SiLabs/Tiva/ATtiny Programming Guide
By Joseph Haas, KEØFF

*joeh-**at**-rollanet-**dot**-org*

This document centralizes the key configuration steps, software identification, and hardware needed to allow a developer to program (and debug) SiLabs 8-bit, TI Tiva (see page 4), and Microchip ATtiny (see page 6 - UPDI variants only) microcontrollers (MCUs). The optional Small-Form-Factor (SFF) connector standard developed and used by the author is also presented in enough detail to allow one to reproduce the needed hardware and accessories.

SiLabs Microcontrollers

Programming the SiLabs 8051 microcontroller (MCU) on a target PCB can be accomplished using the SiLabs USB debug adapter, the SiLabs FLASH Programming Utility program, and a source file for the desired application to program. The steps defined below apply to both C2D-style and JTAG style SiLabs 8051 variants.

The Debug adapter can be obtained from Mouser Electronics (www.mouser.com) or DigiKey Electronics (www.digikey.com) using the part number DEBUGADPTR1-USB. The cost (as of this writing) is about $49 plus applicable tax and shipping charges. A "standard" USB "printer" cable is also needed (these cables feature the rectangular "A" style connector that plugs into the PC at one end, and the square-ish "B"style connector that plugs into something like a printer or scanner at the other end). Mouser or Digikey are also good sources for the USB cable. The Qualtek 3021001-03 lists for about $3.

The Flash Programming Utility software is available at:

http://www.silabs.com/products/development-tools/software/8-bit-8051-microcontroller-software#flash

If this link does not work, go to www.silabs.com and search for "flash programming utility".

Follow the installation instructions on the web site and in the debug adapter documentation.

Connect the debug adapter ribbon cable connector to the target connector either directly or by using an adapter to connect to the Molex 6-pin small-form-factor programming connector (details on this adapter are at the end of this document). To program the MCU, execute the following steps:

- Obtain the object file for the application to program (typically, this is a "*.hex" file)
- Connect the debug adapter to the programming connector on the target
- Plug in the USB cable to the debug adapter and the PC. Some targets may require power to be applied to the target separately, while others may be powered from the USB Programming Adapter.
- Open the flash programming utility software
- See Figures 1 – 3:
- Under the "Download Hex File/Go/Stop" tab, select the object application file (.hex) and check the "Erase all of Code Space" box
- Under the "Connect/Disconnect" tab, make sure all of the check boxes match the image and that the "USB Debug Adapter" is checked (if not, you must make sure the debug adapter is connected and that the drivers are working properly). Click "Connect" and click "OK" when the "Connected" box appears.

- Return to the "Download Hex File/Go/Stop" tab and click "Download" (there should be erase, program, and verify progress screens displayed). Programming should take no more than 15 seconds.
- Return to the "Connect/Disconnect" and click "Disconnect"
- Remove Power from the Orion and disconnect the debug adapter.
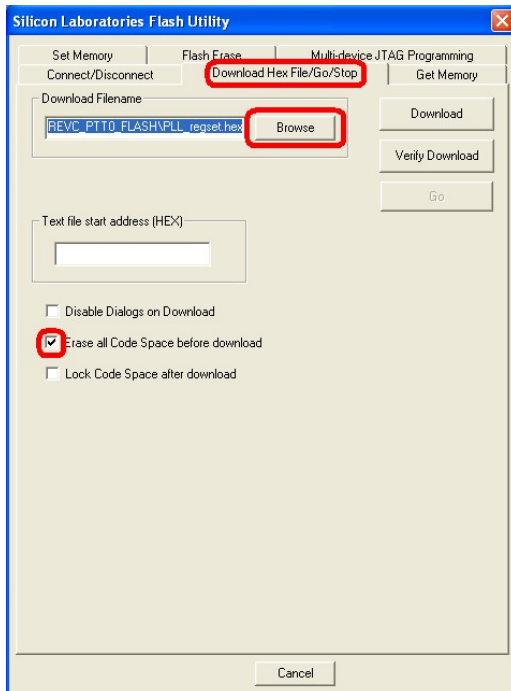- The target device is now ready to operate with the new application software.
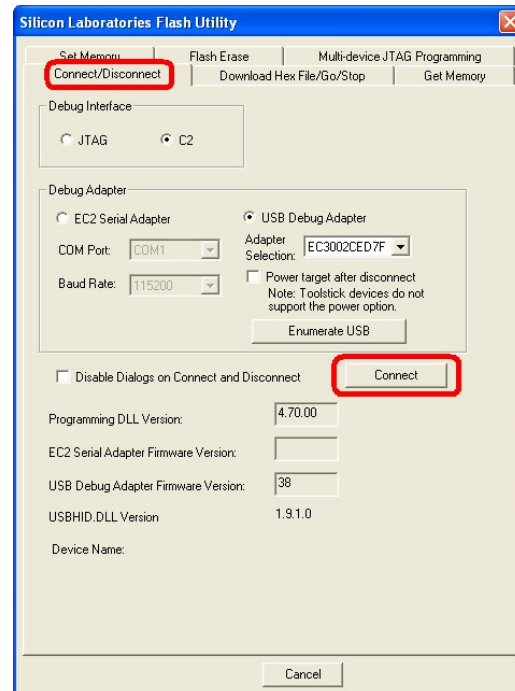

Figure 1. FLASH Utility setup dialog box.

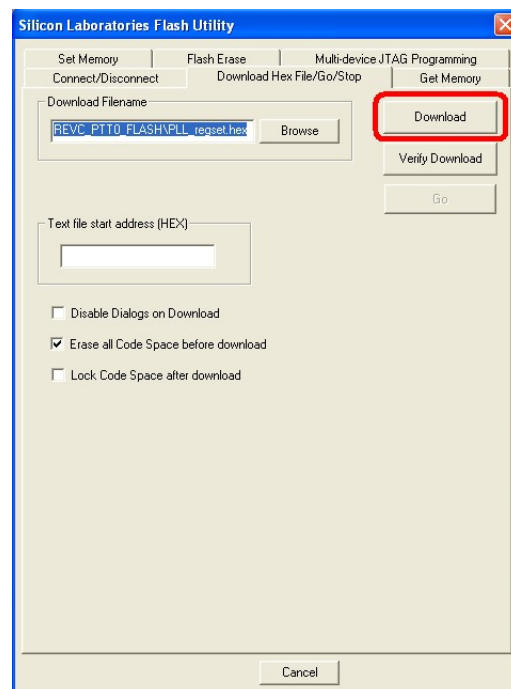
Figure 2. FLASH Utility connect dialog box.

Figure 3. FLASH Utility setup dialog box (download).

2

Small Form-Factor Programming Connections

Many of my designs use a small-form-factor (SFF) connector to minimize the PCB space required for in-circuit programming capability. A connection pinout that I have standardized for my projects uses a Molex, 1.25mm, 6-position connector, PN **53261-0671** (R/A) or PN **53398-0671** (vertical) for the target PCB connector. This connector has a small footprint, and the right-angle and vertical versions can both be soldered to the same PCB footprint. All that is required to connect the SiLabs programming adapter is an intermediate cable which accepts a 10-position ribbon cable connector and converts this to a 6-position cable with the mating Molex connector for the PCB connection.

The transition cable (see Photo 1) consists of a 10-pin, dual row, 0.1" spaced ribbon header (TE Connectivity 5103309-1, or equivalent) and a cable terminated with the appropriate 6-positon Molex connector (Molex PN 15134-0602, or equivalent). The 15134-0602 cable comes with two ends and should be cut in half (more or less – the other half may be saved or discarded). *Note: Other lengths of this cable are generally available and may be used in place of that part# if it is out of stock.* A small piece of pad-per-hole protoboard (approximately 0.75" square) should be used to stabilize the connectors and wires. Strip and tin the leads approximately 0.1", then form each tinned wire into a "hook". Solder the 10-pin header to the protoboard and then solder the GND net connections (pins 2, 3, and 9 – a piece of ¼W leaded resistor lead can do this nicely) then solder each wire according to Table 1.

JTAG Signal	C2D Signal	P1	**10-pin**		P1	C2D Signal	JTAG Signal
-	-	-	**1**	**2**	5	GND	GND
GND	GND	5	**3**	**4**	2	C2D	TCK
TMS	/RESET	1	**5**	**6**	4	P0.6	TDO
TDI	/RST_C2K	3	**7**	**8**	-	-	-
GND	GND	5	**9**	**10**	6	+5V (note)	+3.3V (note)

Table 1. SiLabs to SFF programming adapter. P1 is the target device
connector pinout. *Note: Some targets require +3.3V supply.
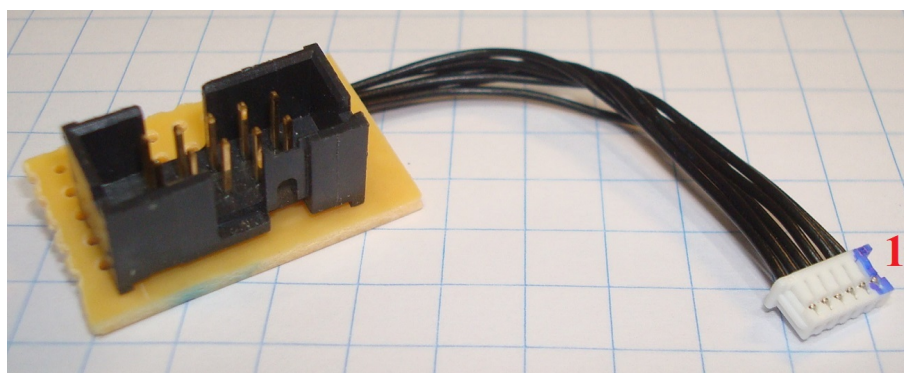For these target devices, DO NOT connect +5V to P1-6.*

Photo 1. Programming adapter cable.

To support Tiva MCUs, a 3-pin, single-row, header must be affixed to the pad-per-hole board. The center-pin of this header would connect to pin 6 of the SFF connector. The "Other" side of the 3-pin header

3

would connect to pin-10 of the 10-pin connector while the remaining side ("Tiva") would connect to pin-1 of the 10-pin connector. Place a jumper-shunt to select "Other" or "Tiva" as needed.

Tiva Programming Hardware Guide

The same SFF connector and 10-pin adapter used for the SiLabs processors is also used in my Tiva designs for programming/debugging. The connections are similar with a couple of exceptions noted in Table 2.

LP Signal	P1	10-pin		P1	LP Signal
X1-RESET	6	1	2	5	GND
GND	5	3	4	2	X1-TCK
X1-TMS	1	5	6	4	X1-TDO
X1-TDI	3	7	8	-	X1-RXD
GND	5	9	10	-	X1-TXD

Table 2. Tiva to SFF programming adapter. P1 is the target device
connector pinout. *Note: X1 is located on the EK-TM4C1294XL LaunchPad.*
An "SiLabs/Tiva" shunt jumper required. Pins 8 & 10 must be separately routed to the target
if the LaunchPad COM port is desired. See Tables 2 & 3 for the Tiva and ATtiny pin maps.

Any of the Tiva LaunchPad evaluation boards can be modified to act as a programming interface for a target board. This is generally much less expensive than buying a commercial programmer. Photo 2 illustrates the connection to an EK-TM4C1294XL Launchpad that has had its target MCU removed (this is not necessary to use the LaunchPad as a programmer).
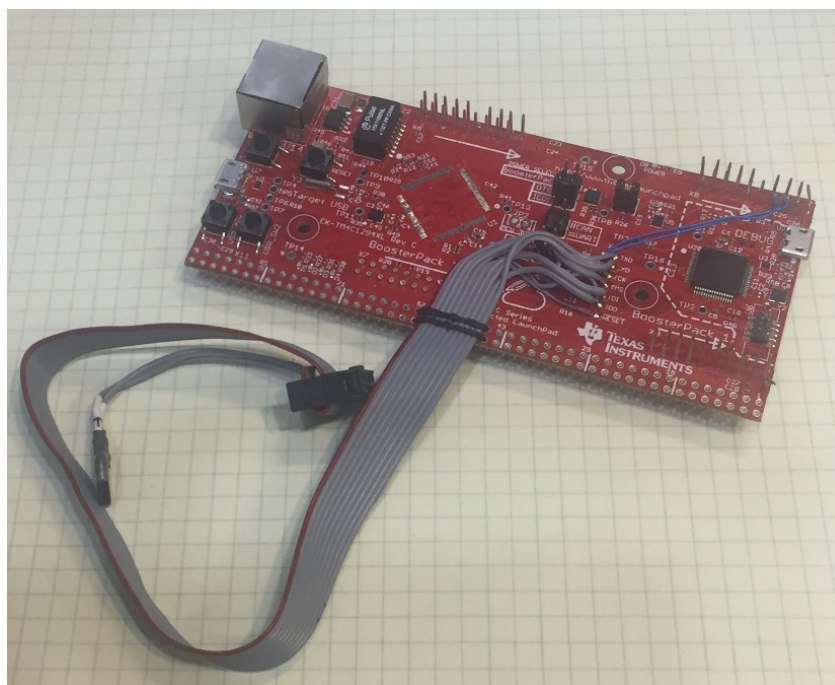

Photo 2. A modified Tiva LaunchPad for use as a programmer (*Note: the*
programming USB connector is on the right as shown in the photo).

To modify a LaunchPad, start by fabricating a 10-pin ribbon cable (8", max) with an IDC connector at one end (e.g., Kobi Conn: 164-9006-E (conn) and 3M: 3801/10 (100') (cable)). Separate the wires at the un-terminated end of the ribbon cable, then strip and tin each wire 0.1". *Note: Some LaunchPads feature a dual row header with shorting jumpers while others have a hole-pattern for such a header-jumper but have traces connecting the holes, and no header is installed*.

For the boards that have no header – cut the traces connecting each pad pair (cut BETWEEN the pads only). Then install a header (so that the LaunchPad target device can be programmed if desired) and solder the ribbon cable wires to the back side of the board on the side closest to the programming MCU. *Note that the LaunchPad shown in Photo 2 has had its dual-row header removed, so the ribbon cable is shown connected to the top side*.

For the boards with a header, remove the jumpers and store them on the same header such that they are secured on only one of the header pins. Solder the ribbon cable wires to the back side of the board on the side closest to the programming MCU.

The LaunchPad programmers also feature a serial port service that can be used for debug or user interface with the target device. This connection is generally a 3-pin, 0.1" spacing header on my target boards. To use this serial connection, connect a 3-conductor cable to the appropriate target connector, and solder the appropriate wires to GND, RXD, and TXD on either the SFF 10-pin connector, or on the LaunchPad board. To use the serial port, simply connect the LaunchPad programmer USB connection to the PC, start the terminal emulator of your choice, and select the appropriate COM port.

To use the programmer with Code Composer Studio (https://www.ti.com/tool/CCSTUDIO), add or edit a target configuration file to the project and select the "Stellaris In-Circuit Debug Interface" along with the target processor – see Figure 4. To program the Tiva target device, simply start a debug session in Code Composer Studio. Use the CCS debug icons to start/stop/reset the target software/hardware.
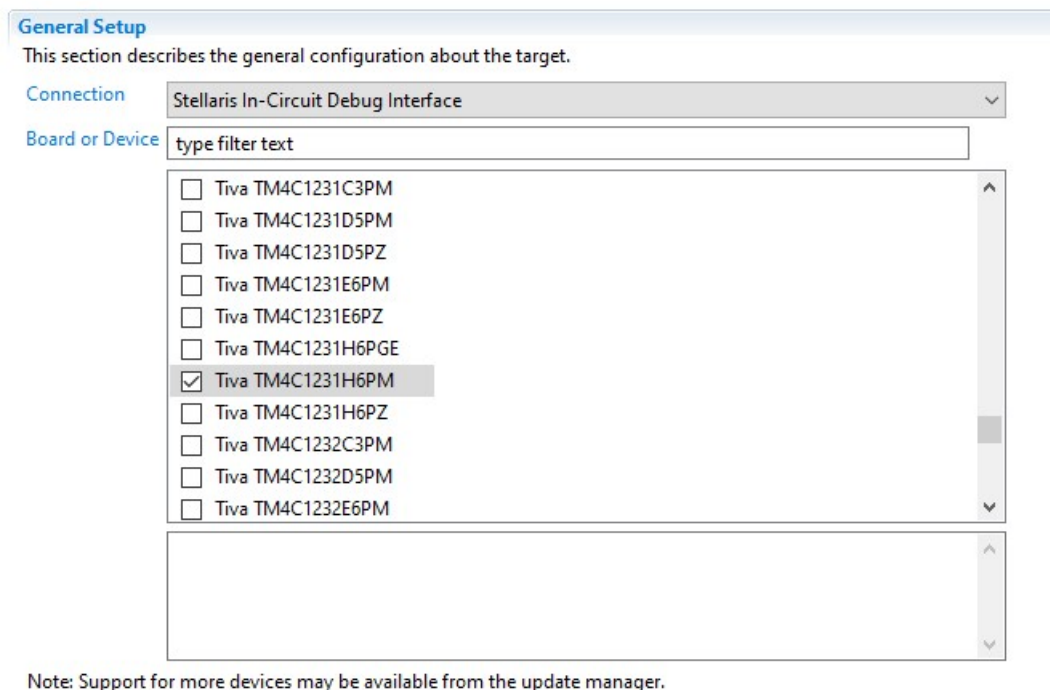

Figure 4. Code Composer Studio target configuration.

5

ATtiny (UPDI) Programming Hardware Guide

The small form-factor connector used for the SiLabs processors is also used in some of my ATtiny designs for programming. The connections are shown in Table 3.

Target pin (notes)	Nano pin	PGMR Signal	P1	**10-pin** pgmr		P1	PGMR Signal	Nano pin	Target pin (notes)
PB3 (opt)	4	TARG_RXD	6	1	2	5	GND	30	GND
GND	15	GND	5	3	4	2	TARG_Vdd	29	Vdd
/RESET_UPDI	4	TARG_UPDI	1	5	6	4	-	-	-
PB2 (opt)	3	TARG_TXD	3	7	8	-	-	-	(Tiva TXD)
GND	20	GND	5	9	10	-	-	-	(Tiva RXD)

Table 3. ATtiny-to-SFF programming adapter. P1 is the SFF target
device connector pinout. *See page 3 for SFF adapter construction details.*

An ATTINY3217 Curiosity Nano evaluation board can be modified to act as a programming interface for any UPDI-based ATtiny target MCU. In addition to performing UPDI programming, the Nano programmer also offers a USB COM-PORT that can be used to communicate with the target device UART, if so connected (these are the TARG_RXD and TARG_TXD signals indicated in Table 3). Using a Nano as a programmer is generally much less expensive than buying a commercial programmer, but requires some "assembly" to get a working system. Photo 3 illustrates the pad-cuts needed to isolate the target MCU on the Nano board.
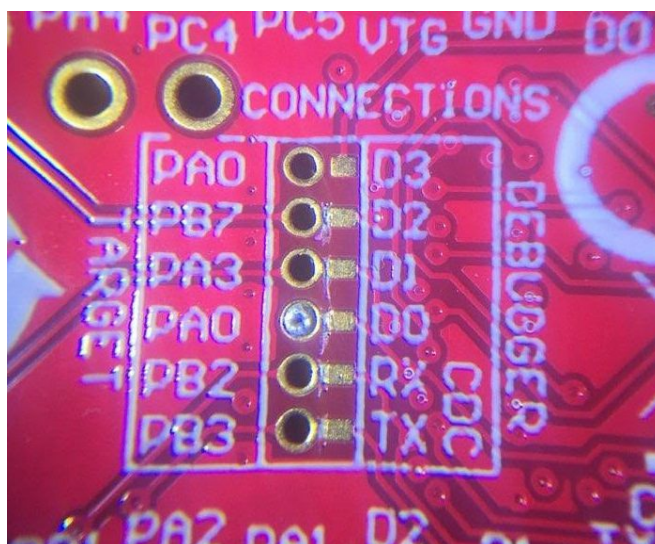

Photo 3. Cut-pads on the Curiosity Nano 3217 (located on the bottom of the board).

Photo 4 illustrates a jumper to allow the UPDI signal to be externally re-routed back to the Nano target MCU if it is later desired to access that device (externally connect Nano pin 31 to Nano pin 1, typically through a switch or shunt-jumper).
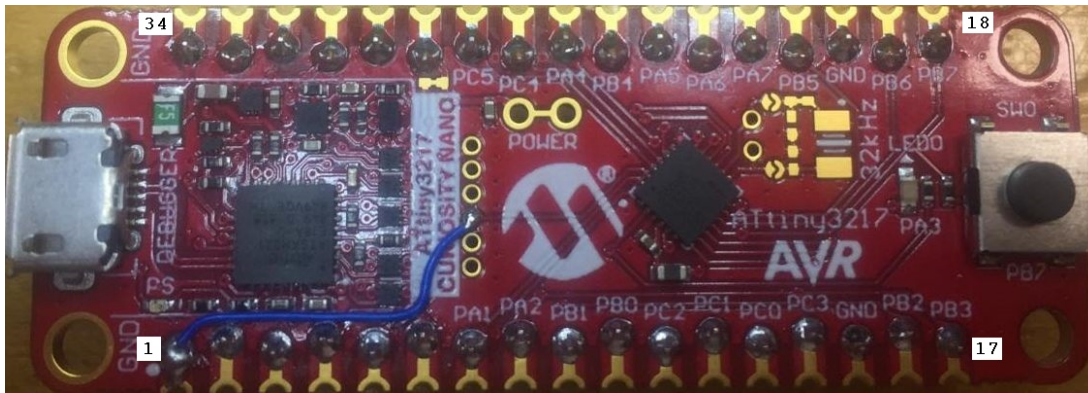
6

Photo 4. A modified Curiosity Nano 3217 showing the jumper needed (from "D0" to header pin 1) to allow the external reconnection of the 3217 target to the programming interface.

Atmel Studio 7 Setup

Atmel Studio 7 is a full-featured IDE which includes Edit, Compile/Link, Debug, and device programming functions. Projects created for the Curiosity Nano 3217 will seamlessly connect to the application allowing easy debug and programming connection. However, using an MCU other than the 3217 is not directly supported and requires some adjustments to the Atmel Studio configuration.

To configure Atmel Studio, start the application and open or create the project for the desired MCU. Connect the Curiosity Nano 3217 that has been configured to operate as a stand-alone programmer to a USB port on the PC (ideally, the actual target MCU will also be connected):

- Go to "Project" and select "<proj> Properties..." (where <proj> is the project name).
- Click on "Device" then "Change Device". Select the ATTINY3217.
- Click on "Tool", then click on the dropdown under "Selected debugger/programmer" and select the nEDBG option that shows up. Make sure that the "Interface" is set to "UPDI".
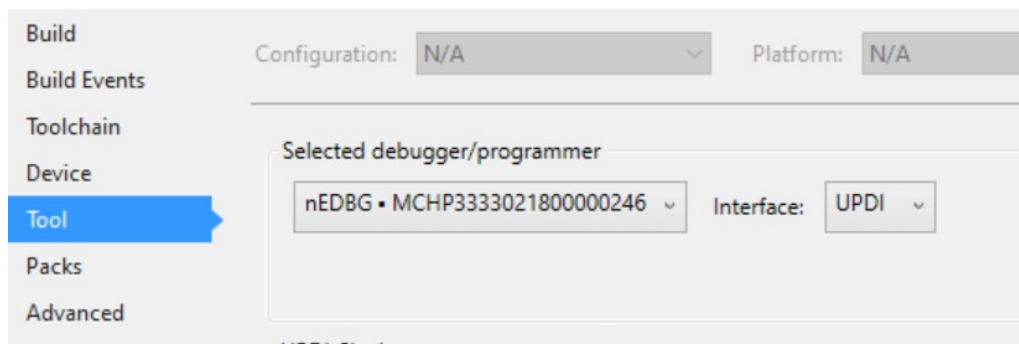

- Click on "Device" then "Change Device" (again). Select the actual target device number (e.g., ATtiny1616).
- Click on "Tool" (again), then click on the dropdown under "Selected debugger/programmer" and select the nEDBG option that is available (if not already showing).

While somewhat convoluted, these steps seem to be a sequence that works allowing one to use the debugger feature of the IDE to load and monitor the device object code using the CuriosityNano 3217 board.

7

Device Programming Settings

Some projects require the device settings to be modified. In particular, the device Vdd voltage can be adjusted. To make this adjustment, go to "Tools" and select "Device Programming":
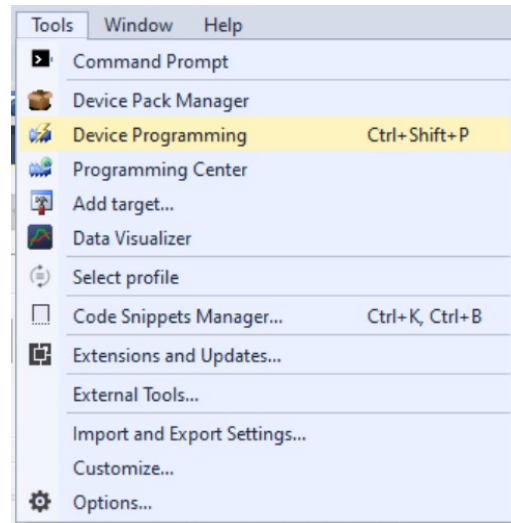
Manually type in the device type in the "Device" drop-down field then click "Apply":
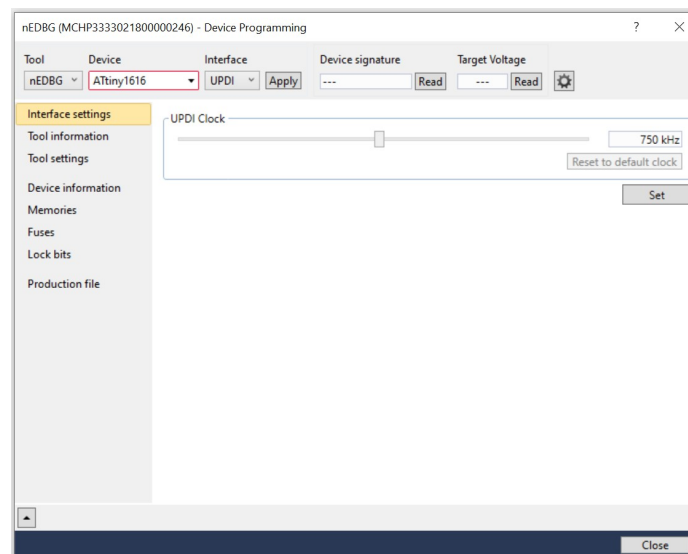
Typically, the device type will be "ATtinyXXXX", where the "XXXX" are the digits of the device part#.

*!! Note: Unfortunately, this step must be performed each
time the "Device Programming" window is selected !!*

8

Adjust the supply voltage to the desired value (0V and values between 1.6V and 5.5V are available) and click "Write":
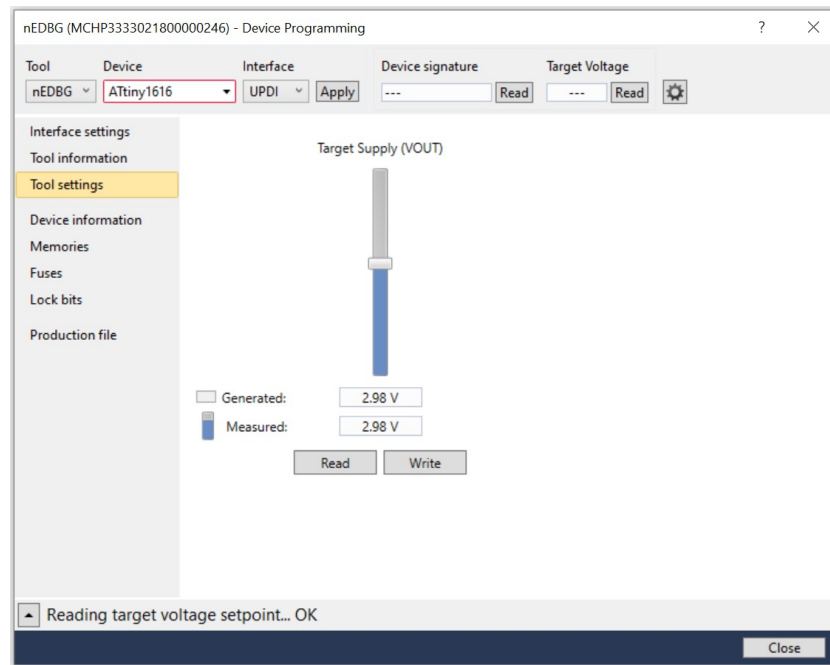

After all required configuration settings are completed, click "Close".

Atmel Studio Optimization Settings

Some projects require the complier optimizations to be removed. If this is needed, go to the "Project" menu and select "<projx> Properties..." (where <projx> is the project name):
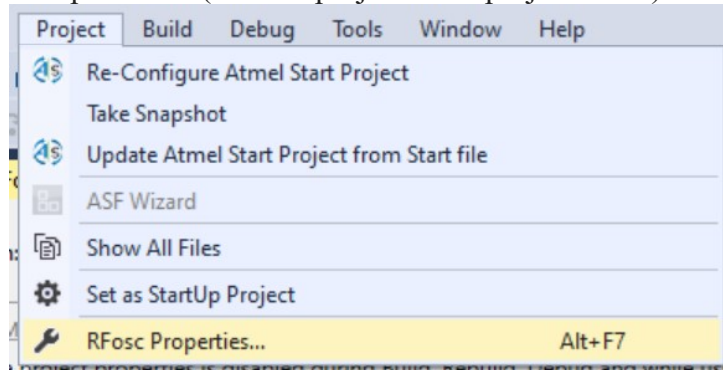
Click on "Toolchain" then, under "AVR/GNU C Compiler", select "Optimization". Click on the "Optimization Level:" drop-down field and select "None":

10
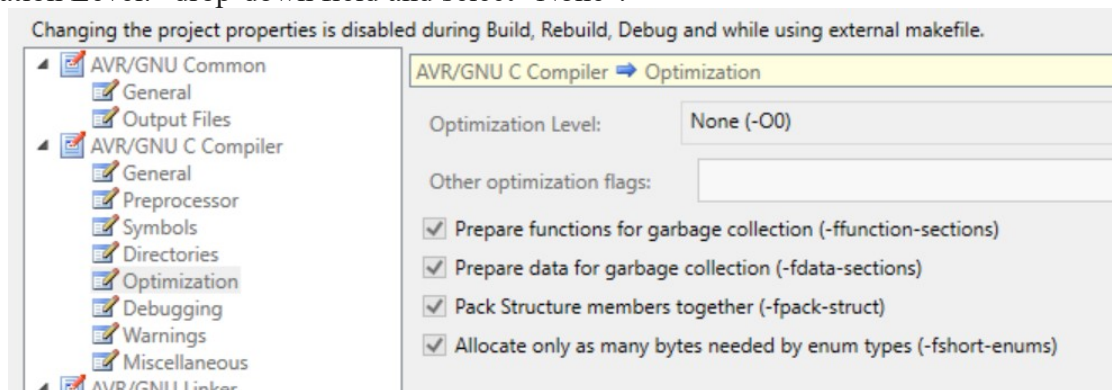
Revision History

Rev 3.0, 06/09/2022:
- Added ATtiny section
- Added JTAG signals to SiLabs SFF pinout
- Added details for modifying the SFF adapter to support Tiva connections
- Cleaned up headers/footers

Rev 2.0, 08/04/2021:
- Added Tiva section

Rev 1.0, circa 2016
- Initial release