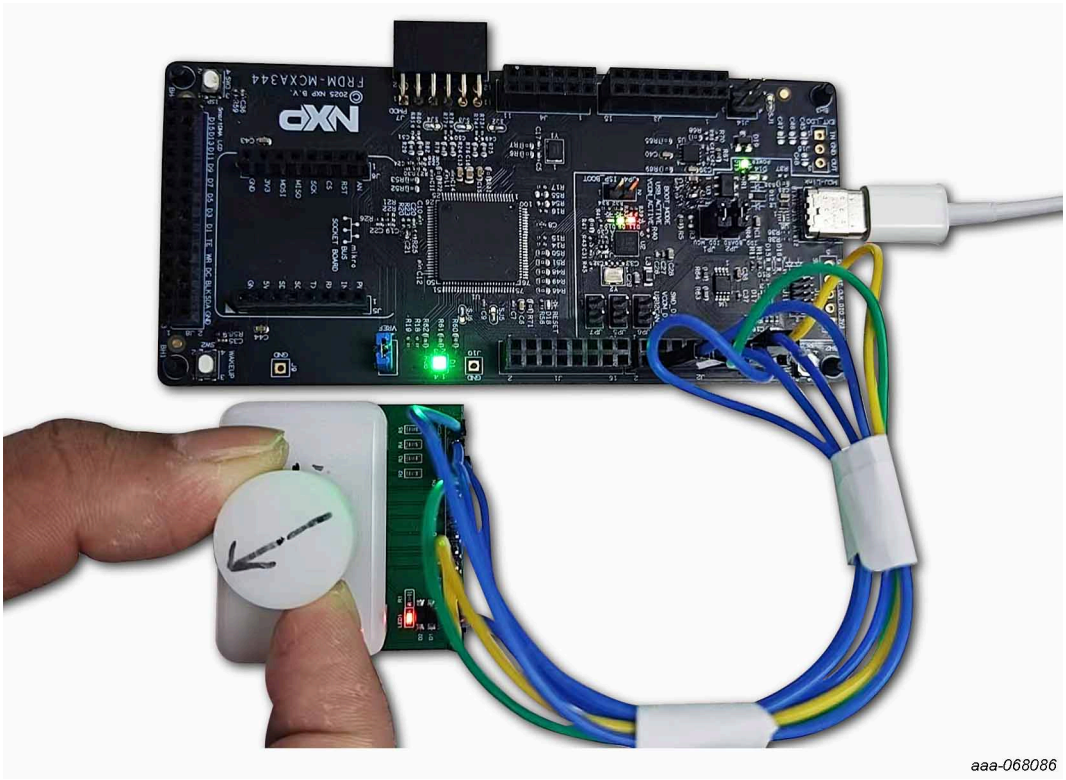


Figure 4. FRDM-MCXA344, MEMSIC sensor board, and manual rotor with interconnecting wiring

4 Getting started

This section describes how to get started with the design.

4.1 Prepare and build the firmware

Install MCUXpresso for VS Code, Arm GNU Toolchain 14.2, Ninja, CMake 3.27 or later and the MCUXpresso Python environment. Obtain the complete FRDM-MCXA344 SDK 26.09.00 from the official catalog[4]. Extract it outside the application; the SDK root contains mcuxsdk, manifests and `.west`. The freestanding `CMake/GCC` project keeps application sources separate from the external SDK while reusing its drivers and board support. Historical Keil bring-up files are not the formal build path.

```
./firmware/build.ps1 -SdkRoot 'C:/SDK/FRDM-MCXA344_26.09.00'
```

```
./firmware/build.ps1 -SdkRoot 'C:/SDK/FRDM-MCXA344_26.09.00' -Configuration
release
```

Replace the example SDK path with the installation directory. Alternatively, use `SDK_ROOT` or the ignored local configuration described in the *firmware/README.md*[5]. The script resolves the SDK, compiler, and LinkServer and writes machine-specific settings locally. Generated builds use an ASCII TEMP directory. When the checkout path contains non-ASCII characters, the script prints an ASCII source entry referring to the same sources. Open that entry in VS Code because this SDK's generated-header pipeline cannot handle the original path.

1. Open the firmware directory, or the ASCII source entry printed by the build script, in VS Code and trust the project.
2. Run the **Build firmware debug** task. Confirm that the selected SDK and compiler are the expected installations.

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

© 2026 NXP B.V. All rights reserved.

3. Select FRDM-MCXA344 MCU-Link in **Run and Debug**. Deliberately select the connected board. The candidate configuration initially names the development bench probe.
4. Start debugging, confirm the stop at main, and continue. Terminate the debug session before normal unattended acquisition.

4.2 Start the local workbench

Install Python 3.10 or later on Windows. Run start_demo.cmd from the project root. First launch creates an independent .venv and installs the pinned Python packages, so package-download access is needed once. Later launches reuse the local environment. The launcher starts one service at http://localhost:8000, verifies its identity, and opens the browser. It reuses an existing workbench instance and does not replace an unrelated process occupying the port.

1. Choose the MCU-Link serial port and click Connect. Opening a page alone does not connect the device.
2. Wait for initialization and valid angle data. The service performs INFO, STOP, INIT, and START in that order.
3. Rotate slowly in each direction and across zero. Confirm that angle, RPM, and direction update. Switch to **How it works** to inspect the same measured rotation.
4. Use **Disconnect** to release the port. Stop the Python terminal with Ctrl+C when finished. Closing a browser tab alone does not stop other tabs.

The browser supports immediate English/Chinese switching. Manual environment setup and an alternate HyperText Transfer Protocol (HTTP) port are documented in *backend/README.md*. Use the Python service rather than opening the production HTML file directly. No Node.js installation, front-end compilation, or vendor PC application is needed for normal operation. Open **Design & Development** to review the system path, wiring, and development resources. This static view also works without a device connection.

5 Firmware implementation

The firmware separates board I/O in `board_io`, sensor access in mda2000, and portable CRC, framing, and speed logic in `encoder_core`. The main loop coordinates commands, acquisition, and telemetry. This separation keeps SDK peripheral handling distinct from sensor transactions and calculations, allowing a different board or sensor to be adapted without moving host presentation logic into the MCU.

5.1 Initialize only the required sensor function

The firmware configures LPSPI1 for Mode 3, 500 kHz, 8-bit transfers, MSB first and active-low PCS0. A transaction keeps chip select continuous. Initialization checks power-ready state and reads alarm and control registers. Its only sensor setup write reads register `0x54`, sets bit 0, and writes the result back, preserving the other bits. Readback must match. This enables angle output without deliberately changing zero, direction, calibration parameters, or nonvolatile memory.

INIT runs only while streaming is stopped. A successful reply means the implemented readiness and readback checks passed; it is not a silicon-identification test or an angular-accuracy test. The host does not automatically retry INIT. This first implementation does not expose arbitrary register writes, calibration, or EEPROM commands to the host.

5.2 Acquire and validate a sample

Each acquisition obtains a six-byte angle response and separately reads alarm register `0x01`. The three angle/status bytes are RAW[2] through RAW[4]. The low 21 bits encode angle and the upper three bits form an angle-status enumeration. Convert counts to degrees as counts × 360/2097152. Register alarm bits 4:0 are independent fault flags; bits 6:5 report NLC state. Do not interpret the angle-status enumeration as a collection of alarm bits.

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

© 2026 NXP B.V. All rights reserved.

The sensor CRC-8 covers all three angle/status bytes, including the status bits, with polynomial `0x07` and initial value `0xFF`. RAW[5] carries the received CRC. The separately read alarm register is not covered by this sensor CRC. A sample is valid only when the SPI access, sensor CRC, angle status, and relevant alarm checks pass. Invalid samples remain represented in telemetry with their raw response and flags. They must not be replaced by a valid zero-angle reading.

5.3 Schedule, estimate speed, and transmit

A 1 ms SysTick interrupt updates the timebase. The foreground advances a 10 ms deadline and performs at most one acquisition when due. If time has passed several deadlines, it counts skipped slots and advances the schedule instead of executing a catch-up burst. A late start is counted separately. The sampling timestamp is the MCU time immediately before the SPI operation, not the exact sensor conversion instant. The foreground waits for completion of the SDK interrupt-driven SPI transfer, with a 100 ms timeout per transaction. Neither its wait nor text formatting runs in SysTick.

Speed calculation unwraps successive valid angle differences and uses actual MCU elapsed time over approximately 100 ms. Positive RPM means increasing raw angle. It does not assert a physical clockwise viewing direction. An invalid angle, duplicate timestamp, or interval of at least 30 ms clears the speed window. Valid angle can return before RPM becomes valid again. This avoids carrying a speed estimate across a discontinuity or assuming the shortest angular path remains known across an excessive gap.

UART operates at 115200 baud, 8N1. Interrupt-driven RX and TX rings contain 512 bytes and 4096 bytes respectively, with 512 TX bytes reserved for control replies. Telemetry is queued as a complete frame or counted as a drop; it does not block sampling while waiting for UART space. A damaged or oversized command is discarded. Sequence numbers count attempted acquisitions, while missed-slot and queue counters expose different loss mechanisms. The RX error counter includes hardware receive errors and a full software ring; it is not an exact count of lost bytes. The TX drop count includes rejected samples and periodic statistics. Statistics are emitted once per second. Sensor CRC-8 and the UART protocol's CRC-16 protect different boundaries.

6 Python service and web workbench

One serial worker owns the device. Fast Application Programming Interface (API) provides connection control and status over HTTP. Server-Sent Events (SSE) deliver current state and batches of samples at up to 20 Hz. Uvicorn binds to 127.0.0.1 with one worker. Browser reloads and additional tabs share the same serial session. The backend independently checks the text-frame CRC, sensor CRC, decoded fields, and validity before publishing angle and RPM. It keeps at most 1000 samples and ten seconds of history in memory.

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

© 2026 NXP B.V. All rights reserved.

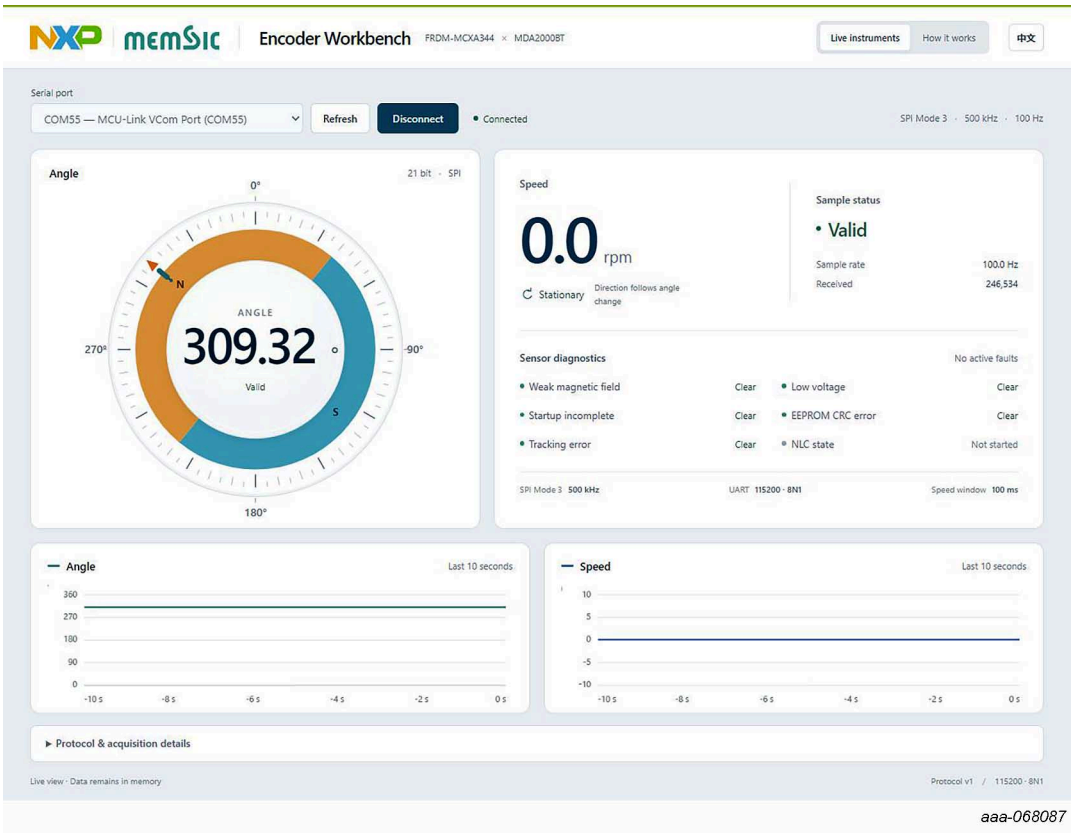


Figure 5. Instrument view with real encoder data from the earlier two-view interface

The instrument view prioritizes angle, signed speed, direction, and validity. The angle originates in the sensor. The speed is calculated by MCX A344. Two plots show recent measurements, with breaks for invalid samples. Expanded details expose raw frames, timestamps, counters, and sensor state. How it works uses the same measured presentation angle and explains the sensing-to-Web chain. Its magnet and field-direction graphics are illustrations, not additional sensor channels. The instrument screenshot retains the earlier interface as historical evidence; current navigation also includes the **Design & Development** view described below.

Design & development presents selectable sensor, MCX A344, MCU-Link, and host nodes. It explains acquisition, software validation, speed calculation and the LPUART2-to-MCU-Link USB path, with six-wire connection details and official development resources[5]. Node selection is explanatory and sends no device commands. Static content remains available while disconnected. Unused platform capabilities are identified separately.

6.1 Present the measured path without predicting motion

Animating only the latest 20 Hz angle can misrepresent direction. At the 1000 rpm design value, the shaft advances 60 degrees per 100 Hz sample but 300 degrees per 20 Hz update. A shortest-path animation of the latter appears to move backward by 60 degrees. The front-end therefore retains all valid samples in each batch, unwraps the measured path, and interpolates between measurements using a nominal 80 ms presentation buffer.

The rotor, numeric angle, and principle graphics share one presentation angle. The renderer never integrates RPM or extrapolates a missing position. Invalid data, sequence discontinuity, reset, or a gap of at least 30 ms clears the path. Late data resynchronizes to a real measurement. Reduced-motion preference displays the latest measured position directly. Raw details and RPM retain current backend values. Display refresh can still alias rapid rotation, so numerical direction and RPM remain necessary.

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

© 2026 NXP B.V. All rights reserved.

6.2 Handle connection state explicitly

An invalid sample is different from an unavailable serial session. The interface marks its value unavailable but does not invent a disconnect. More than 500 ms without new samples marks data stale. A serial failure or an MCU BOOT event invalidates the session and requires the user to reconnect. The firmware precedes BOOT with a synchronization newline so a reset during transmission cannot merge the announcement with an old partial frame. Old START or INIT commands are not replayed.

Disconnect sends STOP when possible and always releases the port. Service shutdown first ends SSE responses, then completes serial cleanup. Slow browser clients do not control acquisition. Normal operation has no recording, save, export, database, or cloud function. Separately invoked engineering verification tools may create explicit acceptance artifacts. Those tools are not an application recording feature.

7 Verification and troubleshooting

Verification distinguishes source/build checks, host algorithm fixtures, real-browser behavior, and physical-DUT observations. The development setup used FRDM-MCXA344 with MCXA344VLL identified by the debugger, SDK 26.09.00, Arm GNU 14.2, MCUXpresso for VS Code 26.9.44, and LinkServer 26.9.130. Debug and release builds passed. The VS Code build task, programming, breakpoint at main, continue, and debug termination were exercised on the board. These results establish the development path, not sensor accuracy[7].

An earlier ten-minute physical campaign used the image before the startup synchronization-newline fix. It captured 10 ms MCU intervals with zero independently detected sensor CRC, transport CRC, parse, or sequence-gap errors and zero listed schedule, queue, and SPI-error counters. It also contained one invalid sample with angle status 1 and alarm `0x02`, interpreted as low voltage by the supplied data sheet. The electrical waveform and cause were not measured. The run therefore supports timing, transport and fault accounting, not an alarm-free platform.

The earlier ten-minute image and final BOOT-synchronization image have separate SHA-256 identities in [7]. The final image has only short-capture and runtime evidence; it has no independently captured ten-minute result. This review adds no hardware retest.

After the newline change, a separate 30.072 s final-image run captured 3013 new samples. All adjacent MCU intervals at 10 ms, with no invalid samples or listed errors. Reset detection then produced an explicit reconnect error, and manual disconnect/connect restored valid data. USB removal/reinsertion also required manual recovery; a subsequent 30.034 s check captured 3007 new samples with zero listed errors. The physical removal instant was not measured, so these checks do not establish a 500 ms physical response bound.

Browser checks covered bilingual switching, page reload during acquisition, two tabs sharing one port, view switching and live presentation. The user confirmed smooth bidirectional movement in both live views. During a later hand manipulation, one sensor CRC error and five invalid samples were recorded while the user also moved the magnet or wiring. That is coincident disturbance, not a proven cause; the transient raw frames were no longer available for detailed diagnosis. This record does not support a fault-free claim.

Table 3. Verification evidence and its limits

Evidence	Demonstrated	Does not establish
Portable C fixtures	CRC vectors, parser bounds, wrap, speed-window recovery	Peripheral or sensor behavior
Backend fixtures	Framing, API state, shared SSE and cleanup	Physical serial reliability
Rotation fixtures	Measured-path interpolation, gaps, and 1000 rpm input	Real 1000 rpm operation
Live bench	Recorded acquisition, hand rotation, and recovery cases	System angular or speed accuracy

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

© 2026 NXP B.V. All rights reserved.

Table 4. First check when the workbench does not show valid data

Symptom	Check or action
No selectable MCU-Link port	Check the MCU-Link USB connection and enumeration. Identify the board rather than assuming COM55.
Busy or unavailable port	Close other serial monitors, then explicitly disconnect and reconnect.
INIT or sensor errors	Power off; verify 3.3 V, ground, CSN, and SPI wiring. Keep the original controller disconnected.
CRC or intermittent invalidity	Secure magnet spacing and wires. Separate sensor CRC, and transport CRC and alarm evidence.
Angle valid, RPM unavailable	Allow a new speed window after startup, invalidity, or a long sample gap.
Stale after USB or MCU reset	Restore hardware, then use Disconnect and Connect. Do not expect an automatic START.

For reproducible diagnosis, retain the source/image identity, board and sensor configuration, stimulus and counter deltas. Request the explicit engineering verifier when a durable record is needed. Do not add a normal acquisition logger merely to reproduce a fault. Preserve the bounded-memory and no-export behavior of the application.

8 Limitations and extension guidance

A 21-bit output defines digital quantization, not system angular accuracy. Magnet geometry, placement, mechanical runout, temperature, supply conditions, and calibration can affect the complete assembly. No independent angular standard or tachometer was used in the recorded hand-rotation work. Do not derive an accuracy, INL improvement or speed-error claim from resolution, smooth animation or agreement with a manually moved shaft.

Validate the 1000 rpm design boundary with a stable-speed platform and an independent reference. Inspect raw validity, sample timing, signed RPM and presentation behavior separately. Establish power and signal integrity under the intended wiring and motion. Before adding zero handling, distinguish a temporary software offset from a hardware or persistent zero change. Before NLC, establish the required motion, state transitions, EEPROM behavior, and post-power-cycle checks from the applicable supplier documentation. These are subsequent milestones, not hidden functions of INIT.

For later applications, MCX A344 provides FlexPWM and ADC resources for motor control, CAN FD for system communication and a Math Accelerator Unit (MAU) for numerical processing [8]. These resources are not enabled by this demo. Evaluate their timing, interfaces and software integration separately when extending the encoder design.

The ACH candidate must retain separate firmware, back-end and front-end responsibilities, the shared protocol, English setup instructions, and a concise verification summary. Keep the SDK external and exclude local path settings, generated caches, and unapproved vendor materials. Finalize licenses, third-party notices, branding assets, and publication metadata before publishing. Future changes must preserve explicit side-effect control: a timeout or reconnect must not silently repeat a persistent operation.

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

© 2026 NXP B.V. All rights reserved.

9 References

- [1] MEMSIC, MDA2000B Preliminary Datasheet, supplied file *MEMSIC MDA2000B Preliminary Datasheet 20260417 V2.07 B.pdf*. Primary local source for sensing architecture, SPI angle/status, CRC and register interpretation. Confirm the applicable revision with MEMSIC for a production design.
- [2] MEMSIC, MDA2000BT product information: https://www.memsic.com/cbm-87
- [3] NXP, FRDM-MCXA344 Development Board: https://www.nxp.com/design/design-center/development-boards-and-designs/FRDM-MCXA344
- [4] NXP, MCUXpresso SDK guidance: https://mcuxpresso.nxp.com/mcux-vscode/latest/Working-with-MCUXpresso-SDK.html; FRDM-MCXA344 SDK catalog: https://mcuxpresso.nxp.com/vscode/FRDM-MCXA344.json; SDK 26.09.00 ARM GCC archive: https://mcuxpresso.nxp.com/vscode/armgcc/26.09.00/FRDM-MCXA344.zip
- [5] Accompanying ACH v0.2 candidate, *README.md*, *firmware/README.md*, *backend/README.md*, and *docs/workbench-design.md*. These define the build, debug and launch procedure and the three-view workbench. A public ACH identifier is not assigned in this draft.
- [6] Accompanying ACH v0.2 candidate, *docs/protocol.md*. Version 1 UART framing, command and sample definitions, local HTTP/SSE contract, and browser-presentation rules.
- [7] Accompanying ACH v0.2 candidate, *docs/verification-summary.md*. Detailed development records and raw engineering artifacts are retained separately. Consult the recorded image identities and test conditions rather than combining results from different revisions.
- [8] NXP, MCX A343/344 data sheet, MCXAP100M180F40, Rev. 4.0, 6 July 2026: https://www.nxp.com/docs/en/data-sheet/MCXAP100M180F40.pdf

10 Appendix A Communication and status summary

The shared contract[6] is authoritative for firmware, Python, and browser consumers. UART uses 115200/8N1, ASCII and LF termination. Each line is BODY*HHHH followed by LF; HHHH is CRC-16/CCITT-FALSE over BODY, polynomial 0x1021, initial value `0xFFFF`, without reflection or final XOR. Data frames are at most 80 bytes. A command carries a protocol version, 16-bit request ID, and verb. Requests are serialized and replies echo their ID.

Table 5. Command sequence and sample interpretation

Item	Meaning
INFO → STOP → INIT → START	Connect handshake: verify protocol/settings, stop, explicitly initialize, then stream.
STOP / STATS	Stop acquisition or request counters. Statistics also arrive once per second.
D,1,seq,t_ms,RAW,AA,FF,rpm_milli	Sample body: attempted sequence, MCU time, six raw bytes, alarm, flags and signed milli-RPM.
Flags 01/02	Angle-valid/RPM-valid. Use each value only when its flag is set.
Flags 04/08/10/20	SPI error/sensor CRC error/sensor alarm/not initialized; values are hexadecimal.
E,1,BOOT	New MCU session. Invalidate an established connection and reconnect explicitly.

The back-end independently checks sensor CRC-8 in addition to line CRC-16. Sensor faults, transport faults, sequence gaps, skipped schedule slots and queue drops remain separate counters. Invalid angle and RPM become null in HTTP/SSE; they are never published as valid zero. `GET/api/ports` lists devices; `POST/api/connect and /api/disconnect` controls ownership; `GET/api/state` gives current state/history; `GET/api/events` streams updates. Multiple browser clients share these data without creating additional serial owners.

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

© 2026 NXP B.V. All rights reserved.

11 Acronyms

Table 6 lists the acronyms used in this document.

Table 6. Acronyms

Acronym	Meaning
ACH	Application Code Hub
ADC	Analog-to-Digital Converter
API	Application Programming Interface
CRC	Cyclic Redundancy Check
DUT	Device Under Test
EEPROM	Electrically Erasable Programmable Read-Only Memory
HTTP	HyperText Transfer Protocol
MAU	Math Accelerator Unit
MCU	Micro Controller Unit
NLC	Non-Linear Calibration
PC	Personal Computer
RPM	Revolutions Per Minute
SDK	Software Development Kit
SSE	Server-Sent Events
SWD	Serial Wire Debug
UART	Universal Asynchronous Receiver/Transmitter

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.

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

© 2026 NXP B.V. All rights reserved.

13 Revision history

Table 7 summarizes the revisions to this document.

Table 7. Revision history

Document ID	Release date	Description
AN15236 v.1.0	10 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.

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.

Tables

Tab. 1. MCU and development resources used by the interface 3
Tab. 2. Implemented rate and timing definitions 3
Tab. 3. Verification evidence and its limits 9
Tab. 4. First check when the workbench does not show valid data 10
Tab. 5. Command sequence and sample interpretation 11
Tab. 6. Acronyms 12
Tab. 7. Revision history 13

Figures

Fig. 1. Data and control paths in the local encoder workbench 2
Fig. 2. Division of responsibility between magnetic sensing and MCU processing 3
Fig. 3. Six-wire connection for the tested 3.3 V sensor-board configuration 4
Fig. 4. FRDM-MCXA344, MEMSIC sensor board, and manual rotor with interconnecting wiring 5
Fig. 5. Instrument view with real encoder data from the earlier two-view interface 8

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

© 2026 NXP B.V. All rights reserved.

Contents

1 Introduction 2
2 System overview and operating principle 2
2.1 MCX A344 resources in the interface 2
2.2 Magnetic angle sensing 3
2.3 Rates and timing domains 3
3 Hardware setup 4
4 Getting started 5
4.1 Prepare and build the firmware 5
4.2 Start the local workbench 6
5 Firmware implementation 6
5.1 Initialize only the required sensor function 6
5.2 Acquire and validate a sample 6
5.3 Schedule, estimate speed, and transmit 7
6 Python service and web workbench 7
6.1 Present the measured path without predicting motion 8
6.2 Handle connection state explicitly 9
7 Verification and troubleshooting 9
8 Limitations and extension guidance 10
9 References 11
10 Appendix A Communication and status summary 11
11 Acronyms 12
12 Note about the source code in the document 12
13 Revision history 13
Legal information 14

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.