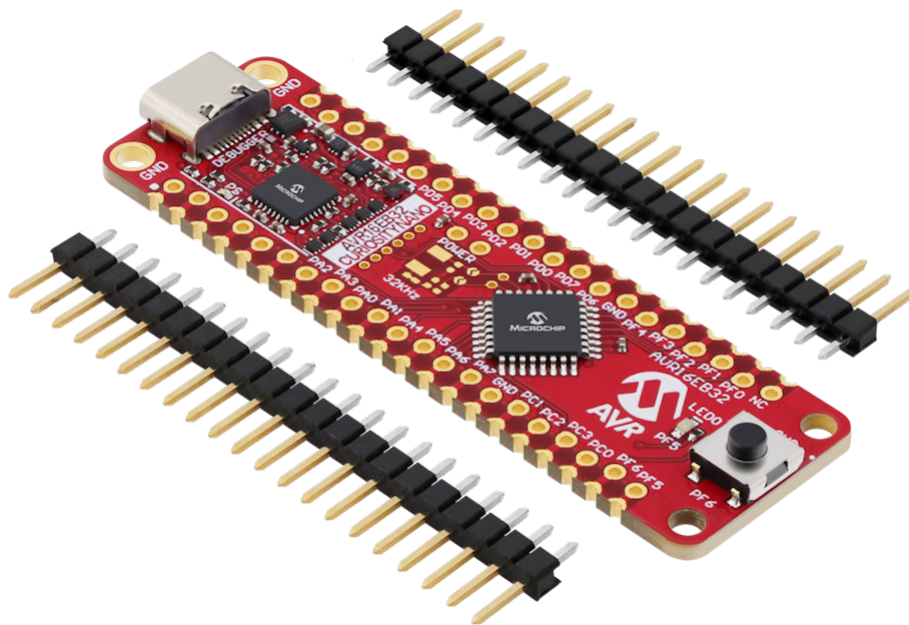**MICROCHIP**

Preface

The AVR16EB32 Curiosity Nano evaluation kit (EV73J36A) is a hardware platform to evaluate the AVR® EB family microcontrollers. This board has the AVR16EB32 microcontroller (MCU) mounted.

The Curiosity Nano series of evaluation boards include an on-board debugger. No external tools are necessary to program and debug the AVR16EB32.

1. **AVR16EB32 Curiosity Nano website** - Kit information, latest user guide, and design documentation.
2. **AVR16EB32 website** - Find documentation, data sheets, sample, and purchase microcontrollers.
3. **Code examples on MPLAB® Discover** - Get started with code examples.
4. **AVR16EB32 Curiosity Nano on Microchip Direct** - Purchase this kit on Microchip Direct.
5. **AVR16EB32 Curiosity Nano Schematics** - Board schematics and history.
6. **AVR16EB32 Curiosity Nano Altium Project** - Latest project revision.
7. **AVR16EB32 Curiosity Nano Design Documentation** - Production files for every revision.

Table of Contents

1. Features and Pinout

Features of MCU and Curiosity Nano, Board Layout Picture, Board Block Diagram, Pinout Diagram

1.1 AVR16EB32 Key Features

The AVR® EB family integrates real-time control and drive functionality into the AVR's functionality. Featuring the latest Core Independent Peripherals (CIPs), these microcontrollers (MCUs) excel as stand-alone processors that will be valuable additions to any modern embedded design.

- Up to 32 KB in-system-programmable Flash memory with a true read-while-write operation
- Programming and Debug Interface Disable (PDID) Security Functionality
- One 16-bit Timer/Counter type E (TCE) with four compare channels for PWM generation and Waveform Extension (WEX)
- One 24-bit Timer/Counter type F (TCF) for frequency generation
- One 16-bit Real-Time Counter (RTC) that can run from an external crystal or internal oscillator
- One 12-bit, 300 ksps, differential Analog-to-Digital Converter (ADC) with a programmable gain amplifier (PGA)
- Two Analog Comparators (ACs) with window compare functions
- Configurable Custom Logic (CCL) with up to four programmable Look-up Tables (LUTs)
- Event System for CPU-independent and predictable inter-peripheral signaling
- Configurable, internally generated reference voltage

1.2 Board Features

- AVR16EB32 Microcontroller
- USB Type-C® Connector
- One Yellow User LED
- One Mechanical User Switch
- One green power and status LED
- Footprint for 32.768 kHz Crystal
- On-Board Debugger support in Microchip MPLAB® X IDE and Microchip Studio:
    - Board identification
    - Virtual serial port (CDC)
    - Programming and debugging
    - Two debug GPIO channels (DGI GPIO)
- USB Powered
- Adjustable Target Voltage:
    - MIC5353 LDO regulator controlled by the on-board debugger
    - 1.8–5.1V output voltage (limited by USB input voltage)
    - 500 mA maximum output current (limited by ambient temperature and output voltage)

1.3 Board Overview

Figure 1-1. AVR16EB32 Curiosity Nano Board Overview
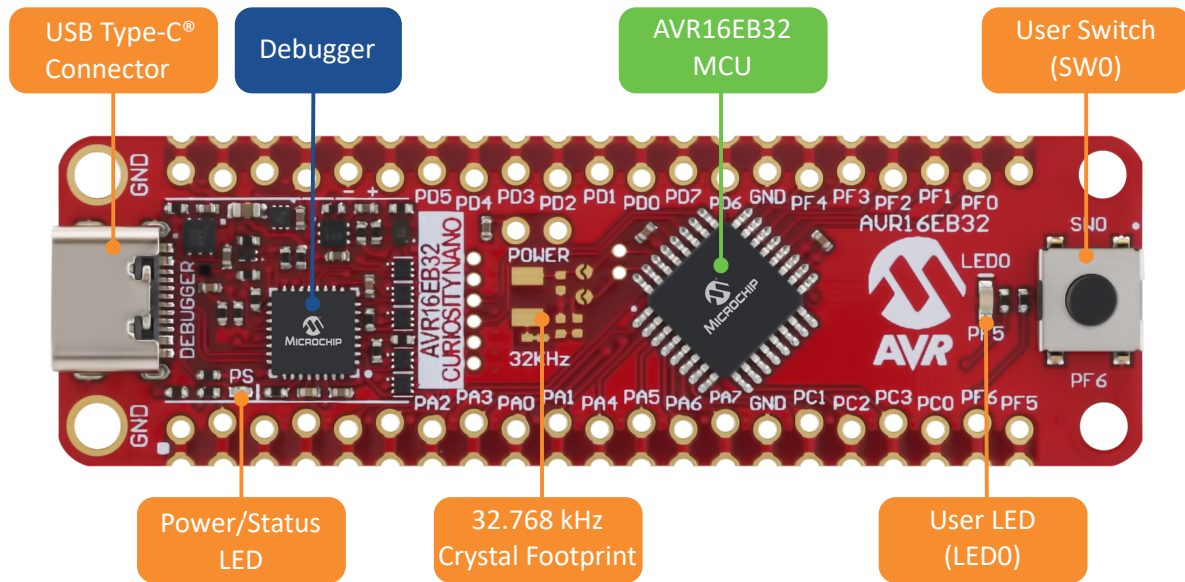
1.4 Block Diagram

Figure 1-2. AVR16EB32 Curiosity Nano Board Block Diagram
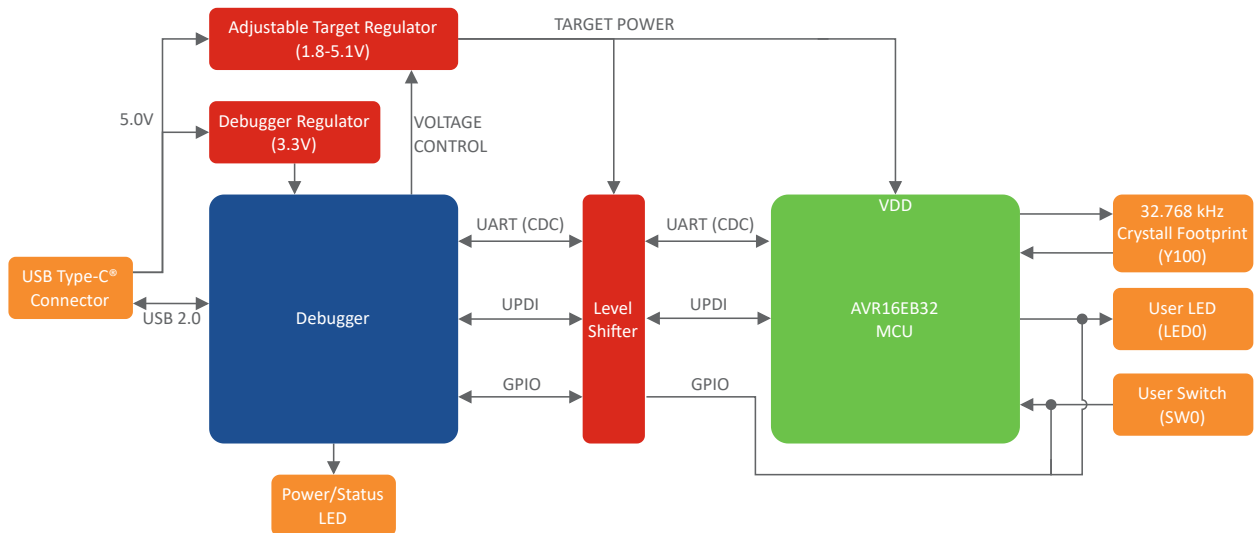
1.5 Pinout

All the AVR16EB32 I/O pins are accessible at the edge connectors on the board. The image below shows the board pinout. Refer to the I/O Multiplexing and Considerations section in the AVR16EB32 data sheet for all available functions on each pin.

Figure 1-3. AVR16EB32 Curiosity Nano Pinout
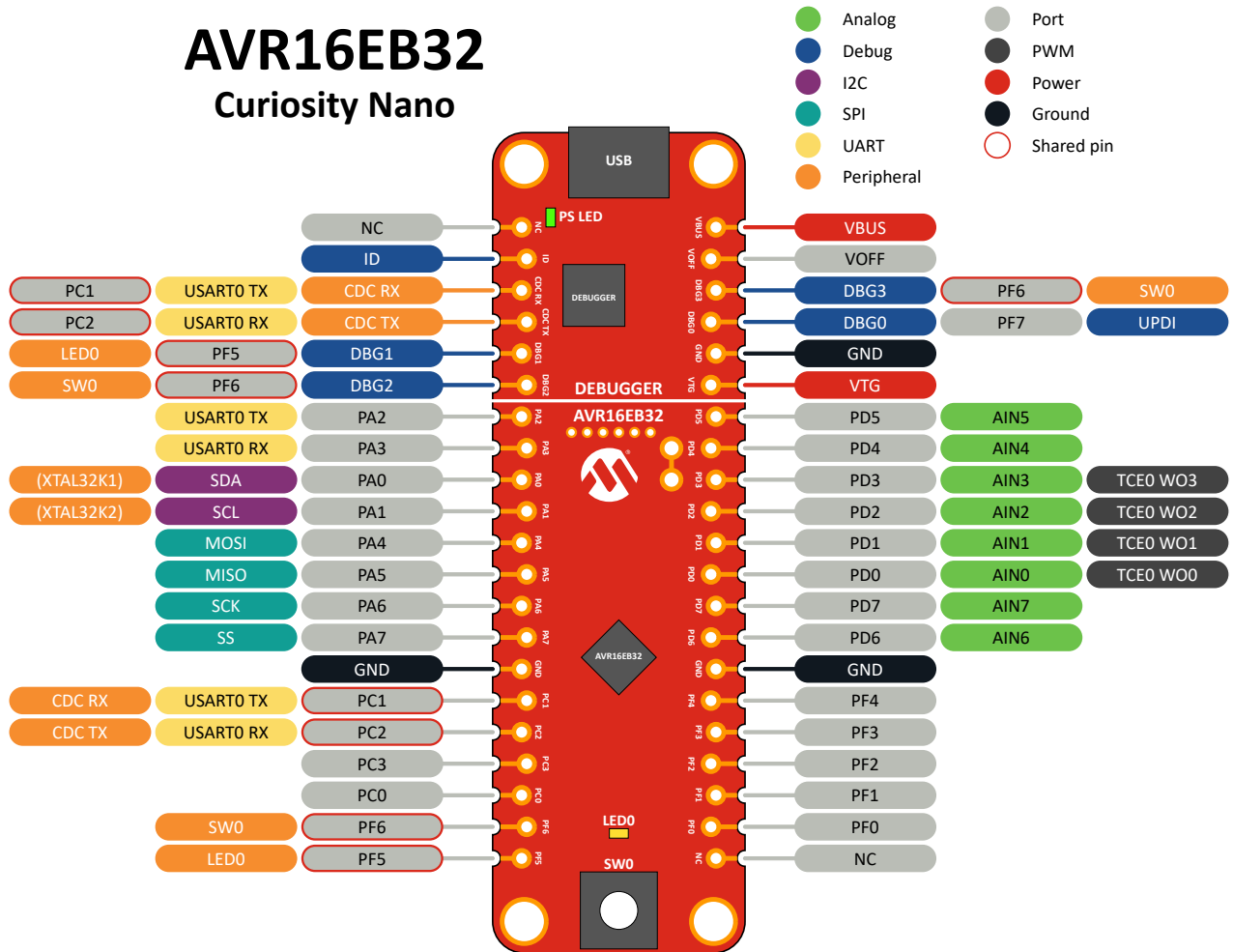
2. Getting Started

Getting started resources for the AVR16EB32 Curiosity Nano board in the MPLAB Tools Ecosystem.

2.1 Getting Started Now with AVR®

Are you new to using AVR microcontrollers (MCUs)? Our comprehensive MPLAB® development ecosystem supports Microchip MCUs, which makes it easy to get your prototype up and running and includes production-ready code generation tools. Try the interactive guide to find the necessary resources: **Get Started Now with AVR MCUs**.

1. **Examples: MPLAB Discover** is a tool to help you find Microchip example projects. It offers several ways to filter projects to efficiently find a Microchip-tested example, as close as possible to your application requirements, to use as a starting point for your development.
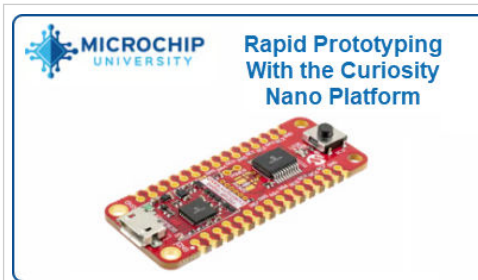
2. **Configure: MCC Melody** provides Libraries, Drivers, and Peripheral Libraries (PLIB) for the development of embedded software for a range of supported Microchip MCUs, which are configured using MPLAB® Code Configurator (MCC).

3. **Develop: MPLAB X IDE** (Integrated Development Environment), which is available for Windows, Linux and macOS, also MPLAB® XC C Compilers and GCC Compiler for AVR®.

4. **Debug:** With the MPLAB X IDE and/or the **MPLAB Data Visualizer**, the AVR16EB32 device on the AVR16EB32 Curiosity Nano board is programmed and debugged by the on-board debugger. Therefore, no external programmer or debugger tool is required.

5. **Boards:** The AVR16EB32 Curiosity Nano board is a supported part of the **Curiosity Nano Development Platform**, which includes the Curiosity Nano Base for Click Boards™.

> 💡 **Tip:** Have a look at this free Microchip University course.
>
>
>
> The Curiosity Nano development platform is well suited to rapid prototyping. This course will introduce you to the defining features of the Curiosity Nano platform and show you how it can assist with using a new microcontroller or developing a new prototype. This course includes an in-depth look at the on-board debugger, which is central to the platform, and how to use its various user interfaces to help you reach your goals quicker.

2.2 How the Curiosity Nano Fits Into the MPLAB Tools Ecosystem

2.2.1 MPLAB X IDE Support for Curiosity Nano

When the board connects to the computer for the first time, the operating system will install the driver software. The drivers for the board are included with MPLAB® X IDE. Once this is done, when connecting the Curiosity Nano to a host PC via USB, if MPLAB X IDE is open, a Kit Window is opened with several key links for that Curiosity Nano.

When creating a new project, the part number on the Curiosity Nano will be detected, as will the debug tool.
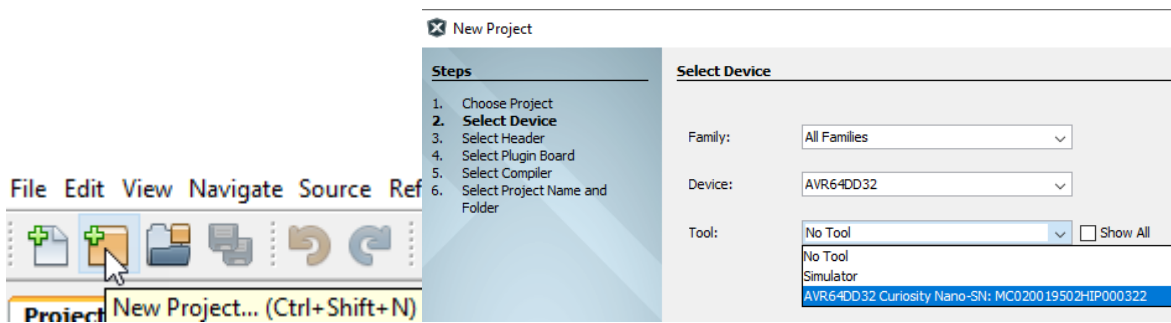
Tip:
- For the AVR16EB32 Curiosity Nano board, use MPLAB® X version 6.15, device family pack AVR-Ex_DFP version 2.5.176, and tool pack nEDBG_TP version 1.13.715 or newer
- The latest device family packs are available through **Tools** > **Packs** in MPLAB® X IDE or online at Microchip MPLAB® X Packs Repository. For more information on packs and how to upgrade them, refer to the MPLAB® X IDE User's guide - Work with Device Packs.
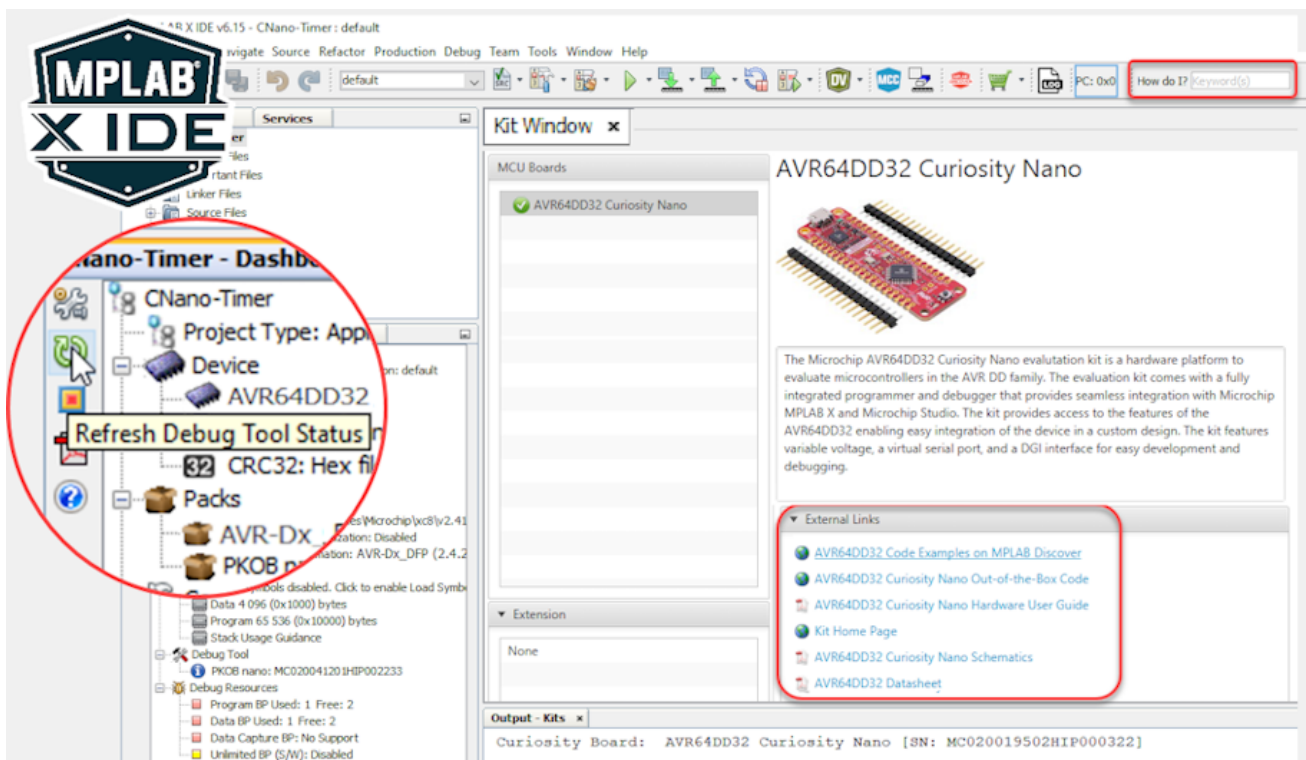
As shown in the image below, additional information about your Curiosity Nano can be seen in the **"Debug Tool"** window once you click **"Refresh Debug Tool Status"**.

Tip:
- If closed, reopen the Kit Window in MPLAB® X IDE through the menu bar **Window** > **Kit Window**
- The 'How do I?' search bar often gives excellent results if you're new to MPLAB X IDE
- **'Debug main project"** will start a debug session
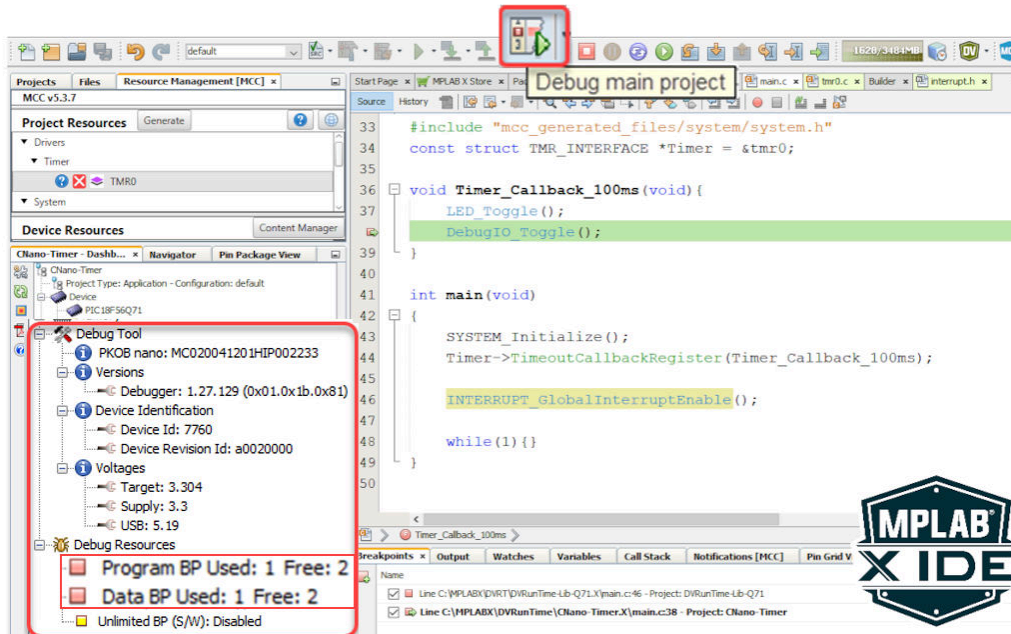
Tip:
- After clicking on 'Refresh Debug Tool Status,' you can see information such as MCU Target Voltage
- What are Program- and Data-Break Points? MPLAB X IDE Advanced Debugging - Breakpoints Demo
- Reference for example above - MCC Melody Timer0 Driver: 100 ms Timer, API Ref Code

2.3 MPLAB Data Visualizer Support for Curiosity Nano

The Curiosity Nano, via a USB/serial bridge, facilitates a connection between a UART on the Target MCU and your computer's COM port. For example, you may use this to connect to the MPLAB Data Visualizer or other terminal programs.
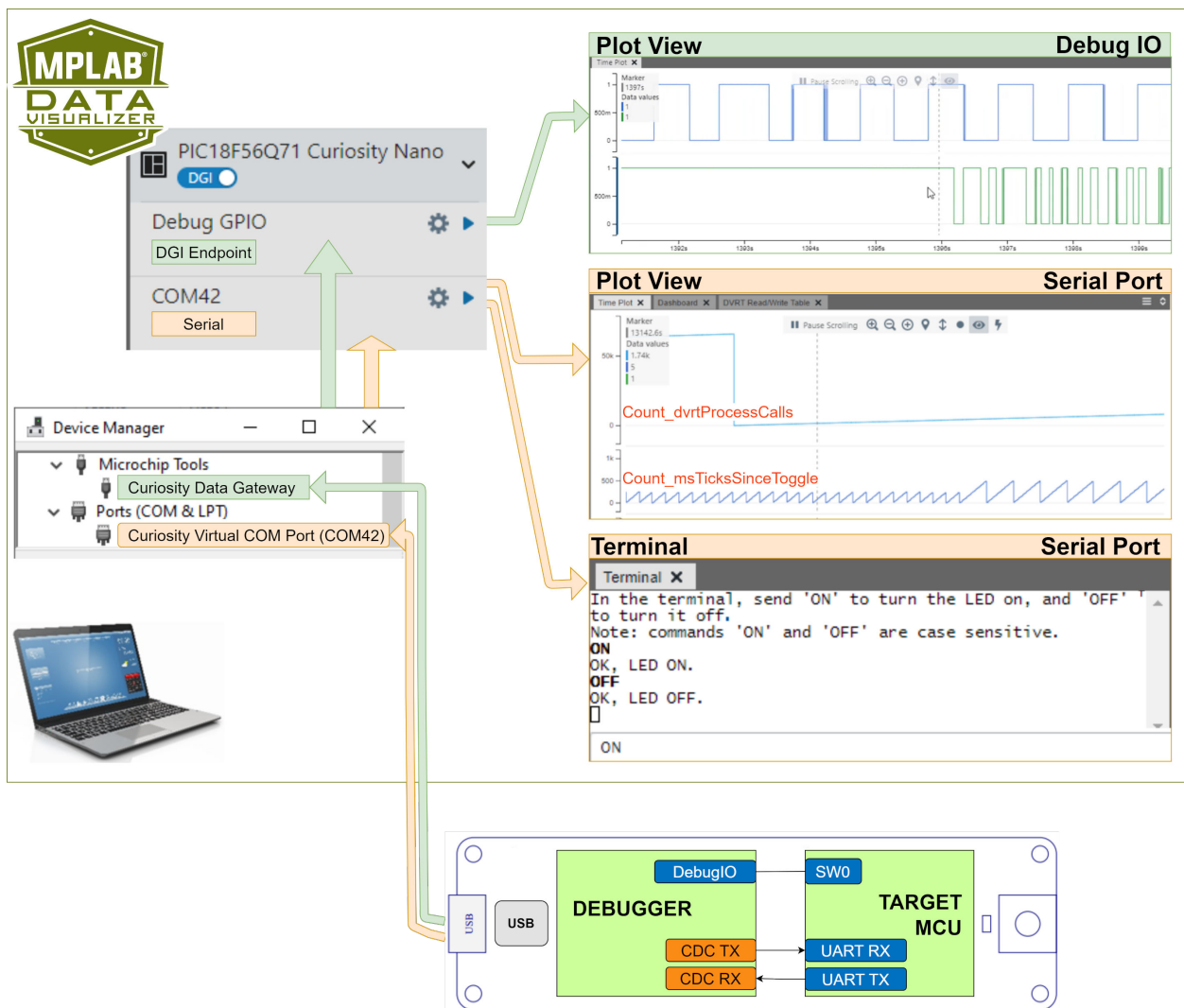
> **Tip:** References for the example Data Visualizer Plot- and Terminal-views above:
> 1. Plot View - DebugIO: DebugIO Hello World (Microchip University).
> 2. Plot View - Serial Port: MCC Melody Use Case Data Visualizer Run Time Use Case 1.
> 3. Terminal - Serial Port: MCC Melody UART Driver: LED Control Commands.

2.4 Using Pin-Headers

The edge connector footprint on the AVR16EB32 Curiosity Nano has a staggered design where each hole is shifted 8 mil (~0.2 mm) off-center. The hole shift allows regular 100 mil pin headers to be used without soldering on the board. The pin-headers can be used in applications like pin sockets and prototyping boards without issues once they are firmly in place.

Figure 2-1. Attaching Pin-Headers to the Curiosity Nano Board

Figure 2-2. Connecting to Curiosity Nano Base for Click boards™
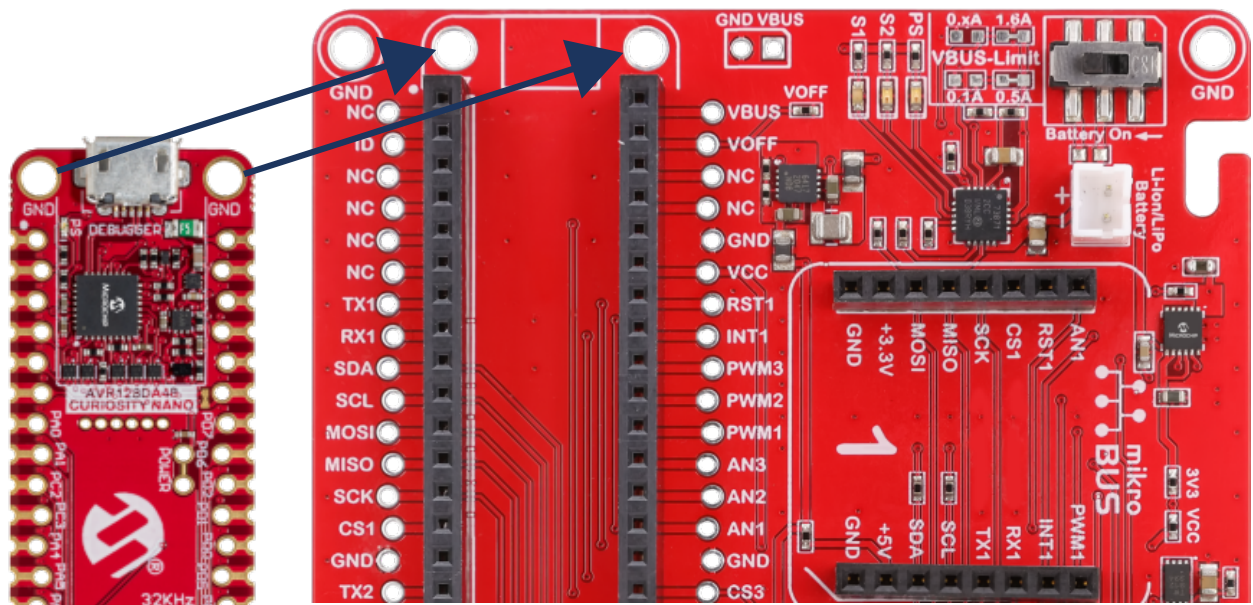
> 💡 **Tip:**
> - Start at one end of the pin-header and gradually insert the header along the length of the board. Once all the pins are in place, use a flat surface to push them in.
>   - For applications using the pin-headers permanently, it is still recommended to solder them in place
>   - Once the pin-headers are in place, they are hard to remove by hand. Use a set of pliers and carefully remove the pin-headers to avoid damage to the pin-headers and PCB.

3. On-Board Debugger

Features and interfaces of the on-board debugger for programming and debugging.

3.1 On-Board Debugger Overview

AVR16EB32 Curiosity Nano contains an on-board debugger for programming and debugging. The on-board debugger is a composite USB device consisting of several interfaces:

- A debugger that can program and debug the AVR16EB32 in MPLAB® X IDE
- A virtual serial port (CDC) that is connected to a Universal Asynchronous Receiver/Transmitter (UART) on the AVR16EB32 and provides an easy way to communicate with the target application through terminal software
- A mass storage device that allows drag-and-drop programming of the AVR16EB32
- A Data Gateway Interface (DGI) for code instrumentation with logic analyzer channels (debug GPIO) to visualize program flow

The on-board debugger controls a Power and Status LED (marked PS) on the AVR16EB32 Curiosity Nano board. The table below shows how the different operation modes control the LED.

Table 3-1. On-Board Debugger LED Control

Operation Mode	Power and Status LED
Boot Loader mode	The LED blinks slowly during power-up
Power-up	The LED is ON
Normal operation	The LED is ON
Programming	Activity indicator: The LED blinks slowly during programming/debugging
Drag-and-drop programming	**Success:** The LED blinks slowly for 2 sec. **Failure:** The LED blinks rapidly for 2 sec.
Fault	The LED blinks rapidly if a power fault is detected
Off	The on-board debugger is powered down, the LED is OFF

> **Info:** Slow blinking is approximately 1 Hz, and rapid blinking is about 5 Hz.

3.2 On-Board Debugger Connections

The table below shows the connections between the target and the debugger section. All the connections between the target and the debugger are tri-stated when the debugger is not using the interface. Hence, there are few contaminations of the signals, e.g., the pins can be configured to anything the user wants.

> **Info:** The 12-edge connections closest to the USB connector on Curiosity Nano boards have a standardized pinout. The program/debug pins have different functions depending on the target programming interface.

Table 3-2. On-Board Debugger Connections

Debugger Pin	AVR16EB32 Pin		Description
CDC **TX**	PC2	UART **RX**	USB CDC TX line
CDC **RX**	PC1	UART **TX**	USB CDC RX line
DBG0	PF7	ICSPDAT	Debug data line
DBG1	PF5	ICSPCLK	Debug clock line

..........continued

Debugger Pin	AVR16EB32 Pin		Description
DBG2	PF6	SW0/GPIO0	Debug GPIO0/SW0
DBG3	PF6	MCLR	Reset line
ID	—	—	ID line for extensions
NC	—	—	No connect
V_{BUS}	—	—	V_{BUS} voltage for external use
VOFF	—	—	Voltage Off input. Disables the target regulator and target voltage when pulled low.
VTG	—	—	Target voltage
GND	—	—	Common ground

Figure 3-1. Curiosity Nano Debugger Pinout
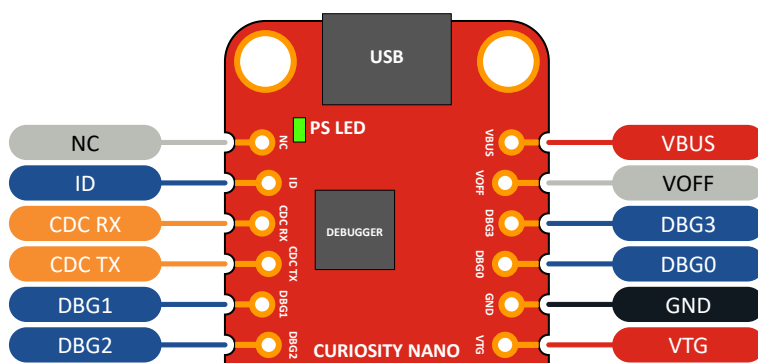
> **Tip:** For the complete AVR16EB32 Curiosity Nano pinout, see the AVR16EB32 Curiosity Nano Pinout.

3.3 Debugger USB Enumeration

The on-board debugger on the AVR16EB32 Curiosity Nano board appears as a Human Interface Device (HID) on the host computer's USB subsystem. The debugger supports full-featured programming and debugging of the AVR16EB32 using MPLAB X IDE and some third-party IDEs.

> **Remember:** Keep the debugger's firmware up-to-date. Firmware upgrades automatically when using MPLAB X IDE.

3.4 Virtual Serial Port (CDC)

The virtual serial port (CDC) is a general purpose serial bridge between a host PC and a target device.

3.4.1 Overview

The on-board debugger implements a composite USB device with a standard Communications Device Class (CDC) interface, which appears on the host as a virtual serial port. Use the CDC to stream arbitrary data between the host computer and the target in both directions: All characters sent through the virtual serial port on the host computer will be transmitted as UART on the debugger's CDC TX pin. The UART characters captured on the debugger's CDC RX pin will be returned to the host computer through the virtual serial port.

Figure 3-2. CDC Connection
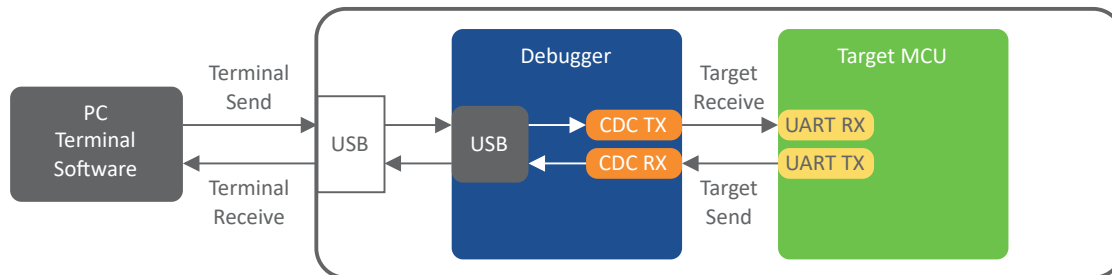
> **Info:** The debugger's CDC TX pin is connected to a UART RX pin on the target for receiving characters from the host computer, as shown in the figure above. Similarly, the debugger's CDC RX pin is connected to a UART TX pin on the target for transmitting characters to the host computer.

3.4.2   Operating System Support

On Windows® machines, the CDC will enumerate as *Curiosity Virtual COM Port* and appear in the Ports section of the Windows Device Manager. The COM port number can also be found there.

> **Info:** On older Windows systems, the CDC requires a USB driver. The MPLAB X IDE installation include this driver.

On Linux® machines, the CDC will enumerate and appear as `/dev/ttyACM#`.

> **Info:** tty* devices belong to the "dialout" group in Linux, so it may be necessary to become a member of that group to have permission to access the CDC.

On Mac® machines, the CDC will enumerate and appear as `/dev/tty.usbmodem#`. Depending on the terminal program used, it will appear in the available list of modems as `usbmodem#`.

> **Info:** For all operating systems, use a terminal emulator that supports DTR signaling. See Signaling.

3.4.3   Limitations

Not all UART features are implemented in the on-board debugger CDC. The constraints are outlined here:

- **Baud rate**: Must be in the range of 1200 bps to 500 kbps. Any baud rate outside this range will be set to the closest limit without warning. Baud rate can be changed on-the-fly.
- **Character format**: Only 8-bit characters are supported.
- **Parity**: Can be odd, even, or none.
- **Hardware flow control**: Not supported.

- **Stop bits**: One or two bits are supported.

3.4.4 Signaling

During USB enumeration, the host OS will start both the communication and data pipes of the CDC interface. At this point, it is possible to set and read back the baud rate and other UART parameters of the CDC, but data sending and receiving will not be enabled.

The terminal must assert the DTR signal when it connects to the host. As this is a virtual control signal implemented on the USB interface, it is not physically present on the board. Asserting the DTR signal from the host will indicate to the on-board debugger that a CDC session is active. The debugger will enable its level shifters (if available) and start the CDC data send and receive mechanisms.

Deasserting DTR in debugger firmware version 1.20 or earlier has the following behavior:
- Debugger UART receiver is disabled, and no further data will be transferred to the host computer
- Debugger UART transmitter will continue to send queued data ready for transfer, but no new data is accepted from the host computer
- Level shifters (if available) are not disabled, and the debugger CDC TX line remains driven

Deasserting DTR in debugger firmware version 1.21 or later has the following behavior:
- Debugger UART receiver is disabled, and no further data will be transferred to the host computer
- Debugger UART transmitter will continue to send queued data ready for transfer, but no new data is accepted from the host computer
- Once the ongoing transmission is complete, level shifters (if available) are disabled, and the debugger CDC TX line will become high-impedance

> **Remember:** Set up the terminal emulator to assert the DTR signal. Without the signal, the on-board debugger will not send or receive data through its UART.

> **Tip:** The on-board debugger's CDC TX pin will not be driven until the CDC interface is enabled by the host computer. Also, there are no external pull-up resistors on the CDC lines connecting the debugger and the target, meaning the lines are floating during power-up. The target device may enable the internal pull-up resistor on the pin connected to the debugger's CDC TX pin to avoid glitches resulting in unpredictable behavior like framing errors.

3.4.5 Advanced Use

CDC Override Mode

In ordinary operation, the on-board debugger is a true UART bridge between the host and the device. However, in certain use cases, the on-board debugger can override the basic Operating mode and use the CDC TX and RX pins for other purposes.

Dropping a text file into the on-board debugger's mass storage drive can be used to send characters out of the debugger's CDC TX pin. The filename and extension are trivial, but the text file will start with the characters:

```
CMD:SEND_UART=
```

Debugger firmware version 1.20 or earlier has the following limitations:
- The maximum message length is 50 characters – all remaining data in the frame are ignored

- The default baud rate used in this mode is 9600 bps, but if the CDC is already active or configured, the previously used baud rate still applies

Debugger firmware version 1.21 and later has the following limitations/features:

- The maximum message length will vary depending on the MSC/SCSI layer timeouts on the host computer and/or operating system. A single SCSI frame of 512 bytes (498 characters of payload) is ensured, and files up to 4 KB will work on most systems. The transfer will complete on the first NULL character encountered in the file.
- The baud rate used is always 9600 bps for the default command:

```
CMD:SEND_UART=
```

- Do not use the CDC Override mode simultaneously with data transfer over the CDC/terminal. If a CDC terminal session is active when receiving a file via the CDC Override mode, it will be suspended for the duration of the operation and resumed once complete.
- Additional commands are supported with explicit baud rates:

```
CMD:SEND_9600=
```

```
CMD:SEND_115200=
```

```
CMD:SEND_460800=
```

USB-Level Framing Considerations

Sending data from the host to the CDC can be done byte-wise or in blocks, chunked into 64-byte USB frames. Each frame will be queued for transfer to the debugger's CDC TX pin. Sending a small amount of data per frame can be inefficient, particularly at low baud rates, as the on-board debugger buffers frames but not bytes. A maximum of four 64-byte frames can be active at any time. The on-board debugger will throttle the incoming frames accordingly. Sending full 64-byte frames containing data is the most efficient method.

When receiving data on the debugger's CDC RX pin, the on-board debugger will queue up the incoming bytes into 64-byte frames, which are sent to the USB queue for transmission to the host when they are full. Incomplete frames are also pushed to the USB queue at approximately 100 ms intervals, triggered by USB start-of-frame tokens. Up to eight 64-byte frames can be active at any time.

An overrun will occur if the host (or the software running on it) fails to receive data fast enough. When this happens, the last-filled buffer frame recycles instead of being sent to the USB queue, and a complete data frame will be lost. To prevent this occurrence, the user will ensure that the CDC data pipe is continuously read, or the incoming data rate will be reduced.

Sending Break Characters

The host can send a UART break character to the device using the CDC, which can be useable for resetting a receiver state-machine or signaling an exception condition from the host to the application running on the device.

A break character is defined as a sequence of at least 11 zero bits transmitted from the host to the device.

Not all UART receivers have support for detecting a break, but a correctly-formed break character usually triggers a framing error on the receiver.

Sending a break character using the debugger's CDC has the following limitations:

- Sending a break must NOT be done simultaneously, as using the CDC Override mode (drag-and-drop). Both these functions are temporary states and must be used independently.
- Sending a break will cause the data currently being sent to be lost. Be sure to wait a sufficient amount of time to allow all characters in the transmission buffer to be sent (see above section)

Microchip

before sending the break, which is also in line with expected break character usage: For example, reset a receiver state-machine after a timeout occurs waiting for returning data to the host.

- The CDC specification allows for debugger-timed breaks of up to 65534 ms in duration to be requested. For simplicity, the debugger will limit the break duration to a maximum of 11 bit-durations at its minimum supported baud rate.
- The CDC specification allows for indefinite host-timed breaks. It is the responsibility of the terminal application/user to release the break state in this case.

Note: Sending break characters is available in debugger firmware version 1.24 and later.

3.5 Mass Storage Device

The on-board debugger includes a simple Mass Storage Device implementation, which is accessible for read/write operations via the host operating system to which it is connected.

It provides:
- Read access to basic text and HTML files for detailed kit information and support
- Write access for programming Intel® HEX formatted files into the target device's memory
- Write access for simple text files for utility purposes

3.5.1 Mass Storage Device Implementation

The on-board debugger implements a highly optimized variant of the FAT12 file system with several limitations, partly due to the nature of FAT12 itself and optimizations made to fulfill its purpose for its embedded application.

The Curiosity Nano USB device is USB Chapter 9-compliant as a mass storage device but does not, in any way, fulfill the expectations of a general purpose mass storage device. This behavior is intentional.

When using the Windows operating system, the on-board debugger enumerates as a Curiosity Nano USB Device found in the disk drives section of the device manager. The CURIOSITY drive appears in the file manager and claims the following available drive letter in the system.

The CURIOSITY drive contains approximately one MB of free space and does not reflect the target device's Flash size. When programming an Intel HEX file, the binary data are encoded in ASCII with metadata providing a vast overhead, so 1 MB is a trivially chosen value for disk size.

It is not possible to format the CURIOSITY drive. The filename may appear in the disk directory listing when programming a file to the target, which is merely the operating system's view of the directory that, in reality, has not been updated. It is not possible to read out the file contents. Removing and replugging the board will return the file system to its original state, but the target will still contain the previously programmed application.

Copy a text file starting with "`CMD:ERASE`" onto the disk to erase the target device.

By default, the CURIOSITY drive contains several read-only files for generating icons as well as reporting status and linking to further information:

- `AUTORUN.ICO` – icon file for the Microchip logo
- `AUTORUN.INF` – system file required for Windows Explorer to show the icon file
- `KIT-INFO.HTM` – redirect to the development board website
- `KIT-INFO.TXT` – a text file containing details about the board's debugger firmware version, board name, USB serial number, device, and drag-and-drop support
- `STATUS.TXT` – a text file containing the programming status of the board

> **Info:** The on-board debugger dynamically updates `STATUS.TXT`. The contents may not reflect the correct status as the OS may cache it.

3.5.2   Limitations of Drag-and-Drop Programming

Lock Bits

Lock bits included in the hex file will be ignored when using drag-and-drop programming. To program lock bits, use MPLAB® X IDE.

Enabling CRC Check in Fuses

It is not advisable to enable the CRC check fuses in the target device when using drag-and-drop programming because a subsequent chip erase (which does not affect fuse bits) will cause a CRC mismatch, and the application will fail to boot. A chip erase must be done using MPLAB® X IDE, which automatically clears the CRC fuses after erasing to recover a target from this state.

3.5.3   Special Commands

Several utility commands are supported by copying text files to the mass storage disk. The filename or extension is irrelevant – the command handler reacts to content only.

Table 3-3. Special File Commands

Command Content	Description
`CMD:ERASE`	Executes a chip erase of the target
`CMD:SEND_UART=`	Sends a string of characters to the CDC UART. See "CDC Override Mode."

`CMD:SEND_9600=`
`CMD:SEND_115200=`
`CMD:SEND_460800=`
	Sends a string of characters to the CDC UART at the specified baud rate. Note that only the baud rates explicitly specified here are supported. See "CDC Override Mode." (Debugger firmware v1.25.6 or newer.)

`CMD:RESET`	Resets the target device by entering Programming mode and exiting Programming mode immediately afterward. The exact timing can vary according to the programming interface of the target device. (Debugger firmware v1.25.6 or newer.)

> **Info:** The content sent to the mass storage emulated disk triggers the commands listed here and provides no feedback in the case of either success or failure.

3.6   Data Gateway Interface (DGI)

Data Gateway Interface (DGI) is a USB interface transporting raw and timestamped data between on-board debuggers and host computer-based visualization tools. MPLAB Data Visualizer is used on the host computer to display any debug GPIO data. It is available as a plug-in for MPLAB X IDE or a stand-alone application that can be used in parallel withMPLAB® X IDE.

Although DGI encompasses several physical data interfaces, the AVR16EB32 Curiosity Nano implementation includes logic analyzer channels:

- Two debug GPIO channels (also known as DGI GPIO)

3.6.1   Debug GPIO

Debug GPIO channels are timestamped digital signal lines connecting the target application to a host computer visualization application. They are typically used to plot low-frequency events on a time axis, such as when given Application state transitions occur.

The figure below shows the monitoring of the Digital state of a mechanical switch connected to a debug GPIO in the MPLAB Data Visualizer.

Figure 3-3. Monitoring Debug GPIO with MPLAB Data Visualizer
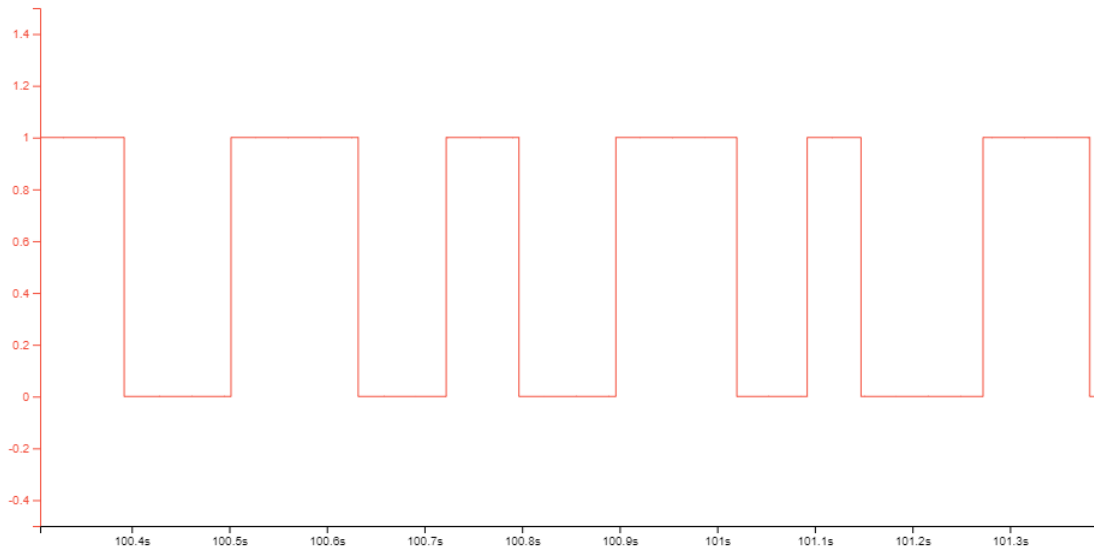
Debug GPIO channels are timestamped, so the resolution of DGI GPIO events is determined by the resolution of the DGI Timestamp module.

> **Important:** Although signal bursts of higher frequency can be captured, the frequency range of signals for which debug GPIO can be used is up to about 2 kHz. Attempting to capture signals above this frequency will result in data saturation and overflow, which may cause the DGI session to be aborted.

3.6.2 Timestamping

When captured by the debugger, DGI sources are timestamped. The timestamp counter implemented in the Curiosity Nano debugger increments at a 2 MHz frequency, providing a timestamp resolution of a half microsecond.

3.7 Device Configuration Protection

The on-board debugger on the AVR16EB32 Curiosity Nano has a protection mechanism intended to prevent the MCU from being rendered unrecoverable, especially when using drag-and-drop programming. Disabling the programming and debugging interface, or permanently locking memories are features of some MCUs that are not conducive to evaluation on a development kit platform.

The protection mechanism works by intercepting write operations to the relevant fuses and conditionally masking the addresses and values written.

> **Info:** The protection mechanism is intended to prevent making *accidental* irreversible changes. Users intentionally making irreversible changes do so at their own risk.

> **Info:** The available feature-set and corresponding prevention mechanism depend on the device. Check the data sheet for further information.

The protection mechanism can be disabled for users wanting to have the complete experience of given features that require irreversible changes. Doing this will result in permanent changes.

The pydebuggerconfig package distributed on pypi.org can be used to tweak many aspects of the debugger, including configuration protection. This procedure requires the installation of a recent release of Python 3 and the pydebuggerconfig package in that environment.

Step 1: Current status

Determine whether protection is currently enabled by executing:

```
pydebuggerconfig read
```

Then check for the section:

```
Register TARGET_DEBUG_FEATURES:   0x0F (15)  # Program/debug features
                                   bit 0, SINGLE_DEVICE: 1 # Single-device
                                   bit 1, PROG_ENABLED: 1 # Programming
                                   bit 2, DEBUG_ENABLED: 1 # Debug
                                   bit 3, FUSE_PROTECTION: 1 # Fuse protection
```

A FUSE_PROTECTION value of '1' indicates that protection is in place.

Step 2: Modifying the protection setting

Calculate the desired setting for TARGET_DEBUG_FEATURES by setting FUSE_PROTECTION to '0'. Replace the current value in the TARGET_DEBUG_FEATURES register by executing, for example:

```
pydebuggerconfig replace -r TARGET_DEBUG_FEATURES=0x07
```

To verify that the new value is set, repeat the process from step 1 and then toggle power on the kit. All protection mechanisms are now disabled.

Step 3: Restoring protection

Protection can be re-enabled either by repeating step 2 with the initial value or using the factory-restore function by executing:

```
pydebuggerconfig restore
```

> **Important:** The restore function will only restore any altered *kit* configuration settings that have changed since manufacturing, which is done by restoring a copy of the *kit* configuration settings stored internally on the debugger. This operation does not affect the MCU and will not undo any irreversible changes that may have been made to its configuration.

4. Hardware Implementation

LED, Mechanical Switch, Power Supply

4.1 Crystal

The AVR16EB32 Curiosity Nano Board has a 32.768 kHz crystal footprint made for standard 3.2 mm by 1.5 mm surface mount crystals with two terminals. The crystal footprint is not connected to the AVR16EB32 by default, as the GPIOs are routed to the edge connector.

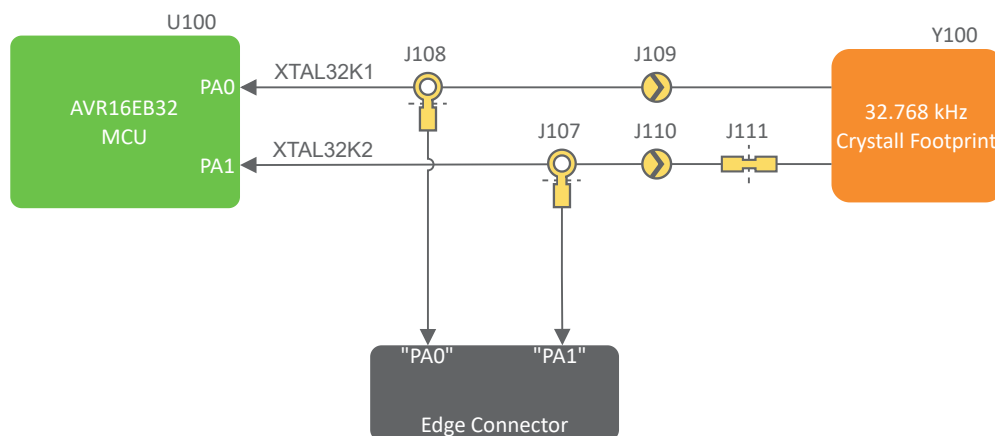**Figure 4-1.** 32.768 kHz Crystal Block Diagram

Table 4-1. Crystal Connections

AVR16EB32 Pin	Function	Shared Functionality
PA0	XTAL32K1	Edge connector
PA1	XTAL32K2	Edge connector

Some hardware modifications are required to use a crystal with the AVR16EB32.

- The two I/O lines routed to the edge connector must be disconnected to reduce the chance of contention to the crystal and to remove excessive capacitance on the lines
- Disconnect the edge connector by cutting the two cut straps on the bottom side of the board, J107 and J108. Marked PA0 and PA1 in the silkscreen
- Connect the crystal by soldering on a solder blob on each circular solder point next to the crystal on the top side of the board, J109 and J110
- Add suitable capacitors to C104 and C105
- Add a suitable series resistor to R100

Figure 4-2. Crystal Cut Straps
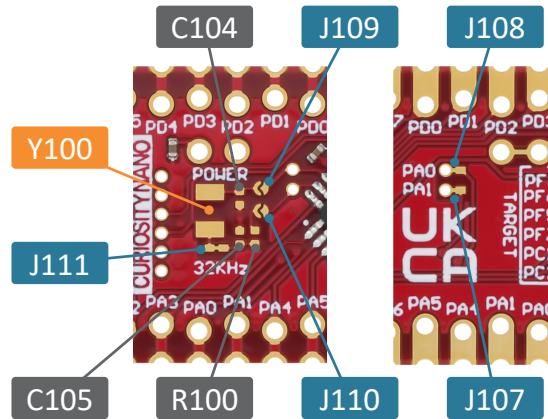
Info: The series resistor R100 may be required to limit the drive strength of the crystal. If no resistor is required, the pads of R100 can be connected with a solder blob.

> **Important:** PA0 and PA1 are used as the I^{2}C pins on the edge connector. Disconnecting PA0 and PA1 from the edge connector may disable some functionality with connected base boards.

The crystal has a cut strap (J111), which can be used to measure the oscillator safety factor and is done by cutting the strap and adding a 0402 SMD resistor across the strap. The AN2648 application note from Microchip contains more information about oscillator allowance and safety factors.

Figure 4-2 shows the cut straps and solder points.

4.2 LED

One yellow user LED is available on the AVR16EB32 Curiosity Nano board. Either GPIO or PWM can control it. Driving the connected I/O line to GND can also activate the LED.

Figure 4-3. AVR16EB32 Curiosity Nano LED0 Block Diagram
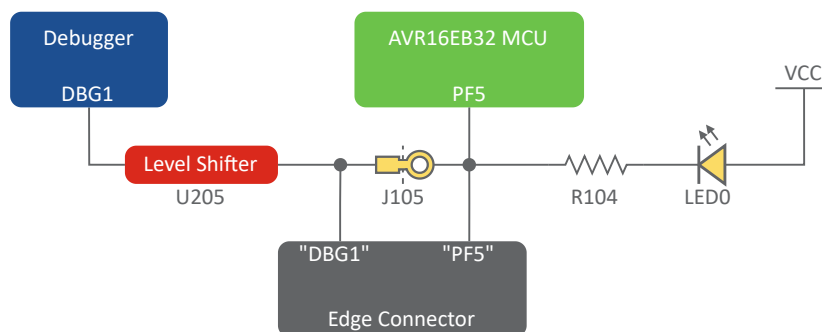
Table 4-2. LED Connection

AVR16EB32 Pin	Function	Shared Functionality
PF5	Yellow LED0	Edge connector, On-board debugger

4.3 Mechanical Switch

The AVR16EB32 Curiosity Nano board has one mechanical switch - a generic user-configurable switch. Pressing it will connect the I/O pin to ground (GND).

Figure 4-4. AVR16EB32 Curiosity Nano SW0 Block Diagram
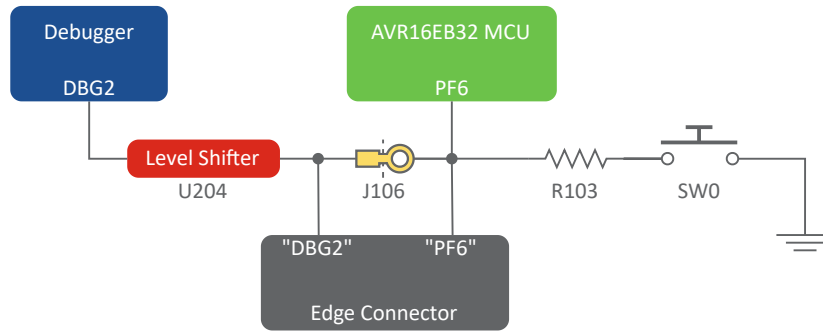
Tip: There is no externally connected pull-up resistor on the switch. Enable the internal pull-up resistor on Pin PF6 to use it.

Table 4-3. Mechanical Switch Connection

AVR16EB32 Pin	Description	Shared Functionality
PF6	User switch (SW0)	Edge connector, On-board debugger

4.4 Power Supply

The USB port powers the board. The input power is delayed to allow USB cable oscillations to die out. The delay is about 1 ms. VBUS is further limited to a 2 V/ms slew rate and is current limited to 500 mA by U202 (MIC2008).

Tip: Changing the values for C206 and R211 can alter the slew rate and current limit set by U202.

The power supply consists of two LDO regulators, one to generate 3.3V for the on-board debugger and an adjustable LDO regulator for the target AVR16EB32 microcontroller and its peripherals. The voltage from a USB connector can vary between 4.4V and 5.25V (according to the USB specification) and will limit the maximum voltage supplied to the target. The figure below shows the entire power supply system on the AVR16EB32 Curiosity Nano.

Figure 4-5. Power Supply Block Diagram
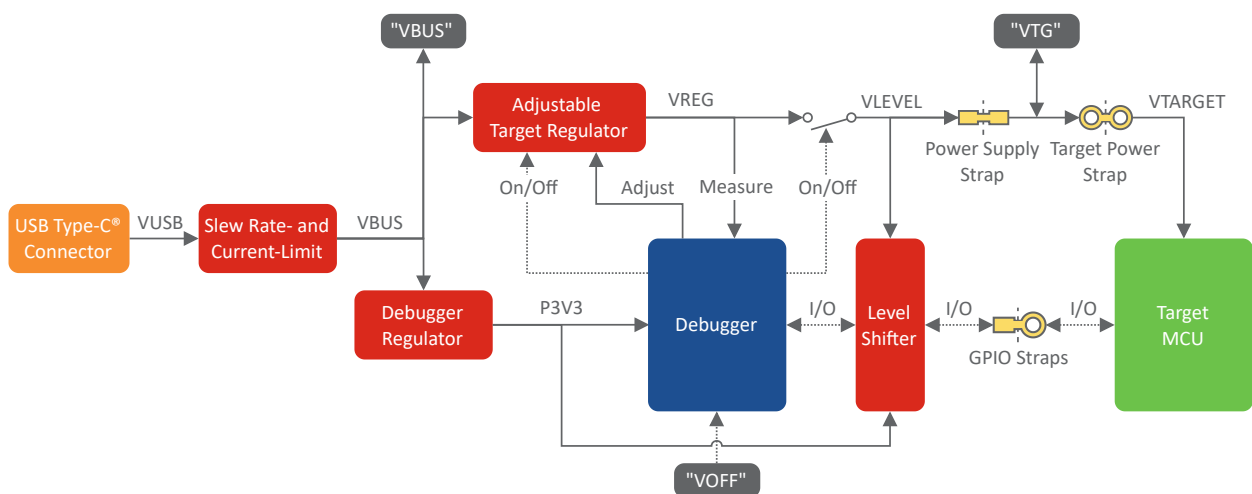
4.4.1 Cut Straps

All power and debugging signals are connected to the target by default. The following cut straps are available to take measurements or separate the debugger from the target:

- Target Power Strap (J201)
- Power Supply Strap (J200)
- Debugger Pins (J101, J102, J104, J105, J106)

Figure 4-6. Common Curiosity Nano Cut Straps
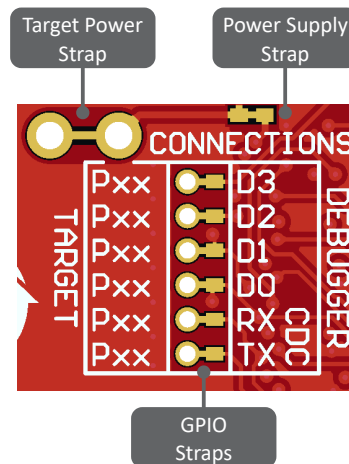
4.4.2 Target Regulator

The target voltage regulator is a MIC5353 variable output LDO. The on-board debugger can adjust the voltage output supplied to the board target section by manipulating the MIC5353's feedback voltage. The hardware implementation is limited to an approximate voltage range from 1.7V to 5.1V. Additional output voltage limits are configured in the debugger firmware to ensure that the output voltage never exceeds the hardware limits of the AVR16EB32 microcontroller. The voltage limits configured in the on-board debugger on AVR16EB32 Curiosity Nano are 1.8–5.5V.

> ℹ **Info:** The factory default target voltage is 3.3V. The project properties in MPLAB® X IDE can change it. Any change to the target voltage is persistent, even after a power toggle. The resolution is less than 5 mV but may be limited to 10 mV by the adjustment program.

> ℹ **Info:** The voltage settings setup in MPLAB® X IDE is not applied immediately to the board. Like clicking the Refresh Debug Tool Status button in the project dashboard tab or programming/reading program memory, the new voltage setting is applied to the board when accessing the debugger.

> ℹ **Info:** There is an easy option to adjust the target voltage with a drag-and-drop command text file to the board, which supports a set of commonly used target voltages. See section Special Commands for further details.

MIC5353 supports a maximum current load of 500 mA. It is an LDO regulator in a small package placed on a small printed circuit board (PCB) and can reach the thermal shutdown condition at lower loads than 500 mA. The maximum current load depends on the input voltage, the selected

output voltage, and the ambient temperature. The figure below shows the safe operating area for the regulator, with an input voltage of 5.1V and an ambient temperature of 23**°**C.

Figure 4-7. Target Regulator Safe Operation Area
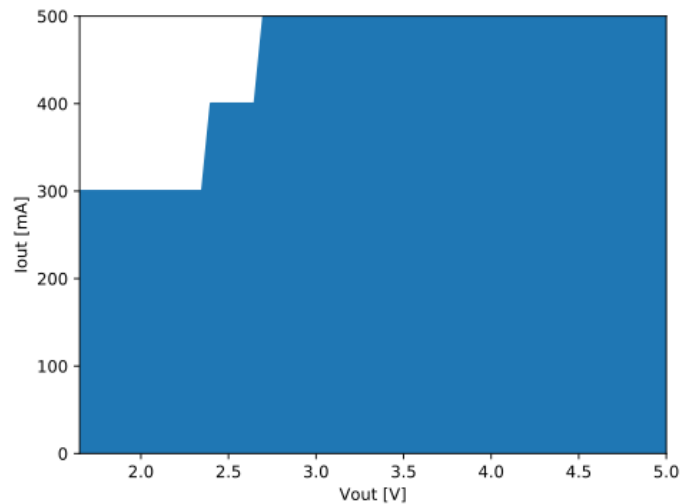
The voltage output of the target regulator is continuously monitored (measured) by the on-board debugger. An error condition will be flagged - and the target voltage regulator will be switched off, detecting and handling any short-circuit conditions if it is more than 100 mV over/under the set device voltage. It will also detect and handle if an external voltage, which causes V_{CC}_TARGET to move outside the voltage setting monitoring window of ±100 mV, is suddenly applied to the VTG pin without setting the V_{OFF} pin low.

> **Info:** The on-board debugger has a monitoring window of V_{CC}_TARGET±100 mV, and the status LED will blink rapidly if the external voltage is under this limit. The on-board debugger status LED will continue to shine if the external voltage surpasses this limit. When removing the external voltage, the status LED will start blinking rapidly until the on-board debugger detects the new situation and turns the target voltage regulator back on.

4.4.3   External Supply

Instead of the on-board target regulator, an external voltage can power the AVR16EB32 Curiosity Nano. When shorting the Voltage Off (VOFF) pin to the ground (GND) pin, the on-board debugger firmware disables the target regulator, and it is safe to apply an external voltage to the VTG pin.

It is also safe to apply an external voltage to the VTG pin when no USB cable is plugged into the DEBUG connector on the board.

The VOFF pin can be tied low/let go at any time, which will be detected by a pin-change interrupt to the on-board debugger, which controls the target voltage regulator accordingly.

> ⚠️ **WARNING**   Applying an external voltage to the VTG pin without shorting VOFF to GND may cause permanent damage to the board.

> ⚠️ **WARNING** Do not apply any voltage to the VOFF pin. Let the pin float to enable the power supply.

> ⚠️ **WARNING** The absolute maximum external voltage is 5.5V for the on-board level shifters, and the standard operating condition of the AVR16EB32 is 1.8–5.5V. Applying a higher voltage may cause permanent damage to the board.

> ℹ️ **Info:** The on-board debugger monitors the voltage supplied to the board. If VOFF is not pulled low and external power is supplied that differs by more than ±100 mV from the target regulator setting, the on-board debugger will shut off the target regulator and begin blinking the status LED rapidly, indicating an error condition. Once the input voltage returns to within ±100 mV of the target regulator setting, the on-board debugger will switch on the target regulator and stop blinking the status LED.

Programming, debugging, and data streaming are still possible with an external power supply. The USB cable will power the debugger and signal level shifters. Both regulators, the debugger, and the level shifters are powered down when the USB cable is removed.

> ℹ️ **Info:** In addition to the power consumed by the AVR16EB32 and its peripherals, approximately 100 µA will be drawn from any external power source to power the on-board level shifters and voltage monitor circuitry when plugging a USB cable into the DEBUG connector on the board. When a USB cable is unplugged, some current is used to supply the level shifter's voltage pins, having a worst-case current consumption of approximately 5 µA. Typical values may be as low as 100 nA.

4.4.4 Power Supply Exceptions

This section sums up most exceptions that can occur with the power supply.

Target Voltage Shuts Down

Not reaching the target voltage setting can happen if the target section draws too much current at a given voltage and causes the thermal shutdown safety feature of the MIC5353 regulator to kick in. To avoid this, reduce the current load of the target section.

Target Voltage Setting is Not Reached

The USB input voltage (specified to be 4.4-5.25V) limits the maximum output voltage of the MIC5353 regulator at a given voltage setting and current consumption. If a higher output voltage is needed, use a USB power source with a higher input voltage or use an external voltage supply on the VTG pin.

Target Voltage is Different From Setting

An externally applied voltage to the VTG pin without setting the VOFF pin low can cause this. If the target voltage varies more than 100 mV over/under the voltage setting, the on-board debugger will detect it, and the internal voltage regulator will shut down. To fix this issue, remove the applied voltage from the VTG pin, and the on-board debugger will enable the on-board voltage regulator when the new condition is detected. Note that the PS LED will blink rapidly if the target voltage is below 100 mV of the setting but will ordinarily turn on when higher than 100 mV above it.

No, or Very Low Target Voltage and PS LED is Blinking Rapidly

A full or partial short circuit can cause this and is a particular case of the issue above. Remove it, and the on-board debugger will re-enable the on-board target voltage regulator.

No Target Voltage and PS LED is Lit 1

This situation occurs if the target voltage is set to 0.0V. Set the target voltage to a value within the specified voltage range for the target device to fix this.

No Target Voltage and PS LED is Lit 2

This situation can be the issue when cutting power jumper J200 and/or J201 and setting the target voltage regulator to a value within the specified voltage range for the target device. To fix this, solder a wire/bridge between the pads for J200/J201 or add a jumper on J201 if a pin-header is mounted.

V$_{BUS}$ Output Voltage is Low or Not Present

If the VBUS output voltage is low or missing, the reason is probably a high-current drain on V$_{BUS}$, and the current limit set by U202 (MIC2008) is tripped and has cut off V$_{BUS}$ completely. Reduce the current consumption on the VBUS pin to fix this issue.

4.4.5 Low-Power Measurement

Power to the AVR16EB32 comes from the on-board power supply and *VTG* pin through a 100 mil pin-header marked with "POWER" in silkscreen (J201). To measure the power consumption of the AVR16EB32 and other peripherals connected to the board, cut the *Target Power strap* (J201) on the bottom side and connect an ammeter across it.

> **Tip:** A 100-mil pin-header can be soldered into the *Target Power strap* (J201) footprint for a simple connection of an ammeter. Place a jumper cap on the pin-header once the ammeter is no longer needed.

To measure the lowest possible power consumption, follow these steps:

1. Cut the POWER strap with a sharp tool.
2. Solder a 1x2 100 mil pin-header in the footprint.
3. Connect an ammeter to the pin header.
4. Write firmware that:
   a. Tri-states any I/O connected to the on-board debugger.
   b. Sets the microcontroller in its lowest Power sleep mode.
5. Program the firmware into the AVR16EB32.

> **Info:** The on-board level shifters will draw a small amount of current even when unused. Each level shifter has a maximum of 2 µA leakage current. Therefore, the worst-case maximum current draw for the five on-board level shifters is 10 µA. Prevent leakage current through an I/O pin connected to a level shifter by keeping the I/O pin tri-stated. All I/Os connected to the on-board debugger are listed in On-Board Debugger Connections. The on-board level shifters can be completely disconnected, preventing leakage, as described in Disconnecting the On-Board Debugger.

4.4.6 VBUS Output Pin

AVR16EB32 Curiosity Nano has a VBUS output pin that can be used to power external components that need a 5V supply. The VBUS output pin is protected by the same start-up delay with a slew rate and current limiter as the rest of the power supply. A side effect is a voltage drop on the VBUS

output with higher current loads. The chart below shows the VBUS output voltage versus the current load of the VBUS output.

Figure 4-8. VBUS Output Voltage vs. Current
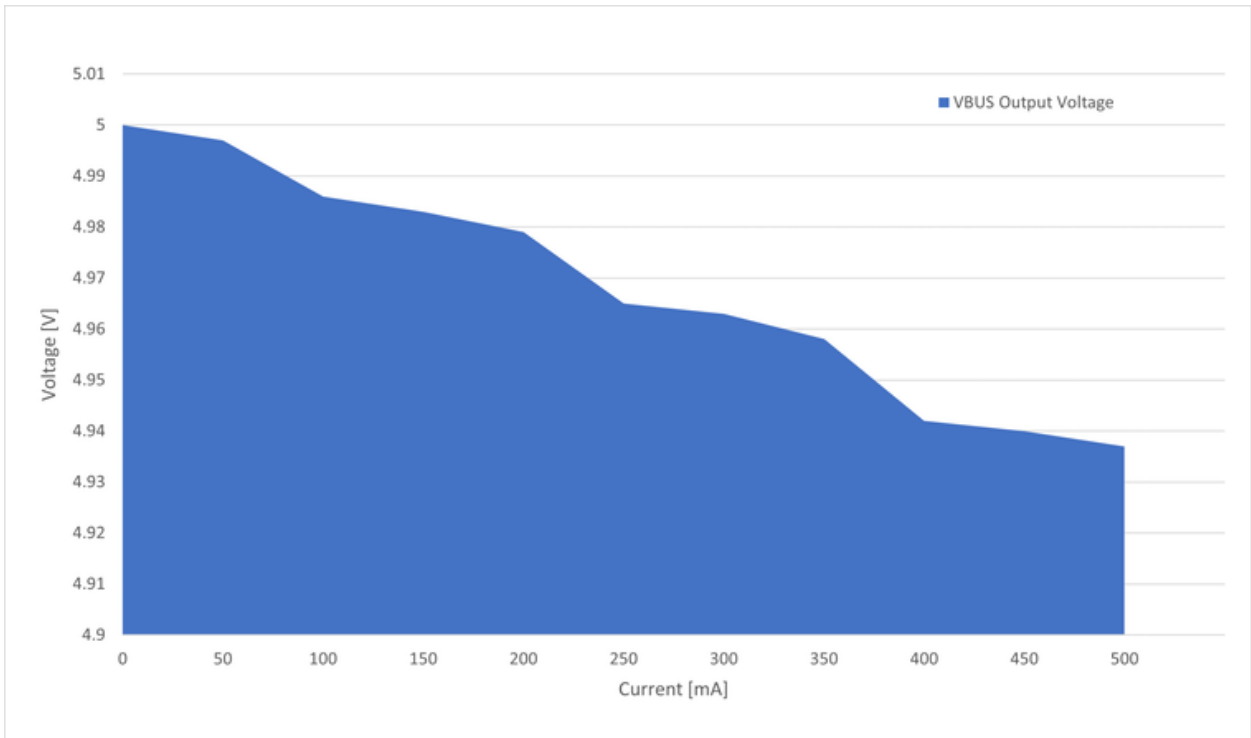
5.   Revision History

Hardware and Document Revision History.

5.1   Hardware Revision History and Known Issues

This user guide provides information about the latest available revision of the board. The following sections contain information about known issues, a revision history of older revisions, and how older revisions differ from the latest revision.

5.1.1   Identifying Product ID and Revision

There are two ways to find the revision and product identifier of the AVR16EB32 Curiosity Nano: The MPLAB X IDE Kit Window or the sticker on the bottom of the PCB.

The Kit Window will pop up when connecting AVR16EB32 Curiosity Nano to a computer with MPLAB X IDE running. The first six digits of the serial number, listed under kit information, contain the product identifier and revision.

> **Tip:**   If closed, the Kit Window can be opened in MPLAB X IDE through the menu bar **Window** > **Kit Window**.

The same information is found on the sticker on the bottom side of the PCB. Most boards will have the identifier, revision, serial number, and manufacturing date printed in plain text as 02-nnnn\rr SN: mmmssssss [Manufacturing date], where "nnnn" is the identifier, "rr" is the revision, "mmm" is the manufacturer, and "ssssss" is the serial number.

The serial number string has the following format:

"nnnnnnrrmmmssssss"

n = product identifier

r = revision

m = manufacturer

s = serial number

The product identifier for AVR16EB32 Curiosity Nano is 02-00618.

5.1.2   Revision 2

Revision 2 is the initially released board revision. It has AVR16EB32 rev. A0 mounted.

5.1.2.1   Flash Verification Fails in MPLAB® X

Verification errors may be reported in MPLAB® X when writing fuses/configuration bits on AVR16EB32 Curiosity Nano revision 2, which is caused by the debugger incorrectly masking/protecting the configuration memory of the AVR device. There are several possible workarounds for this issue. Workarounds 1 and 2 are persistent on the AVR16EB32 Curiosity Nano, while workaround 3 affects only the current project and does not persist on the kit.

> **Info:**   You can see this message during the verification of reserved locations not required to be written for regular operation. You may assume that Flash programming has succeeded unless seeing a Flash memory verification failure.

Workaround 1

The Curiosity Nano debugger can be re-provisioned by the end-user with updated masks using a binary utility.

Procedure:
1. Navigate to github.com/microchip-pic-avr-tools/debuggerconfigfix and download the debuggerconfigfix utility for the appropriate operating system.
2. Connect the Curiosity Nano kit to the host PC and reprovision the debugger by opening a command-line/terminal and executing the binary

    ```
    debuggerconfigfix.exe
    ```

3. Check the console output for success indication:

    ```
    *** Debugger configuration SUCCESS ***
    ```

4. Unplug the Curiosity Nano kit and plug it in again for the new mask to take effect.

Workaround 2

The Curiosity Nano debugger can be re-provisioned by the end-user with updated masks using a Python-based utility.

Requirements:
•   Python 3.8 or later installed

Procedure:
1. Install the pydebuggerconfig Python package using:

    ```
    pip install pydebuggerconfig
    ```

    Note: If the pydebuggerconfig package is already installed, update it to version 3.11 or newer.
2. Connect the Curiosity Nano kit to the host PC and reprovision the debugger using

    ```
    pydebuggerconfig write -d AVR16EB32-device-blob.xml
    ```

3. Unplug the Curiosity Nano kit and plug it in again for the new mask to take effect.

Workaround 3

When writes to reserved locations are skipped by the debugger, this issue occurs, and the verification fails as unexpected values read back. You can avoid verification errors by providing the reserved fuse values already in the MCU in the project being programmed.

For example, in a project, a structure like this specifies the fuse values:

```
FUSES =
{
  ...
};
```

The values already programmed in reserved locations can be added to this structure (in this case 0xFF) by adding:

```
.reserved_1 = {0xFF, 0xFF},
.reserved_2 = 0xFF,
```

During verification, the specified values will match the values already programmed, and no verification issues will be seen.

5.2   Document Revision History

Doc. Rev.	Date	Comments
B	9/2024	Added revision 2 ERRATA

...........continued		
Doc. Rev.	Date	Comments
A	10/2023	Initial document release

6. Appendix

Schematic, Assembly Drawing, Adapter Pinout, Programming External MCUs, External Debuggers

Figure 6-1. AVR16EB32 Curiosity Nano MCU Schematic

AVR16EB32 Curiosity Nano

Appendix

Figure 6-2. AVR16EB32 Curiosity Nano Debugger Schematic
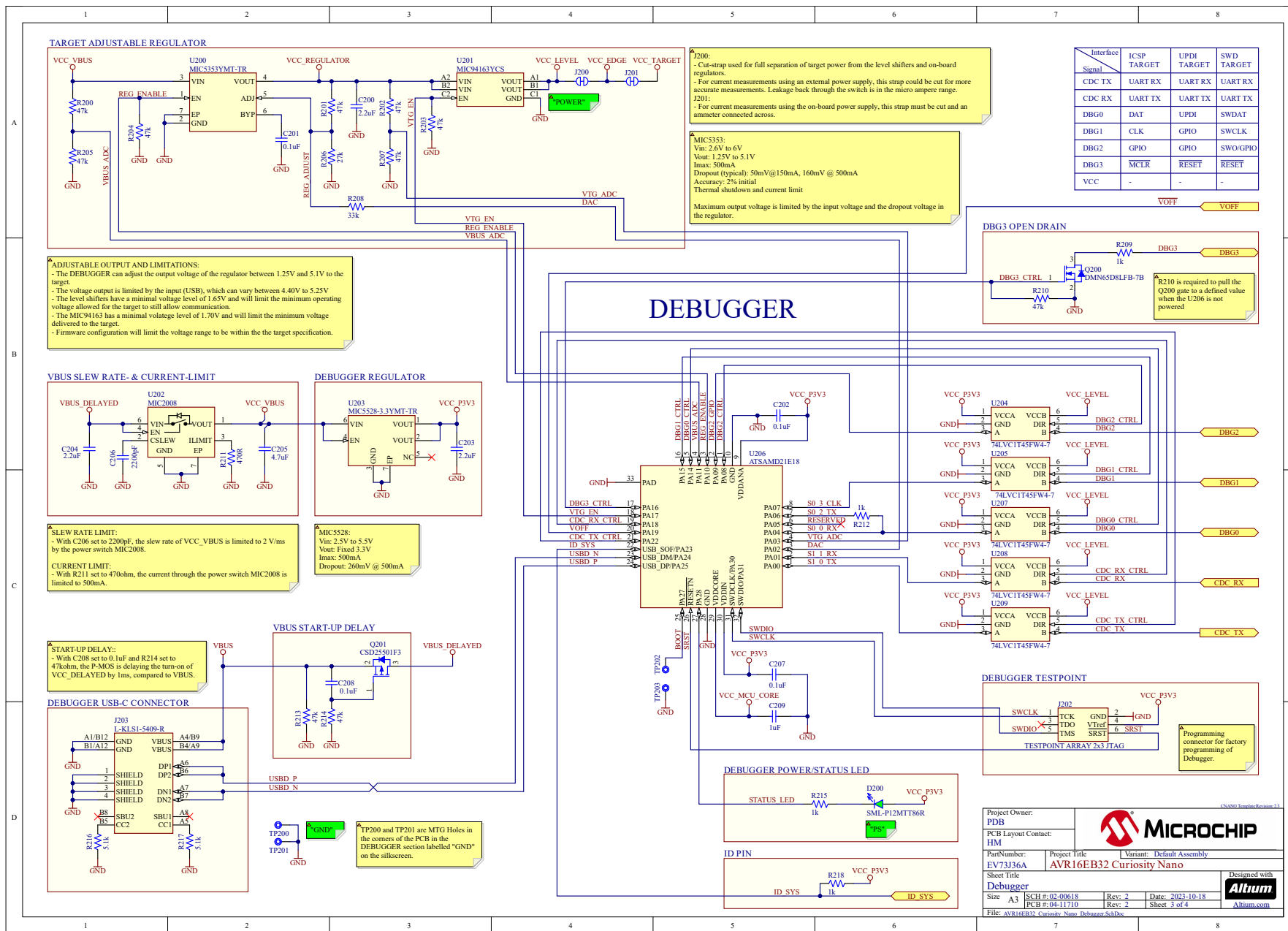
6.2   Assembly Drawing

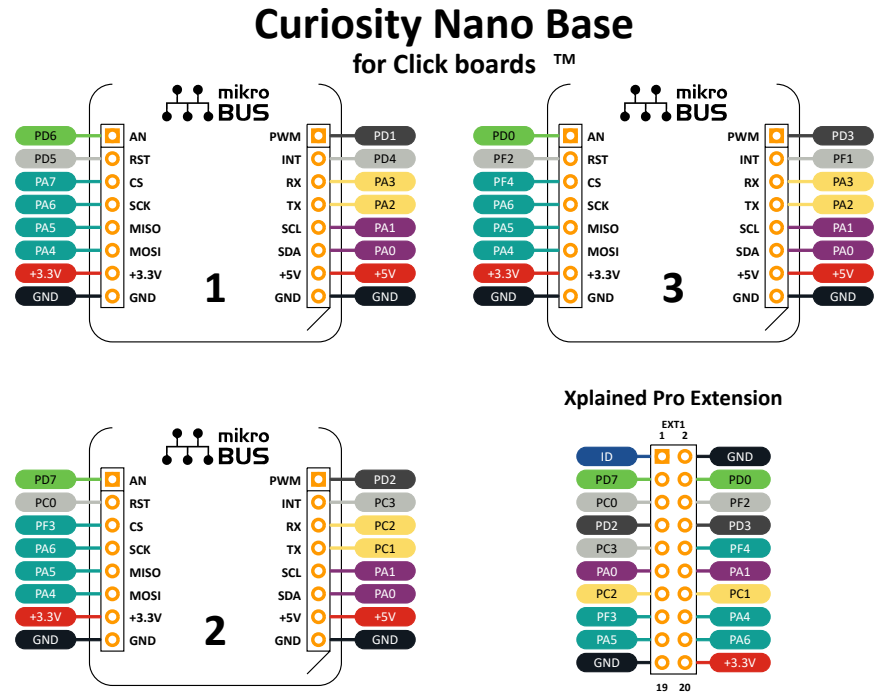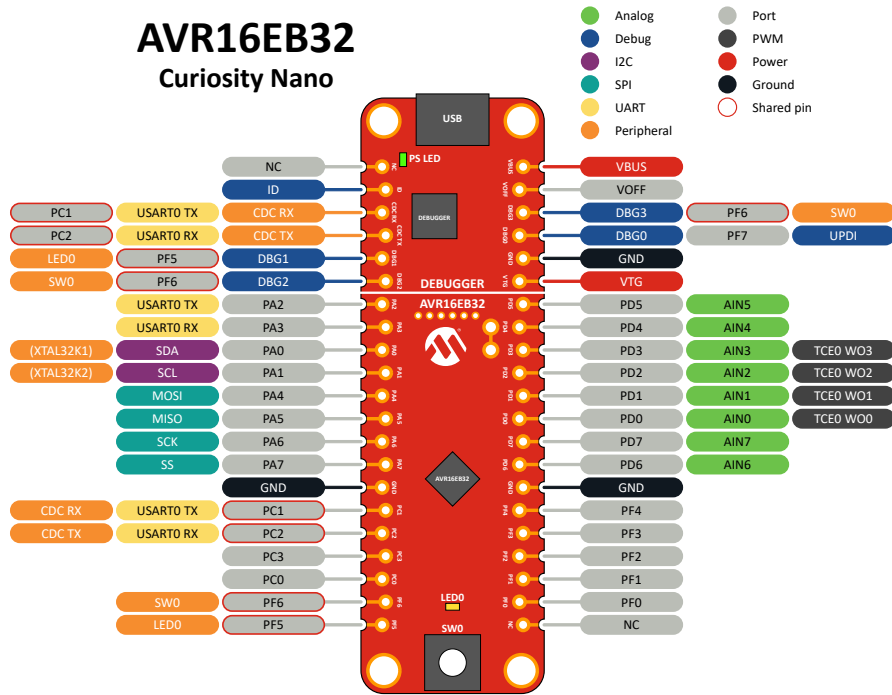**Figure 6-3.** AVR16EB32 Curiosity Nano Assembly Drawing Top
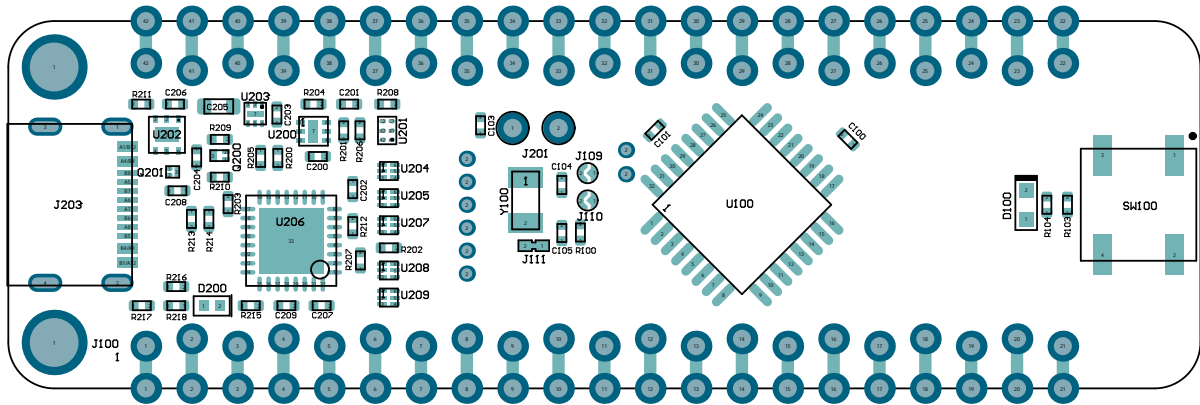
Figure 6-4. AVR16EB32 Curiosity Nano Assembly Drawing Bottom
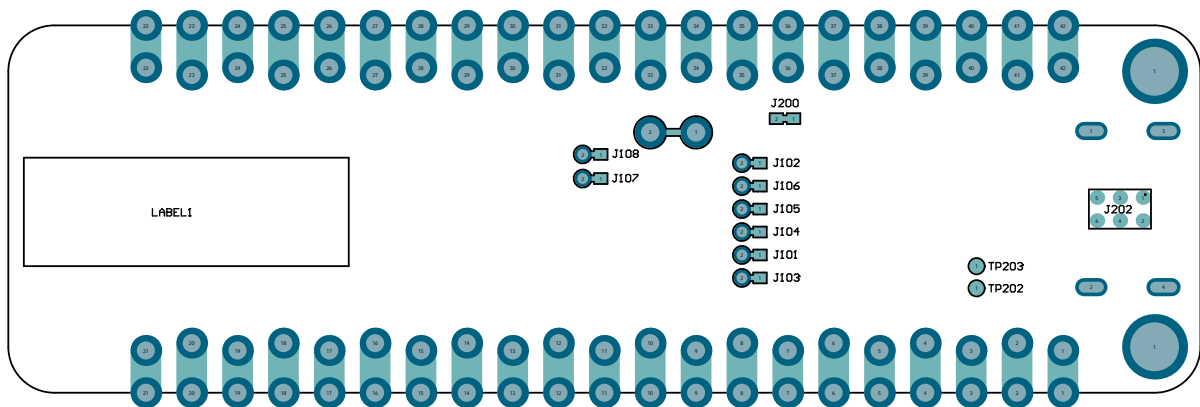
6.3 Curiosity Nano Base for Click boards™

Figure 6-5. AVR16EB32 Curiosity Nano Pinout Mapping

User Guide

6.4 Programming External Microcontrollers

Use the on-board debugger on AVR16EB32 Curiosity Nano to program and debug microcontrollers on external hardware.

6.4.1 Supported Devices

All external AVR microcontrollers with the UPDI interface can be programmed and debugged with the on-board debugger with Microchip Studio.

External SAM microcontrollers having a Curiosity Nano Board can be programmed and debugged with the on-board debugger with Microchip Studio.

AVR16EB32 Curiosity Nano can program and debug external AVR16EB32 microcontrollers with MPLAB X IDE.

6.4.2 Software Configuration

No software configuration is required to program and debug the same device mounted on the board.

To program and debug a different microcontroller than the one mounted on the board, configure Microchip Studio to allow an independent selection of devices and programming interfaces.

1. Navigate to **Tools** > **Options** through the menu system at the application top.
2. Select the **Tools** > **Tool settings** category in the options window.
3. Set the **Hide unsupported devices** option to **False**.

Figure 6-6. Hide Unsupported Devices
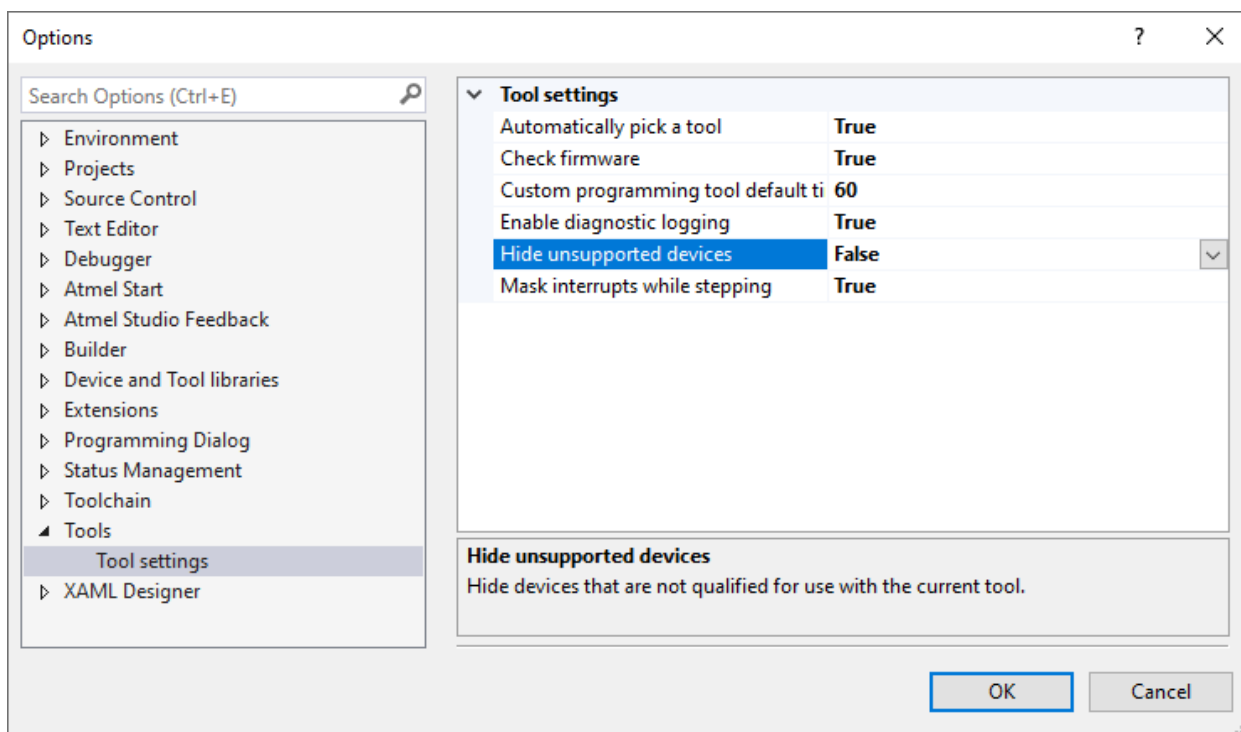
> **Info:** Microchip Studio allows any microcontroller and interface to be selected when the **Hide unsupported devices** setting is set to **False** - also microcontrollers and interfaces not supported by the on-board debugger.

6.4.3 Connecting to External Microcontrollers

The figure and table below show where to connect the programming and debugging signals to program and debug external microcontrollers. The on-board debugger can supply power to the external hardware or use an external voltage reference for its level shifters. Read more about the power supply in Power Supply.

The on-board debugger and level shifters actively drive data and clock signals used for programming and debugging (DBG0, DBG1, and DBG2). Pull-down resistors are required on the ICSP™ data and clock signals to debug PIC® microcontrollers. All other interfaces are functional with or without pull-up or pull-down resistors.

DBG3 is an open-drain connection and requires a pull-up resistor to function.

> **Remember:**
> - Connect GND and VTG to the external microcontroller
> - Tie the VOFF pin to GND if the external hardware has a power supply
> - Make sure there are pull-down resistors on the ICSP data and clock signals (DBG0 and DBG1) to support the debugging of PIC microcontrollers

Figure 6-7. Curiosity Nano Standard Pinout
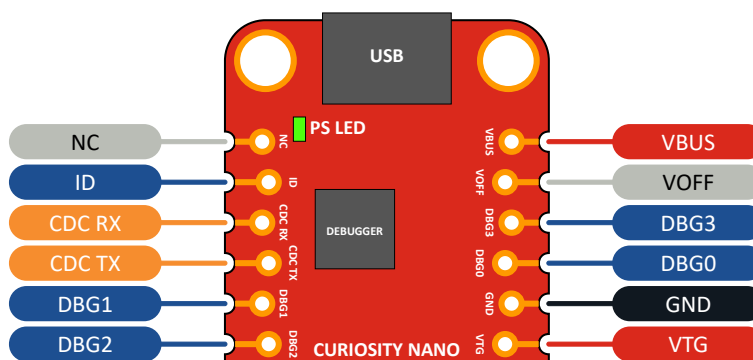
Table 6-1. Programming and Debugging Interfaces

Curiosity Nano Pin	UPDI	ICSP™	SWD
DBG0	UPDI	DATA	SWDIO
DBG1	—	CLK	SWCLK
DBG2	—	—	—
DBG3	—	$\overline{\text{MCLR}}$	$\overline{\text{RESET}}$

6.5 Connecting External Debuggers

Even though there is an on-board debugger, external debuggers can be connected directly to the AVR16EB32 Curiosity Nano to program/debug the AVR16EB32. When not actively used, the on-board debugger keeps all the pins connected to the AVR16EB32 and board edge in tri-state. Therefore, the on-board debugger will not interfere with any external debug tools.

Figure 6-8. Connecting the MPLAB® PICkit™ 5 In-Circuit Debugger/Programmer to AVR16EB32 Curiosity Nano
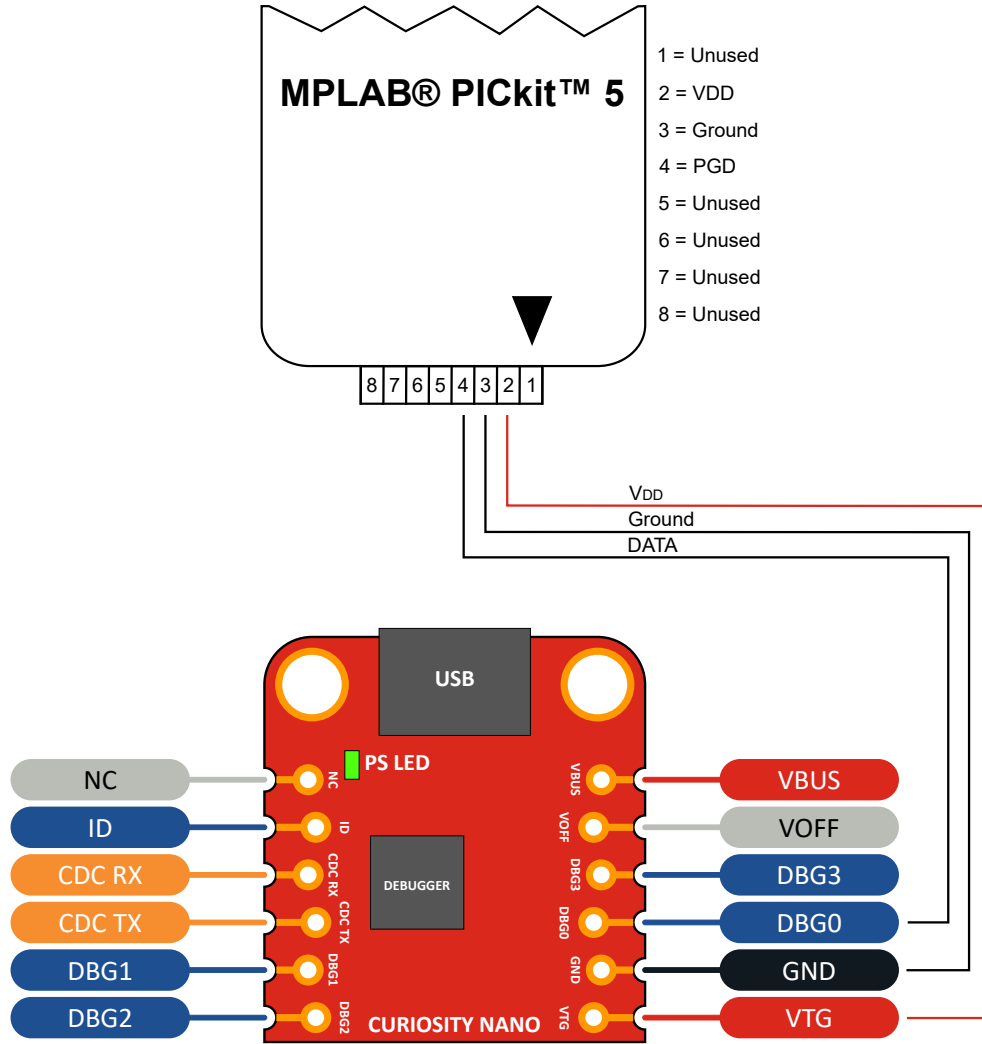
Figure 6-9. Connecting the Atmel-ICE to AVR16EB32 Curiosity Nano
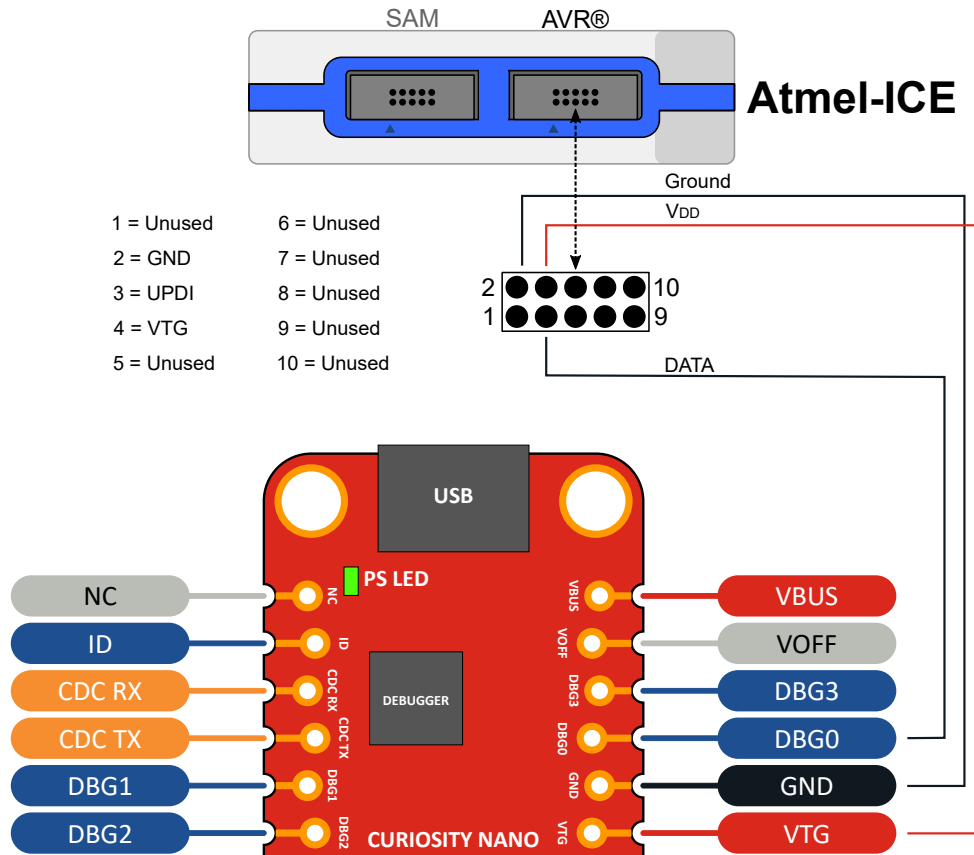
To avoid contention between the external debugger and the on-board debugger, do not start any programming/debug operation with the on-board debugger through MPLAB® X IDE or mass storage programming while the external tool is active.

6.6 Disconnecting the On-Board Debugger

The on-board debugger and level shifters can be disconnected from the AVR16EB32.

The power supply block diagram (Figure 4-5) shows all connections between the debugger and the AVR16EB32. The signal names are also printed in silkscreen on the top or bottom side of the board.

To disconnect the debugger, cut the GPIO straps shown in Cut Straps.

Attention: Cutting the GPIO straps to the on-board debugger will disable the virtual serial port, programming, debugging, and data streaming. Cutting the power supply strap will disconnect the on-board power supply.

Tip: Reconnect any cut connection by using solder. Alternatively, mount a 0Ω 0402 resistor.

> **Tip:** When the debugger is disconnected, an external debugger can be connected to the holes. Connecting External Debuggers describes how to connect an external debugger.

6.7   Getting Started with IAR™

IAR Embedded Workbench® for AVR® is a proprietary high-efficiency compiler not based on GCC. Programming and debugging of AVR16EB32 Curiosity Nano is supported in IAR™ Embedded Workbench for AVR using the Atmel-ICE interface. To get the programming and debugging to work, some initial settings must be set up in the project.

The following steps will explain how to get the project ready for programming and debugging:

1. Make sure that the project to be configured is opened. Open the **OPTIONS** dialog for the project.
2. In the category **General Options**, select the **Target** tab. Select the device for the project, or if not listed, the core of the device, as shown in Figure 6-10.
3. In the category **Debugger**, select the **Setup** tab. Select **Atmel-ICE** as the driver, as shown in Figure 6-11.
4. In the category **Debugger** > **Atmel-ICE**, select the **Atmel-ICE 1** tab. Select **UPDI** as the interface. Optionally select the **UPDI** frequency, as shown in Figure 6-12.

> **Info:** If the selection of Debug Port (mentioned in step 4) is grayed out, the interface is preselected, and the user can skip this configuration step.

Figure 6-10. Select Target Device
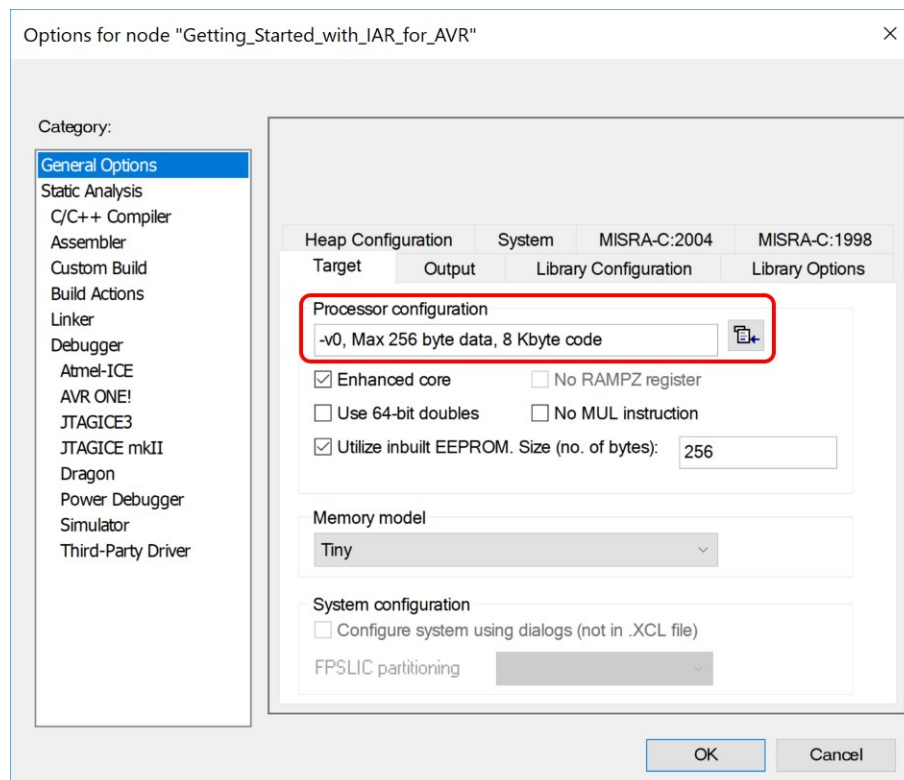
Figure 6-11. Select Debugger
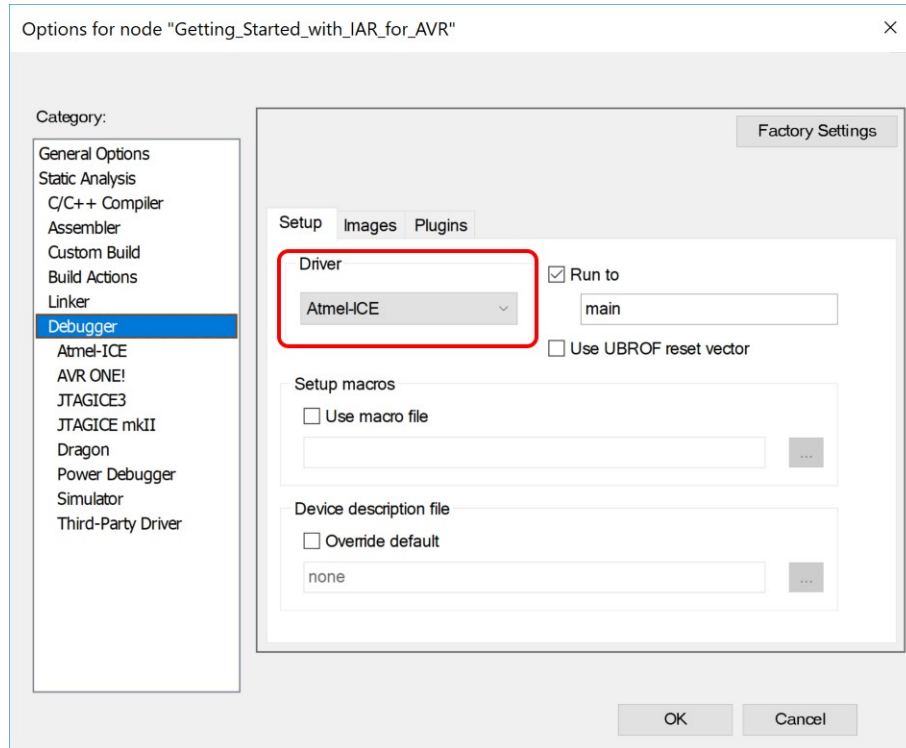
Figure 6-12. Configure Interface
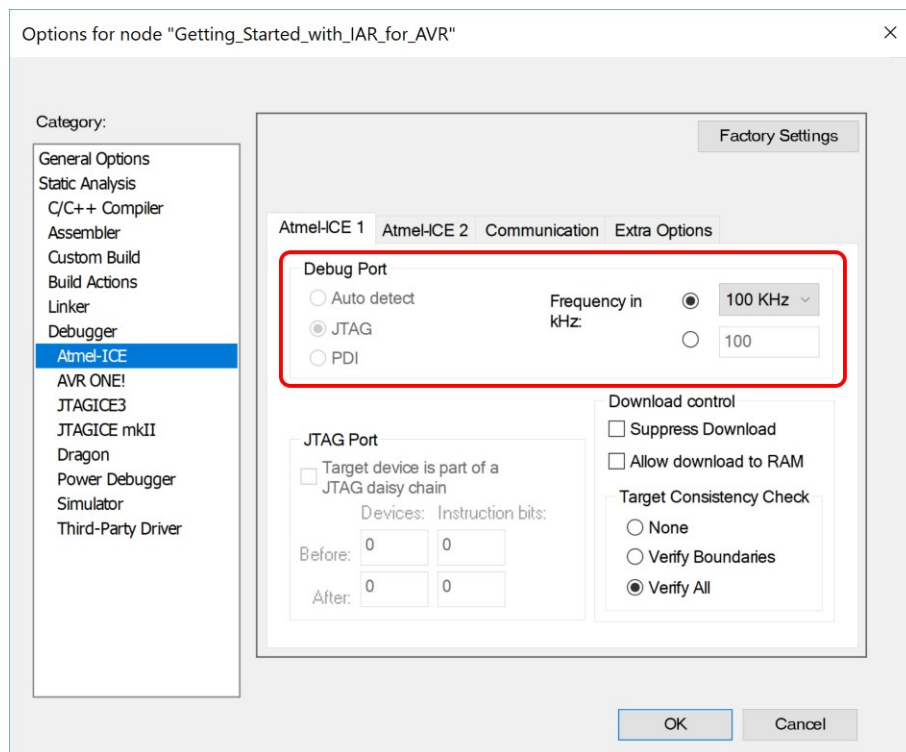
Microchip Information

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable". Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure

that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

Trademarks

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: www.microchip.com/support Web Address: www.microchip.com	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4485-5910 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820
Atlanta Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455	**China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029	**Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301	**France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700
Austin, TX Tel: 512-257-3370	**China - Hangzhou** Tel: 86-571-8792-8115	**Korea - Seoul** Tel: 82-2-554-7200	**Germany - Haan** Tel: 49-2129-3766400
Boston Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088	**China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460	**Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870	**Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370
Chicago Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075	**China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000	**Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870	**Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560
Dallas Addison, TX Tel: 972-818-7423 Fax: 972-818-2924	**China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200	**Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830	**Israel - Hod Hasharon** Tel: 972-9-775-5100
Detroit Novi, MI Tel: 248-848-4000	**China - Suzhou** Tel: 86-186-6233-1526	**Taiwan - Taipei** Tel: 886-2-2508-8600	**Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781
Houston, TX Tel: 281-894-5983	**China - Wuhan** Tel: 86-27-5980-5300	**Thailand - Bangkok** Tel: 66-2-694-1351	**Italy - Padova** Tel: 39-049-7625286
Indianapolis Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380	**China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138	**Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388
Los Angeles Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800	**China - Zhuhai** Tel: 86-756-3210040		**Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50
Raleigh, NC Tel: 919-844-7510			**Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91
New York, NY Tel: 631-435-6000			**Sweden - Gothenberg** Tel: 46-31-704-60-40
San Jose, CA Tel: 408-735-9110 Tel: 408-436-4270			**Sweden - Stockholm** Tel: 46-8-5090-4654
Canada - Toronto Tel: 905-695-1980 Fax: 905-695-2078			**UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820