Helix 520 Series Industrial Computer MCU User Manual

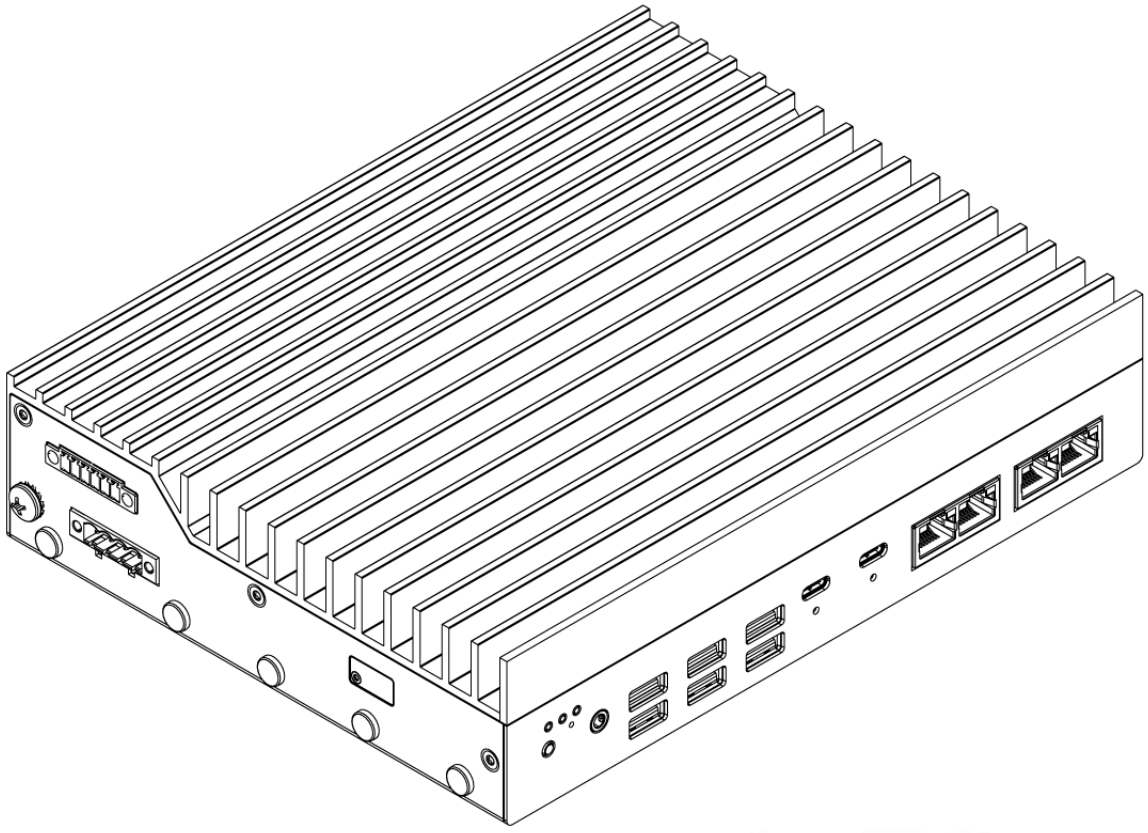


Table of Contents

1 - Revision History

Revision	Description	Date
1.0	Initial Release	05/27/2025

2 - Documents and Downloads

Description	Link
LPMCU Tool	Download
DIO Python Package	Download

3 - Feature Overview

The Helix 520 Series has an embedded power sequencing controller and supports isolated Digital Input Output (DIO) and Controller Area Network (CAN) add-in-cards. OnLogic may provide updates for the embedded sequencing controller over the product's lifetime for feature enablement or product improvements.

The DIO module has 8 input and 8 output pins, an intrusion detection pin, two contact modes, and supports firmware updates. OnLogic provides our Low Power Micro Controller Unit (LPMCU) command line tool to interact with the DIO microcontroller. Additionally, a custom Python package is provided in Section 2 to programmatically embed LPMCU tool functionality in scripting environments.

The Helix 520 Series CAN add-in-card supports a 2 channel CAN 2.0 A/B interface with configurable bitrates from 100k – 1M. It can be programmed on Linux using the socket-can interface, or on Windows via a custom C++ API. More information can be found on the CAN module in Section 5.

3.1 - MCU Firmware Update Process

The Helix 520 Series supports In-System Firmware Updates to both the DIO and power sequence microcontroller using the LPMCU tool from the download link provided in Section 2. To update the embedded sequence controller, the communications port must be enabled in the BIOS (see 3.1.1) before running either the Linux or Windows tool (3.1.2 / 3.1.3). The DIO controller VCOM port is always enabled when the add-in card is installed.

3.1.1 - Enable Communication Port in Sequence MCU

1. Power on the system and press Del a few times to access the “Front Page” menu
2. Choose “Setup Utility”
3. Navigate to Advanced > Serial Configuration
4. Locate “UART0 Controller”
5. Change it to “Communication port <COM>” to enable communication port.
6. Press F10 to Save & Exit

3.1.2 - Using Windows command prompt:

```
Shell
$ lpmcu-tool.exe -p COMx version
[yyyy-mm-ddThh:mm:ssZ INFO  lpmcu_actions] Opening COMx...
[yyyy-mm-ddThh:mm:ssZ INFO  lpmcu_actions::connection] Reading the firmware version...
0.0.2

$ lpmcu-tool.exe -p COMx flash path-to-binary/xxxx.bin
[yyyy-mm-ddThh:mm:ssZ INFO  lpmcu_actions] Opening COMx...
[yyyy-mm-ddThh:mm:ssZ INFO  lpmcu_actions] Reading update file: "path-to-binary/xxxx.bin"
[yyyy-mm-ddThh:mm:ssZ INFO  lpmcu_actions::connection] Erasing flash region
000xxxxx-000yyyyy
[yyyy-mm-ddThh:mm:ssZ INFO  lpmcu_actions::connection] Writing binary
[yyyy-mm-ddThh:mm:ssZ INFO  lpmcu_actions::connection] Requesting MCU reset at next reboot
Done! Shut down system to apply the update.
```

After completing the firmware update, the system must be shut down (reaching S5 state) in order to allow the new firmware to be loaded and executed properly.

3.1.3 - Using Linux-Ubuntu Terminal

```
Shell
// access the lpmcu-tool tool
$ chmod +x ./lpmcu-tool
// to find the ttyS number at MMIO that has baud rate 115200
$ dmesg | grep -i ttyS
$ ./lpmcu-tool -p /dev/ttySx version
$ ./lpmcu-tool -p /dev/ttySx flash xxxx.bin
```

4 - Helix 520 Series Isolated DIO Module

The optional Helix 520 Series digital input/output (DIO) add-in-card (USB-16DIO-01) adds 8 digital inputs, 8 digital outputs, and an additional intrusion (INT) pin to the system.

4.1 - DIO Pinouts and Definitions

Pin	Definition	Pin	Definition
-	ISO_GND	+	Power (VIN)/VIO+
DI0	Digital input pin 0	DO0	Digital output pin 0
DI1	Digital input pin 1	DO1	Digital output pin 1
DI2	Digital input pin 2	DO2	Digital output pin 2
DI3	Digital input pin 3	DO3	Digital output pin 3
DI4	Digital input pin 4	DO4	Digital output pin 4
DI5	Digital input pin 5	DO5	Digital output pin 5
DI6	Digital input pin 6	DO6	Digital output pin 6
DI7	Digital input pin 7	DO7	Digital output pin 7
INT	Intrusion	GND*	ISO_GND

**GND is provided as a return path for the intrusion detection switch. It is shared with the DIO - pin (ISO_GND).*

4.2 - DIO Functionality

The outputs function as open drains. The inputs are high impedance. Both DI and DO can be configured in two modes as detailed in the table below.

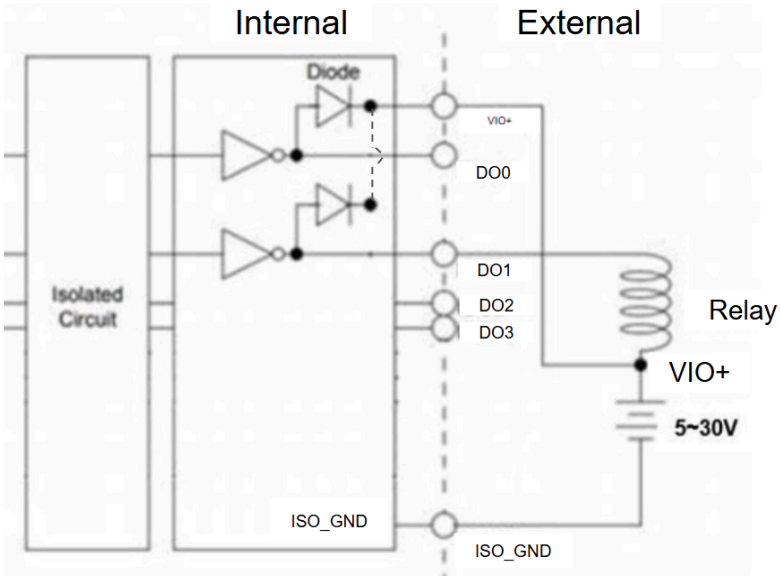
	Wet contact mode (default)	Dry contact mode

DI	logic 0 - 10 to 30VDC logic 1 - 0 to 3VDC	logic 0 - shorted to GND logic 1 - Open
DO	power is supplied externally (5 - 30V)	internal powered

4.2.1 - DO Wet Contact Mode (Suitable for Inductive Load Operation)

To function properly, pin V+ of the module should be connected to external power and ground. The high side of the load should be connected to the external power source, and the low side to the module DO pin. Load current should not exceed 150 mA. Voltage ranges should be 5 V to 30 V.

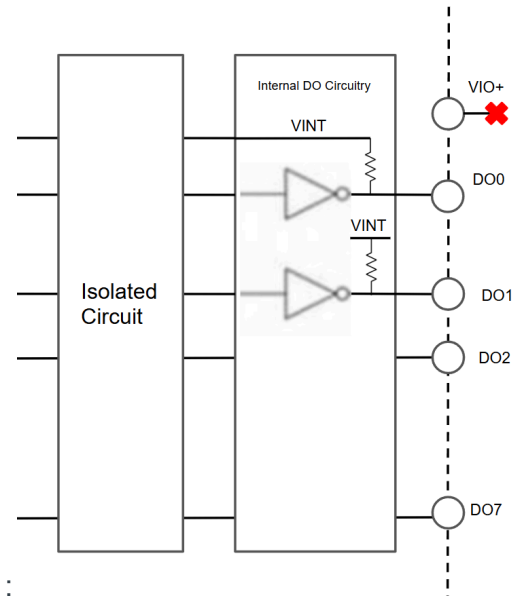
Setup required for Output:



4.2.2 - DO Dry-Contact Mode

Voltage is provided by the system. Each DO will output 11 V - 12.6 V when active.

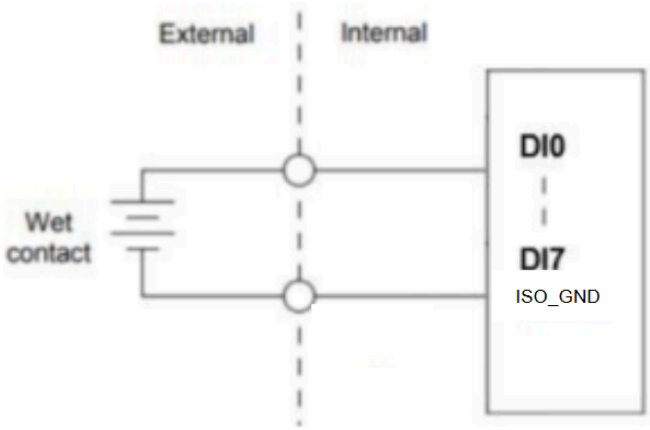
Setup required for Output:



4.2.3 - DI Wet Contact Mode

There is no internal pull up to the DI[0:7] pins when set to WET mode. Externally supplied 5 - 30V is recognized as logic 0 and 0 - 3V as logic 1 when DI[0:7] pins are set to wet contact mode.

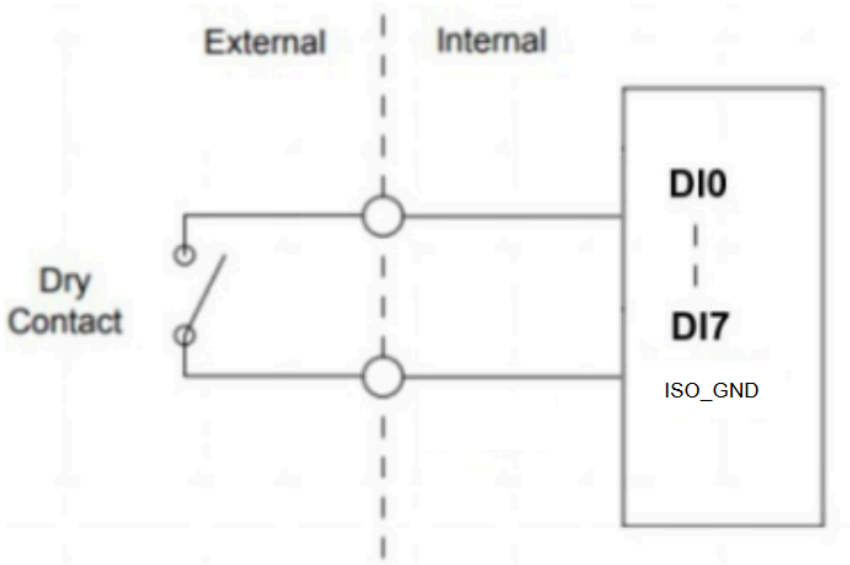
Setup required for Input:



4.2.4 - DI Dry Contact Mode

When the contact type is set to DRY mode, DI[0:7] are pulled up to the internal isolated ~12V supply. An open/floating connection is recognized as logic 0 and a short to GND as logic 1 when DI[0:7] pins are set to dry contact mode.

Setup required for Input:



4.3 - Device Usage

The DIO card uses the USB-CDC communication protocol. On Windows, it will show up as “USB Serial Device (COMx)” in the device manager. On Linux, it will show up as “/dev/ttyACMx” in the serial device list.

4.3.1 - DIO Programming

The pin states and contact types of DIO add-in-card can be controlled and read from using the LPMCU and Python API.

Command Summary of LPMCU

Command	Description	Parameters	Returns
get-di	Read digital input pin state	Pin val (0-7)	(false:logic 0, true: logic 1)
get-do	Read digital output pin state	Pin val (0-7)	(false:logic 0, true: logic 1)
set-do	Set digital output pin state	Pin val (0-7) | state (false:low, true:high)	
set-di-contact	Set digital input contact type*	(false:wet, true:dry)	
set-do-contact	Set digital output contact type*	(false:wet, true:dry)	
get-di-contact	Read digital input contact type*		(false:wet, true:dry)
get-do-contact	Read digital output contact type*		(false:wet, true:dry)

**See section 4.2 for mode definitions.*

From Windows Command Prompt:

```shell
// Set digital output contact type as dry
lpmcu-tool.exe -p COMx set-do-contact true

// Set digital output 0
lpmcu-tool.exe -p COMx set-do 0 true

// Clear digital output 0
lpmcu-tool.exe -p COMx set-do 0 false

// Read the state of digital input 0
lpmcu-tool.exe -p COMx get-di 0
```

Using Ubuntu Terminal:

```shell
// Access the lpmcu-tool tool
$ chmod +x ./lpmcu-tool

// Read the state of digital output 0
$ ./lpmcu-tool -p /dev/ttyACMx get-do 0
```

Python API Link:

Example code, setup instructions, and API specific documentation for the Python DIO utility are available below:

Full Link: https://github.com/onlogic/onlogic-m031-manager

4.4 - Intrusion Detection

The Helix 520 Series provides an intrusion detection feature which is disabled by default. The DIO `INT` signal is assigned for intrusion detection. The intrusion detection feature must be enabled from the BIOS setup menu.

4.4.1 - Enable Intrusion Detection Steps in BIOS

1. Navigate to the BIOS **Setup Utility Menu**.
2. Select **Advanced**.
3. Choose **OnLogic Feature Configuration**.
4. Go to **Intrusion Detect**.
5. Set to **Enabled**.

4.4.2 - Intrusion Alert Mode

Power Button Emulation (SCI#)

When an intrusion event is detected in the Operating System environment by shorting the INT pin to ground, a power button press is triggered. The resulting action can be a shutdown, hibernation, sleep or nothing, depending on the OS configuration for power button events.

Perform Power Cycle (SMI#)

As soon as an intrusion event is detected, the system will shut down immediately.

4.5 - DIO Firmware Update

The Helix 520 Series supports In-System Firmware Updates for the DIO add-in-card using the lpmcu-tool which can be accessed within the link provided in Section 2. The commands are shown below with example outputs:

4.5.1 - Using Windows Command Prompt

```
Shell
$ lpmcu-tool.exe -p COMx version
[yyyy-mm-ddThh:mm:ssZ INFO  lpmcu_actions] Opening COMx...
[yyyy-mm-ddThh:mm:ssZ INFO  lpmcu_actions::connection] Reading the firmware version...
0.0.2

$ lpmcu-tool.exe -p COMx flash path-to-binary/xxxx.bin
[yyyy-mm-ddThh:mm:ssZ INFO  lpmcu_actions] Opening COMx...
[yyyy-mm-ddThh:mm:ssZ INFO  lpmcu_actions] Reading update file: "path-to-binary/xxxx.bin"
[yyyy-mm-ddThh:mm:ssZ INFO  lpmcu_actions::connection] Erasing flash region
000xxxxx-000yyyyy
[yyyy-mm-ddThh:mm:ssZ INFO  lpmcu_actions::connection] Writing binary
[yyyy-mm-ddThh:mm:ssZ INFO  lpmcu_actions::connection] Requesting MCU reset at next reboot
Done! Shut down system to apply the update.
```

4.5.2 - Using Ubuntu-Linux Terminal

```
Shell
// access the lpmcu-tool tool
```

```
$ chmod +x ./lpmcu-tool
$ dmesg | grep -i ttyACM
$ ./lpmcu-tool -p /dev/ttyACMx version
$ ./lpmcu-tool -p /dev/ttyACMx flash xxxx.bin
```

After updating the firmware, an AC power cycle is required to allow the new firmware to be loaded and executed properly.

5 - Helix 520 CAN Module

5.1 - CAN-Bus Overview

The optional Helix 520 CAN (Controller Area Network) add-in-card (USB-02CAN-01) provides a two channel 2.0 A/B interface. The CAN bus consists of two main signal lines: CAN High and CAN Low. CAN High is biased at a high voltage potential of ~3.5 Volts and CAN Low is biased at a low voltage potential of ~1.5 Volts. A nominal voltage measured between the two signal lines will default to ~2.5 Volts, serving as a reliable indicator of the Helix 520 Series CAN bus operational status. Additionally, the CAN bus operates in two states: dominant and recessive. The dominant state is represented by logic level 0, while the recessive state is represented by logic level 1. The CAN interface supports configurable bitrates from 10k – 1M.

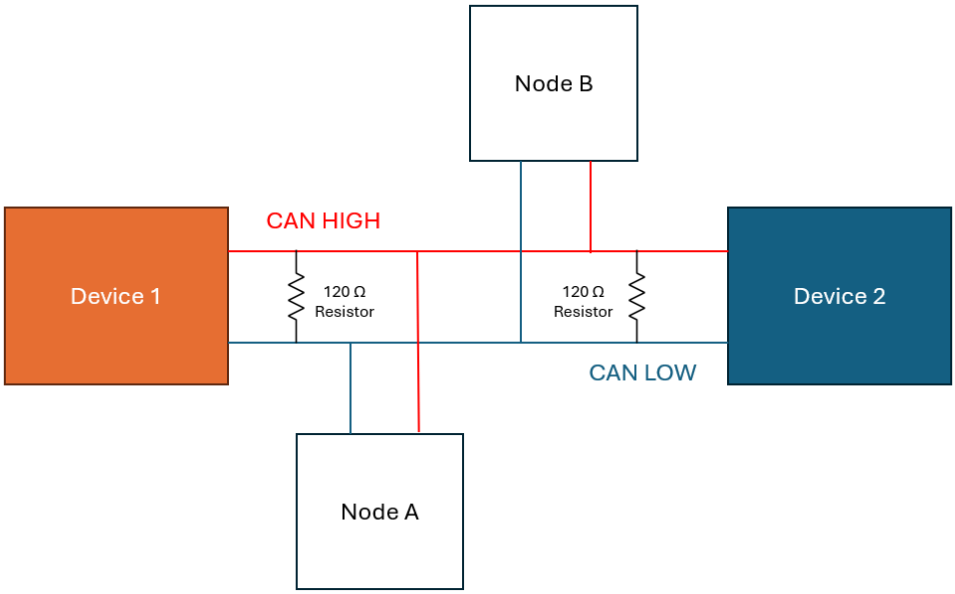


Diagram of a simplified CAN bus network.

The figure above shows: 1) One termination resistor at each end of the bus, 2) required endpoints of the bus acting as either transmitters or receivers, 3) High and Low CAN bus lines, and 4) additional (optional) network members connected on the same signal lines.

Further details about the CAN bus and its operation can be found on the OnLogic technical resource page: https://support.onlogic.com/hx520-series-technical-resources.

5.2 - Driver Installation and Program Environment Setup

GitHub Repository Location: Driver installation instructions, program environment setup instructions, and example source code are found in this GitHub repository.

Full Link: https://github.com/onlogic/onlogic-f81604n-utilities

Linux: The driver for the CAN add-on-card is included in the Linux kernel and should be automatically present on Linux Kernel 2.6.2+. If not, refer to the Github README in the link above for instructions on installation and usage.

Windows: Helix 520 Series systems purchased with Windows pre-installed ship with the drivers pre-installed as well. For post-sale Windows deployments, please refer to the README in the link above.