

# ATtiny3227 Curiosity Nano User Guide

## ATtiny3227 Curiosity Nano

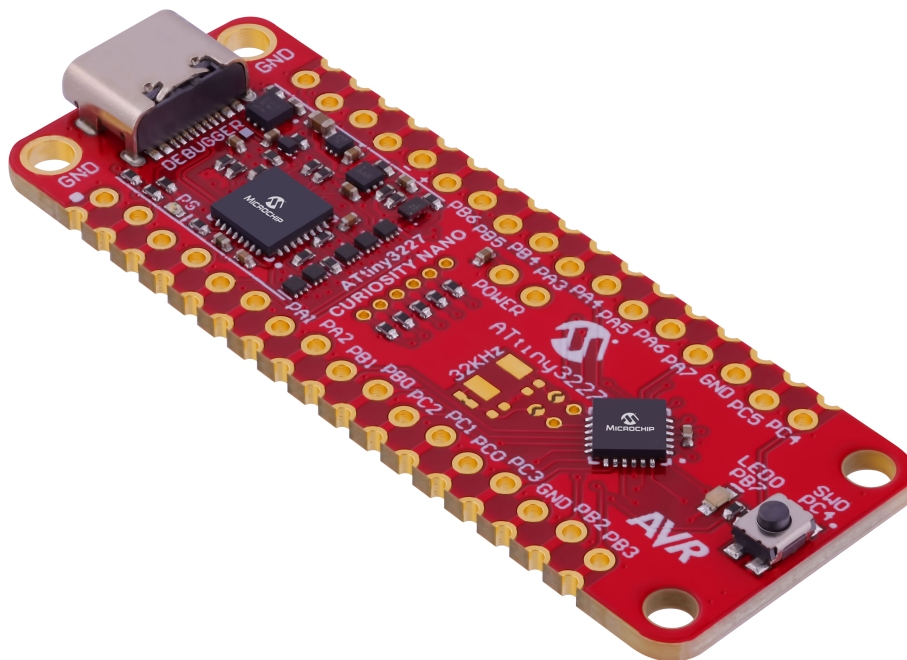


[www.microchip.com](http://www.microchip.com) Product Pages: [ATtiny3227](#)

## Preface

The ATtiny3227 Curiosity Nano evaluation kit (EV58A83A) is a hardware platform to evaluate the tinyAVR<sup>®</sup> 2 family microcontrollers. This board has the ATtiny3227 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 ATtiny3227.



- [ATtiny3227 Curiosity Nano website](#) - Kit information, latest user guide, and design documentation.
- [ATtiny3227 website](#) - Find documentation, data sheets, sample, and purchase microcontrollers.
- [Code examples on MPLAB<sup>®</sup> Discover](#) - Get started with code examples.
- [ATtiny3227 Curiosity Nano on Microchip Direct](#) - Purchase this kit on Microchip Direct.
- [ATtiny3227 Curiosity Nano Schematics](#) - Board schematics and history.
- [ATtiny3227 Curiosity Nano Altium Project](#) - Latest project revision.
- [ATtiny3227 Curiosity Nano Design Documentation](#) - Production files for every revision.

## Table of Contents

Preface.....	1
1. Features and Pinout.....	4
1.1. ATtiny3227 Key Features.....	4
1.2. Board Features.....	4
1.3. Board Overview.....	5
1.4. Block Diagram.....	6
1.5. Pinout.....	6
2. Getting Started.....	7
2.1. Getting Started Now with AVR <sup>®</sup> .....	7
2.2. How the Curiosity Nano Fits Into the MPLAB Tools Ecosystem.....	7
2.3. MPLAB Data Visualizer Support for Curiosity Nano.....	9
2.4. Using Pin Headers.....	10
3. On-Board Debugger.....	12
3.1. On-Board Debugger Overview.....	12
3.2. On-Board Debugger Connections.....	12
3.3. Debugger USB Enumeration.....	13
3.4. Virtual Serial Port (CDC).....	13
3.5. Mass Storage Device.....	17
3.6. Data Gateway Interface (DGI).....	19
3.7. Device Configuration Protection.....	20
4. Hardware Implementation.....	22
4.1. Crystal.....	22
4.2. LED.....	23
4.3. Mechanical Switch.....	23
4.4. Power Supply.....	24
5. Revision History.....	30
5.1. Hardware Revision History and Known Issues.....	30
6. Document Revision History.....	31
7. Appendix.....	32
7.1. Schematic.....	33
7.2. Assembly Drawing.....	35
7.3. Curiosity Nano Base for Click boards <sup>™</sup> .....	36
7.4. Programming External Microcontrollers.....	37
7.5. Connecting External Debuggers.....	38
7.6. Disconnecting the On-Board Debugger.....	40
7.7. Getting Started with IAR <sup>™</sup> .....	41
Microchip Information.....	43
Trademarks.....	43
Legal Notice.....	43

Microchip Devices Code Protection Feature.....43

## 1. Features and Pinout

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

### 1.1 ATtiny3227 Key Features

The tinyAVR® 2 family of microcontrollers use the AVR® CPU with hardware multiplier, running at up to 20 MHz, with 32 KB Flash, 3 KB of SRAM, and 256B of EEPROM. The family uses the latest technologies from Microchip with a flexible and low-power architecture, including an Event System, advanced digital peripherals, and accurate analog features such as a 12-bit differential ADC with Programmable Gain Amplifier (PGA).

High-Performance AVR® RISC CPU with Hardware Multiplier

- Running at up to 20 MHz
- Single Cycle I/O Access
- Two-level Interrupt Controller
- Sleep modes: Idle, Standby, Power-down
- Six channels Event System
- Configurable Custom Logic (CCL)
- Single-pin programming and debugging interface (UPDI)

Analog Features

- 12-bit Differential ADC with Programmable Gain Amplifier (PGA)
- Analog Comparator with internal DAC reference

Timer/Counters

- 1x 16-bit Timer/Counter type A (TCA)
- 2x 16-bit Timer/Counter type B (TCB)
- 1x 16-bit Real Time Counter (RTC) with Periodic Interrupt Controller (PIT)

Built-in Safety Functions

- Power-On Reset (POR)
- Brown-Out-Detection (BOD)
- Voltage Level Monitor (VLM)
- Cyclic Redundancy Check (CRC) Scan
- Watchdog Timer (WDT) with Window Mode

Communication

- Serial communication interfaces: SPI, I<sup>2</sup>C, 2x USART

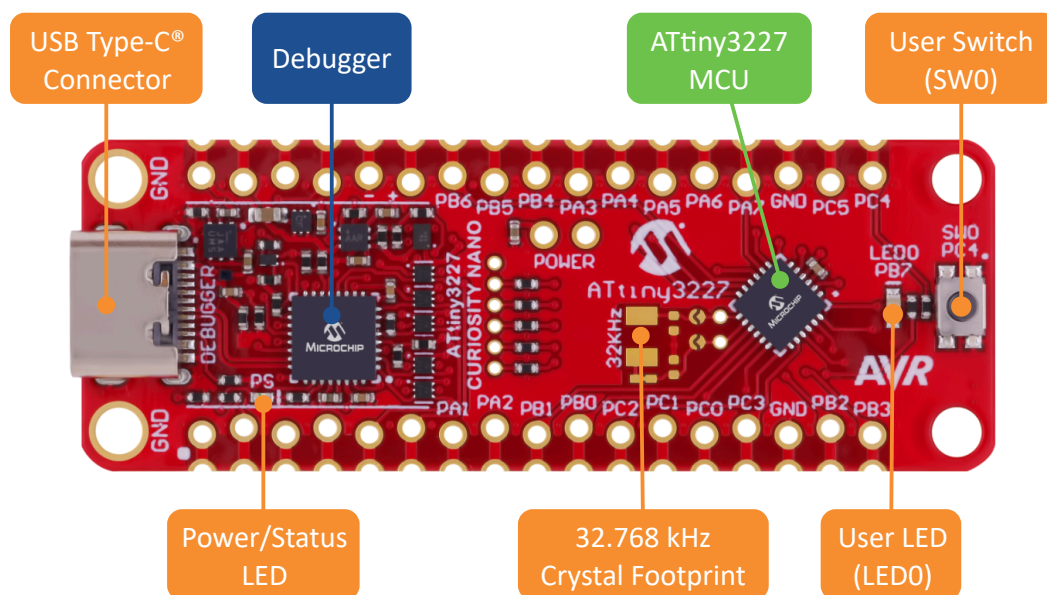
### 1.2 Board Features

- ATtiny3227 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:

- 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)

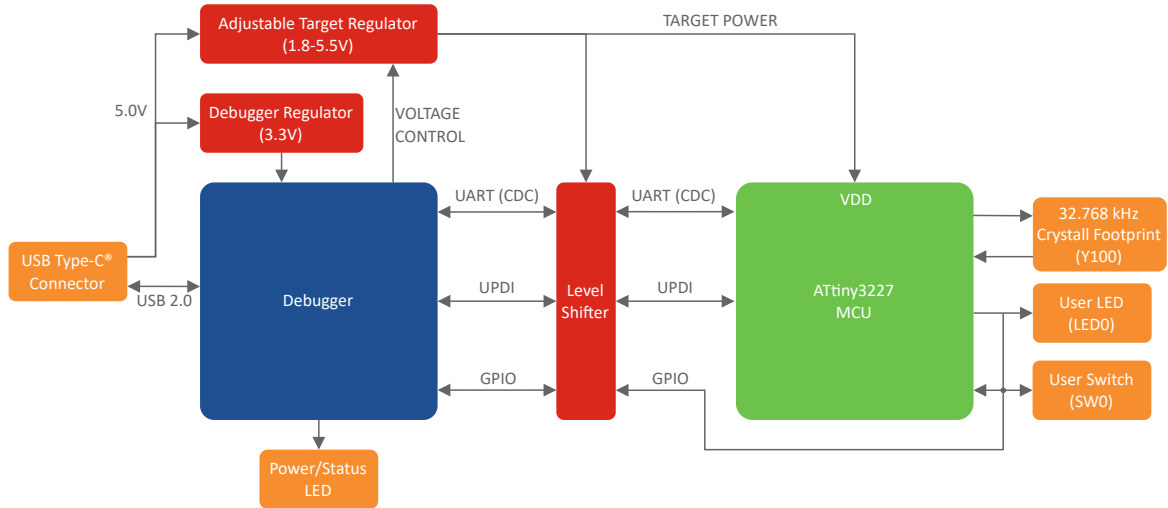
### 1.3 Board Overview

Figure 1-1. ATtiny3227 Curiosity Nano Board Overview



## 1.4 Block Diagram

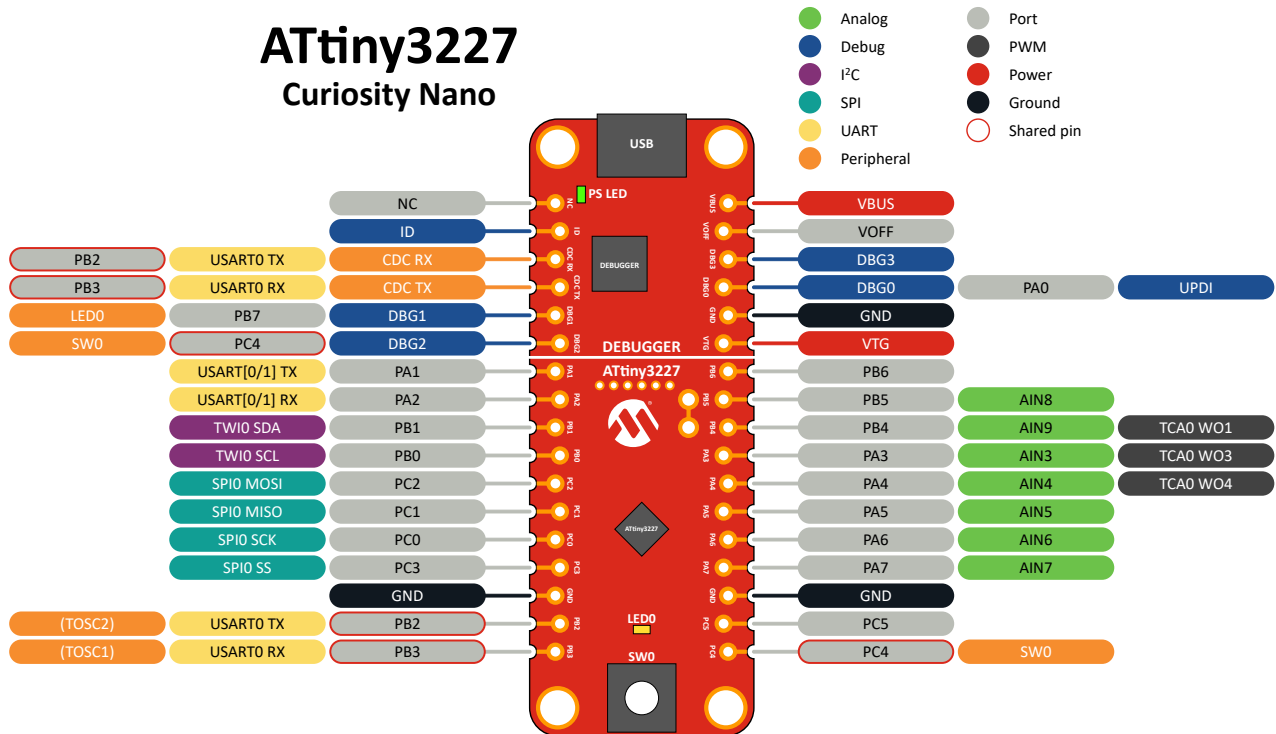
Figure 1-2. ATtiny3227 Curiosity Nano Board Block Diagram



## 1.5 Pinout

All the ATtiny3227 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 ATtiny3227 data sheet for all available functions on each pin.

Figure 1-3. ATtiny3227 Curiosity Nano Pinout



## 2. Getting Started

Getting started resources for the ATtiny3227 Curiosity Nano board in the MPLAB Tools Ecosystem.

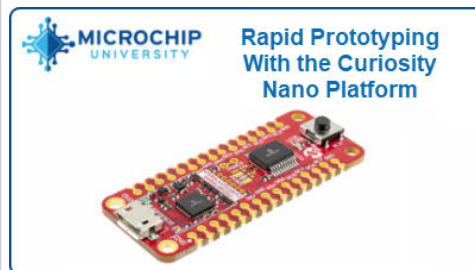
### 2.1 Getting Started Now with AVR<sup>®</sup>

Are you new to using AVR microcontrollers (MCUs)? Our comprehensive MPLAB<sup>®</sup> 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) to develop embedded software for a range of [supported Microchip MCUs](#) configured using [MPLAB<sup>®</sup> Code Configurator \(MCC\)](#).
3. **Develop:** [MPLAB X IDE](#) (Integrated Development Environment), which is available for Windows, Linux and macOS, also [MPLAB<sup>®</sup> XC C Compilers](#) and [GCC Compiler for AVR<sup>®</sup>](#).
4. **Debug:** With the MPLAB X IDE and/or the [MPLAB Data Visualizer](#), the ATtiny3227 device on the ATtiny3227 Curiosity Nano board is programmed and debugged by the on-board debugger. Therefore, no external programmer or debugger tool is required.
5. **Boards:** The ATtiny3227 Curiosity Nano board is a supported part of the [Curiosity Nano Development Platform](#), which includes the [Curiosity Nano Base for Click Boards<sup>™</sup>](#).



**Tip:** Take a look at [this](#) free Microchip University course.



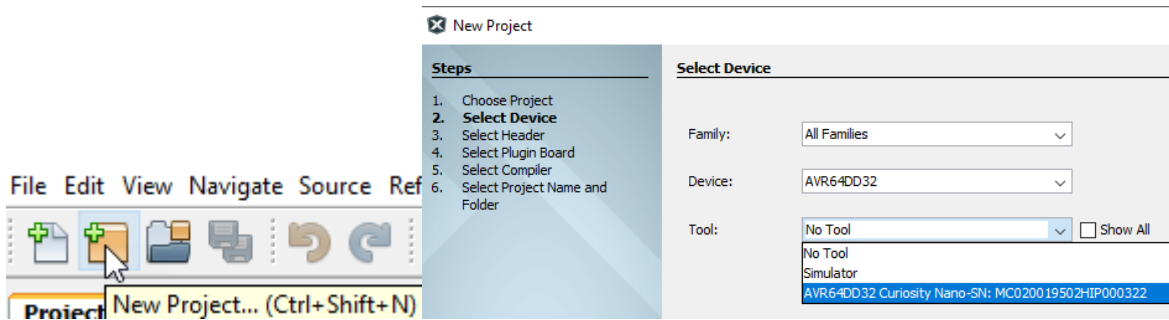
The Curiosity Nano development platform is well-suited to rapid prototyping. This course will introduce you to the defining features of the Curiosity Nano platform and show you how it can assist with using a new microcontroller or developing a new prototype. This course includes an in-depth look at the 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

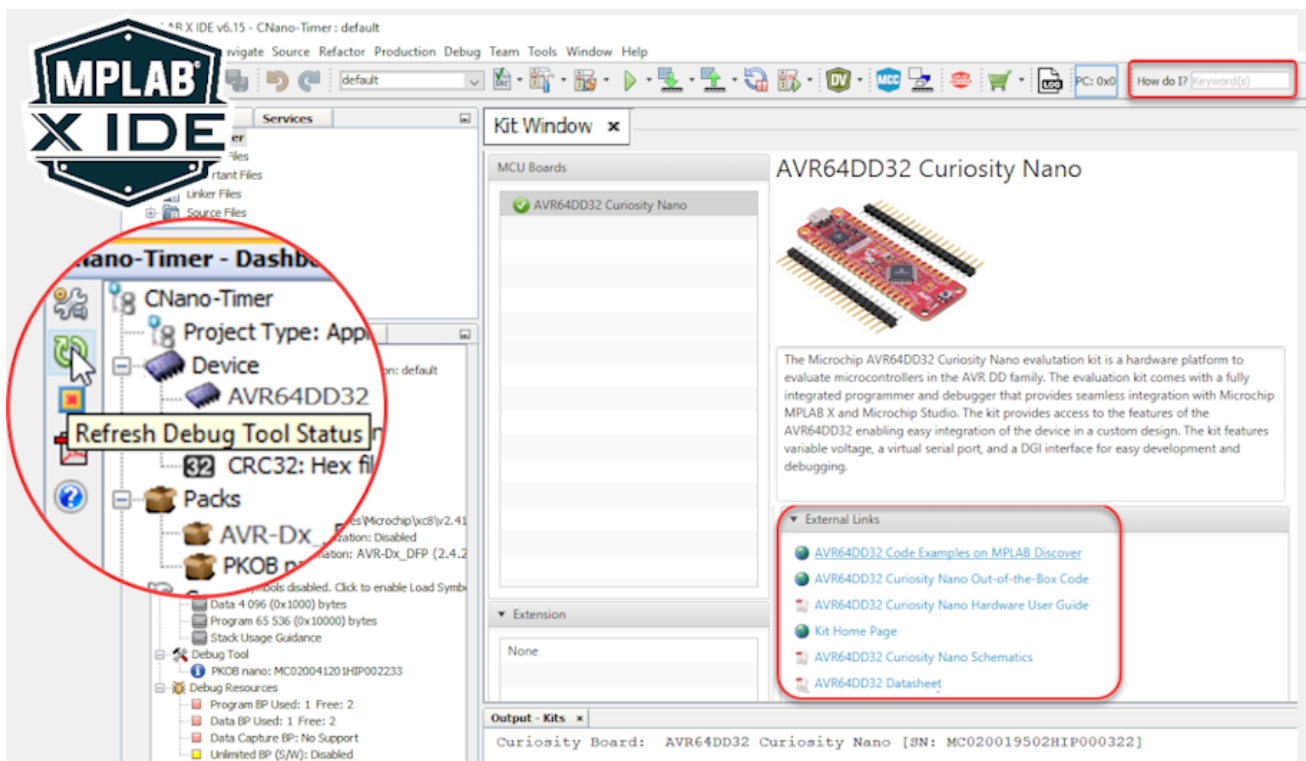
When the board connects to the computer for the first time, the operating system will install the driver software. The drivers for the board are included with MPLAB<sup>®</sup> X IDE. Once this is done, when connecting the Curiosity Nano to a host PC via USB, if MPLAB X IDE is open, a Kit Window is opened 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 ATtiny3227 Curiosity Nano board, use MPLAB® X version 6.20, device family pack "ATtiny\_DFP" version 2.6.122, 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](#).

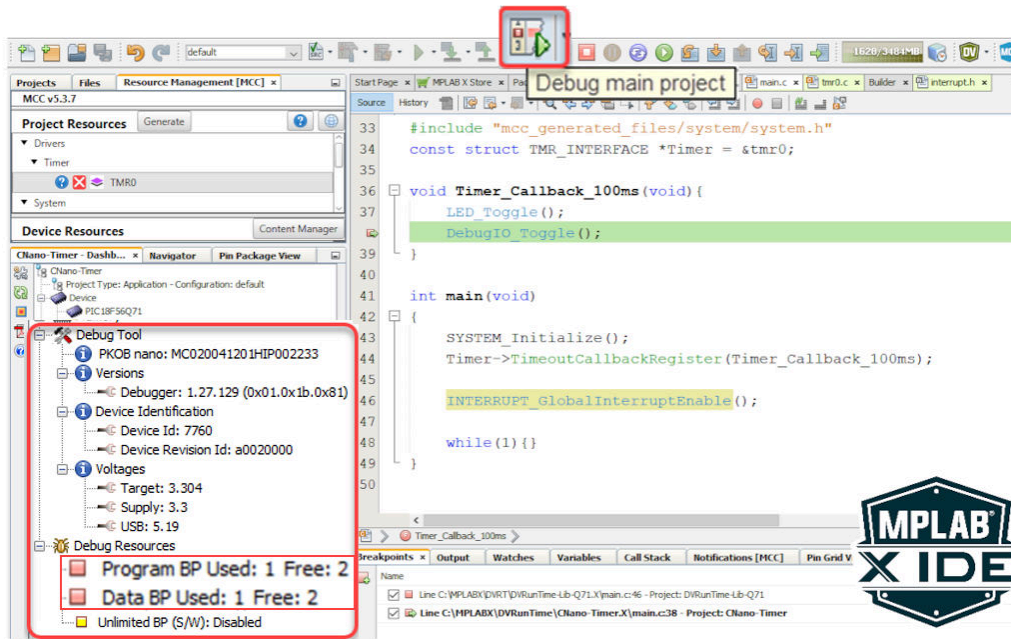


As shown in the image below, additional information about your Curiosity Nano can be seen in the "Debug Tool" window once you click "Refresh Debug Tool Status".



**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

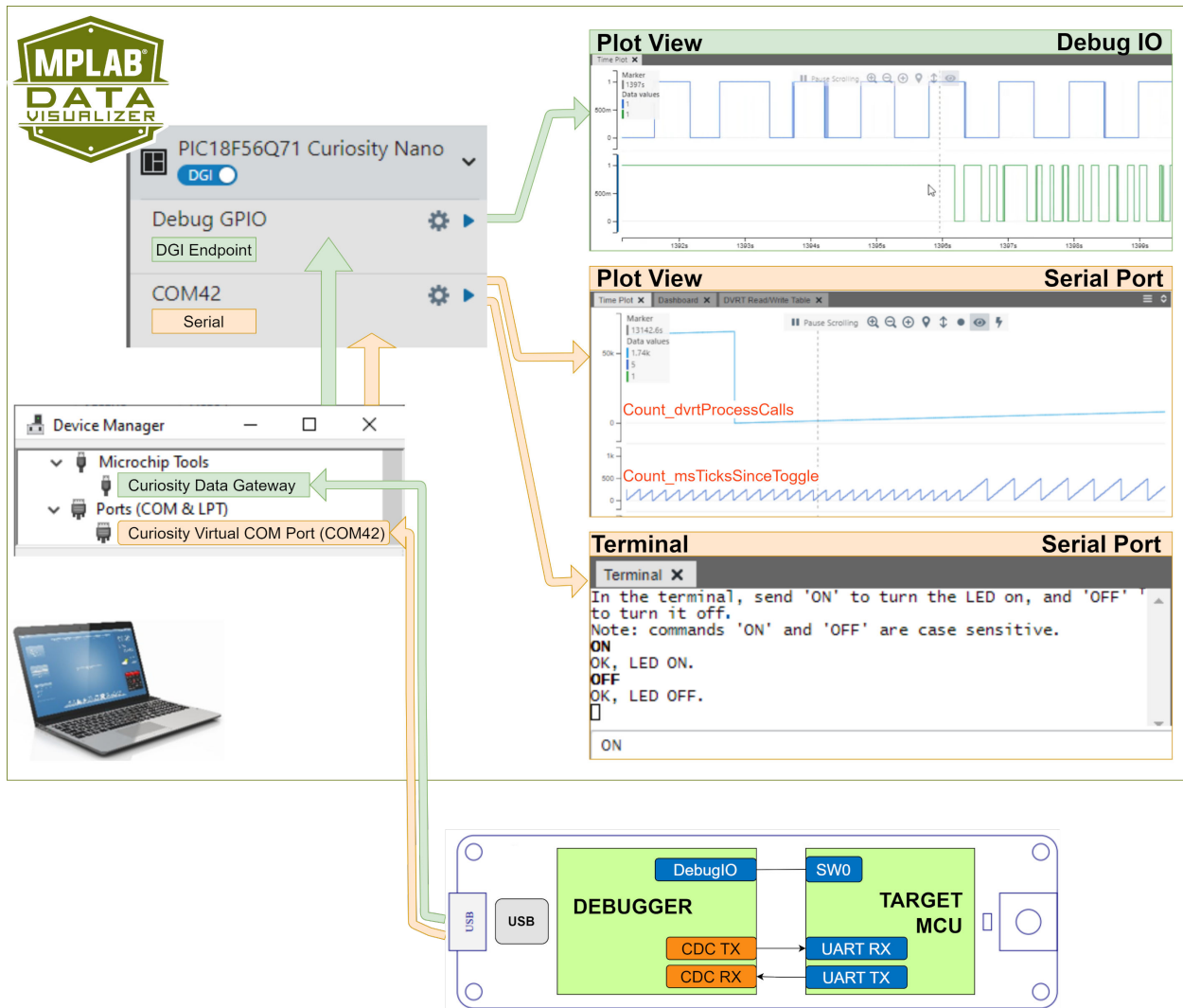


**Tip:**

- 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](#).

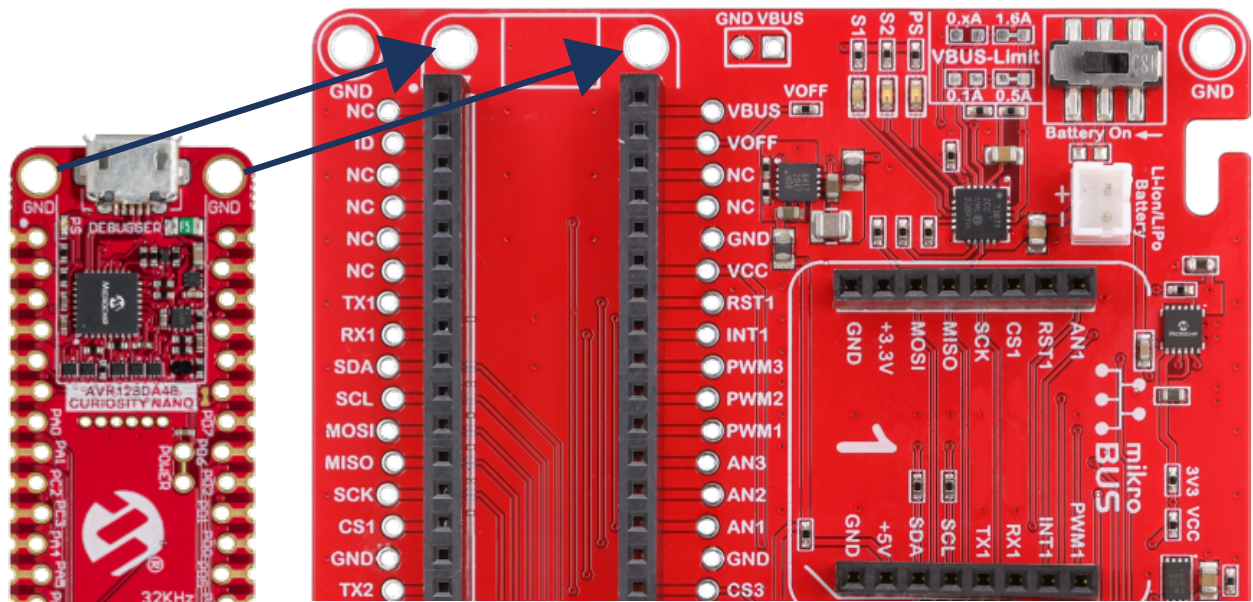
## 2.4 Using Pin Headers

The edge connector footprint on the ATtiny3227 Curiosity Nano has a staggered design where each hole is shifted 8 mils (~0.2 mm) off-center. The hole shift allows using regular 100 mil pin headers 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 permanently using pin headers, 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

ATtiny3227 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 ATtiny3227 in MPLAB® X IDE
- A virtual serial port (CDC) that is connected to a Universal Asynchronous Receiver/Transmitter (UART) on the ATtiny3227 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 ATtiny3227
- 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 ATtiny3227 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	<b>Success:</b> The LED blinks slowly for 2 sec. <b>Failure:</b> 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.

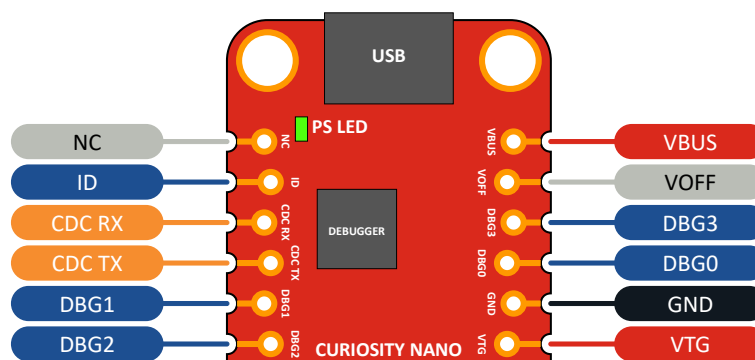
**Table 3-2.** On-Board Debugger Connections

Debugger Pin	ATtiny3227 Pin	Description
CDC TX	PB3	UART RX
CDC RX	PB2	UART TX
DBG0	PA0	UPDI
		Program and Debug Interface

**Table 3-2.** On-Board Debugger Connections (continued)

Debugger Pin	ATtiny3227 Pin		Description
DBG1	PB7	LED0/GPIO1	Debug GPIO1/LED0
DBG2	PC4	SW0/GPIO0	Debug GPIO0/SW0
DBG3	—	—	—
ID	—	—	ID line for extensions
NC	—	—	No connect
V <sub>BUS</sub>	—	—	V <sub>BUS</sub> 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 ATtiny3227 Curiosity Nano pinout, see the [ATtiny3227 Curiosity Nano Pinout](#).

### 3.3 Debugger USB Enumeration

The on-board debugger on the ATtiny3227 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 ATtiny3227 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.

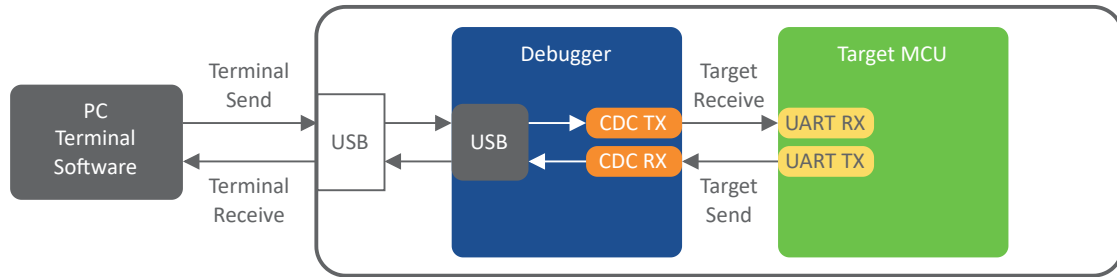
### 3.4 Virtual Serial Port (CDC)

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<sup>®</sup> 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 includes this driver.

On Linux<sup>®</sup> 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<sup>®</sup> 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. The 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 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 sending and receiving data 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 more 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 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 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 be completed 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) 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, 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 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 any data 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) 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 terminal application/user's responsibility 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<sup>®</sup> HEX and UF2 formatted files into the target device's memory
- Write access for simple text files for utility purposes

**Note:** Support for UF2 format is available in debugger firmware version 1.31 or later.

### 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 or UF2 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 Drag-and-Drop Programming Using UF2 Format

#### UF2 format for Drag-and-Drop

Drag-and-Drop programming provides a simple mechanism for programming the non-volatile memories of the microcontroller on board the Curiosity Nano kit. This is typically done using the Intel® hex format, which contains the necessary addresses and segmentation information as part of the format. Intel hex files contain the memory contents encoded as ASCII characters, which must be parsed in strictly sequential order, meaning that the host operating system must send the content in the correct sequence. This is not always the case for all variants of all operating systems. The UF2 format was developed by Microsoft as a means to allow memory to be transferred out of sequence. This is done by forcing a fixed block size to be used throughout the entire data transfer path so that partial writes are prevented.

More information on the UF2 format is available on [GitHub](#).

#### Generating a UF2 file

The result of a project compilation procedure is typically an Intel hex file, which can be converted into a UF2 file using a post-build step.

The [pymcuprog](#) package distributed on pypi.org includes a function to convert an Intel hex file to a UF2 file in version 3.17 or later.

**Note:** This procedure requires the installation of a recent release of Python 3 and the `pymcuprog` package in that environment and that the Python scripts folder is included in the system or user path.

An Intel hex file can be converted into a UF2 file using the `pymcuprog` command-line interface:

```
pymcuprog makeuf2 -f app.hex --uf2file app.uf2
```

This process can be streamlined by adding a post-build step in MPLAB X IDE. In the **Project Properties** configuration dialog, select the **Building** tab and check **Execute this line after build**. Add the command to the edit box:

```
pymcuprog makeuf2 -f ${ImagePath} --uf2file ${ImageDir}\${ProjectName}.X.${IMAGE_TYPE}.uf2
```

Each time the application is built, the produced Intel hex file will automatically be converted to a UF2 file with the same filename but with a `.uf2` file extension.

### 3.5.3 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.

### 3.5.4 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 target chip erase
CMD:SEND_UART=	Sends a string of characters to the CDC UART. See “ <a href="#">CDC Override Mode.</a> ”
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 “ <a href="#">CDC Override Mode.</a> ” (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.)
CMD:POWERTOGGLE	Powers down the target and restores it after a 100 ms delay. If external power is provided, this has no effect. (Debugger firmware v1.25.6 or newer.)
CMD:0V	Powers down the target device by disabling the target supply regulator. If external power is provided, this has no effect. (Debugger firmware v1.25.6 or newer.)
CMD:1V8	Sets the target voltage to 1.8V. If using external power, this has no effect. (Debugger firmware v1.25.6 or newer.)
CMD:3V3	Sets the target voltage to 3.3V. If using external power, this has no effect. (Debugger firmware v1.25.6 or newer.)



**Info:** The content sent to the mass storage emulated disk triggers the commands listed in the table above 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 with MPLAB<sup>®</sup> X IDE.

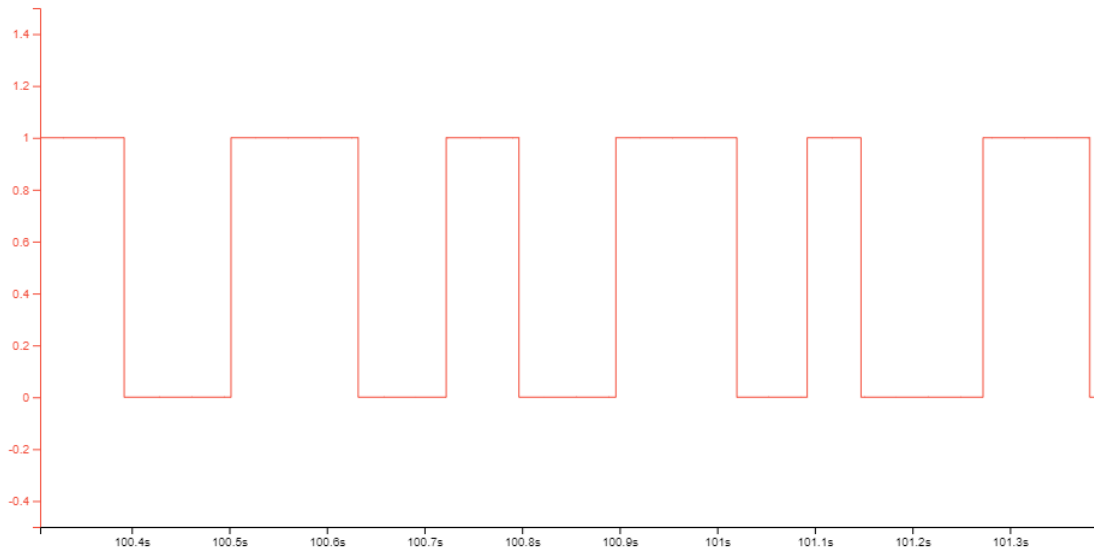
The ATtiny3227 Curiosity Nano has two available debug GPIO channels DGI GPIO0 and DGI GPIO1.

### 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 DGI Timestamp module resolution.

---

**➔ Important:** Although capturing higher frequency signal bursts is possible, the signals' frequency range where the 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 aborting the DGI session.

---

### 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 ATtiny3227 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.

---

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

---

**i 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 the new value, 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 made to its configuration.

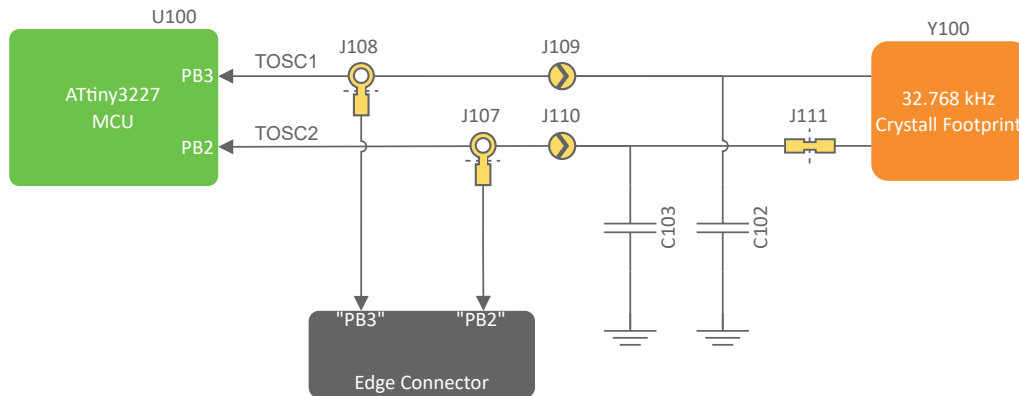
## 4. Hardware Implementation

LED, Mechanical Switch, Power Supply, Crystal.

### 4.1 Crystal

The ATtiny3227 Curiosity Nano Board has a 32.768 kHz crystal footprint for standard 3.2 x 1.5 mm surface mount crystals with two terminals. The crystal footprint is not connected to the ATtiny3227 by default, as the GPIOs are routed to the edge connector.

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



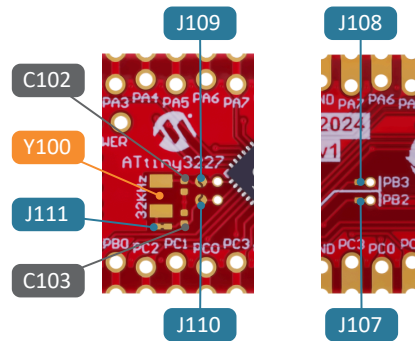
**Table 4-1.** Crystal Connections

ATtiny3227 Pin	Function	Shared Functionality
PB3	TOSC1	Edge connector
PB2	TOSC2	Edge connector

Some hardware modifications are required to use a crystal with the ATtiny3227.

- 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 PB2 and PB3 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 C102 and C103

Figure 4-2. Crystal Cut Straps



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.

## 4.2 LED

One yellow user LED is available on the ATtiny3227 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. ATtiny3227 Curiosity Nano LED0 Block Diagram

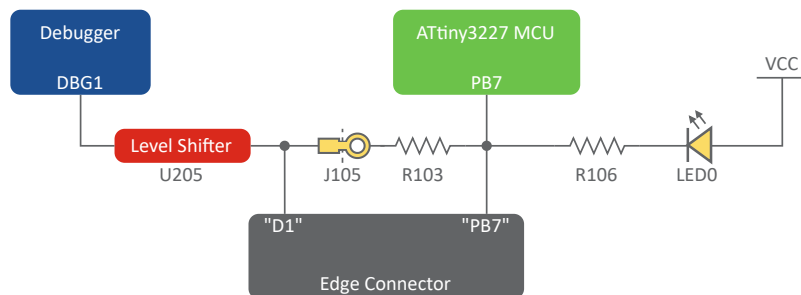


Table 4-2. LED Connection

ATtiny3227 Pin	Function	Shared Functionality
PB7	Yellow LED0	Edge connector, On-board debugger

## 4.3 Mechanical Switch

The ATtiny3227 Curiosity Nano board has one mechanical switch - a generic user-configurable switch. Pressing it will connect the I/O pin to ground (GND).

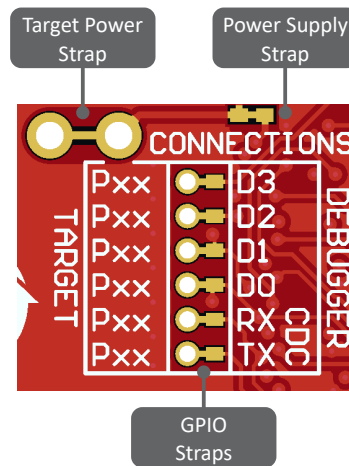


#### 4.4.1 Cut Straps

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, J103, 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 ATtiny3227 microcontroller. The voltage limits configured in the on-board debugger on ATtiny3227 Curiosity Nano are 1.8–5.5V.

**Info:** The factory default target voltage is 3.3V. The project properties in MPLAB<sup>®</sup> 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.

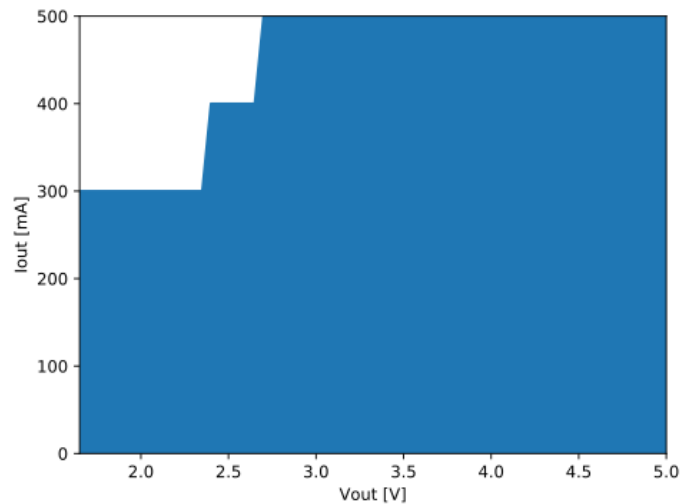
**Info:** The voltage settings setup in MPLAB<sup>®</sup> X IDE is not applied immediately to the board. Like clicking the Refresh Debug Tool Status button in the project dashboard tab or programming/reading program memory, the new voltage setting is applied to the board when accessing the debugger.

**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  $\pm 100$  mV, is suddenly applied to the VTG pin without setting the  $V_{OFF}$  pin low.

**i Info:** The on-board debugger has a monitoring window of  $V_{CC\_TARGET} \pm 100$  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 ATtiny3227 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 ATtiny3227 is 1.8–5.5V. Applying a higher voltage may cause permanent damage to the board.



**Info:** The on-board debugger monitors the voltage supplied to the board. If VOFF is not pulled low and the external power supplied differs by more than  $\pm 100$  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 within  $\pm 100$  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.



**Info:** In addition to the power consumed by the ATtiny3227 and its peripherals, approximately 100  $\mu$ A 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  $\mu$ A. Typical values may be as low as 100 nA.

#### 4.4.4 Power Supply Exceptions

This section summarizes most issues that can arise with the power supply.

##### Target Voltage Shuts Down

Not reaching the set target voltage can happen if the target section draws too much current at a given voltage, causing 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 fluctuates by over 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 it is more 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. To fix this, set the target voltage to a value within the specified voltage range for the target device.

### No Target Voltage and PS LED is Lit 2

This situation can be an 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<sub>BUS</sub> Output Voltage is Low or Not Present

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

## 4.4.5 Low-Power Measurement

Power to the ATtiny3227 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 ATtiny3227 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 ATtiny3227.



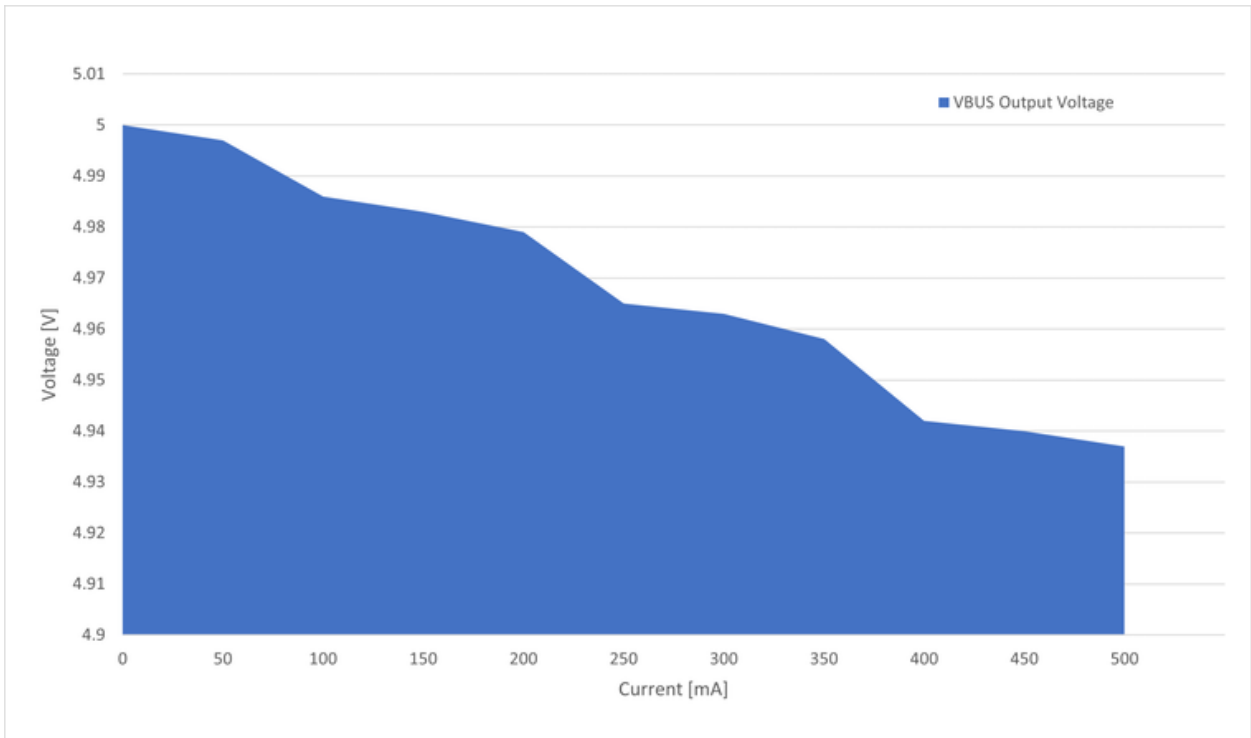
**Info:** The on-board level shifters will draw a small amount of current even when unused. Each level shifter has a maximum of 2  $\mu$ A leakage current. Therefore, the worst-case maximum current draw for the five on-board level shifters is 10  $\mu$ A. Prevent leakage current through an I/O pin connected 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 V<sub>BUS</sub> Output Pin

ATtiny3227 Curiosity Nano has a V<sub>BUS</sub> output pin that can be used to power external components that need a 5V supply. The V<sub>BUS</sub> 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 V<sub>BUS</sub>

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 ATtiny3227 Curiosity Nano: The MPLAB® X IDE Kit Window or the sticker on the bottom of the PCB.

The Kit Window appears in MPLAB X IDE when connecting ATtiny3227 Curiosity Nano to the computer.

The first nine 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. The data matrix code on the sticker contains a string with the product identifier 02-01280, revision, and serial number.

The string in the data matrix code has the following format:

```
"nnnnnnnrssssssss"
```

n = product identifier

r = revision

s = serial number

#### 5.1.2 Revision 1

Revision 1 is the initially released board revision. It has ATtiny3227 rev. A mounted.

**Known issue:** The slew rate limiting for the U208 - MIC2008 is implemented incorrectly. The capacitor on the CSLEW pin is currently connected to GND, which causes the slew rate for VCC\_P5V0 to be approximately 50 V/ms. For the correct implementation, the capacitor may instead be connected between the VIN and CSLEW pins to achieve the intended slew rate limiting.

## 6. Document Revision History

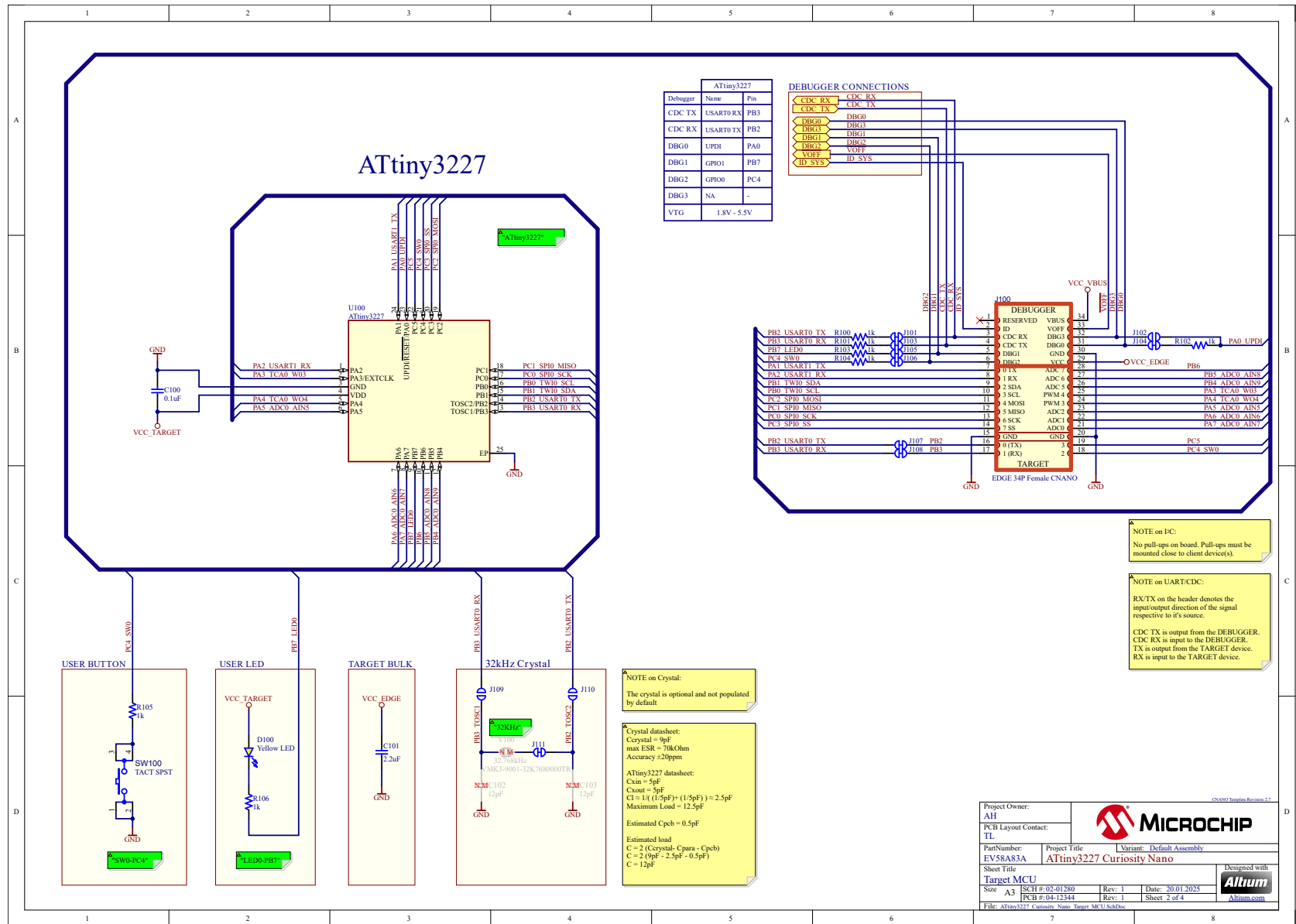
Doc. Rev.	Date	Comments
A	3/2025	Initial document release

## 7. Appendix

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

## 7.1 Schematic

Figure 7-1. ATtiny3227 Curiosity Nano MCU Schematic





## 7.2 Assembly Drawing

Figure 7-3. ATtiny3227 Curiosity Nano Assembly Drawing Top

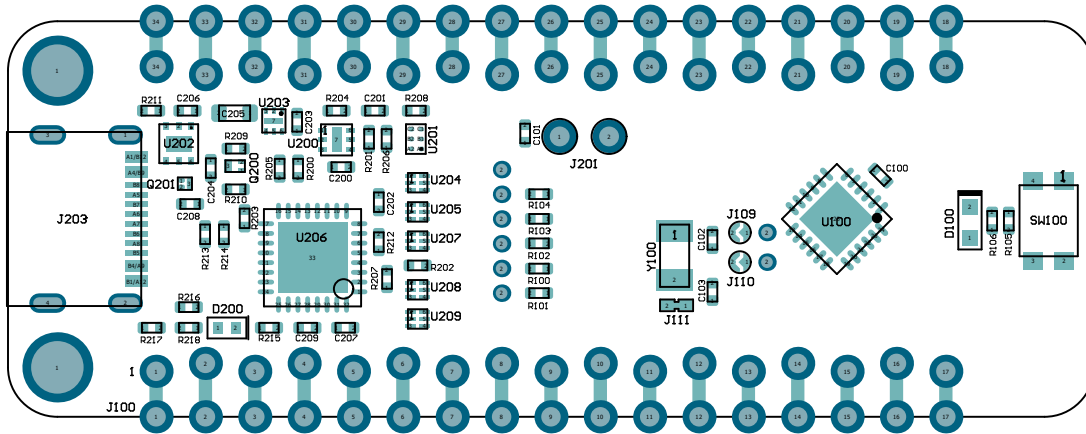
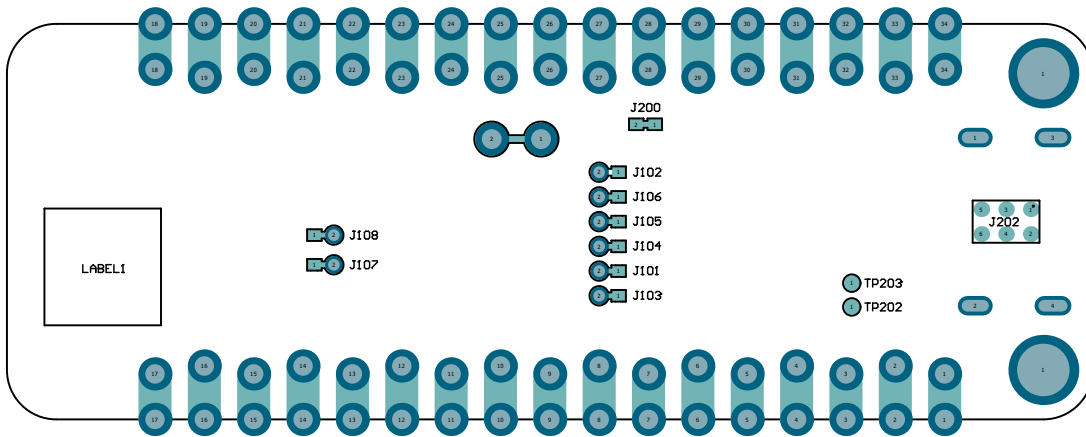
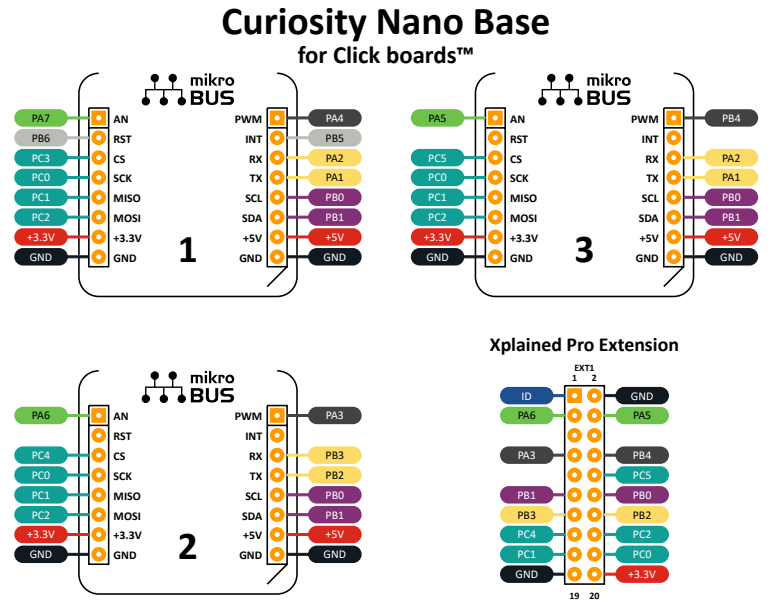
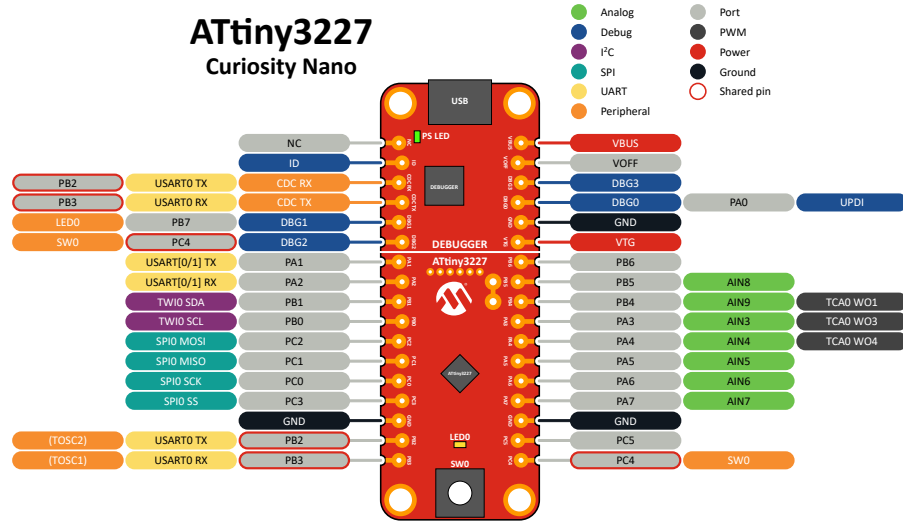


Figure 7-4. ATtiny3227 Curiosity Nano Assembly Drawing Bottom



## 7.3 Curiosity Nano Base for Click boards™

Figure 7-5. ATtiny3227 Curiosity Nano Pinout Mapping



## 7.4 Programming External Microcontrollers

Use the on-board debugger on ATtiny3227 Curiosity Nano to program and debug microcontrollers on external hardware.

### 7.4.1 Supported Devices

All external AVR<sup>®</sup> microcontrollers with the UPDI interface can be programmed and debugged with the on-board debugger with Microchip MPLAB X IDE.

External SAM, PIC16 and PIC18 microcontrollers having a Curiosity Nano Board can be programmed and debugged with the on-board debugger with Microchip MPLAB X IDE.

ATtiny3227 Curiosity Nano can program and debug external ATtiny3227 microcontrollers with MPLAB X IDE.

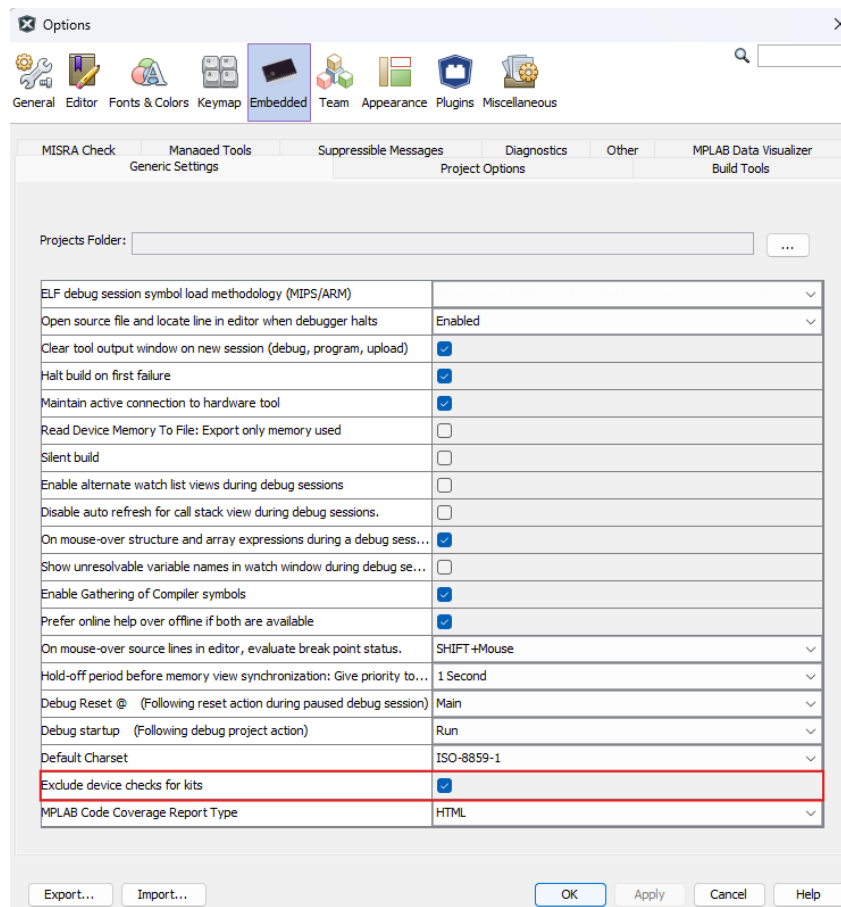
### 7.4.2 Software Configuration

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 MPLAB X IDE 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 **Embedded > Generic Settings** category in the options window.
3. Check the **Exclude device checks for kits** option.

Figure 7-6. Exclude Device Checks For Kits



**Info:** Microchip MPLAB X IDE allows any microcontroller and interface to be selected when the **Exclude device checks for kits** setting is **checked** - also microcontrollers and interfaces not supported by the on-board debugger.

### 7.4.3 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 [Figure 4-5](#).

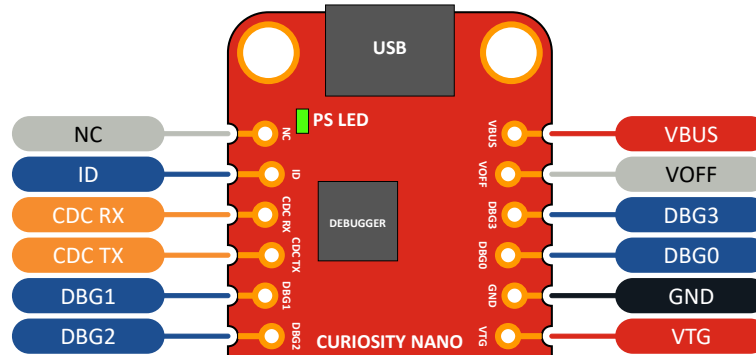
The on-board debugger and level shifters actively drive data and clock signals used for programming and debugging (DBG0, DBG1, and DBG2). Pull-down resistors are required on the ICSP™ data and clock signals to debug PIC® microcontrollers. All other interfaces are functional with or without pull-up or 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 7-7.** Curiosity Nano Standard Pinout



**Table 7-1.** Programming and Debugging Interfaces

Curiosity Nano Pin	UPDI	ICSP™	SWD
DBG0	UPDI	DATA	SWDIO
DBG1	—	CLK	SWCLK
DBG2	—	—	—
DBG3	—	MCLR	RESET

### 7.5 Connecting External Debuggers

Even though there is an on-board debugger, external debuggers can be connected directly to the ATtiny3227 Curiosity Nano to program/debug the ATtiny3227. When not actively used, the on-board debugger keeps all the pins connected to the ATtiny3227 and board edge in tri-state. Therefore, the on-board debugger will not interfere with any external debug tools.

Figure 7-8. Connecting the MPLAB® PICKit™ 5 In-Circuit Debugger/Programmer to ATtiny3227 Curiosity Nano

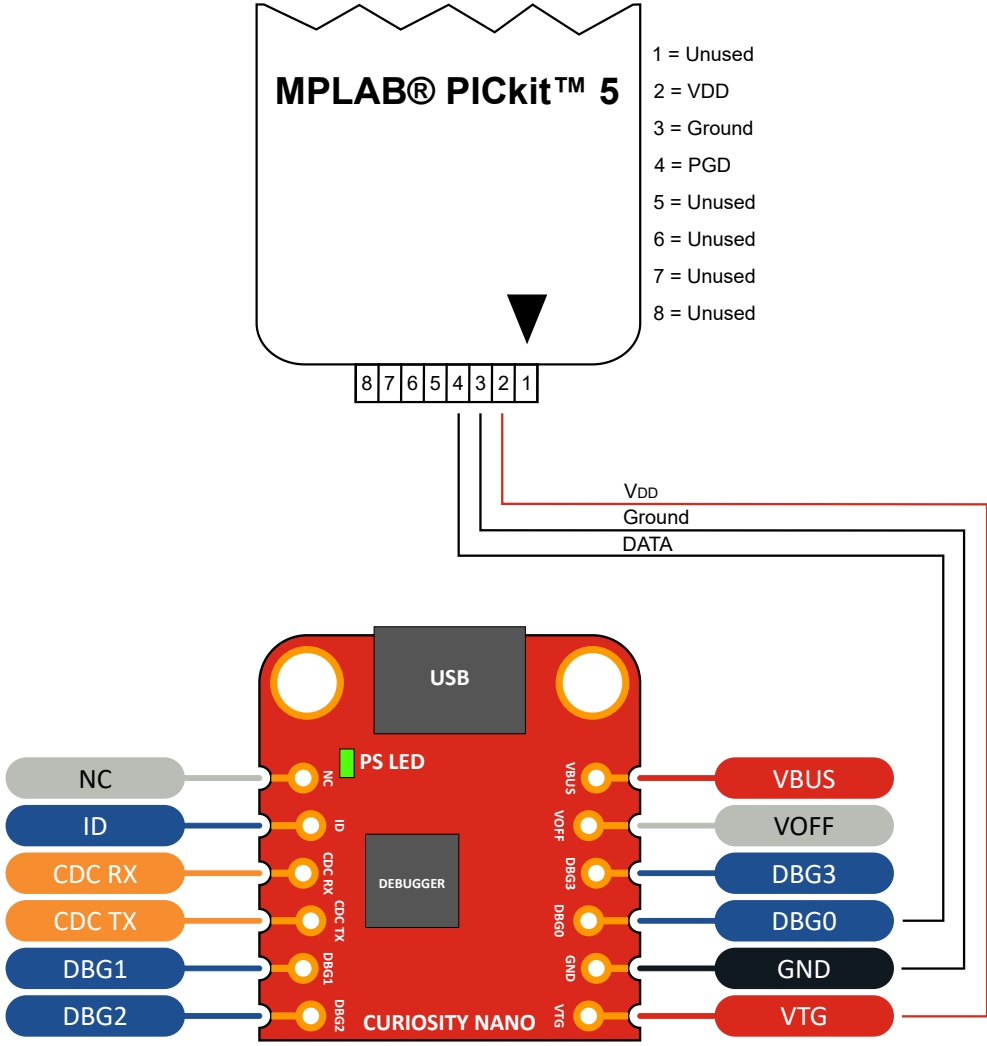
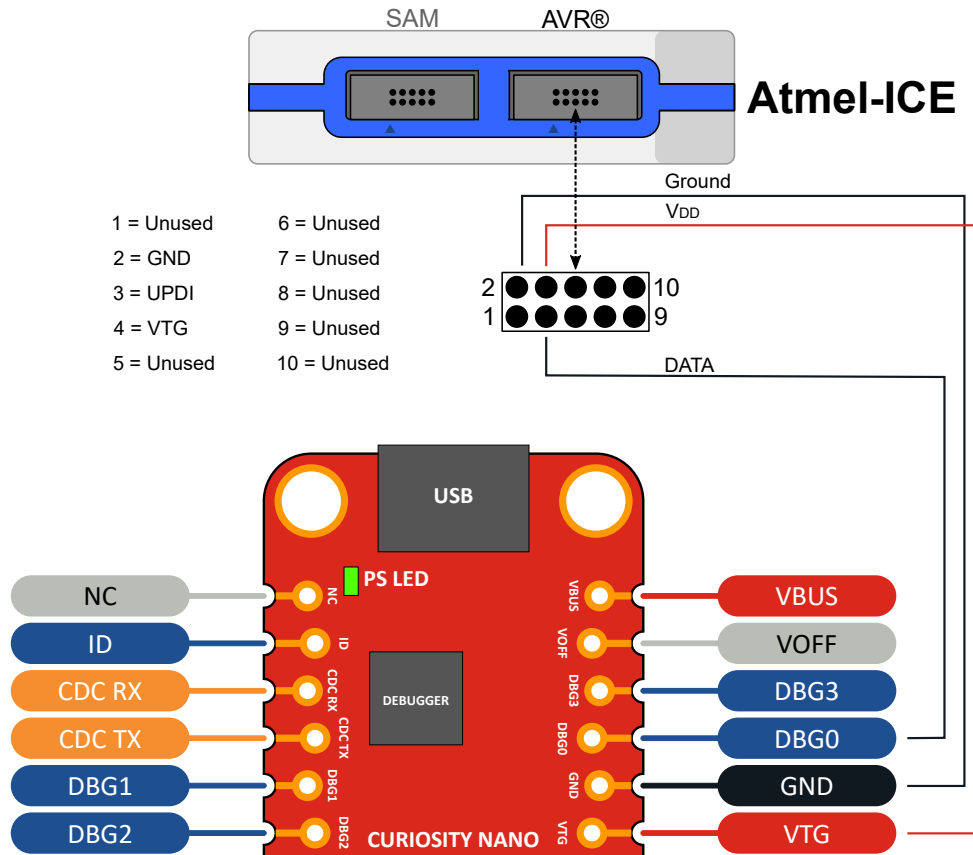


Figure 7-9. Connecting the Atmel-ICE to ATTiny3227 Curiosity Nano



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.

## 7.6 Disconnecting the On-Board Debugger

The on-board debugger and level shifters can be disconnected from the ATTiny3227.

The power supply block diagram (Figure 4-5) shows all connections between the debugger and the ATTiny3227. The signal names are printed in silkscreen on the top or bottom side of the board.


Cut the GPIO straps in Figure 4-6 to disconnect the debugger.



**Attention:** Cutting the GPIO straps to the on-board debugger prevents the virtual serial port, programming, debugging, and data streaming from functioning. Cutting the power supply strap disconnects 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.

## 7.7 Getting Started with IAR™

IAR Embedded Workbench® for AVR® is a proprietary high-efficiency compiler not based on GCC. Programming and debugging of ATtiny3227 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 7-10](#).
3. In the category **Debugger**, select the **Setup** tab. Select **Atmel-ICE** as the driver, as shown in [Figure 7-11](#).
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 7-12](#).

 **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 7-10.** Select Target Device

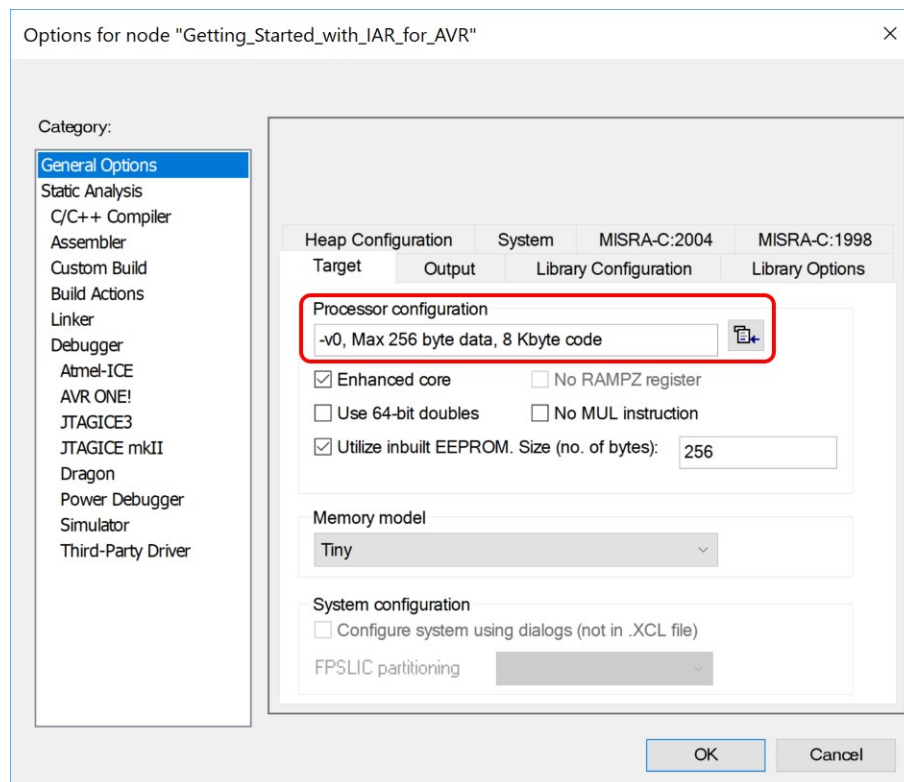


Figure 7-11. Select Debugger

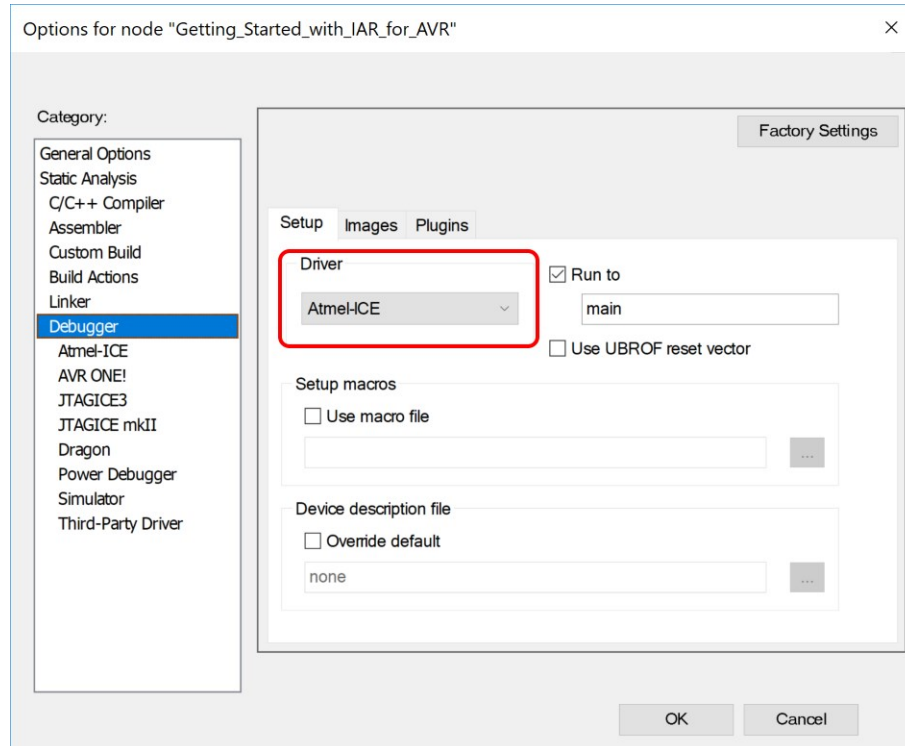
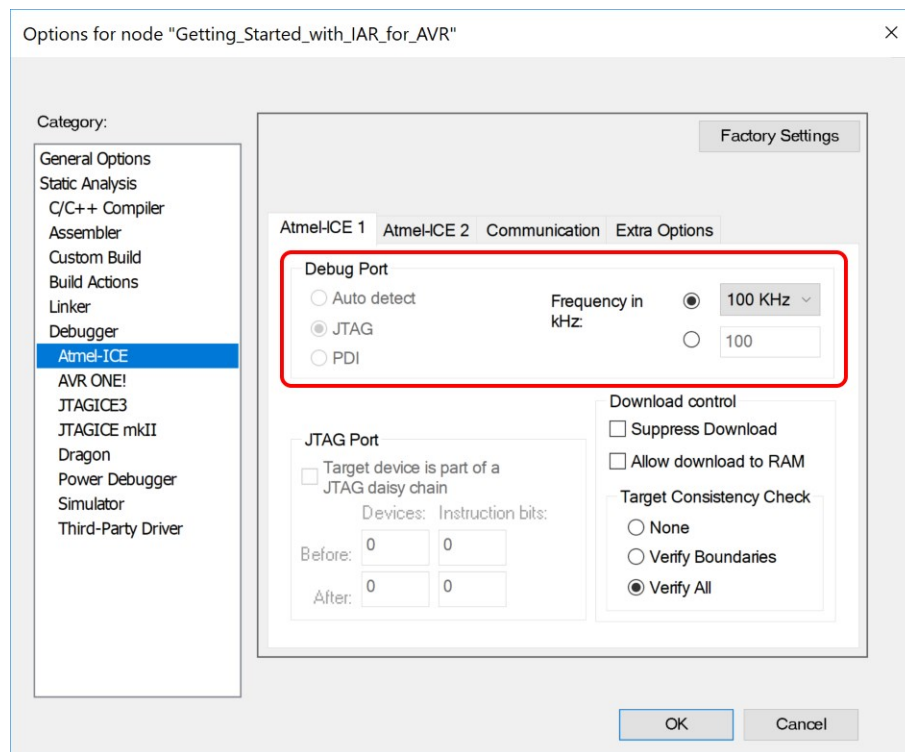


Figure 7-12. Configure Interface



## Microchip Information

### Trademarks

The “Microchip” name and logo, the “M” logo, and other names, logos, and brands are registered and unregistered trademarks of Microchip Technology Incorporated or its affiliates and/or subsidiaries in the United States and/or other countries (“Microchip Trademarks”). Information regarding Microchip Trademarks can be found at <https://www.microchip.com/en-us/about/legal-information/microchip-trademarks>.

ISBN: 979-8-3371-0833-9

### 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](http://www.microchip.com/en-us/support/design-help/client-support-services).

THIS INFORMATION IS PROVIDED BY MICROCHIP “AS IS”. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP’S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

### 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 products are 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.