**https://github.com/SpenceKonde/AVR-Guidance/blob/master/UPDI/jtag2updi.md**

**将串行适配器或 uno/nano/pro mini 变成 UPDI 编程器的分步指南**

tinyAVR 0/1/2 系列、megaAVR 0 系列和 AVR Dx 系列部件通过统一编程和调试接口 (UPDI) 进行编程。这是使用 AVR Dx 系列部件上的 UPDI 引脚的 1 线接口。需要 UPDI 程序员来更改保险丝（“烧录引导加载程序”）、上传引导加载程序（如果需要）并在不使用引导加载程序时上传草图。不能使用经典的 ISP 编程器——只能使用 UPDI 来对这些部分进行编程。幸运的是，做一个 UPDI 程序员比做一个 ISP 程序员要容易得多！

有两种非常简单的方法可以以 3 美元或更少的价格获得 UPDI 编程硬件。

制作 3 针电缆时，我们建议使用此引脚排列：

UPDI - GND - VCC（或等效地，VCC - GND - UPDI）。

原因很简单，如果向后插入，它可以最大限度地减少损坏的机会。

**Serial-UPDI：来自串行适配器的 UPDI 编程器（推荐）**

从 DxCore 1.3.0 和 megaTinyCore 2.2.6 开始，现在可以使用串行适配器和肖特基二极管！（或 4.7k 电阻 - 但二极管效果更好）从 megaTinyCore 2.3.2 和 DxCore 1.3.6 开始，它们比 jtag2updi 快得多，并且是推荐的编程方法。

**所需组件**

1. USB 串行适配器在 ebay 和 aliexpress 基于 CH340G 上的价格低至 1 美元，CP210 的价格略高。理想情况下，您希望将串行适配器专用于此目的以方便使用，而不是每次您想要使用它时都必须连接和断开连接。
2. 快速信号肖特基二极管，例如 1N4148 或许多其他二极管中的任何一个（推荐）或电阻器（请参阅下文以计算值））。
3. 1 个电阻，几百欧姆 - 220 到 1k，甚至 2k 也可以（可能不需要）。
4. 几根跳线。

**串行适配器要求**

几乎任何串行适配器都可以用于 pyupdi 风格的程序员，只要您注意避免这些陷阱：

1. FTDI FT232（包括正品和假货）默认配置为通过减少轮询适配器使用更少的 CPU 时间。对于 UPDI 编程，性能影响是严重的。您可以轻松解决此问题（至少在 Windows 上 - 我不确定该问题是否发生在 Linux 上）：打开设备管理器，在端口（COM 和 LPT）下找到 FTDI 适配器。右键单击-> 属性。单击端口设置选项卡，然后单击高级按钮。在左侧中间，有一个名为“延迟计时器”的选项，可能设置为 16 毫秒。将其设置为 1 毫秒）。单击“确定”足够次数以离开对话框。您将看到适配器消失并在应用更改时重新出现。正确配置后，FT232RL 具有出色的性能。配置不当，性能是彻头彻尾的糟糕。
2. 大多数串行适配器已经有自己的电阻器与 Tx 串联！通常在 1k 到 2.2k 之间；如果使用推荐的二极管方法，则只需添加二极管，而无需添加第二个电阻。如果由于某种原因您不使用二极管方法，请使用一个总电阻为 4.7k 的电阻器（除非您可以跟踪电阻器的轨迹并读出代码（三个），否则您需要用万用表测量它数字代码 - 前两个数字是最高有效数字，第三个数字是它们后面的零数 - 222 是 2200 - 2.2k，471 是 470，（这当然意味着 101 表示 100 欧姆电阻，以及100 表示一个 10 欧姆的电阻！））有三种方法可以通过将万用表的一端夹在 Tx 上来测量所有数据。要么在线查找串行适配器芯片的引脚排列并测量该引脚的电阻，检查芯片上的所有引脚。许多可能具有非常高的电阻，但只有一个的电阻值只有几 k 或更小，只需选择电阻器的两端，期望一个一侧的 Tx 为 1-2.2k，另一侧为 0 /（如果你真的遇到 Tx 和芯片之间没有电阻器的，请告诉我！我只知道一种在野外流传的设计：它是绿色 PCB 上的 CH340G，带有 3.3v/5v 开关和 microUSB 端口；那个板似乎是某人的第一个设计——尽管如此，他们多年来一直在生产而没有修改。该设计有 2 个非常理想的功能——电压开关和微型 USB——以及至少 4 个严重的设计缺陷。）我只知道一种在野外流传的设计：它是绿色 PCB 上的 CH340G，带有 3.3v/5v 开关和 microUSB 端口；那块板子似乎是某人的第一个设计——尽管如此，他们多年来一直在生产而没有修改。该设计有 2 个非常理想的特性——电压开关和微型 USB——以及至少 4 个严重的设计缺陷。）我只知道一种在野外流传的设计：它是绿色 PCB 上的 CH340G，带有 3.3v/5v 开关和 microUSB 端口；那块板子似乎是某人的第一个设计——尽管如此，他们多年来一直在生产而没有修改。该设计有 2 个非常理想的特性——电压开关和微型 USB——以及至少 4 个严重的设计缺陷。）
3. 一些串行适配器有一个专用 LED 来指示 Rx。虽然一些花哨的芯片有一个 I/O 引脚来驱动 RX LED（例如 FT232），但带有 RX LED 的廉价适配器可能只是在 RX 线上放置了一个 LED 和电阻器（实际上这就是上面提到的带有开关和 microusb 端口的绿色板子做了。。来自 UPDI 线上的 LED 的负载将压倒来自目标的任何信号并阻止通信（TX 上的 LED 很好 - 适配器具有足够的驱动强度。）

我推荐 bog 标准 CH340G 作为首选串行适配器，原因如下：

* 它们非常便宜，而且很容易在 ebay/aliexpress/amazon 上买到。7/5 美元在 aliexpress 上运送，用于最基本的设计（黑色那种，末端带有愚蠢的电压跳线。带有圆角和 ENIG 表面处理的较宽的那些在电气上并没有更好，尽管构建质量必须更好（第一个那种你应该计划将 USB 连接器焊接得更牢固），并且体积更大。更好的是你在 ebay 上搜索“CH340 6pin”时找到的 - 你想要带有电压开关的黑色。直到，那是，我将自己的设计投入生产 - 它将*正确*完成电压切换 和一个微型 USB 连接器，而不是一个愚蠢的全尺寸 USB-A、全尺寸 1117 3.3v 稳压器和一个颜色编码的电源灯来指示当前电压设置，并且总体上没有明显的设计缺陷。
* 裸芯片非常便宜且易于获得 - 而且设计起来也非常简单。任何人，甚至是设计那些我一直在骂的绿板的小丑，都可以做到（坦率地说，这看起来像是某人的第一块板）。哎呀，一个有效的 CH340G 串行适配器也是我设计的第一块板（至少我没有尝试出售我的第一个版本！）。
* 它们 - 最多 - 写入完整闪存图像慢半秒。对于一个真正的 FT232RL 适配器的价格，您可以拥有六个 CH340。

而且因为替代品都有问题：

* 大多数出售的FT232RL板都是假的。我有几个订单可以与我的真实产品进行比较。如果它们的性能相当，我会更新这个（尽管即使是假货也是 CH340 价格的两倍多）
* CP2102 不支持 Dx 系列的最高速度（345600 波特） - 或者，如果您使用 Silicon Labs 的实用程序将其配置为支持最高速度，您将失去对 tinyAVR 460800 的支持。
* HT42B534 有多种不同寻常的属性，有好有坏。
  + 它不支持任意波特率——Dx 系列不支持 345600，或 256 到 460 之间的任何波特率。
  + 一方面，它使用标准的 windows CDC 驱动程序；这并不是一件坏事——我从来没有发现为其他适配器安装串行驱动程序是一个可怕的负担。不幸的是，它们不能很好地工作。将它设置为它不喜欢的速度不仅不会产生预期的效果，而且尝试会阻止而不是优雅地终止。serialupdi 具有相当短的内置超时和错误；许多终端程序会挂起，直到您重新启动它们或拔下串行适配器并重新插入。有时拔出后它会留在设备管理器中，尤其是在它挂起之后。
  + 调制解调器控制输入错误：全部显示，直到任何一个改变状态，然后都显示与实际值相反。反向行为一直持续到下一次重置。
  + 该芯片（虽然不是可用的适配器）有一个 VDDIO 引脚来设置输出电压，即内置电平转换器。CH340只能做3.3V-5V，HT42可以做1.8-5V。如果它暴露在分线板中，那将是一个非常好的功能；
  + 他们几乎是 CH340 便宜。
  + 延迟明显低于其他串行适配器。早在 2021 年春季优化工作期间，在针对写入 Dx 系列的“更简单情况”进行优化后（更简单，因为页面更大，因此花费的时间块更大，上传速度在很大程度上决定了通过延迟，而不是原始速度。这个版本在 HT42B534 上的表现明显更好。随后的优化使得其他适配器的速度几乎与 HT42B534 的速度相等，而没有其他的怪癖。

**关于分线板的说明**

一些 tinyAVR 和其他基于 UPDI 的部件分线板具有板载电阻器。有时这是一个 4.7k 的。这是不合适的。有一段时间我是问题的一部分。我认为最初的错误来自人们将 pyupdi 电阻器与一般合适的电阻器混为一谈。当我开始 megaTinyCore 时，我的早期合作者正在制作带有 4.7k 电阻的硬件；我以为他做对了。虽然这确实适用于专用程序员，包括 jtag2updi，但它不适用于串行 UPDIO。只要他们没有自己的电阻器，它就可以与像 jtag2updi 这样的专用程序员一起使用。可以说，有一段时间这是一个非常普遍的信念。我现在使用 470 欧姆，但我找不到它的设计有什么问题，根本没有。

**连接：**

* Vcc, 串行适配器的 Gnd 到 Vcc, 目标的 Gnd
* 适配器的 Tx 和 Rx 之间的 4.7k 电阻器（许多适配器内置 1k、1.5k 或 2.2k 电阻器与 Tx 串联；这些应使用成比例较小的电阻器）
  + 为了获得更好的结果，可以使用一个较小的电阻器（上面提到的大多数适配器上的内置电阻器，在这里可以完美地工作）和一个小的肖特基二极管（带向 Tx，另一端连接到 Rx）（使用“小信号二极管”——较大的通用二极管可能具有使其不太适合此的特性）该二极管显着扩大了这种编程方法的容差，并显着提高了可靠性。
  + 我的首选是 BAT54C,235；它在一个很小的 ​​SOT-23 封装中（它是 2 个二极管，两个二极管都指向一侧单独的引脚）为什么？因为，假设您的串行适配器的引脚位于 0.1" 接头上，并且 TX 和 RX 彼此相邻（两者都非常常见），则二极管正好适合它们。并且没有导致以后疲劳和断裂的引线，结果会更小可能会因粗暴处理而损坏。然后如果我想更明显地将其标记为 UPDI 编程器，我可能会切断 Tx、DTR 和 CTS 引脚；永远记住，您可以在 ebay 上花一块钱买到串行适配器。
* 适配器的 Rx 到目标的 UPDI 引脚。一个小电阻（低于 1k - 就像我们通常推荐的 470 ohm 一样）与它串联就可以了。

USB Serial Adapter

With internal 1-2k resistor on TX

This is the case in 90% of USB serial adapters.

Ideal:

internal resistor in adapter: 2.2k >= Ra

resistor on target: 2k >= Rt >= 100 (100-470 rec.)

-------------------- To Target device

DTR| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

internal Rx |--------------,------------------| UPDI---\/\/---------->

Tx---/\/\/\---Tx |-------|<|---' .--------| Gnd 470 ohm (100 ~ 1k)

resistor Vcc|---------------------------------| Vcc

typ 1-2k CTS| .` |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Gnd|--------------------' If you make a 3-pin connector, use this pinout

--------------------

Near Ideal:

Yes internal resistor in adapter: 2.2k >= Ra

No resistor on target

-------------------- To Target device

DTR| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

internal Rx |--------------,------------------| UPDI----------------->

Tx---/\/\/\---Tx |-------|<|---' .--------| Gnd

resistor Vcc|---------------------------------| Vcc

typ 1-2k CTS| .` |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Gnd|--------------------'

--------------------

Yes internal resistor on adapter

Yes resistor on target: small, under 2k

Also works great, convenient if still using jtag2updi without resistor built into it. Resistorss should sum to less than 4.7k, preferalby much less

-------------------- To Target device

DTR| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

internal Rx |--------------,------------------| UPDI----\/\/\/---->

Tx---/\/\/\---Tx |-------|<|---' .--------| Gnd < 2.2k

resistor Vcc|---------------------------------| Vcc

typ 1-2k CTS| .` |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Gnd|--------------------' Series resistor between header and chip UPDI pin on target PCB

-------------------- I use 470 ohm resistors.

Yes internal resistor on adapter

Yes resistor on target: several k or more:

This will often by 4.7k: it must be bypassed, replaced with a smaller one or shorted out.

-------------------- To Target device

DTR| ,----------------------------------.

internal Rx |--------------/ | UPDI----\/\/\/--\*-

Tx---/\/\/\---Tx |-------|<|---' .--------| Gnd 4.7k

resistor Vcc|---------------------------------| Vcc

typ 1-2k CTS| .' |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Gnd|--------------------'

--------------------

No internal resistor on adapter.

Yes resistor on target >= 100 ohms and not more than a few k.

-------------------- To Target device

DTR| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

No resistor? Rx |--------------,------------------| UPDI----\/\/\/------>

Are you sure? Tx |----|<|------` .--------| Gnd > 100

This is rare! Vcc|---------------------------------| Vcc < 2.2k

CTS| .` |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Resistor of around a few hundred to a few k

Gnd|--------------------'

--------------------

No resistor on target OR adapter:

Include A resistor in ONE of the three places shown below, whichever is easier to wire in,

the first and second positions, with nothing between Rx and target UPDI, are slightly preferable

Value of resistor should be a few hundred ohms, I'd default 470.

No internal resistor

-------------------- To Target device

DTR| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

No resistor? Rx |---------------------,--/\/\-----| UPDI---------------->

Are you sure? Tx |--/\/\---|<|----\/\/' .---| Gnd

This is rare! Vcc|---------------------------------| Vcc

CTS| .` |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Gnd|-------------------------'

--------------------

Resistor-based schemes - these have a narrower window of parameters under which they work reliably.

If you look at the UPDI line on a 'scope while it is malfunctioning, you will see that sometimes

the voltage is not going all the way down to ground when one side tries to assert it.

They are not recommended unless there is something keeping you from using a diode cofiguration.

The PyUPDI classic:

4.7k resistor

No internal resistor

-------------------- To Target device

DTR| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

No resistor? Rx |--------------,------------------| UPDI---------------->

Are you sure? Tx |--/\/\/\/\---` .--------| Gnd

This is rare! Vcc|---------------------------------| Vcc

CTS| .` |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Gnd|--------------------'

--------------------

Very much like the classic, except for the possiility of a resistor on the target. Must be 470 or under on the target.

Resistance should sum to 4.7k

-------------------- To Target device

DTR| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

internal Rx |--------------,------------------| UPDI----\/\/\/------>

Tx---/\/\/\---Tx |--/\/\/\/\---` .--------| Gnd =< 470

resistor Vcc|---------------------------------| Vcc \If resistor present, not more than 470 ohms.

typ 1-2k CTS| .` |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Gnd|--------------------'

--------------------

If the resistor on the target is much more than 470 ohms, you're going to want to bypass it. Alternately,

it may be easier to replace it with a 0 ohm resistor or bridge it with a piece of wire or even

just a blob of solder, and do the classic pyupdi. Note that using a diode will work with resistances oin the target

that are too much for it to work using a resistor.

The resistor (if any) in serial adapter, and the one you add should total 4.7k.

-------------------- To Target device

DTR| ,--------------------------.

internal Rx |--------------,-------' | UPDI----\/\/\/- \*--->

Tx---/\/\/\---Tx |--/\/\/\/\---` .--------| Gnd >470

resistor Vcc|---------------------------------| Vcc

typ 1-2k CTS| .` |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \resistor of more than around 470 ohms - must be bypassed, replaced, or shorted.

or no resistor Gnd|--------------------'

--------------------

If there's no resistor in the serial adapter and the target happens to have a 4.7k resistor, you can do it without

any extrta components, though you've got 4 wires involved instead of 3:

-------------------- To Target device

DTR| .-----------------------------------.

No resistor? Rx |-------------' ,------------| UPDI----\/\/\/--\*---->

Are you sure?-Tx |-------------------' .-------| Gnd 4.7k

This is rare! Vcc|---------------------------------| Vcc

OR resistor CTS| .' |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

bypassed Gnd|---------------------' Excessively - but conveniently - sized resistor.

-------------------- These were (incorrectly) popularized for the first few years of UPDI

为什么我们在做二极管方法时需要一点电阻（几百个）？经典的反应是“如果目标和程序员完全不同步并尝试以相反的方向驱动线路”请注意，在 tinyAVR 上这不是一个问题，因为 UPDI/reset 引脚上的输出驱动器是如此弱以至于它们似乎不太可能超过每个引脚规格的最大电流。除此之外，还有另一个原因：因此，当（不是如果）您将连接器向后插入，但目标也有外部电源时，就不会有适配器的 TX 引脚因尝试驱动而损坏的风险正供应轨低位；

**软件**

从支持此上传方法的内核（megaTinyCore、DxCore 以及即将推出的 MegaCoreX）的 Tools -> Programmer 菜单中选择 Serial-UPDI 选项。tinyAVR 内核提供 57600 波特、230400 波特和 460800 波特选项。DxCore 提供 57600、230400、345600 和一个“特殊”的 460800 w/chunking 用于 HT42B534 串行适配器，它不能达到 345600 波特（和任何其他适配器使用这个选项运行得更慢）并从工具 -> 端口菜单。

请注意，这不会为您提供串行监视器 - 您需要以正常方式连接串行适配器（我建议使用两个，以及外部串行终端应用程序）。这种技术适用于来自 ebay、aliexpress 等的 1 美元 CH340 串行适配器。您是否不小心购买了一些没有损坏 DTR 引脚的适配器，因此对您希望与它们一起使用的 2 美元 Pro Mini 不是很有用? 他们是完美的。虽然 CH340 部件的性能最低，但部件之间的差异现在非常小（在 2.3.2 版本之前的优化之前不是这种情况，在此之前，速度从每秒 125 字节到略高于 1k .

**上传并验证性能**

| **波特率** | **FT232RL kb/s** | **CP2102 kb/s** | **CH340 kb/s** | **HT42B534 kb/s** |
| --- | --- | --- | --- | --- |
| 115200 | 8.7 W / 8.8 R | 8.7 W / 8.8 R | 8.4 W / 8.5 R | 9.0 W / 9.1 R |
| 230400 | 16.6.W / 16.4 R | 16.3 W / 16.5 R | 14.6 W / 16.6 R | 17.7 W / 17.9 R |
| 345600\* | 24.3 W / 23.4 R | 23.2 W / 23.0 R | 22.5 W / 22.1 R | 不支持 |
| 460800\*\* | 不适用 | 不适用 | 不适用 | 24.7W / 32.7 R |
| \*\* HT42B534 使用 32 字节块大小运行，以有限块大小运行导致其他部分成功传输，尽管阈值块大小不同 - 但总体速度大幅下降，类似于 115200 波特。以 460800 波特的速度超过了 NVM 控制器的写入速度 - 我只需要看看它与 FT232RL 相比如何。它们都在芯片向闪存写入数据的能力极限下运行 - FT232RL 不需要采取任何特殊措施，也可以与 tinyAVR 部件配合使用。另一方面，HT42B534 以 230400 波特的（1.3.6 新版本）默认值领先，而且非常便宜（CH340 级别的价格）。 |  |  |  |  |

作为比较，在Dx系列部分（由于它们具有更多闪存，因此更易于用作测试对象，因此上传时间更长且更容易计时。这些数字是使用128k测试图像拍摄的，这是最佳情况。

| **程序员** | **读** | **写** | **笔记** |
| --- | --- | --- | --- |
| jtag2updi | 6.6 KB/秒 | 5.9 KB/秒 | 在 16 MHz Nano 上运行 |
| 好奇号纳米 | 5.9 KB/秒 | 3.3 KB/秒 | 通过 avrdude - 这可能并不理想 |
| Optiboot DX | 10.6 KB/秒 | 6.9 KB/秒 | 115200 波特由 DxCore 提供 |

由于较小的页面尺寸，ATtiny 部分的编程速度较慢；页越小，数据速率越慢——但是对于较小的部分，对整个闪存进行编程的时间仍然更少，因为要写入的数据更少。这两个写入编号分别用于具有 64 字节和 128 字节页面的部分。

| **波特率** | **FT232RL kb/s** | **CP2102\* kb/s** | **- CH340 kb/s** | **HT42B534 kb/s** |
| --- | --- | --- | --- | --- |
| 115200 | 4.6、6.2 瓦/8.8 瓦 | 4.3,6.2 瓦 / 8.8 瓦 | 3.6、5.2 瓦/8.5 瓦 | 3.6,7.2 W / 9.1 R |
| 230400 | 7.5,10.0 W / 16.5 R | 7.1,9.6 W / 16.4 R | 4.9、7.8 瓦/15.6 瓦 | 错误输出 |
| 345600\* | 9.1,13.6 W / 23.4 R | 8.5,13.2 瓦 / 23.1 瓦 | 5.6, 9.5 瓦 / 22.0 R | 不支持 |
| 460800 | 10.4,15.6 瓦 / 28.2 瓦 | 未测试 | 6.0,10.4 瓦 / 26.8 R | 错误输出 |

* 默认情况下，CP2102 不支持 256kbaud 和 460800 波特之间的任何速度 - 但是来自 Silicon Labs 的免费配置实用程序可以在每个请求速度范围内自定义波特率（但遗憾的是，您无法定义这些范围）。我将我的重新配置为 345600 波特，用于开发 Dx 系列部件，但在 460800 下不起作用，并且没有费心将其恢复为出厂设置只是为了填写表格；我希望看到大约 10kb/s 和 15kb/s 的写入速度以及大约 26kb/s 的读取速度。

**速度观察**

以下是根据上面的数字，不包括测试数据，以及示波器轨迹的检查。请注意，Dx 系列的 460800 波特数据不能直接比较，因为它使用块大小人为地减慢写入速度，以防止芯片以恒定速率清空的目标（几乎不是缓冲区）上的缓冲区溢出。

大约，写入时间可以建模为： T = ceil(size / pagesize) \* C + size / (baud in bits/sec / 12 bits/byte) 其中 C 又可以建模为： C = (c \* n) + d/(波特率/12)

12 位到 1 个字节，因为 UPDI 协议使用 1 个奇偶校验位和 2 个停止位。

相同的关系 - 使用不同的常数 - 用于读取和写入，读取块的大小代替页面大小。

d是设置每个页面所涉及的以字节为单位的开销；这取决于相关部件的系列。

c是取决于串行适配器的特性时间 - USB 延迟。这实际上不是一个一致的、恒定的值；它的潜在值似乎是量化的，上传过程中的某些 USB 延迟暂停具有不同的持续时间。

n在设置每个页面写入所需的多个单独 USB 事务中，编程工具使用的程序的一个功能。大多数优化都涉及减少这个数字；读取和 Dx 系列写入为 2。对于 tinyAVR 和 megaAVR，写入时为 4。

监视范围内的事务，可以暂停小交换 p（小开销）->（延迟），每页重复 n 次，然后写入持续一段时间与（页面大小 + d）成正比

在感兴趣的区域内，波特率相对于 d 较大，因此 C 几乎恒定

在低波特率下，T倾向于与波特率成反比。在高波特率下，仅T趋向于大小、NVM 版本和串行适配器的函数。克服此限制需要减少每页写入或块读取的 USB 事务数。

T read在所有部分都相同；通过将 N 从 2 一直降低到 1，未实现的更改可能会将其推低一点。然而，实际影响非常小；在性能提升推送结束后，T read，512b 块中的读取以第二项为主。

对于上表所示的 Dx 系列部件，经过全套优化后，T read = T write（此处可能发生相同的变化；改进可能是 10%）

对于 tinyAVR，T read与 Dx 相同，但 T write差得多——不仅页面大小更小，而且是n4 而不是 2，因此C是两倍高。在 460800 波特时，第二项大约是写入第一项的幅度的三分之一（并且微小将接受 691.2 kbaud 的写入）。要记住的重要一点是，这些芯片仍然在几秒钟内编程 - 并且在这些暂停之一正在进行期间正在提交页面写入。

**jtag2updi - Nano 或 Pro Mini 的 UPDI 程序员**

一种完全不同的方法是使用 Arduino 草图可将基于 ATmega328(p) 的 Arduino，如 Arduino UNO 和 Nano，转变为 UPDI 编程器（它不适用于基于其他部件的电路板，如 32u4（Micro /Leo) 或任何非 AVR 板)。以下步骤显示了如何制作这些程序员之一。虽然这是我们以前推荐的编程器，但 SerialUPDI 现在更可靠，构建成本更低，并且比 jtag2updi 性能更好，即使在小页面大小应该为 jtag2updi 提供优势的 tinyAVR 部件上也是如此。jtag2updi 的一个主要限制是它被限制为所用波特率的大约一半，因为它必须通过软件串行端口重新传输到目标......而波特率又被限制为 115200 波特，8 位

**第 1 部分：将草图上传到您的 Arduino**

1. 可以在此处找到 UPDI 程序员草图：[https](https://github.com/SpenceKonde/jtag2updi) : [//github.com/SpenceKonde/jtag2updi](https://github.com/SpenceKonde/jtag2updi) 下载并解压缩，或将 repo 克隆到您的本地机器。
2. 浏览到下载位置并打开 jtag2updi 文件夹
3. 打开草图 jtag2updi.ino 并将其上传到您的 Arduino。.ino 文件本身是空的，这很好 - 所有代码都包含在同一文件夹中的其他文件中，但需要空的 .ino 以便 IDE 可以编译它。

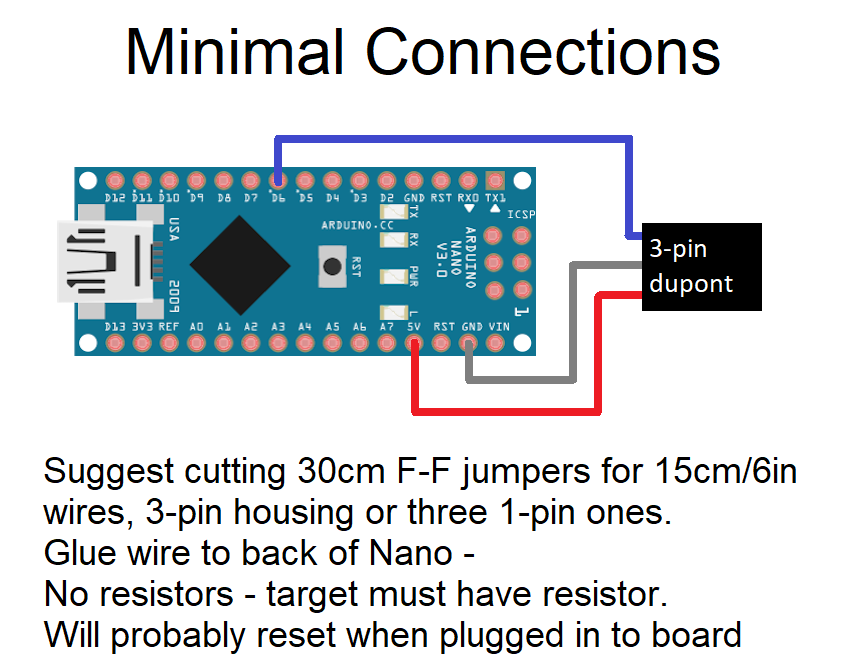
**第 2 部分：连接硬件**

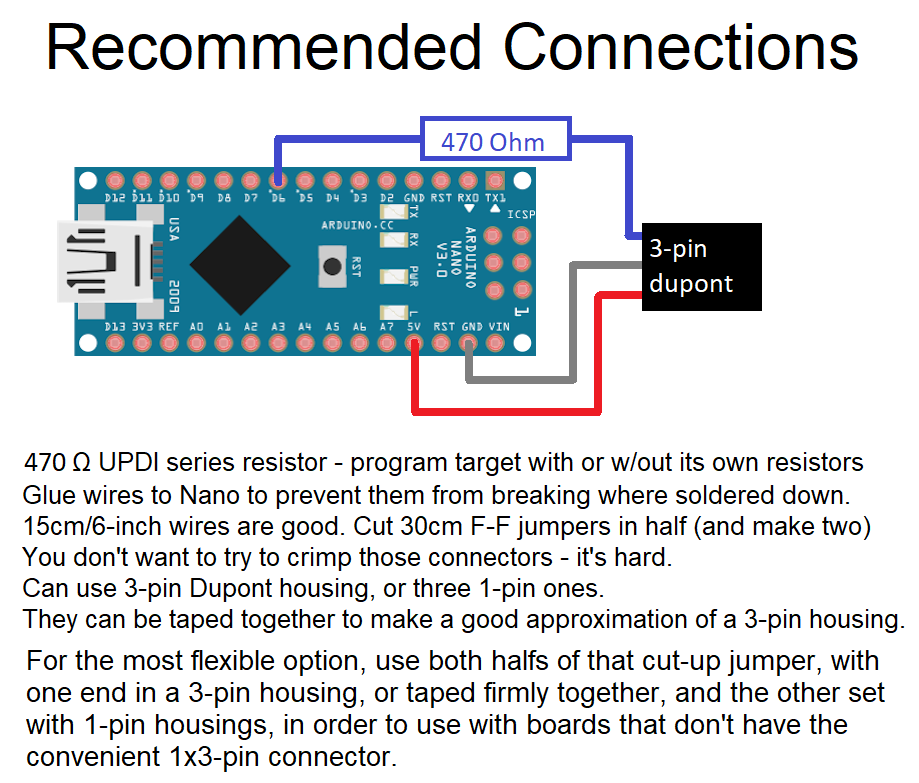
*本指南的先前版本指定了编程后复位和接地之间的上限。测试表明这是不必要的*

1. 将 Arduino 的地连接到 ATtiny 的地
2. 将 Arduino 的引脚 6 连接到 ATtiny 的 UPDI 引脚 - 如果使用裸芯片，请通过[470 ohm 电阻器连接](https://github.com/SpenceKonde/AVR-Best-Practices/blob/master/HardwareNotes/UPDISeriesResistors.md)。许多分线板将提供一个单独的 UPDI 引脚，该引脚内置了该电阻器；在这种情况下，该引脚可以直接连接到编程引脚。
3. 除非 ATtiny 有自己的电源，否则将 Arduino 的 5v 引脚连接到 ATtiny 的 Vcc 引脚

现在，您应该可以从 Tools -> Board 中选择 ATtiny megaAVR 系列板，并通过 IDE 上传草图。同样的编程器也可以用来烧录Bootloader（一定要从Tools -> Programmer 菜单中选择jtag2updi (megaTinyCore) 编程器）

**如果该过程在上传开始时似乎挂起，请按下并释放 UPDI 程序员上的重置按钮，**我在 jtag2updi 上敲了一个月，试图摆脱所有这样的错误，最后将我的修复合并到，发现不知何故，这仍然可能发生。

[](https://github.com/SpenceKonde/AVR-Guidance/blob/master/UPDI/NanoUPDI_Minimal.png)

[](https://github.com/SpenceKonde/AVR-Guidance/blob/master/UPDI/NanoUPDI_Recommended.png)

**忽略有关“闪存”和“引导”内存的警告**

上传过程中将显示avrdude: jtagmkII\_initialize(): Cannot locate "flash" and "boot" memories in description警告 - 此警告是虚假的，可以安全地忽略。

**永久程序员汇编建议：**

* 为方便起见，我们建议将串行适配器、Nano 或 Pro Mini 专用于此目的，并焊接连接。Nano 和 Pro Mini 克隆可以在 ebay 上购买，运费为 2-5 美元。使用一个没有预先安装接头的。串行适配器的运费为 1-2 美元。
* 将电线焊接到位 - 我们建议使用 0.1" 杜邦跳线，因为您需要在另一端使用杜邦连接器，而压接您自己的杜邦连接器非常不愉快：将跳线切成两半，剥离并焊接到位。
* 将电线焊接到位后，用热熔胶将它们粘在板子底部，否则处理时容易疲劳和断裂。
* 我们建议按以下顺序排列连接器：UPDI、GND、Vcc - 这样，如果将连接器向后连接，则不会造成任何伤害。使用 3 针杜邦外壳，或用透明胶带将三个 1 针外壳固定在一起。

**引导加载程序和“刻录引导加载程序”**

“Burn Bootloader 做三件事——它设置保险丝以匹配在工具子菜单中选择的选项（由保险丝控制的选项在菜单中标记），擦除闪存，并将引导加载程序写入闪存，如果您选择了一块板使用引导加载程序定义（板菜单中标有（Optiboot）的那些）。在 2.3.x megaTinyCore 和 1.3.x DxCore 中，在正常上传过程中也会设置一些保险丝。

如果您希望在安装后停止使用引导加载程序并返回通过 UPDI 上传，只要您使用的是 megaTinyCore 2.3.0+ 或 DxCore 1.3.0+，您只需选择一个非引导加载程序板即可上传将自动清除引导加载程序 - 不再需要“刻录引导加载程序”，除非您更改了标记为需要刻录引导加载程序设置的其他设置。对于其他内核，您必须在选择了非引导加载程序板的情况下“刻录引导加载程序”。

**故障排除**

**忽略有关“闪存”和“引导”内存的警告**

上传过程中将显示avrdude: jtagmkII\_initialize(): Cannot locate "flash" and "boot" memories in description警告 - 此警告是虚假的（avrdude 中的错误，或为支持 jtag2updi 所做的修改中的错误）。它可以而且应该被忽略——它会在通过 jtag2updi 上传时显示

**对 jtag2updi 使用详细上传**

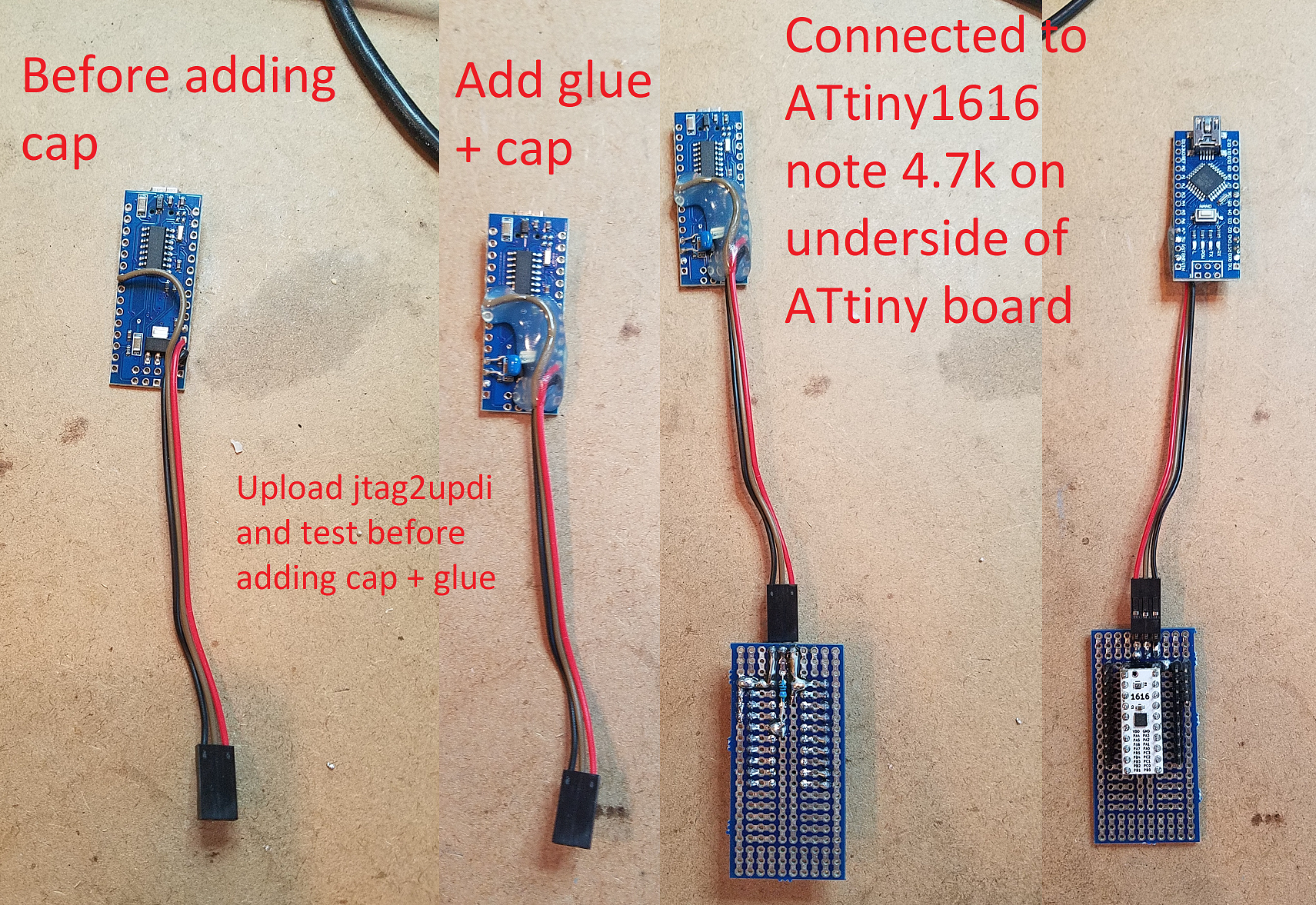
默认情况下，avrdude 输出非常简洁，尤其是使用 jtag2updi。启用详细上传可确保您在上传（和引导加载过程）期间获得有用的反馈

**串行适配器上的详细模式作为 UPDI**

详细模式太冗长了，生成的日志比上传的程序大几倍，普通模式和详细模式下的 AVRdude 一样健谈；在 DxCore 1.3.6 和 megaTinyCore 2.3.2 中，现在正常输出和详细输出之间没有区别，您可以一直保持详细输出。

**照片**

**Nano作为UPDI编程器，组装使用**

[](https://github.com/SpenceKonde/AVR-Guidance/blob/master/UPDI/NanoAsUPDI.png) “Nano 转换为 UPDI 程序员”）

**Pro Mini（和串行适配器）作为 UPDI**

(ProMiniAsUPDI.png["Pro Mini 转换为 UPDI 编程器")

**典型开发配置**

由于使用串行端口进行调试通常很有用，因此我通常会发现自己使用这样的配置，同时连接串行适配器和 UPDI 编程器。显然，也可以使用 Optiboot，但无需禁用 UPDI 进行重置，或使用其他笨拙的技巧（参见 [AlternativeReset.So