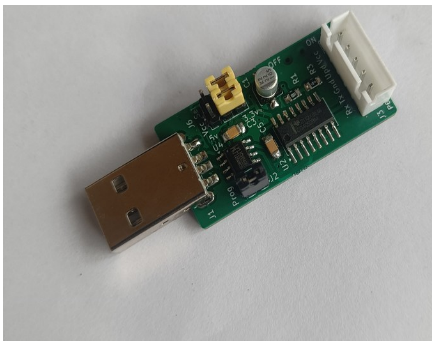
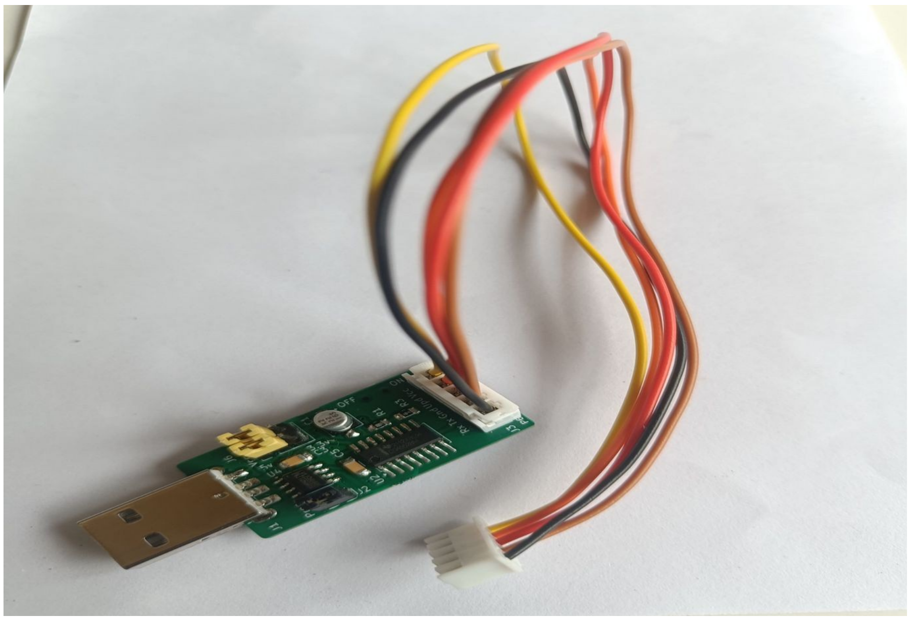
Provenant Systems

Simplifying hardware-software integrations

Getting started with UPDI programmer

Provenant Systems

Simplifying hardware-software integrations

Overview

This document covers the steps needed to get started with programming the ATtiny3224 Micro-controller using UPDI programmer

Below are some of the components needed in process of programming

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.

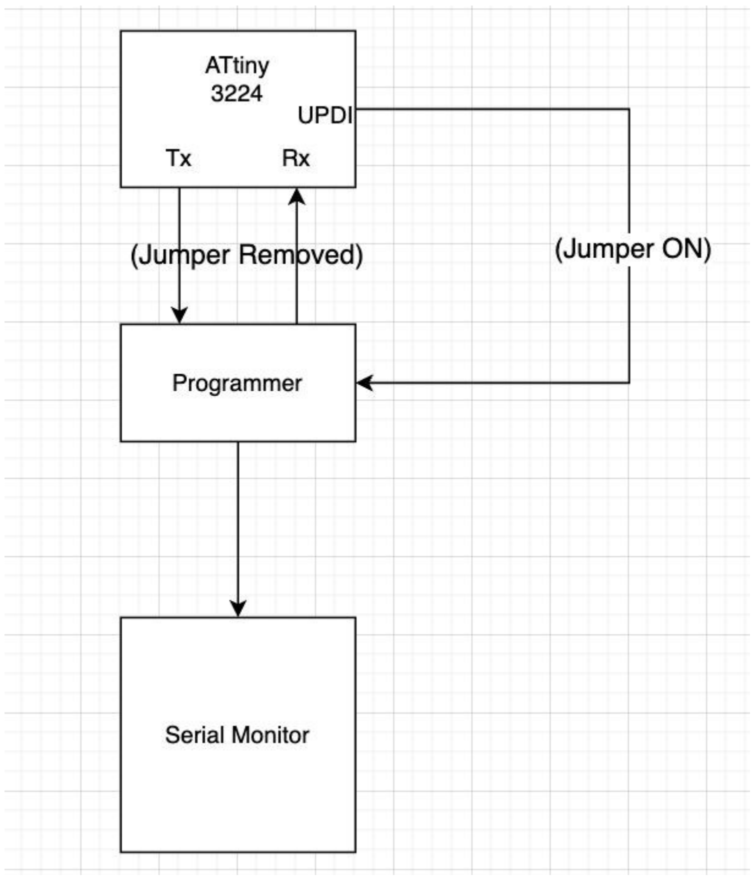
2. UPDI Programmer & Serial monitor
   a) This versatile tool enables you to both program and monitor your ATtiny microcontroller using the Arduino IDE.
   b) Supports UPDI programming for ATtiny microcontrollers.
   c) Includes a jumper for switching between Programming and Serial monitor for microcontroller UART.
   d) Offers selectable 3.3V or 5V operation for compatibility.
   e) Power switch to Enable or Disable target board power.
      (1) Disable power: Use this setting when board is powered with an external power source and UPDI Programmer is only used to Program or perform Serial monitoring of board UART
      (2) Enable power: Use this setting when there is NO external power applied to the board and Programmer is expected to provide power (3.3v or 5v).

      Note: Developer board and/or Programmer may get damaged if the power switch is used improperly.

Usage

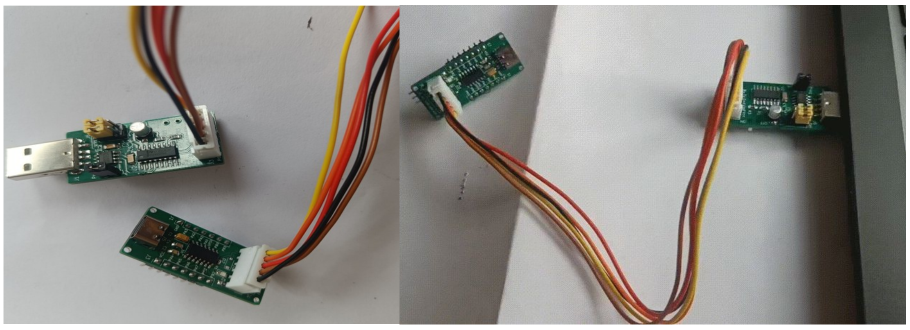
- Use the Arduino IDE to create your program. For example, start with a simple LED blinking sketch.
- Ensure the jumper cap is in the **ON** position to enable programming mode.
- Verify and upload your code to the microcontroller using the Arduino IDE.
- Remove the jumper cap to switch to serial monitoring 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.

SETUP



Purpose:

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

Components:

- **PC:** The main computer used to control the programming process and run software like the Arduino IDE.
- **USB Hub**(optional)**:** A device that expands the number of USB ports available on the PC.
- **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 and is used for supplying power, programming and serial port monitoring.

Connections:

- he USB hub.
- The PC is connected to the microcontroller board via UPDI programmer.

Use Cases:

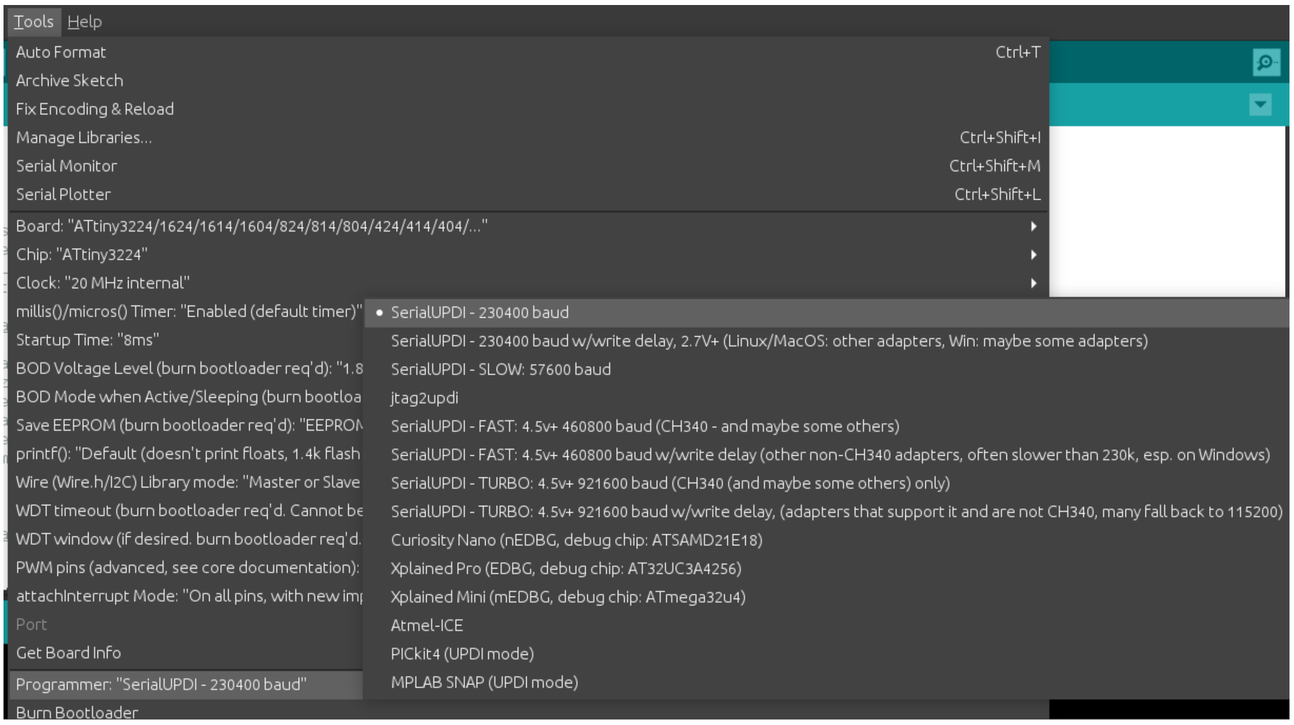
- Programming the microcontroller with code developed on the PC.
- Monitoring the microcontroller's serial port by monitoring its output or interacting with it through the computer.
- Experimenting with different microcontroller configurations and circuits using the breadboard.

Programming the microcontroller

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.

The programmer also doubles as a Serial port monitor permitting developers to view the Serial print output on the Arduino IDE serial monitor.



```
Tools  Help
Auto Format                                              Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries...                                      Ctrl+Shift+I
Serial Monitor                                           Ctrl+Shift+M
Serial Plotter                                           Ctrl+Shift+L
Board: "ATtiny3224/1624/1614/1604/824/814/804/424/414/404/..."
Chip: "ATtiny3224"
Clock: "20 MHz internal"
millis()/micros() Timer: "Enabled (default timer)"
Startup Time: "8ms"
BOD Voltage Level (burn bootloader req'd): "1.8
BOD Mode when Active/Sleeping (burn bootloa
Save EEPROM (burn bootloader req'd): "EEPROM
printf(): "Default (doesn't print floats, 1.4k flash
Wire (Wire.h/I2C) Library mode: "Master or Slave
WDT timeout (burn bootloader req'd. Cannot be
WDT window (if desired. burn bootloader req'd.
PWM pins (advanced, see core documentation):
attachInterrupt Mode: "On all pins, with new im
Port
Get Board Info
Programmer: "SerialUPDI - 230400 baud"
Burn Bootloader
```

```
• SerialUPDI - 230400 baud
  SerialUPDI - 230400 baud w/write delay, 2.7V+ (Linux/MacOS: other adapters, Win: maybe some adapters)
  SerialUPDI - SLOW: 57600 baud
  jtag2updi
  SerialUPDI - FAST: 4.5v+ 460800 baud (CH340 - and maybe some others)
  SerialUPDI - FAST: 4.5v+ 460800 baud w/write delay (other non-CH340 adapters, often slower than 230k, esp. on Windows)
  SerialUPDI - TURBO: 4.5v+ 921600 baud (CH340 (and maybe some others) only)
  SerialUPDI - TURBO: 4.5v+ 921600 baud w/write delay, (adapters that support it and are not CH340, many fall back to 115200)
  Curiosity Nano (nEDBG, debug chip: ATSAMD21E18)
  Xplained Pro (EDBG, debug chip: AT32UC3A4256)
  Xplained Mini (mEDBG, debug chip: ATmega32u4)
  Atmel-ICE
  PICkit4 (UPDI mode)
  MPLAB SNAP (UPDI mode)
```

Jumper configuration

When ON, enables flashing the sketch to the microcontroller.

When removed, routes the microcontroller UART pins to Arduino IDE Serial Monitor.

Note:

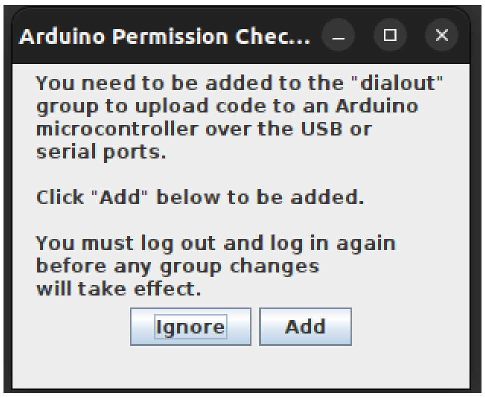
At 3.3V the microcontroller frequency is limited to maximum of 8 – 10 Mhz. At 5v upto 20MHz operation is possible.

<u>Note: While the instructions below are for Provenant Systems Attiny3224 development board, they should be applicable with any other similar development board</u>.

Setting up Arduino IDE on Linux (Ubuntu)

A) Only For first time Arduino users

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)



Arduino Permission Chec...

You need to be added to the "dialout" group to upload code to an Arduino microcontroller over the USB or serial ports.

Click "Add" below to be added.

You must log out and log in again before any group changes will take effect.

Fig.1

3. To program ATtiny we need to use 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)"

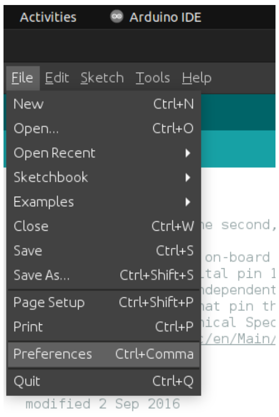
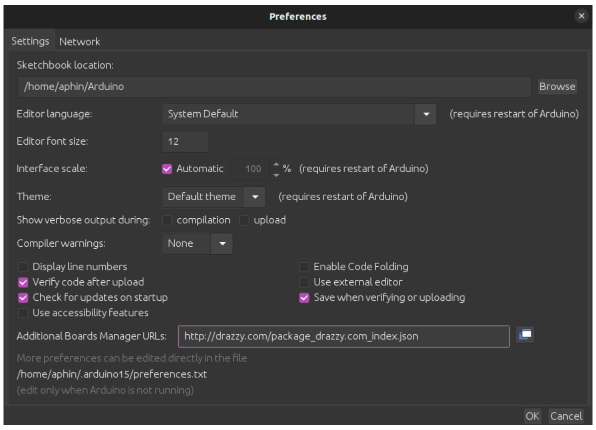


Fig.2



Preferences

Settings | Network

Sketchbook location: /home/aphin/Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Interface scale: [✓ Automatic] 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: compilation upload

Compiler warnings: None

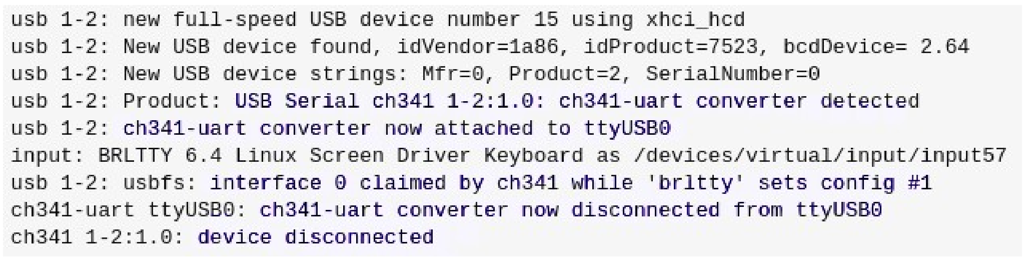
Display line numbers
[✓ Verify code after upload]
[✓ Check for updates on startup]
Use accessibility features
Enable Code Folding
Use external editor
[✓ Save when verifying or uploading]

Additional Boards Manager URLs: http://drazzy.com/package_drazzy.com_index.json

More preferences can be edited directly in the file
/home/aphin/.arduino15/preferences.txt
(edit only when Arduino is not running)

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.



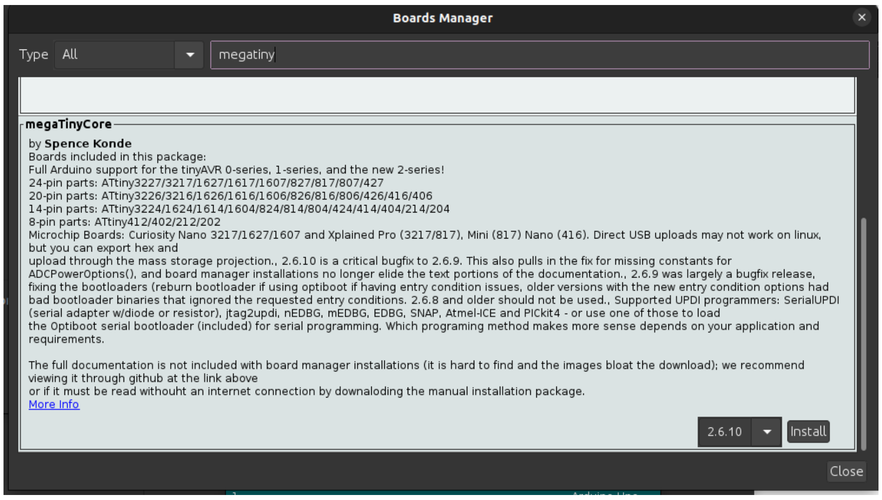
```
usb 1-2: new full-speed USB device number 15 using xhci_hcd
usb 1-2: New USB device found, idVendor=1a86, idProduct=7523, bcdDevice= 2.64
usb 1-2: New USB device strings: Mfr=0, Product=2, SerialNumber=0
usb 1-2: Product: USB Serial ch341 1-2:1.0: ch341-uart converter detected
usb 1-2: ch341-uart 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.

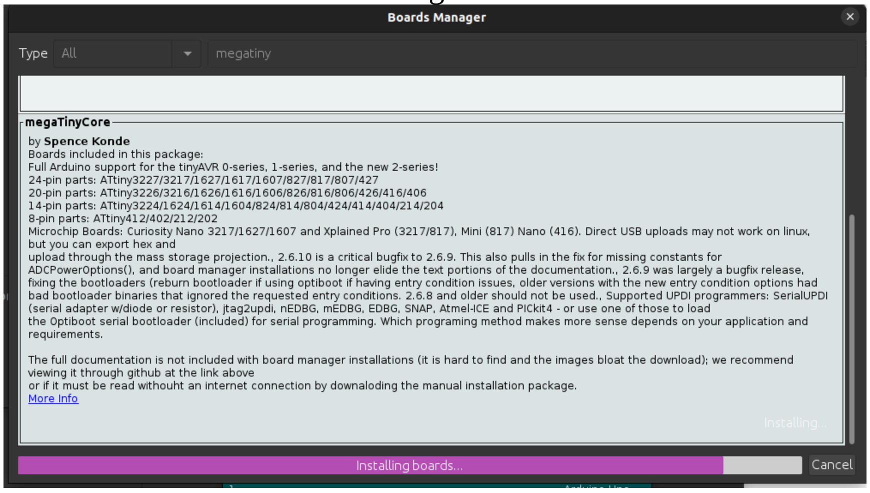


megaTinyCore

by **Spence Konde**
Boards included in this package:
Full Arduino support for the tinyAVR 0-series, 1-series, and the new 2-series!
24-pin parts: ATtiny3227/3217/1627/1617/1607/827/817/807/427
20-pin parts: ATtiny3226/3216/1626/1616/1606/826/816/806/426/416/406
14-pin parts: ATtiny3224/1624/1614/1604/824/814/804/424/414/404/214/204
8-pin parts: ATtiny412/402/212/202
Microchip Boards: Curiosity Nano 3217/1627/1607 and Xplained Pro (3217/817), Mini (817) Nano (416). Direct USB uploads may not work on linux, but you can export hex and
upload through the mass storage projection., 2.6.10 is a critical bugfix to 2.6.9. This also pulls in the fix for missing constants for ADCPowerOptions(), and board manager installations no longer elide the text portions of the documentation., 2.6.9 was largely a bugfix release, fixing the bootloaders (reburn bootloader if using optiboot if having entry condition issues, older versions with the new entry condition options had bad bootloader binaries that ignored the requested entry conditions. 2.6.8 and older should not be used., Supported UPDI programmers: SerialUPDI (serial adapter w/diode or resistor), jtag2updi, nEDBG, mEDBG, EDBG, SNAP, Atmel-ICE and PICkit4 - or use one of those to load
the Optiboot serial bootloader (included) for serial programming. Which programing method makes more sense depends on your application and requirements.

The full documentation is not included with board manager installations (it is hard to find and the images bloat the download); we recommend viewing it through github at the link above
or if it must be read withouht an internet connection by downaloding the manual installation package.
More Info

Fig.6



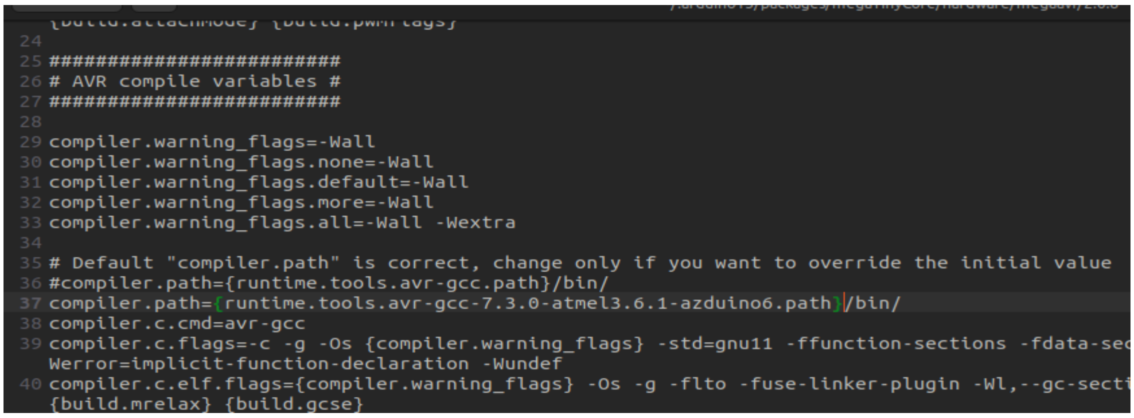
megaTinyCore

by **Spence Konde**
Boards included in this package:
Full Arduino support for the tinyAVR 0-series, 1-series, and the new 2-series!
24-pin parts: ATtiny3227/3217/1627/1617/1607/827/817/807/427
20-pin parts: ATtiny3226/3216/1626/1616/1606/826/816/806/426/416/406
14-pin parts: ATtiny3224/1624/1614/1604/824/814/804/424/414/404/214/204
8-pin parts: ATtiny412/402/212/202
Microchip Boards: Curiosity Nano 3217/1627/1607 and Xplained Pro (3217/817), Mini (817) Nano (416). Direct USB uploads may not work on linux, but you can export hex and
upload through the mass storage projection., 2.6.10 is a critical bugfix to 2.6.9. This also pulls in the fix for missing constants for ADCPowerOptions(), and board manager installations no longer elide the text portions of the documentation., 2.6.9 was largely a bugfix release, fixing the bootloaders (reburn bootloader if using optiboot if having entry condition issues, older versions with the new entry condition options had bad bootloader binaries that ignored the requested entry conditions. 2.6.8 and older should not be used., Supported UPDI programmers: SerialUPDI (serial adapter w/diode or resistor), jtag2updi, nEDBG, mEDBG, EDBG, SNAP, Atmel-ICE and PICkit4 - or use one of those to load
the Optiboot serial bootloader (included) for serial programming. Which programing method makes more sense depends on your application and requirements.

The full documentation is not included with board manager installations (it is hard to find and the images bloat the download); we recommend viewing it through github at the link above
or if it must be read withouht an internet connection by downaloding the manual installation package.
More Info

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.



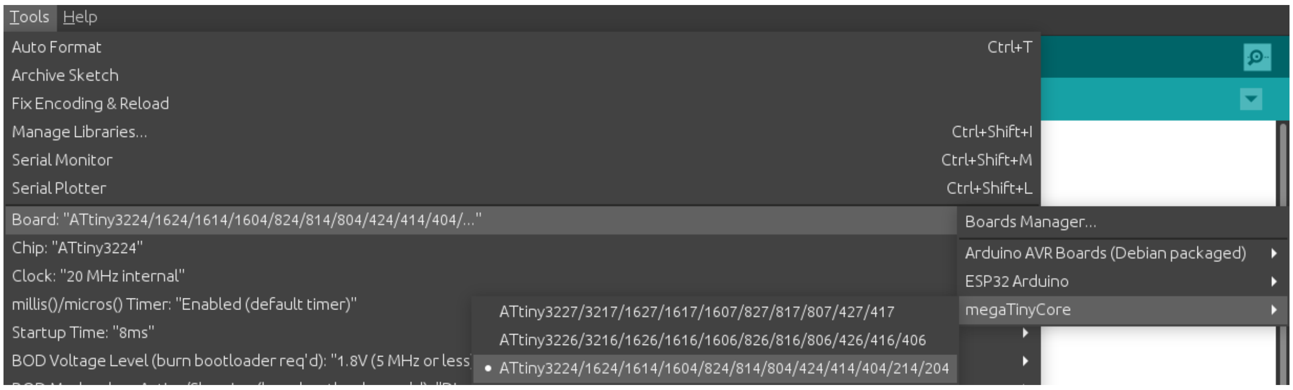
```
##########################
# AVR compile variables #
##########################

compiler.warning_flags=-Wall
compiler.warning_flags.none=-Wall
compiler.warning_flags.default=-Wall
compiler.warning_flags.more=-Wall
compiler.warning_flags.all=-Wall -Wextra

# Default "compiler.path" is correct, change only if you want to override the initial value
#compiler.path={runtime.tools.avr-gcc.path}/bin/
compiler.path={runtime.tools.avr-gcc-7.3.0-atmel3.6.1-azduino6.path}/bin/
compiler.c.cmd=avr-gcc
compiler.c.flags=-c -g -Os {compiler.warning_flags} -std=gnu11 -ffunction-sections -fdata-sec
Werror=implicit-function-declaration -Wundef
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



```
Tools  Help
Auto Format                                          Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries...                                  Ctrl+Shift+I
Serial Monitor                                       Ctrl+Shift+M
Serial Plotter                                       Ctrl+Shift+L
Board: "ATtiny3224/1624/1614/1604/824/814/804/424/414/404/..."   Boards Manager...
Chip: "ATtiny3224"                                   Arduino AVR Boards (Debian packaged)
Clock: "20 MHz internal"                             ESP32 Arduino
millis()/micros() Timer: "Enabled (default timer)"   megaTinyCore
Startup Time: "8ms"
BOD Voltage Level (burn bootloader req'd): "1.8V (5 MHz or less

ATtiny3227/3217/1627/1617/1607/827/817/807/427/417
ATtiny3226/3216/1626/1616/1606/826/816/806/426/416/406
• ATtiny3224/1624/1614/1604/824/814/804/424/414/404/214/204
```

Fig.9

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

Provenant Systems
Simplifying hardware-software integrations

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
}
```