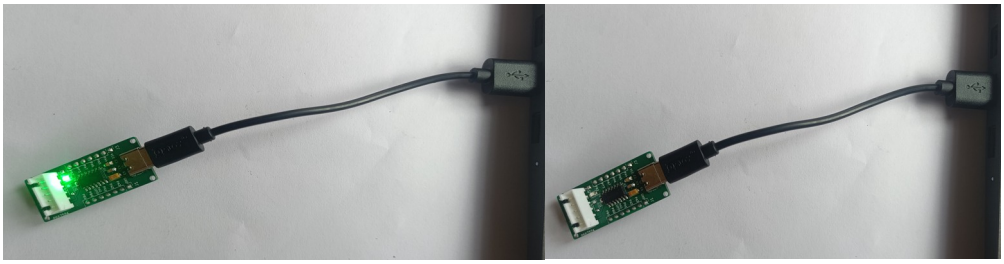
Provenant Systems
Simplifying hardware-software integrations

Getting started with ATtiny3224 development board



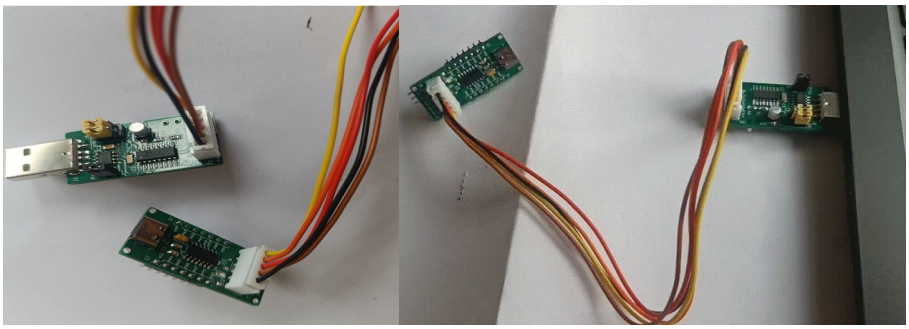
Overview

This document covers the steps needed to get started with programming the ATtiny3224 Microcontroller
Below are some of the components needed to successfully program the ATtiny3224 development board using Arduino IDE

1. MegaTinyCore
   a) It is an excellent library written by SpenceKonde that provides support for the tinyAVR microcontrollers (Which is a newer series of AVR) out of which for our specific application, we use the ATtiny series of microcontrollers (ATtiny 0, ATtiny 1 and ATtiny 2).
   b) Simplifies programming ATtiny microcontrollers using the familiar Arduino environment.
   c) By using MegaTinyCore, you can efficiently develop software for ATtiny microcontrollers without needing to delve into complexities of low-level programming.

2. UPDI Programmer with Serial monitor feature: This tool is essential for programming the development board. The Serial monitor feature provides ability of monitoring serial output from the ATtiny microcontroller UART using the Arduino IDE.

SETUP



Purpose:

This is the setup for programming and monitoring a microcontroller (ATtiny 3224).

Components:

- **Computer:** Computer (can be Windows, Linux or Mac) used to run the Arduino IDE. It is used in programming the microcontroller board.
- **Microcontroller Board:** The target device being programmed and monitor is ATtiny3224 development board.
- **Programmer Cable:** This cable is used to connect the microcontroller board to the UPDI programmer. At a minimum this cable has VCC, UPDI and Ground connections to the microcontroller board.

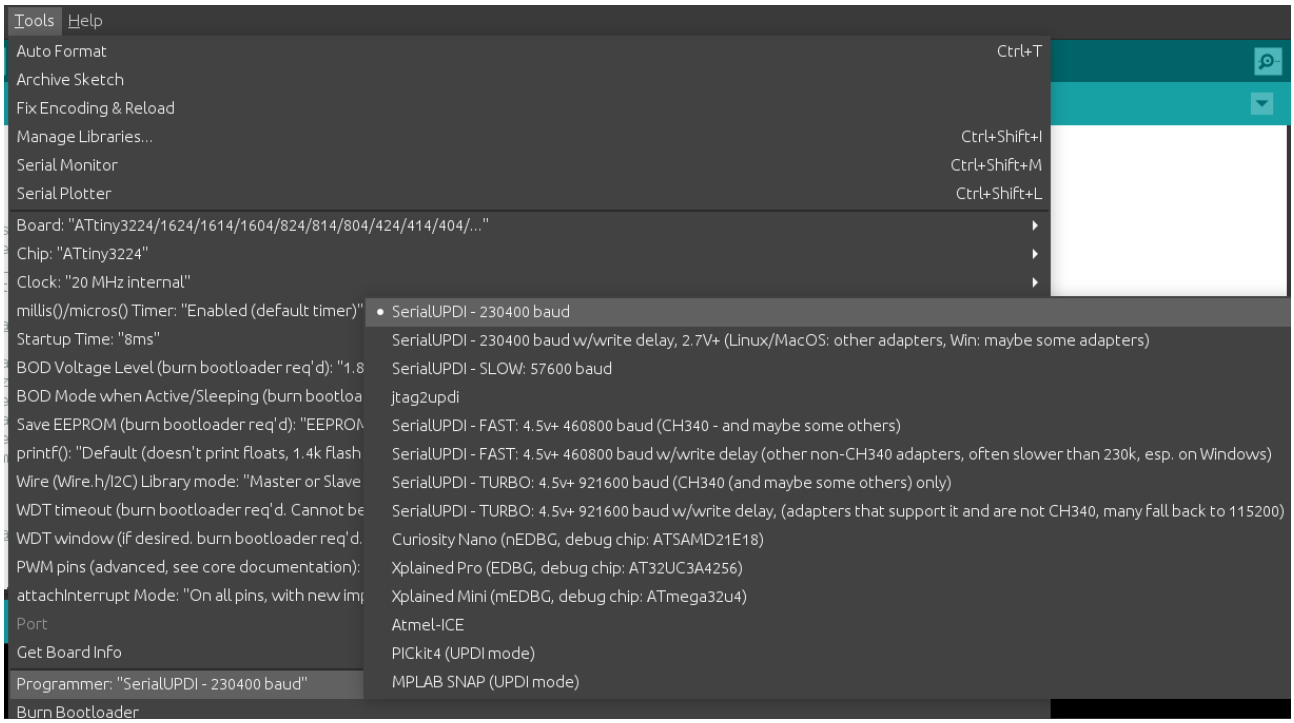
Connections:

- The UPDI programmer is connected to the computer using the USB port. The microcontroller board is connected to the UPDI programmer via a Programmer cable.

Setting up the Arduino IDE for programming ATtiny3224

The USB Serial UPDI programmer is used to download the sketch to the development board.

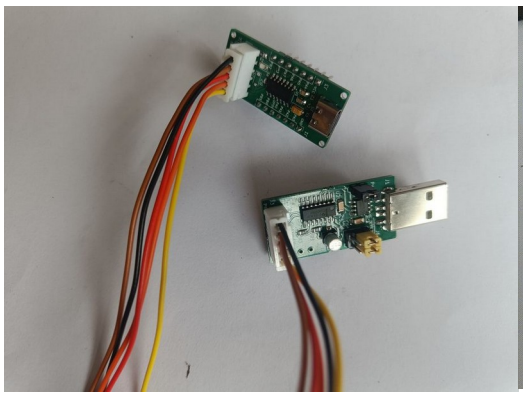
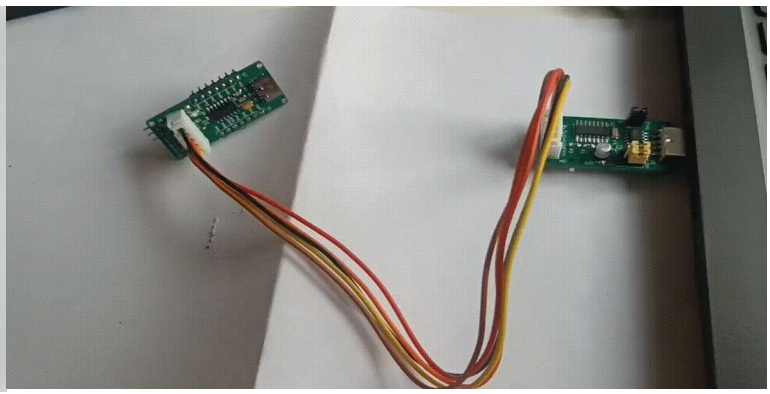
Choose "SerialUPDI (230400 baud)" in the “Programmer” option under “Tools” menu of Arduino IDE.



Note:

At 3.3V the microcontroller frequency is limited to maximum of 8 – 10 Mhz. At 5v 20MHz operation is possible.

Setting up Arduino IDE on Linux (Ubuntu)

A) Needed only once to setup Arduino IDE on Ubuntu Linux

1. Install Arduino IDE 1.8.19 from Arduino website or respective distro/OS software center.
2. Launch the Arduino IDE you shall get following pop-up as fig.1, click add. (to take effect logout from your user account login back to the user account, now it is good to go)

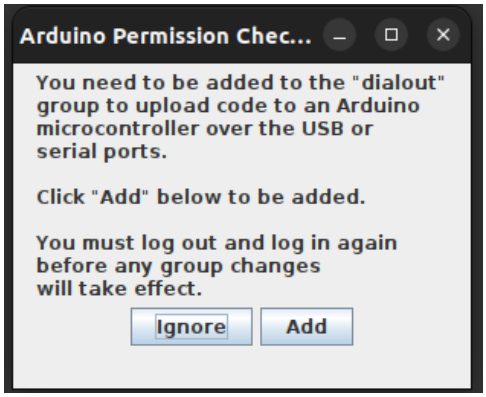


Fig.1

3. Install library of “megaTinyCore” by Spence Konde on to Arduino IDE, to add use the URL “https://drazzy.com/package_drazzy.com_index.json” library in the Arduino additional library option present in “Files --> Preferences(Fig.2) --> Additional Boards Manager URLs (Fig.3)”

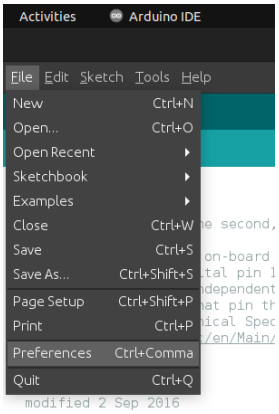


Fig.2

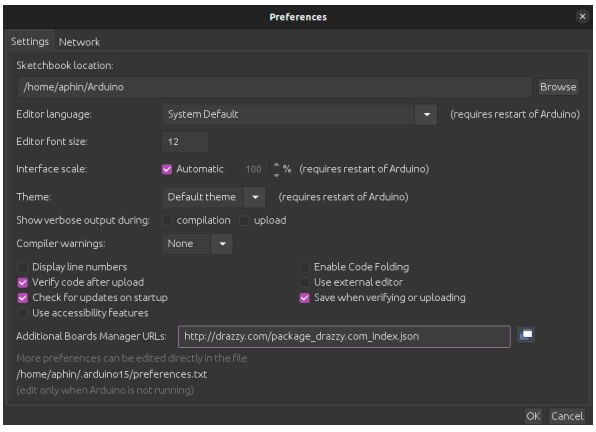


Fig.3

Note: In Ubuntu it has been observed that “brltty” package does not allow the UPDI’s USB to be attached to the machine, following error is observed in ***dmesg*** output.

```
usb 1-2: new full-speed USB device number 15 using xhci_hcd
usb 1-2: New USB device found, idVendor=1a86, idProduct=7523, bcdDevice= 2.64
usb 1-2: New USB device strings: Mfr=0, Product=2, SerialNumber=0
usb 1-2: Product: USB Serial ch341 1-2:1.0: ch341-uart converter detected
usb 1-2: ch341-uart converter now attached to ttyUSB0
input: BRLTTY 6.4 Linux Screen Driver Keyboard as /devices/virtual/input/input57
usb 1-2: usbfs: interface 0 claimed by ch341 while 'brltty' sets config #1
ch341-uart ttyUSB0: ch341-uart converter now disconnected from ttyUSB0
ch341 1-2:1.0: device disconnected
```

Fig. 4

Use the following command to resolve and the UPDI will be attached to ‘/dev/ttyUSB0’

```
sudo apt remove brltty
```

4. Now goto “Tool” --> Board --> Boards Manager and search for “megaTinyCore” (using 2.6.8 version) and click install.

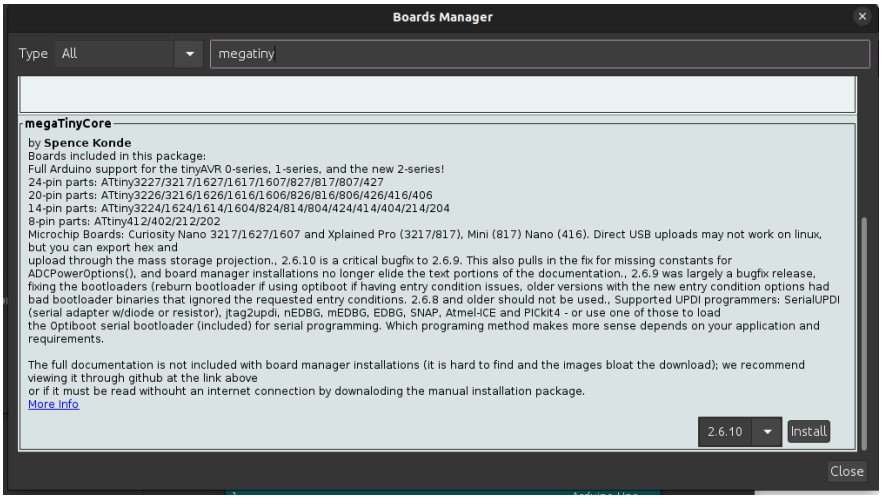


Fig.6

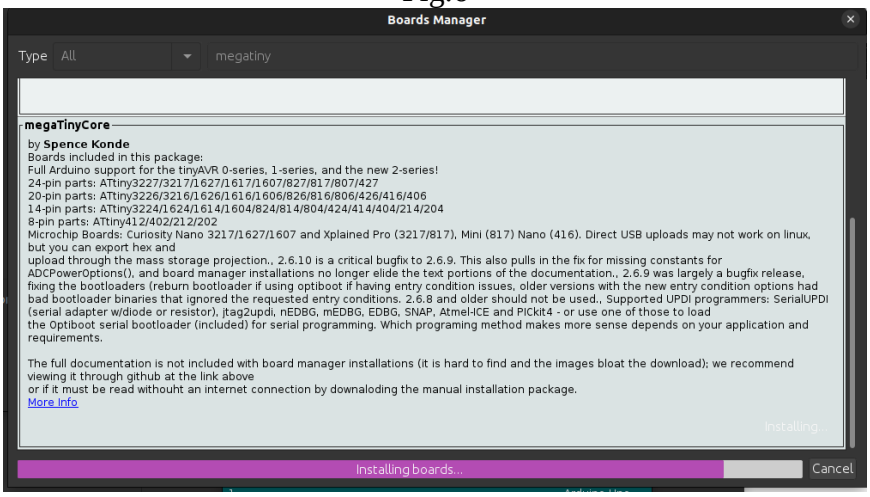


Fig.7

5. Now edit “platform.txt” file in folder location as “*/home/USER_NAME/.arduino15/packages/megaTinyCore/hardware/megaavr/2.6.8/*” (in linux) comment the following line “compiler.path={runtime.tools.avr-gcc.path}/bin/” and add below it following line “compiler.path={runtime.tools.avr-gcc-7.3.0-atmel3.6.1-azduino6.path}/bin/” as shown in Fig.8.

```
24
25 ########################
26 # AVR compile variables #
27 ########################
28
29 compiler.warning_flags=-Wall
30 compiler.warning_flags.none=-Wall
31 compiler.warning_flags.default=-Wall
32 compiler.warning_flags.more=-Wall
33 compiler.warning_flags.all=-Wall -Wextra
34
35 # Default "compiler.path" is correct, change only if you want to override the initial value
36 #compiler.path={runtime.tools.avr-gcc.path}/bin/
37 compiler.path={runtime.tools.avr-gcc-7.3.0-atmel3.6.1-azduino6.path}/bin/
38 compiler.c.cmd=avr-gcc
39 compiler.c.flags=-c -g -Os {compiler.warning_flags} -std=gnu11 -ffunction-sections -fdata-sec
   Werror=implicit-function-declaration -Wundef
40 compiler.c.elf.flags={compiler.warning_flags} -Os -g -flto -fuse-linker-plugin -Wl,--gc-secti
   {build.mrelax} {build.gcse}
```

Fig.8

6. Now goto “Tools” menu --> Boards --> megaTinyCore --> ATtiny3224

Fig.9

Now Arduino IDE is ready to program and monitor ATtiny3224 microcontroller.

Sample Sketch & Arduino IDE pin assignment

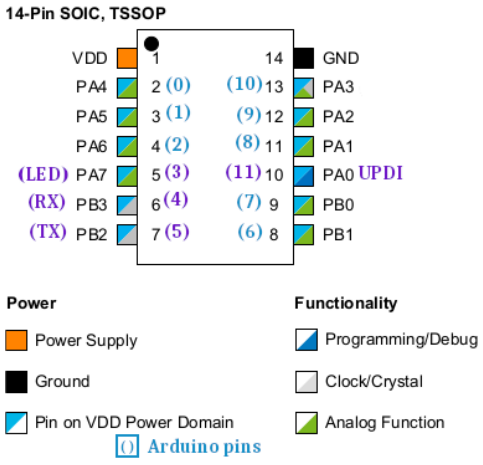
Sample Code: LED BLINKING

```
void setup() {
  Serial.begin(115200);  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}
// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  Serial.printf("LED ON\r\n");
  delay(100);                        // wait for a fraction of second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off (LOW is the voltage level)
  Serial.printf("LED OFF\r\n");
  delay(1000);                       // wait for a second
}
```

Below is the Arduino Pin configuration with refrence ATtiny 3224 pinout

Arduino Pin assignment	ATtiny Pin number	Comments
0	2	PA4
1	3	PA5
2	4	PA6
3	5	PA7 [LED_BUILTIN]
4	6	PB3 [Rx]
5	7	PB2 [Tx]
6	8	PB1
7	9	PB0
8	11	PA1
9	12	PA2
10	13	PA3
11	10	PA0 [UPDI]

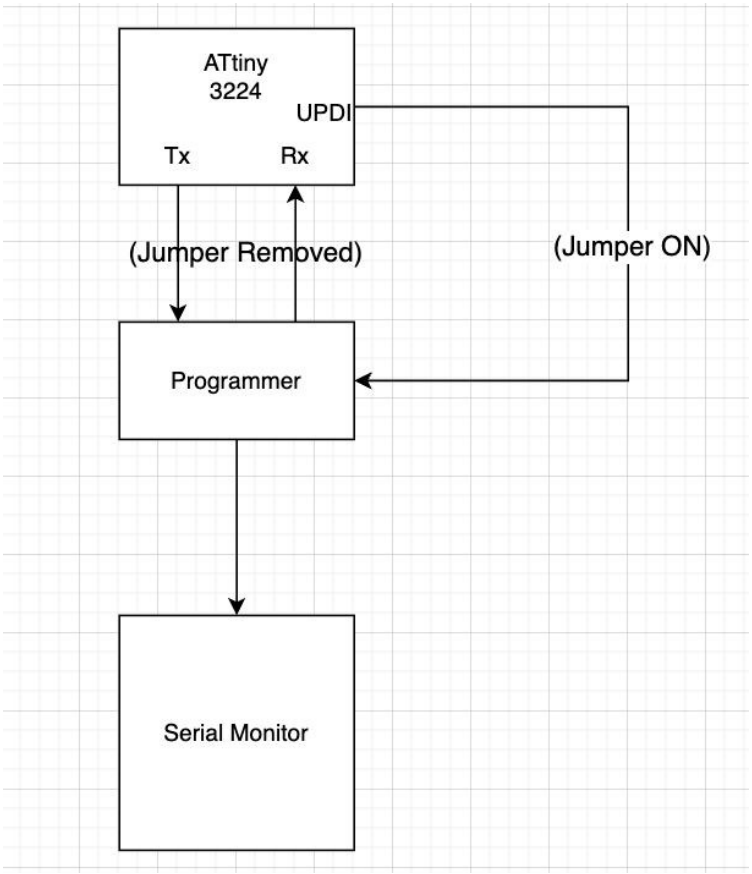
Image reference of Arduino IDE pin assignment for ATtiny 3224 14-pin microcontroller



Steps below show the programming steps using Provenant systems UPDI programmer

- Use the Arduino IDE to create your program. For example, start with a simple LED blinking sketch.
- Ensure the jumper is in the **ON** position to enable programming mode.
- Connect the programmer to USB port on the computer
- Connect the programmer to the development board using the provided cable
- Ensure the power switch on the UPDI programmer is set to OFF if the development board gets external power, else select it to ON position to have the UPDI programmer provide power the development board
- Select the appropriate communication port under “Tools” menu (todo: need a picture here)
- Upload your code to the microcontroller using “Sketch” => “Upload” option.
- To observe serial output from the microcontroller program, remove the jumper to switch to serial monitoring mode.
- Open the Arduino Serial Monitor to view the output from your microcontroller's code.
- **Note:** *When the jumper cap is **ON**, the receiver and transmitter pins are disabled, and the USB connection is redirected to the UPDI pins for programming.*

Flow Diagram of the UPDI & Serial monitoring mode



1.Tx: Transmitter
2.Rx : Receiver
3.UPDI (Unified Program Debug Interface): Used to program when the jumper is ON.