uDEV Soluções Tecnológicas

Multi-Sensor Sigfox Embedded Development Kit (EDK)

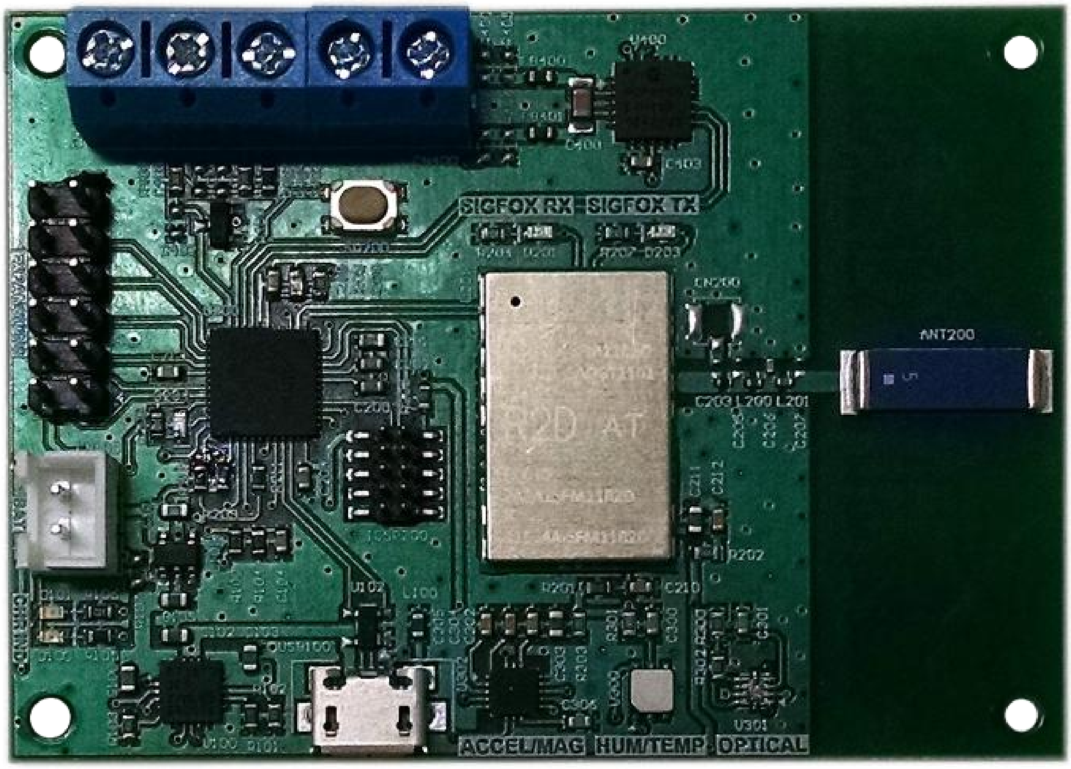


Table of Contents

Revision History

Version	Description	Date
1.0	First version	2017-11-24
1.1	Added Sigfox Activation Section	2018-02-22

1. Introduction

The SIGFOX Embedded Development Kit (EDK) is a multi-sensor board, designed to be employed by developers aiming to use SIGFOX as a means of easy data transmission, especially for Internet of Things (IoT) applications.

Using the on-board sensors in the EDK allows for the user or developer to easily obtain a plethora of data without the need for additional devices or external sensors, and send this data to the cloud seamlessly via the SIGFOX modem. A great deal of applications are possible, such as monitoring different environment variables and signaling any desired changes.

2. Features

The EDK contains the following features:

- High-performance ARM Cortex-M0+ ATSAMD21G18A-M
- Temperature and humidity sensor
- Ambient light sensor
- Connectors for analog input
- Integrated Real-Time Clock
- On-board SIGFOX modem
- On-board Antenna
- Micro USB connection
- External battery connector, with Integrated Battery Charge Management
- 6-axis motion sensor[1]
- Input-ready for external thermocouples[2]

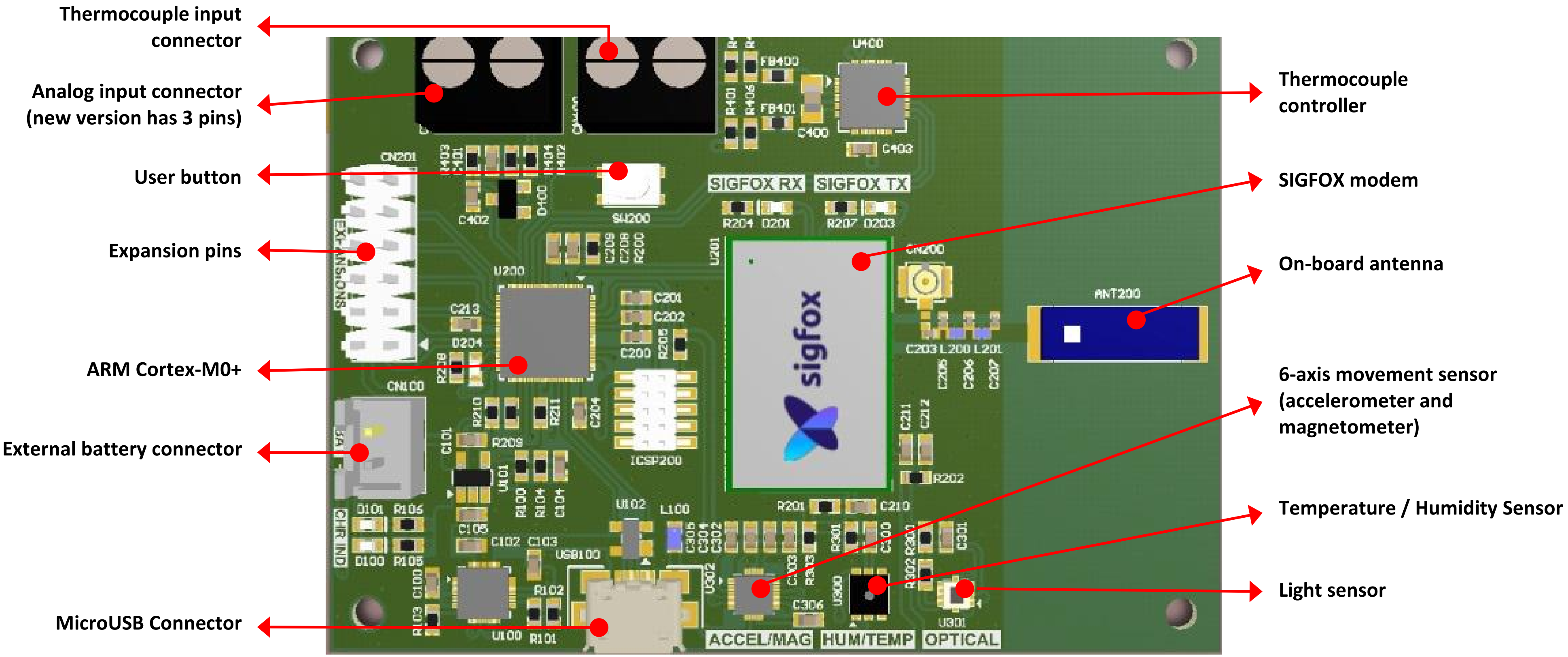
[1] *Not all models include the motion sensor (accelerometer + magnetometer)*

[2] *Not all models include the thermocouple controller and input connectors*

3. Hardware

The EDK is fully designed at and by uDEV, and allows developers to create quick solutions for many IoT-related systems.

The kit is centered around the ARM Cortex-M0+ ATSAMD21G18A-M processor, and has a ready-to-use SIGFOX modem, complete with on-board antenna for readily available communication with the SIGFOX network.



Thermocouple input connector

Analog input connector (new version has 3 pins)

User button

Expansion pins

ARM Cortex-M0+

External battery connector

MicroUSB Connector

Thermocouple controller

SIGFOX modem

On-board antenna

6-axis movement sensor (accelerometer and magnetometer)

Temperature / Humidity Sensor

Light sensor

3.1. General

Main Microprocessor:

- **ARM Cortex-M0+ ATSAMD21G18A-M**
    - The most energy-efficient Cortex processor to date
    - Integrated power management with many low-power functions
    - **More information and full specification page available at https://developer.arm.com/products/processors/cortex-m/cortex-m0-plus**

3.2. Sensors

Humidity and Temperature Sensor:

- **Silicon Labs Si7006-A20**
- **Precision Relative Humidity Sensor:**
    - 0-90% Relative Humidity (RH), ± 5% RH (max.)
    - Operating Range: 0 to 100% RH
- **High Accuracy Temperature Sensor:**
    - -10 to +85 ºC, ± 1 ºC (max.)
    - Operating Temperature: -40 to +125 ºC
- **Low power consumption**
    - 150 µA active current
    - 60 nA standby current
- **Factory-calibrated**
- **Full datasheet available at: https://www.silabs.com/documents/public/data-sheets/Si7006-A20.pdf**

Light Sensor:

- **Silicon Labs Si1133**
  - **High accuracy UV index sensor**
  - **Ambient Light Sensor**
    - Resolution of under 100 mlx possible
  - **Operating Temperature:** -40 to +85 ºC
  - **Low power consumption**
    - 500 nA standby current (max.)
  - **Full datasheet available at:** **https://www.silabs.com/documents/public/data-sheets/Si1133.pdf**

Thermocouple input[1]:

- **Microchip MCP9600**
  - **Supports Thermocouple types K, J, T, N, S, E, B, R**
  - **Integrated Cold-Junction Compensation**
  - **Four Programmable Temperature Alert Outputs**
  - **I2C Compatible**
  - **Low Power:**
    - Shutdown mode: 2 µA typical current
    - Burst Mode: 1 to 128 Temperature Samples
  - **Full datasheet available at:** **http://ww1.microchip.com/downloads/en/DeviceDoc/20005426C.pdf**

One-Wire Thermometer:

- **Maxim Integrated DS18B20**
  - **Temperature Sensor:**
    - -55 to +125 ºC
    - Accuracy of ± 0.5 ºC from -10 to +85ºC
    - Operating / Storage Temperature Range: -55 to +125 ºC
  - **Programmable resolution**
  - **Full datasheet available at:** **https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf**

6-axis sensor[1]:

- **Digikey FXOS8700CQ**
  - **Integrated linear 3-axis accelerometer**
  - **Integrated 3-axis magnetometer**
  - **Complete e-compass hardware solution**
  - **Embedded programmable acceleration event functions**
    - Freefall and motion detection
    - Transient detection
    - Vector-magnitude change detection
    - Pulse and tap detection (single and double)
    - Orientation detection (portrait / landscape)
  - **Embedded programmable magnetic event functions**
    - Threshold detection
    - Vector-magnitude change detection
    - Autonomous magnetic min/max detection
    - Autonomous hard-iron calibration
  - **Low power consumption**
    - 240 µA current consumption at 100 Hz
    - 80 µA current consumption at 25 Hz with both sensors active
- **Full datasheet available at: https://media.digikey.com/pdf/Data%20Sheets/NXP%20PDFs/FXOS8700CQ.pdf**

[1] *Optional sensors, not included in every EDK version. Please check prior to purchasing.*

3.3. Power

Stand-Alone System Load Sharing and Li-Ion / Li-Polymer Battery Charge Management Controller:

- **Integrated System Load Sharing and Battery Charge Management**
  - Simultaneously power the system and charge the battery
  - Voltage Proportional Current Control (VPCC) ensures system load has priority over battery charge current
- **Complete Linear Charge Management Controller**
  - Integrated Pass Transistors
  - Integrated Current Sense
  - Integrated Reverse Discharge Protection
  - Selectable Input power sources
- **Automatic End-of-Charge control**
- **Refer to the complete datasheet for more information:** **http://ww1.microchip.com/downloads/en/DeviceDoc/20002090C.pdf**

This controller allows for this circuit to be powered by either an external battery, or directly via the Micro USB port (for debugging purposes).

Any capacity battery can be used, but it **must conform to the following specifications**:

- Type: Lithium-Ion (Li-ion) or Lithium-Ion Polymer (Li-Po)
- Voltage: 3.7 V
- Recharge Current: 100mA
- Connector: JST-XH-A (DigiKey Part No. 455-2247-ND)

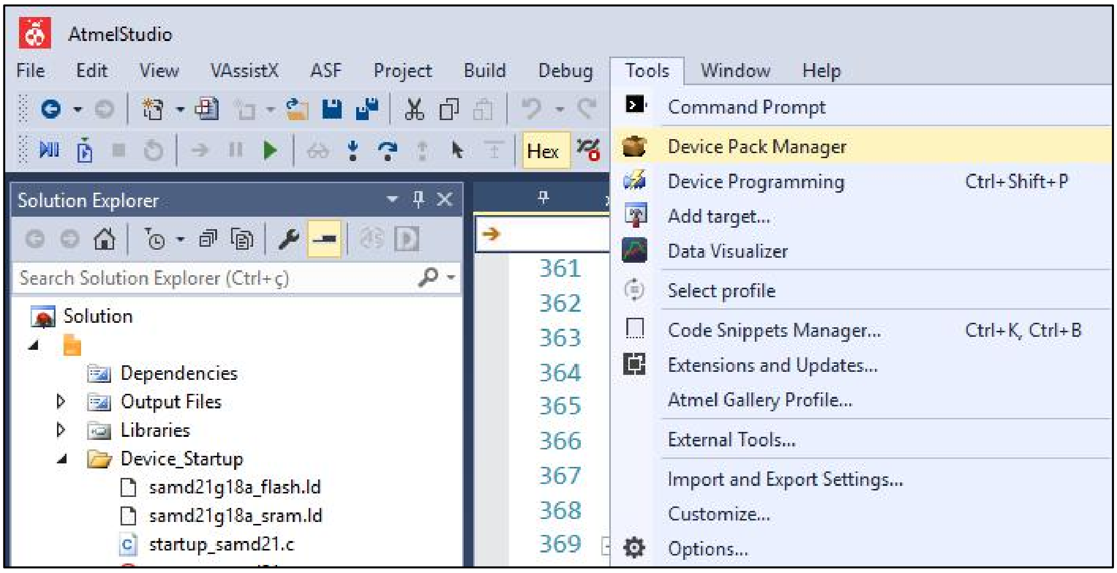
Figure: JST-B2B-XH-A Male Connector

4. Installing the SAM D21 processor files

4.1. Via Device Pack Manager

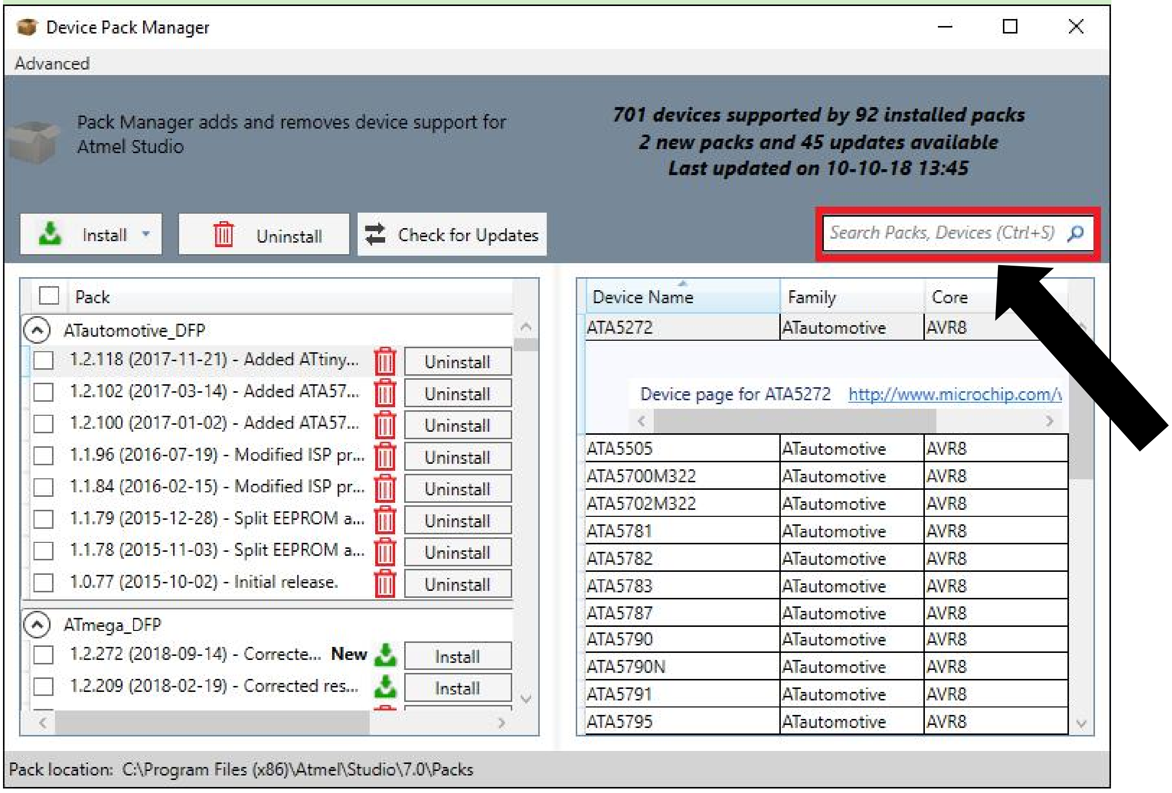
By default, the Atmel Studio installation does not include the SAM D21 processor files. These files need to be included to solve the “**samd21.h not included**” or “**samd21g18a.h not included**” errors. To install them, follow the guide below.

First of all, open the Device Pack Manager, under Tools.



If the “Device Pack Manager” option is not available in your Tools tab, please jump to section 4.2 and come back after completing that.

After clicking on **Device Pack Manager**, a new window should pop up (pictured below)



Device Pack Manager

Advanced

Pack Manager adds and removes device support for Atmel Studio

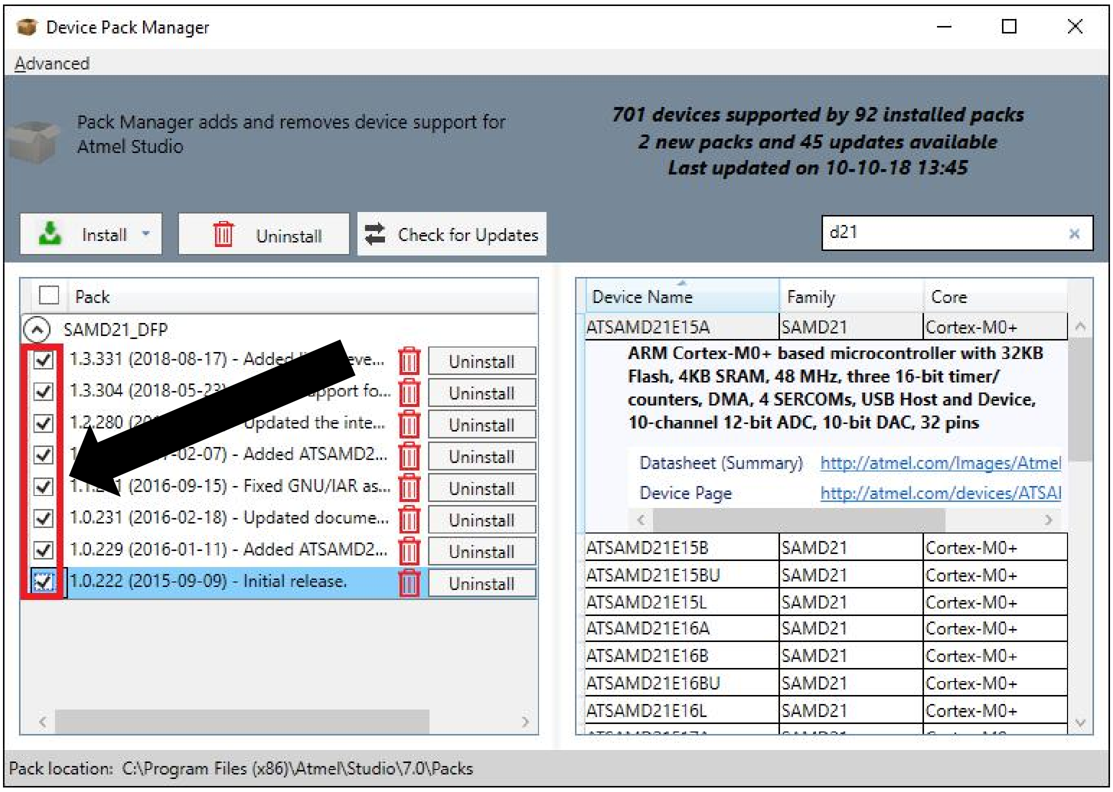
701 devices supported by 92 installed packs
2 new packs and 45 updates available
Last updated on 10-10-18 13:45

□	Pack	
	ATautomotive_DFP	
□	1.2.118 (2017-11-21) - Added ATtiny...	Uninstall
□	1.2.102 (2017-03-14) - Added ATA57...	Uninstall
□	1.2.100 (2017-01-02) - Added ATA57...	Uninstall
□	1.1.96 (2016-07-19) - Modified ISP pr...	Uninstall
□	1.1.84 (2016-02-15) - Modified ISP pr...	Uninstall
□	1.1.79 (2015-12-28) - Split EEPROM a...	Uninstall
□	1.1.78 (2015-11-03) - Split EEPROM a...	Uninstall
□	1.0.77 (2015-10-02) - Initial release.	Uninstall
	ATmega_DFP	
□	1.2.272 (2018-09-14) - Correcte... New	Install
□	1.2.209 (2018-02-19) - Corrected res...	Install

Device Name	Family	Core
ATA5272	ATautomotive	AVR8
Device page for ATA5272	http://www.microchip.com/	
ATA5505	ATautomotive	AVR8
ATA5700M322	ATautomotive	AVR8
ATA5702M322	ATautomotive	AVR8
ATA5781	ATautomotive	AVR8
ATA5782	ATautomotive	AVR8
ATA5783	ATautomotive	AVR8
ATA5787	ATautomotive	AVR8
ATA5790	ATautomotive	AVR8
ATA5790N	ATautomotive	AVR8
ATA5791	ATautomotive	AVR8
ATA5795	ATautomotive	AVR8

Pack location: C:\Program Files (x86)\Atmel\Studio\7.0\Packs

In this window, there is a search bar. Insert the search terms “**D21**” and only one family of processors should appear. <u>**Manually**</u> select every package in the **SAMD21_DFP** family, as pictured below:



Device Pack Manager

Advanced

Pack Manager adds and removes device support for Atmel Studio

701 devices supported by 92 installed packs
2 new packs and 45 updates available
Last updated on 10-10-18 13:45

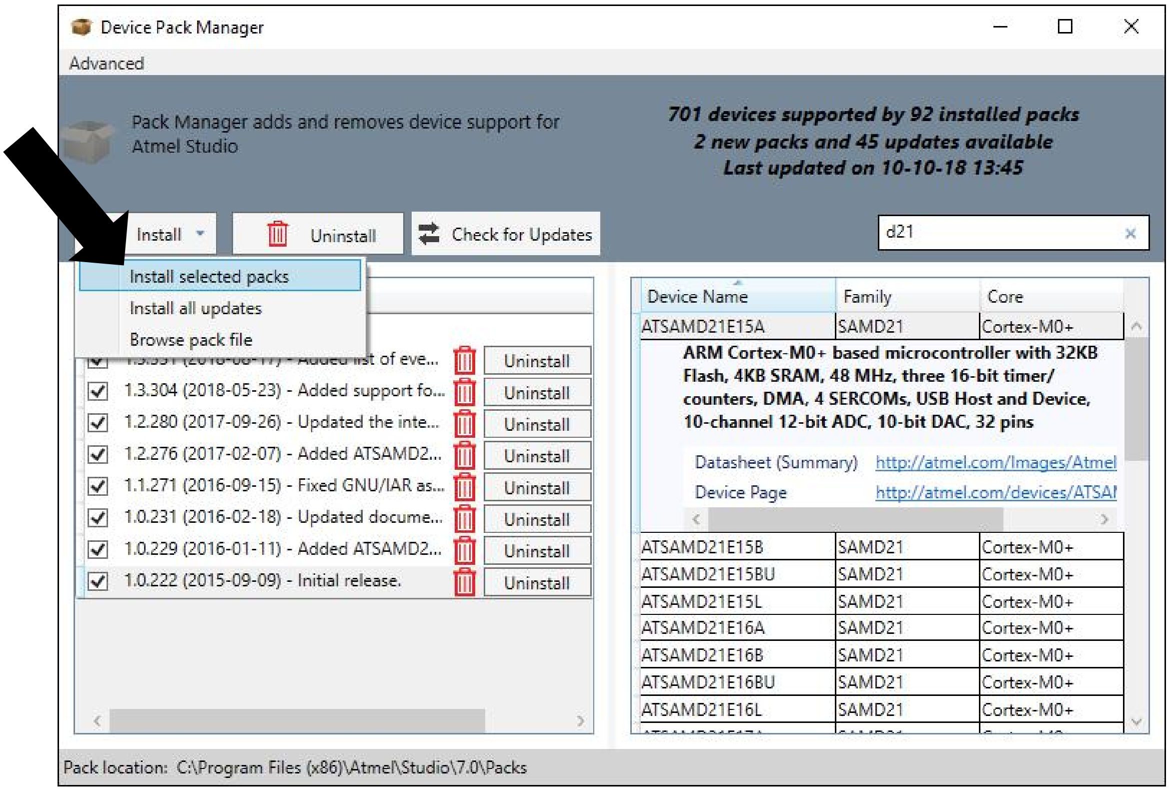
□	Pack	
	SAMD21_DFP	
☑	1.3.331 (2018-08-17) - Added ...eve...	Uninstall
☑	1.3.304 (2018-05-2... ...pport fo...	Uninstall
☑	1.2.280 (20... Updated the inte...	Uninstall
☑	1...-02-07) - Added ATSAMD2...	Uninstall
☑	1.1...1 (2016-09-15) - Fixed GNU/IAR as...	Uninstall
☑	1.0.231 (2016-02-18) - Updated docume...	Uninstall
☑	1.0.229 (2016-01-11) - Added ATSAMD2...	Uninstall
☑	1.0.222 (2015-09-09) - Initial release.	Uninstall

Device Name	Family	Core
ATSAMD21E15A	SAMD21	Cortex-M0+
ARM Cortex-M0+ based microcontroller with 32KB Flash, 4KB SRAM, 48 MHz, three 16-bit timer/ counters, DMA, 4 SERCOMs, USB Host and Device, 10-channel 12-bit ADC, 10-bit DAC, 32 pins		
Datasheet (Summary)	http://atmel.com/Images/Atmel	
Device Page	http://atmel.com/devices/ATSAI	
ATSAMD21E15B	SAMD21	Cortex-M0+
ATSAMD21E15BU	SAMD21	Cortex-M0+
ATSAMD21E15L	SAMD21	Cortex-M0+
ATSAMD21E16A	SAMD21	Cortex-M0+
ATSAMD21E16B	SAMD21	Cortex-M0+
ATSAMD21E16BU	SAMD21	Cortex-M0+
ATSAMD21E16L	SAMD21	Cortex-M0+

Pack location: C:\Program Files (x86)\Atmel\Studio\7.0\Packs

After that, click on “Install” and choose “**Install selected packages**” <u>only</u>.

Be careful to **NOT** click on “Install all updates”, as that will install every update available for every family of processors and every package that can be updated. This would take a very long time and is unnecessary.



Pack Manager adds and removes device support for Atmel Studio

701 devices supported by 92 installed packs
2 new packs and 45 updates available
Last updated on 10-10-18 13:45

☑	1.3.331 (2018-08-17) - Added list of eve...	Uninstall
☑	1.3.304 (2018-05-23) - Added support fo...	Uninstall
☑	1.2.280 (2017-09-26) - Updated the inte...	Uninstall
☑	1.2.276 (2017-02-07) - Added ATSAMD2...	Uninstall
☑	1.1.271 (2016-09-15) - Fixed GNU/IAR as...	Uninstall
☑	1.0.231 (2016-02-18) - Updated docume...	Uninstall
☑	1.0.229 (2016-01-11) - Added ATSAMD2...	Uninstall
☑	1.0.222 (2015-09-09) - Initial release.	Uninstall

Device Name	Family	Core
ATSAMD21E15A	SAMD21	Cortex-M0+
ATSAMD21E15B	SAMD21	Cortex-M0+
ATSAMD21E15BU	SAMD21	Cortex-M0+
ATSAMD21E15L	SAMD21	Cortex-M0+
ATSAMD21E16A	SAMD21	Cortex-M0+
ATSAMD21E16B	SAMD21	Cortex-M0+
ATSAMD21E16BU	SAMD21	Cortex-M0+
ATSAMD21E16L	SAMD21	Cortex-M0+

ARM Cortex-M0+ based microcontroller with 32KB Flash, 4KB SRAM, 48 MHz, three 16-bit timer/counters, DMA, 4 SERCOMs, USB Host and Device, 10-channel 12-bit ADC, 10-bit DAC, 32 pins

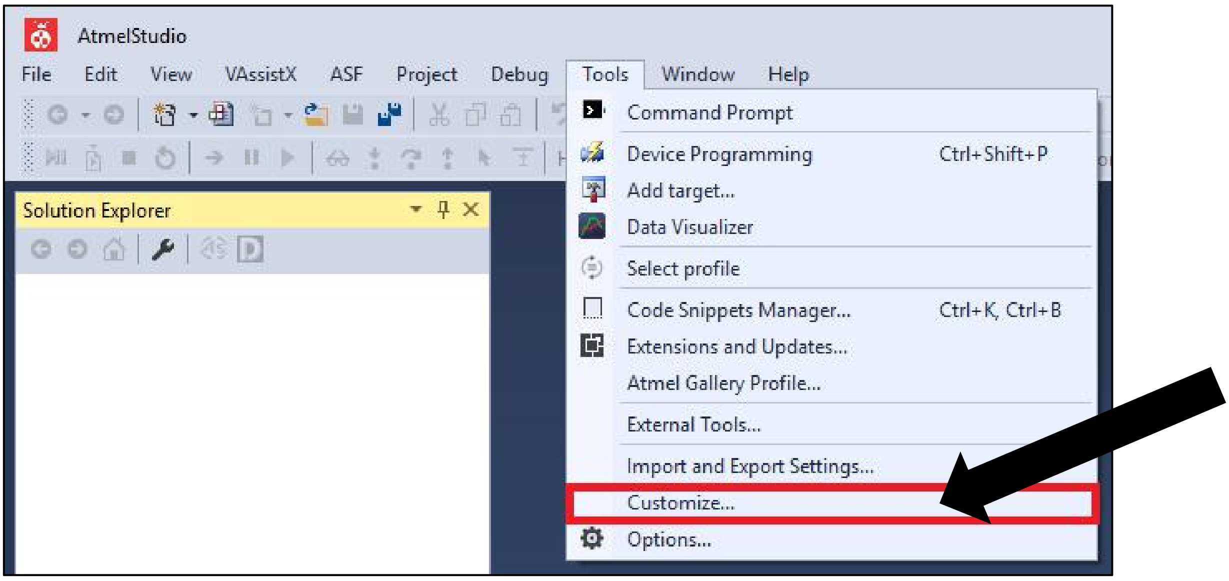
Datasheet (Summary) http://atmel.com/Images/Atmel
Device Page http://atmel.com/devices/ATSAI

Pack location: C:\Program Files (x86)\Atmel\Studio\7.0\Packs

When this process completes, the SAM D21 processor files will be ready for compiling and use.

4.2. Enabling the Device Pack Manager menu

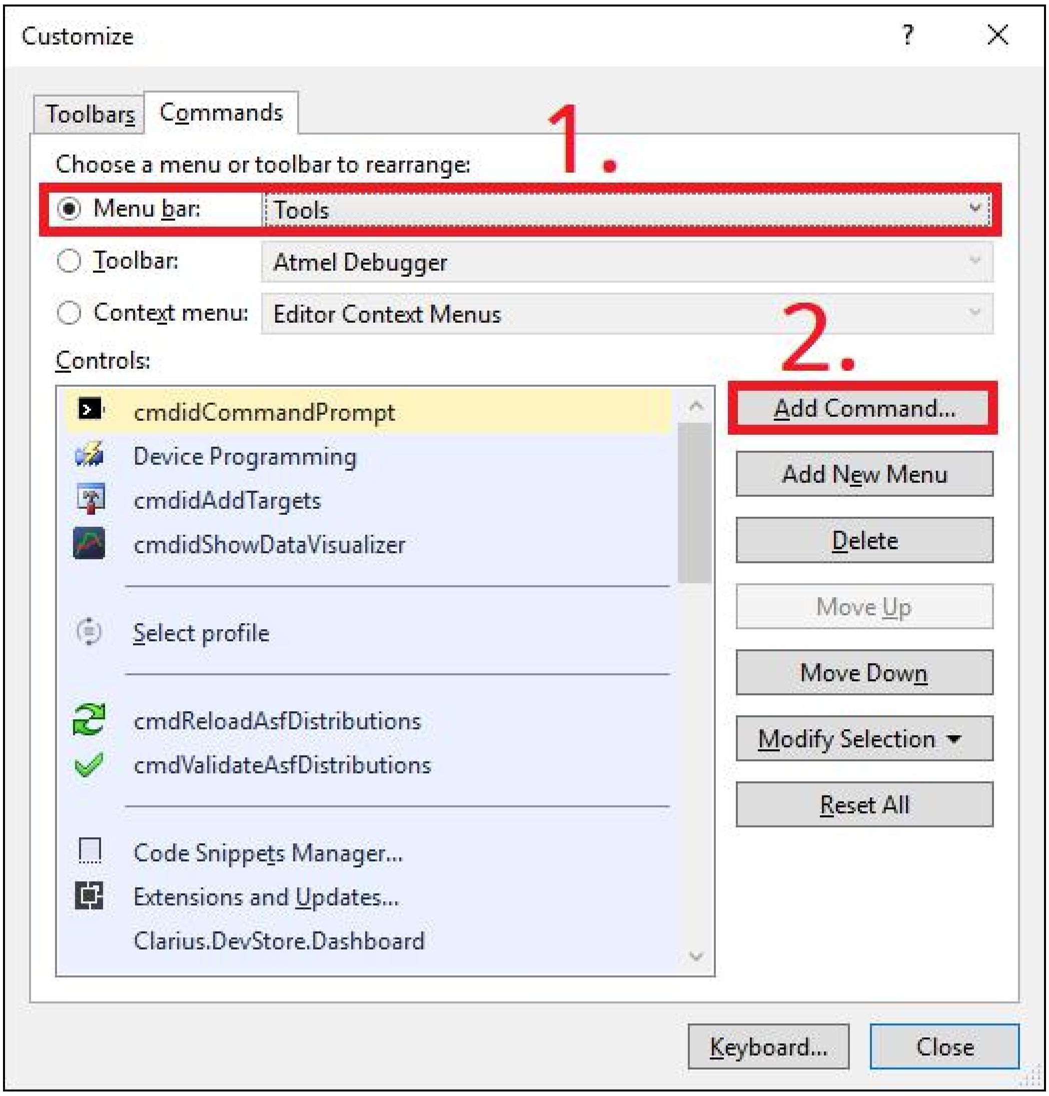
If your **Tools** tab doesn't have the **Device Pack Manager**, it is necessary to include it through a customization option.
Still on the **Tools** tab, choose **Customize…** (pictured below)



This opens a new window with two tabs: **Toolbars** and **Commands**.
Go to the **Commands** tab.

The window is pictured on the next page.

On this window:
1. Choose **Tools** under the drop-down menu on **Menu Bar**.
2. Click **Add Command...**



Customize

Choose a menu or toolbar to rearrange:

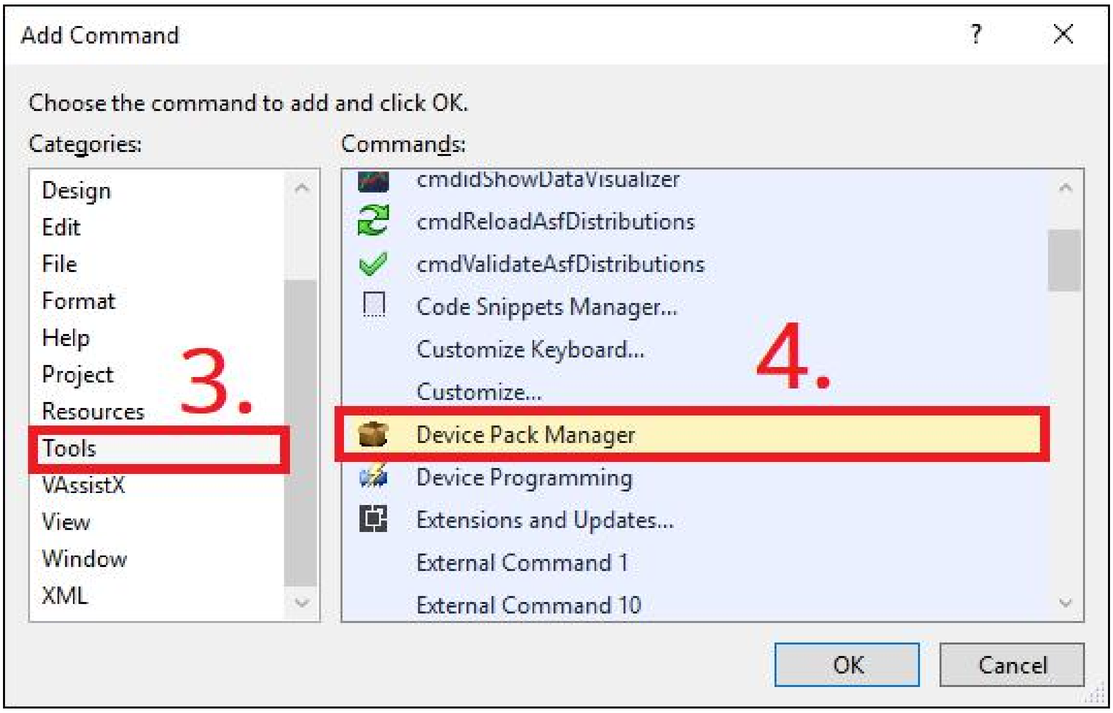
● Menu bar: Tools

○ Toolbar: Atmel Debugger

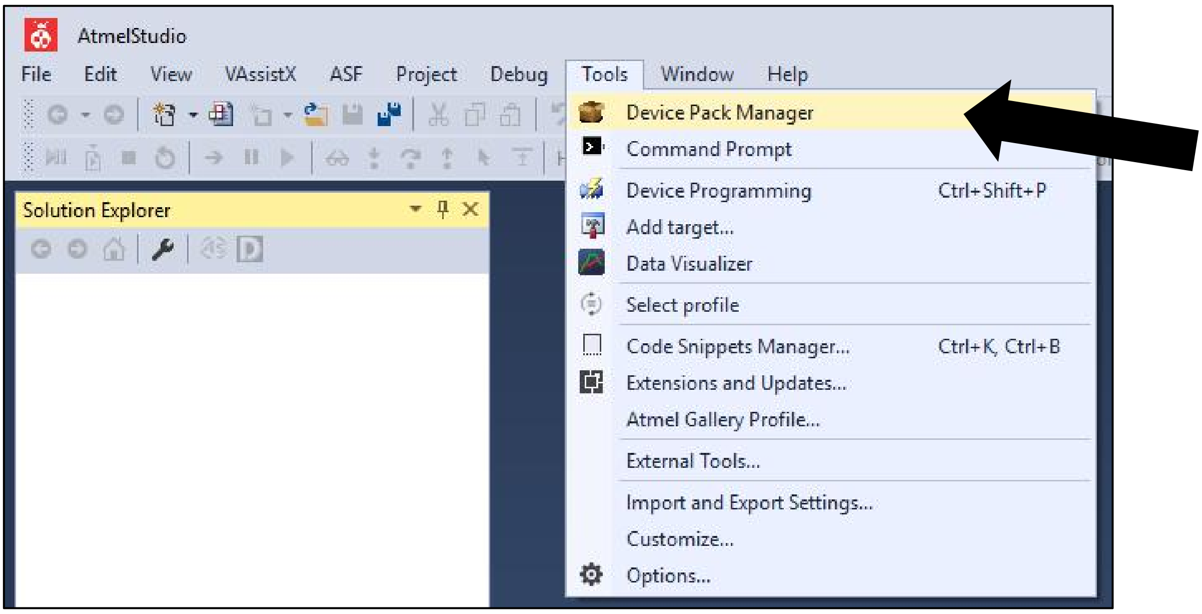
○ Context menu: Editor Context Menus

Clicking **Add Command** will open a new window, pictured in the next page.

In the **Add Command** window:
3. Find **Tools** under **Categories**.
4. Select **Device Pack Manager** on the list under **Commands**.



Click "OK", close the **Customize** window, and now the **Tools** tab should have the **Device Pack Manager** option under it:



Now you can use the **Device Pack Manager** to install the **SAM D21** processor files, go back to section 4.1 to find the procedure on how to do that!

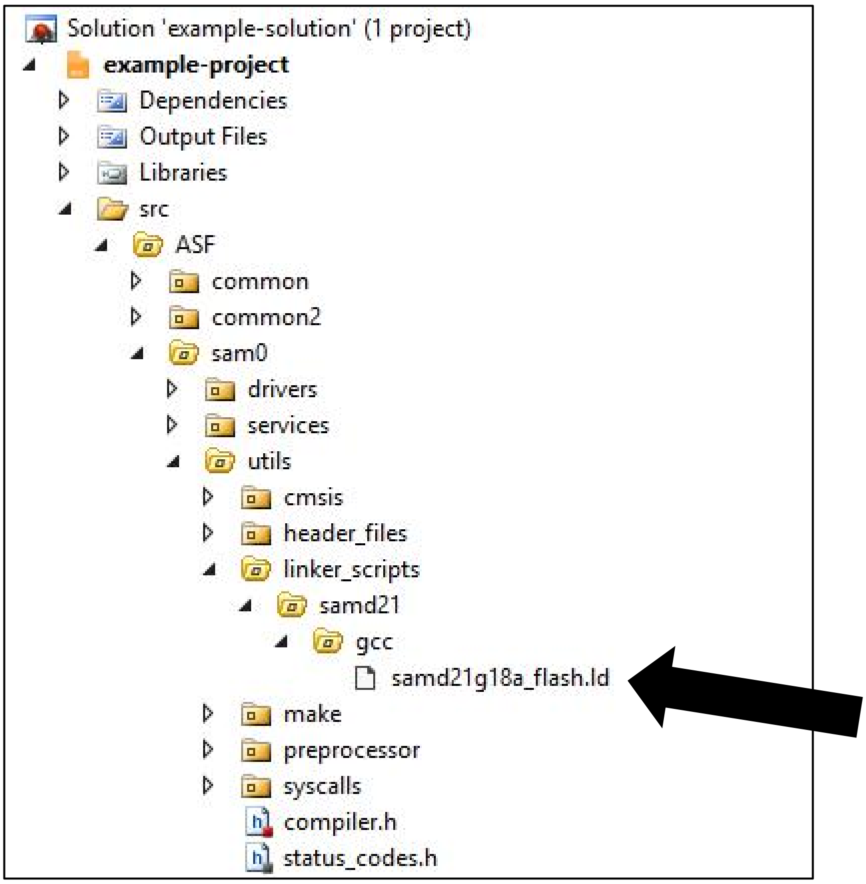
5. Memory Addressing

In order for your application to work correctly, it is necessary to set its starting address at 0x6000. This is due to the custom *Bootloader* that is present on the EDK board, which reserves FLASH memory addresses from 0x0000 to 0x6000.

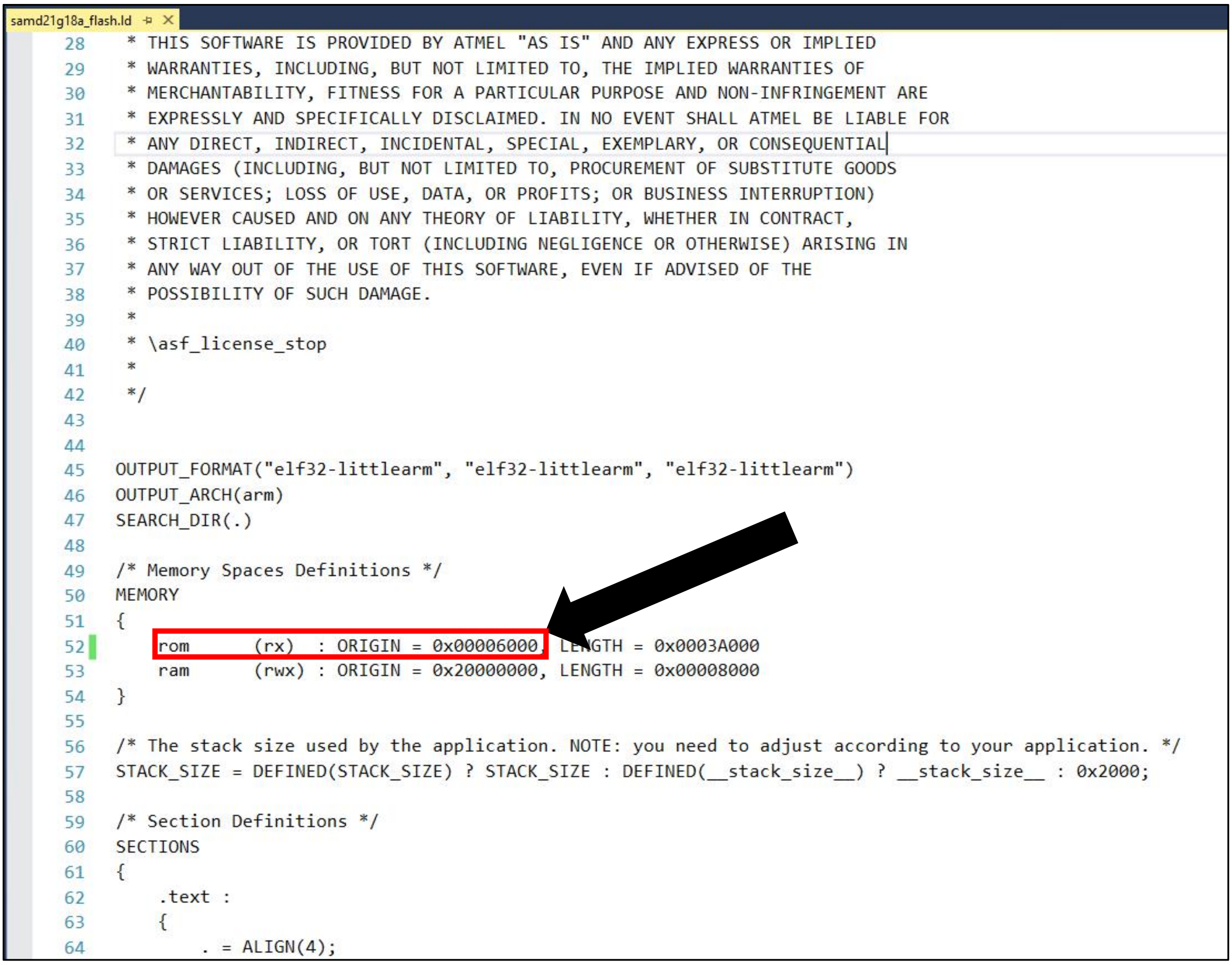
The address must be indicated **both** on the memory definitions within your application's code, and on SAM-BA, before uploading to the board. (See next chapter)
Below is an example on how to do this, building your application using Atmel Studio 7.0 and the SAMD21G18A microprocessor.
There are three methods to set a starting address:

Method A.

You can access the Linker file directly, going to the following path:
Example-solution/example-project/src/ASF/sam0/utils/linker_scripts/samd21/gcc



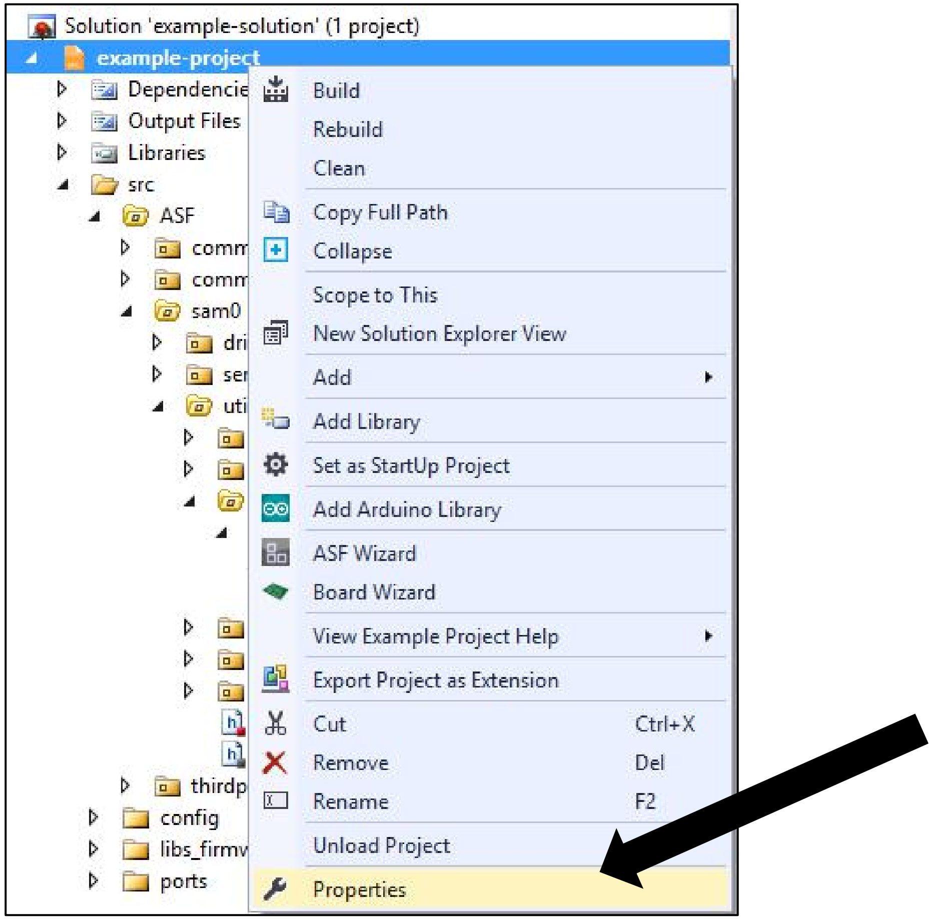
Then, you can edit the ROM ORIGIN attribute, changing the value to **0x000060000**:



```
 * THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY EXPRESS OR IMPLIED
 * WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT ARE
 * EXPRESSLY AND SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
 * ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGE.
 *
 * \asf_license_stop
 *
 */


OUTPUT_FORMAT("elf32-littlearm", "elf32-littlearm", "elf32-littlearm")
OUTPUT_ARCH(arm)
SEARCH_DIR(.)

/* Memory Spaces Definitions */
MEMORY
{
    rom      (rx)  : ORIGIN = 0x00006000, LENGTH = 0x0003A000
    ram      (rwx) : ORIGIN = 0x20000000, LENGTH = 0x00008000
}

/* The stack size used by the application. NOTE: you need to adjust according to your application. */
STACK_SIZE = DEFINED(STACK_SIZE) ? STACK_SIZE : DEFINED(__stack_size__) ? __stack_size__ : 0x2000;

/* Section Definitions */
SECTIONS
{
    .text :
    {
        . = ALIGN(4);
```

This is the way we recommend setting the Application start address.

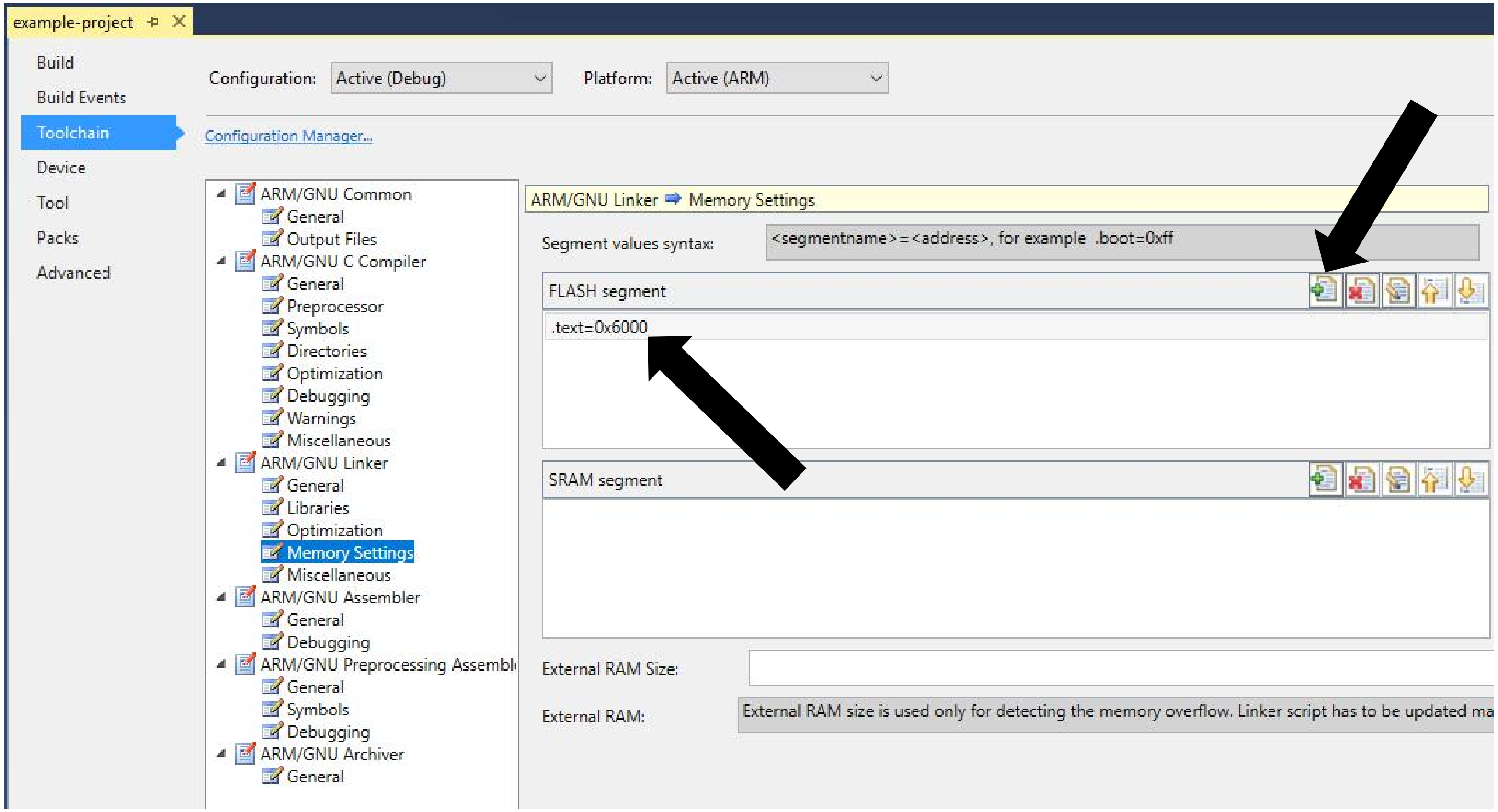
Another way to do this involves adding a command to the Toolchain in the project settings.
Right-click the project, go to **Properties**, and then **Toolchain**.



Then, you have two choices. You can either:

Method B.

- Go to **Memory Settings**, under **ARM/GNU Linker**
  - click “Add Item” under **FLASH segment**
  - write “**.text=0x6000**” and click **OK**



Configuration: Active (Debug)

Platform: Active (ARM)

Segment values syntax: <segmentname>=<address>, for example .boot=0xff

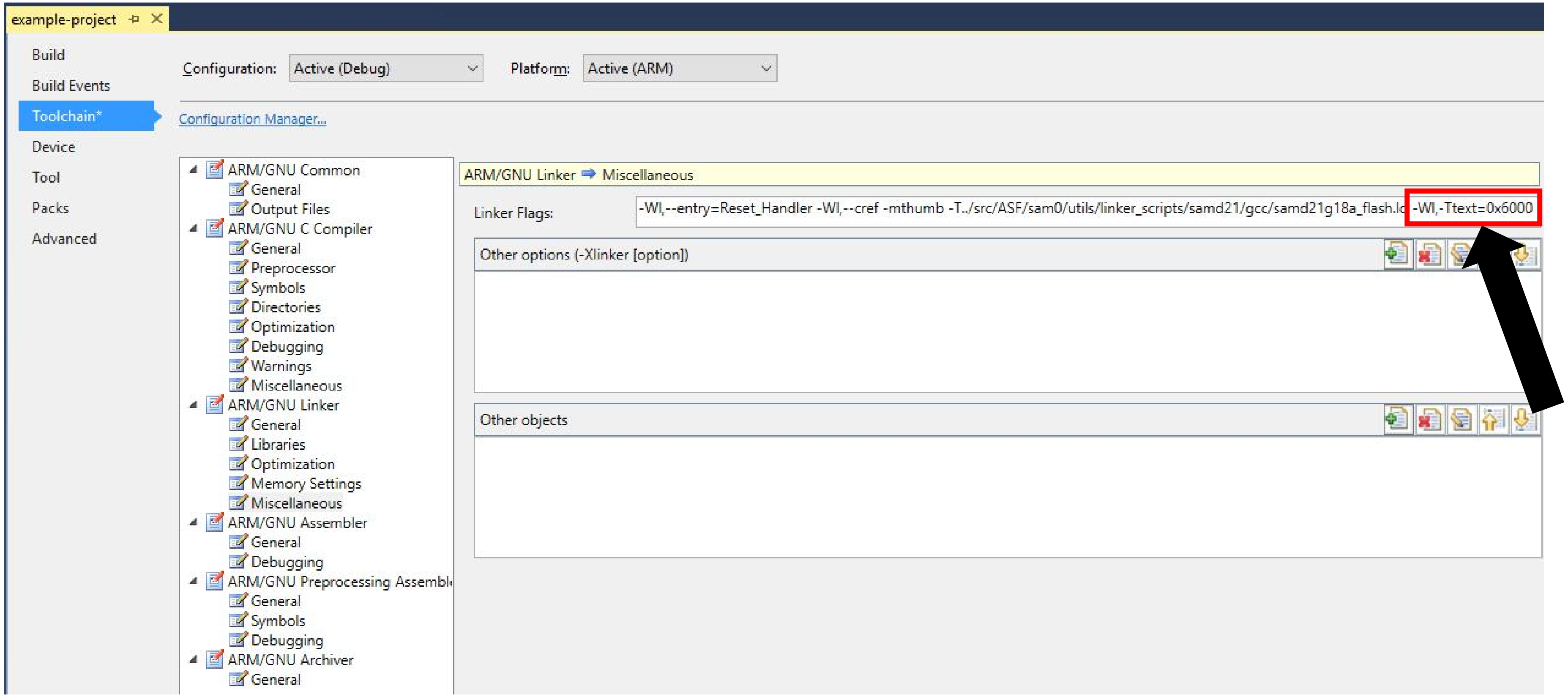
.text=0x6000

External RAM Size:

External RAM: External RAM size is used only for detecting the memory overflow. Linker script has to be updated ma

Method C.

- Go to **Miscellaneous**, under **ARM/GNU Linker**
  - Directly on **Linker Flags**, add a space before the last command
  - Write “**-Wl,-Ttext=0x6000**”



Configuration: Active (Debug)

Platform: Active (ARM)

ARM/GNU Linker ⇒ Miscellaneous

Linker Flags: -Wl,--entry=Reset_Handler -Wl,--cref -mthumb -T../src/ASF/sam0/utils/linker_scripts/samd21/gcc/samd21g18a_flash.ld -Wl,-Ttext=0x6000

Any of the previous methods will build your project at the correct address, and your program will be ready to upload.

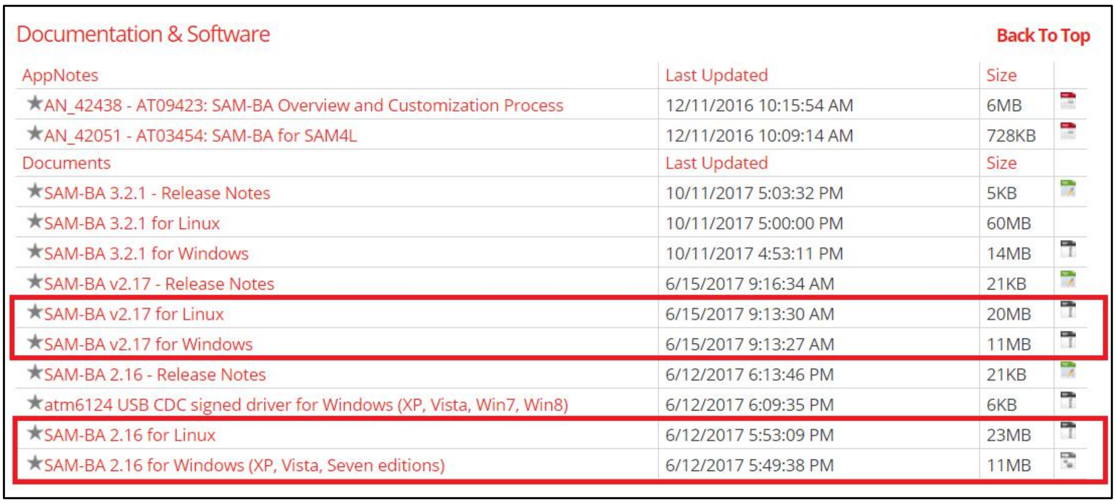
6. Installing SAM-BA and the EDK board files

To write your program on the board, you must use the Microchip SAM Boot Assistance (SAM-BA).

The complete installation packages are available at the bottom of the following page, under "Documentation & Software":

http://www.microchip.com/DevelopmentTools/ProductDetails.aspx?PartNO=Atmel%20SAM-BA%20In-system%20Programmer

We recommend you download version 2.17 preferably, but any one of the highlighted options below will work, according to your operating system.



Documentation & Software

Back To Top

AppNotes	Last Updated	Size
★AN_42438 - AT09423: SAM-BA Overview and Customization Process	12/11/2016 10:15:54 AM	6MB
★AN_42051 - AT03454: SAM-BA for SAM4L	12/11/2016 10:09:14 AM	728KB
Documents	Last Updated	Size
★SAM-BA 3.2.1 - Release Notes	10/11/2017 5:03:32 PM	5KB
★SAM-BA 3.2.1 for Linux	10/11/2017 5:00:00 PM	60MB
★SAM-BA 3.2.1 for Windows	10/11/2017 4:53:11 PM	14MB
★SAM-BA v2.17 - Release Notes	6/15/2017 9:16:34 AM	21KB
★SAM-BA v2.17 for Linux	6/15/2017 9:13:30 AM	20MB
★SAM-BA v2.17 for Windows	6/15/2017 9:13:27 AM	11MB
★SAM-BA 2.16 - Release Notes	6/12/2017 6:13:46 PM	21KB
★atm6124 USB CDC signed driver for Windows (XP, Vista, Win7, Win8)	6/12/2017 6:09:35 PM	6KB
★SAM-BA 2.16 for Linux	6/12/2017 5:53:09 PM	23MB
★SAM-BA 2.16 for Windows (XP, Vista, Seven editions)	6/12/2017 5:49:38 PM	11MB

Except for the 2.16 Windows version, the other packages come in a .zip file, requiring you to unzip the files before installing.

Complete the installation, and before opening the installed program, download the "**board-files.zip**" file from **our GitHub page.**

This zip will contain one single folder called "**tcl_lib**", that you must paste at the **root** of the install location of your SAM-BA application, and overwrite any files. This will add a couple of customization files for your EDK board, and update a file of a list of devices.

The default installation folder is C:\Program Files (x86)\Atmel\sam-ba_2.1**X**\

*(**X** being **6** for SAM-BA 2.16, and **7** for SAM-BA 2.17)*

7. Connecting your EDK

To connect your EDK to your computer via Micro USB, you can use any Micro USB - USB cable. The board should start automatically in *Bootloader Mode*.

> If there is no valid application present, the board will <u>always</u> start in *Bootloader Mode.* To reboot the board in *Bootloader Mode* at any time (such as after uploading a valid application):
>
> - Remove any power supply from the board
> - While **holding the user button**, power on the board.

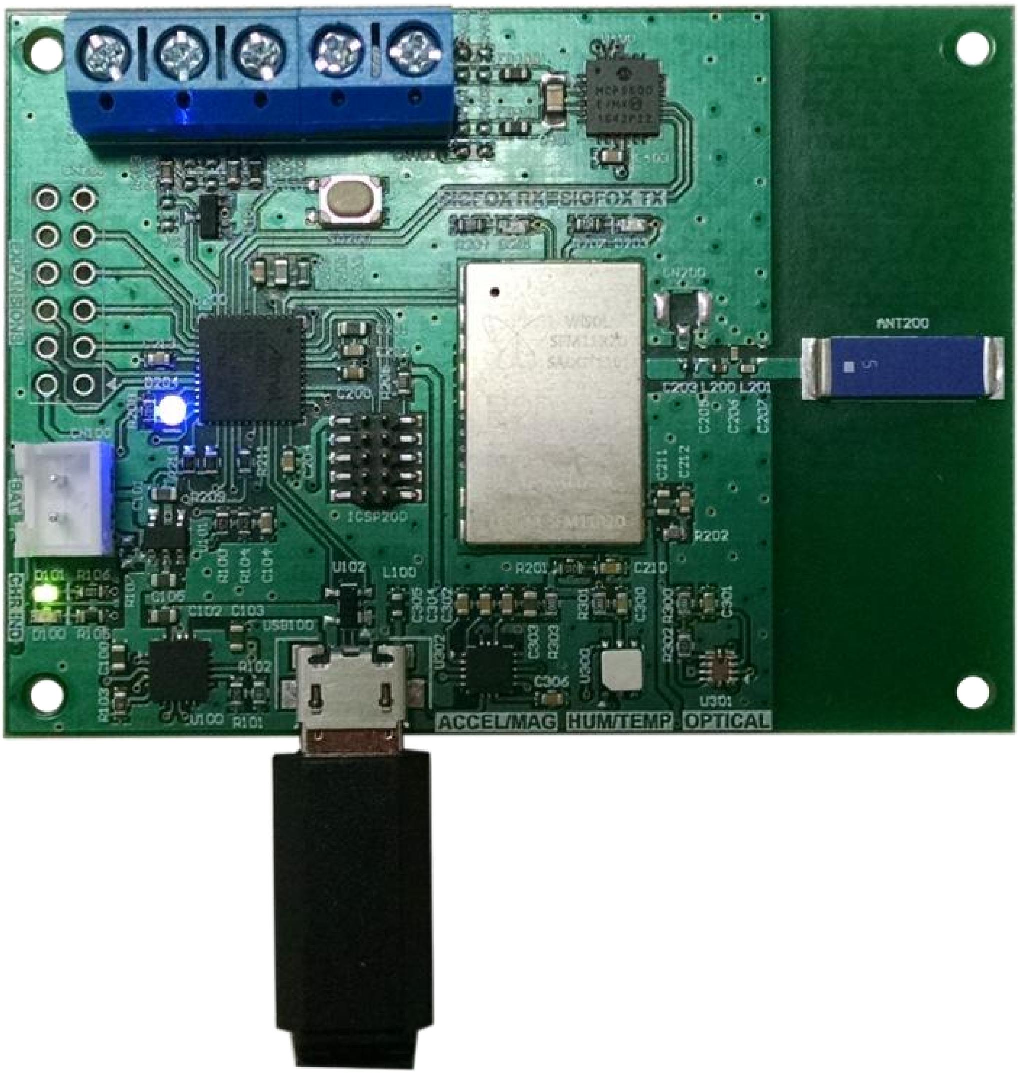
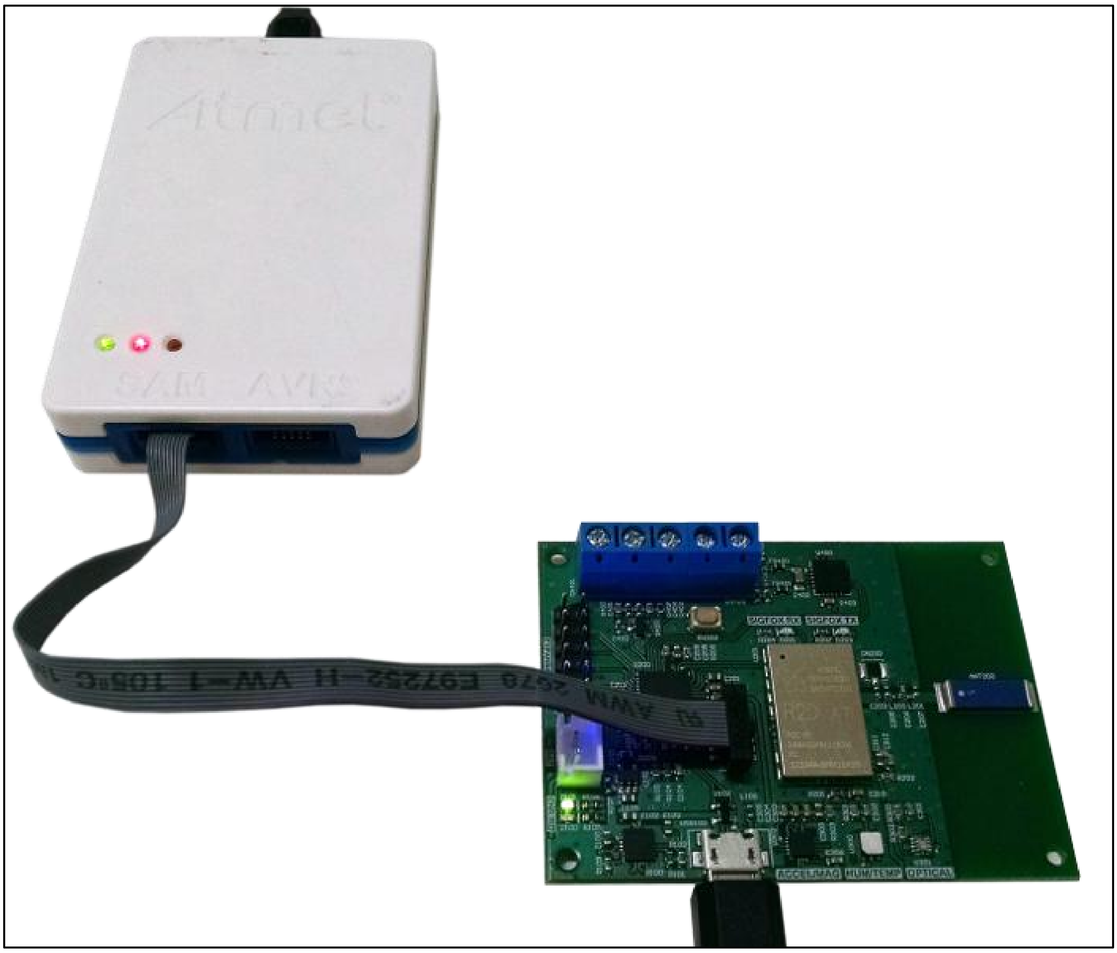


Figure: EDK connected via USB, in Bootloader Mode *(solid blue light)*

While the board is in *Bootloader Mode* via USB, you can upload the binary file of your application to it using the SAM-BA application (see next section for details).

Another way to connect your board is via a Programmer, such as the ATMEL-ICE (pictured below). This allows you to use Atmel Studio 7.0 and upload applications more freely to your board.



EDK connected via an ATMEL-ICE programmer, using the SAM interface.

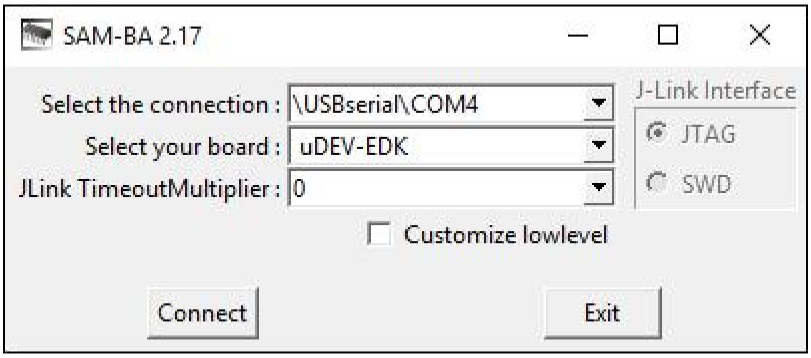
Please double-check the ribbon cable orientation before connecting your board. If using the standard ribbon cable that comes bundled with the ATMEL-ICE, the cable's red edge should be facing the Micro USB - meaning that PIN #1 should be to the side of the user button.

8. Using SAM-BA

Make sure your uDEV EDK board is connected, and has a **solid blue light**, indicating it is in *Bootloader Mode*.

If the board is on but the blue light isn't, reset the board whilst keeping the user button pressed, and it will boot in *Bootloader Mode*.

After opening the **sam-ba.exe** application, the proper connection should be automatically selected.



SAM-BA 2.17

Select the connection : \USBserial\COM4

Select your board : uDEV-EDK

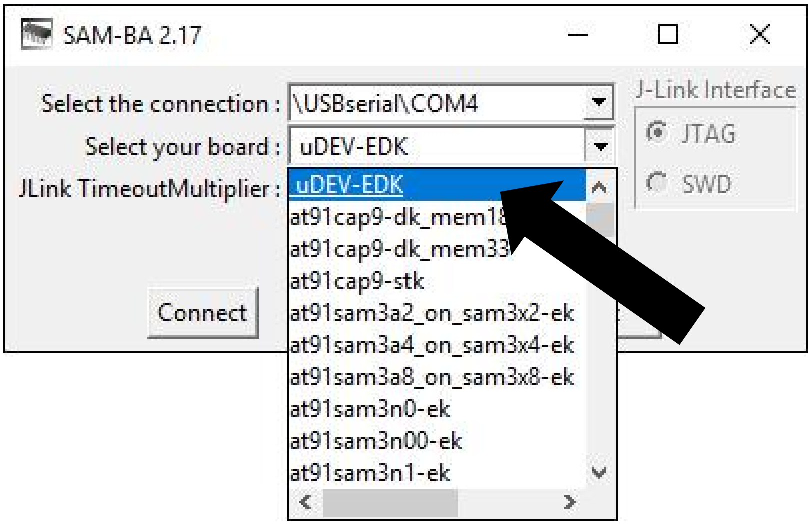
JLink TimeoutMultiplier : 0

J-Link Interface: ● JTAG ○ SWD

□ Customize lowlevel

If you copied the board files correctly, the SAM-BA application should start with the uDEV-EDK board automatically selected, under the "Select your board" option.

If that doesn't happen, select **uDEV-EDK** as your board and click "Connect".



SAM-BA 2.17

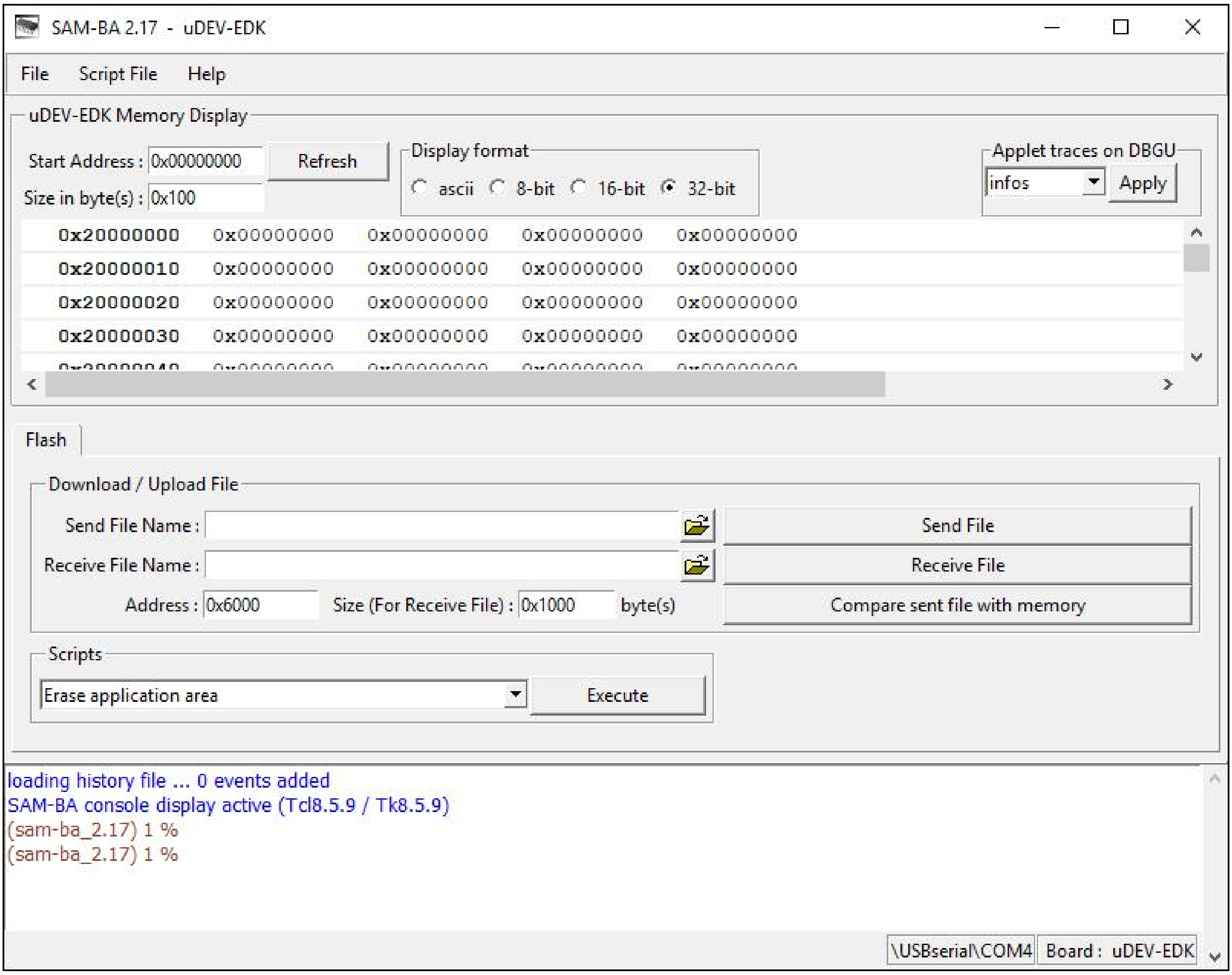
Select the connection : \USBserial\COM4

Select your board : uDEV-EDK

JLink TimeoutMultiplier :

J-Link Interface: ● JTAG ○ SWD

After connecting, the application should display the following window:



Start Address : 0x00000000

Size in byte(s) : 0x100

○ ascii ○ 8-bit ○ 16-bit ● 32-bit

infos

```
0x20000000   0x00000000   0x00000000   0x00000000   0x00000000
0x20000010   0x00000000   0x00000000   0x00000000   0x00000000
0x20000020   0x00000000   0x00000000   0x00000000   0x00000000
0x20000030   0x00000000   0x00000000   0x00000000   0x00000000
```

Send File Name :

Receive File Name :

Address : 0x6000 Size (For Receive File) : 0x1000 byte(s)

Erase application area

```
loading history file ... 0 events added
SAM-BA console display active (Tcl8.5.9 / Tk8.5.9)
(sam-ba_2.17) 1 %
(sam-ba_2.17) 1 %
```

\USBserial\COM4 Board : uDEV-EDK

In this window, you can verify sections of the memory, and send the binary code of your program via the Send File command.

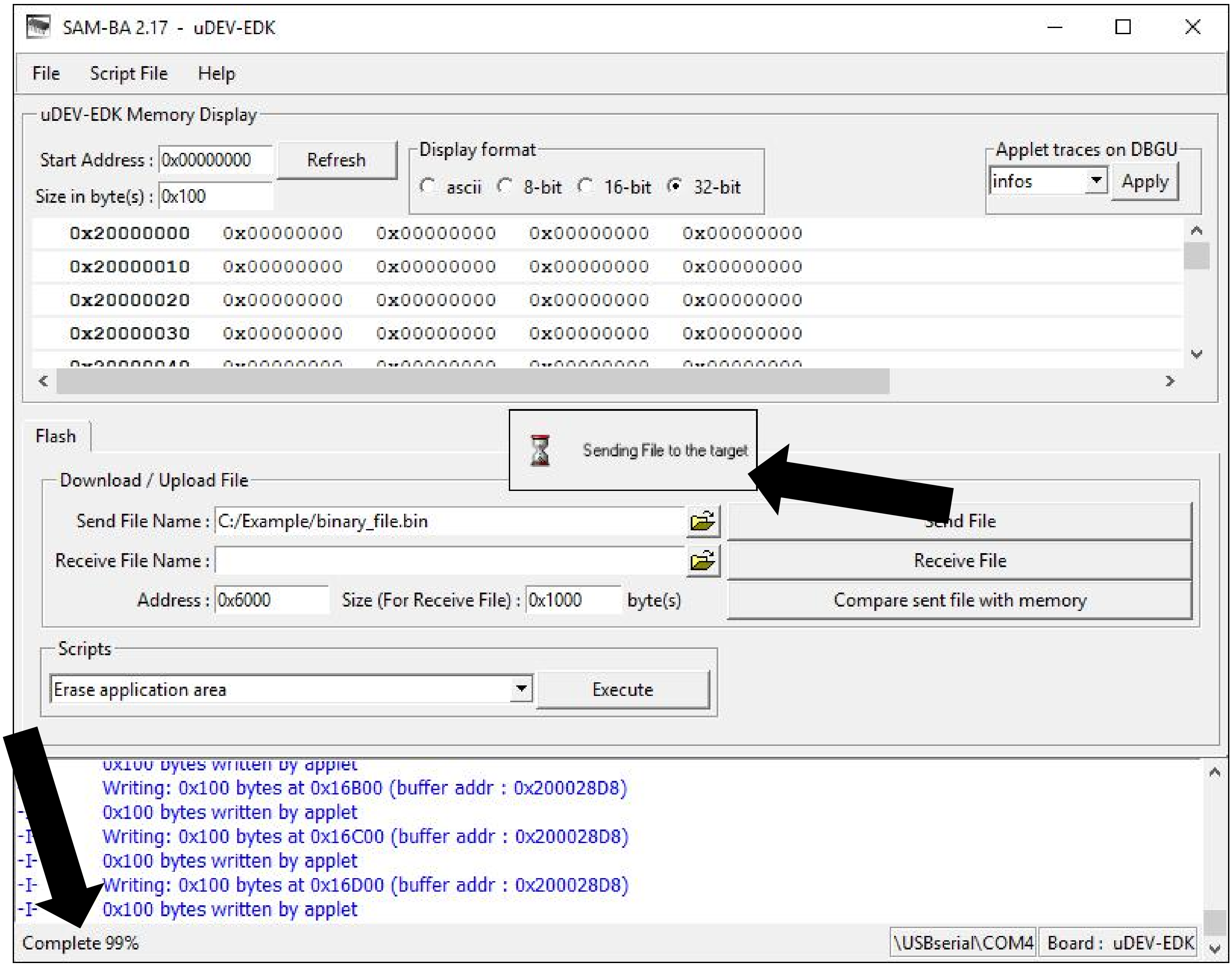
ATTENTION: Your program or application **MUST** start at flash memory address 0x6000 or higher! Any addresses lower than that will fail. Memory addresses 0x0000 - 0x6000 are reserved for the *Bootloader*.

Per default, when choosing the uDEV-EDK board, the address will be automatically set to 0x6000. Make sure this value is not changed before uploading your program to the board.

See the next section for an example of how to build your project at the correct address using Atmel Studio 7.0.

After selecting the .bin file and sending it with “Address” **set to 0x6000**, the SAM-BA application will load the program into the board’s flash memory.

A window will pop-up, indicating that the upload is in progress. Pay attention to the **lower status bar**, as it will display the progress of the upload.



Start Address : 0x00000000

Size in byte(s) : 0x100

Display format: ○ ascii ○ 8-bit ○ 16-bit ● 32-bit

Applet traces on DBGU: infos Apply

```
0x20000000   0x00000000   0x00000000   0x00000000   0x00000000
0x20000010   0x00000000   0x00000000   0x00000000   0x00000000
0x20000020   0x00000000   0x00000000   0x00000000   0x00000000
0x20000030   0x00000000   0x00000000   0x00000000   0x00000000
0x20000040   0x00000000   0x00000000   0x00000000   0x00000000
```

Send File Name : C:/Example/binary_file.bin

Receive File Name :

Address : 0x6000   Size (For Receive File) : 0x1000 byte(s)

Scripts: Erase application area

```
    0x100 bytes written by applet
    Writing: 0x100 bytes at 0x16B00 (buffer addr : 0x200028D8)
-   0x100 bytes written by applet
-I- Writing: 0x100 bytes at 0x16C00 (buffer addr : 0x200028D8)
-I- 0x100 bytes written by applet
-I- Writing: 0x100 bytes at 0x16D00 (buffer addr : 0x200028D8)
-I- 0x100 bytes written by applet
```

Complete 99%   \USBserial\COM4   Board : uDEV-EDK

If it reaches 100% and doesn’t display any errors, your program should have been loaded into the flash memory and will be ready to use. Simply reboot the uDEV EDK board, and it will automatically begin in *Application Mode*.

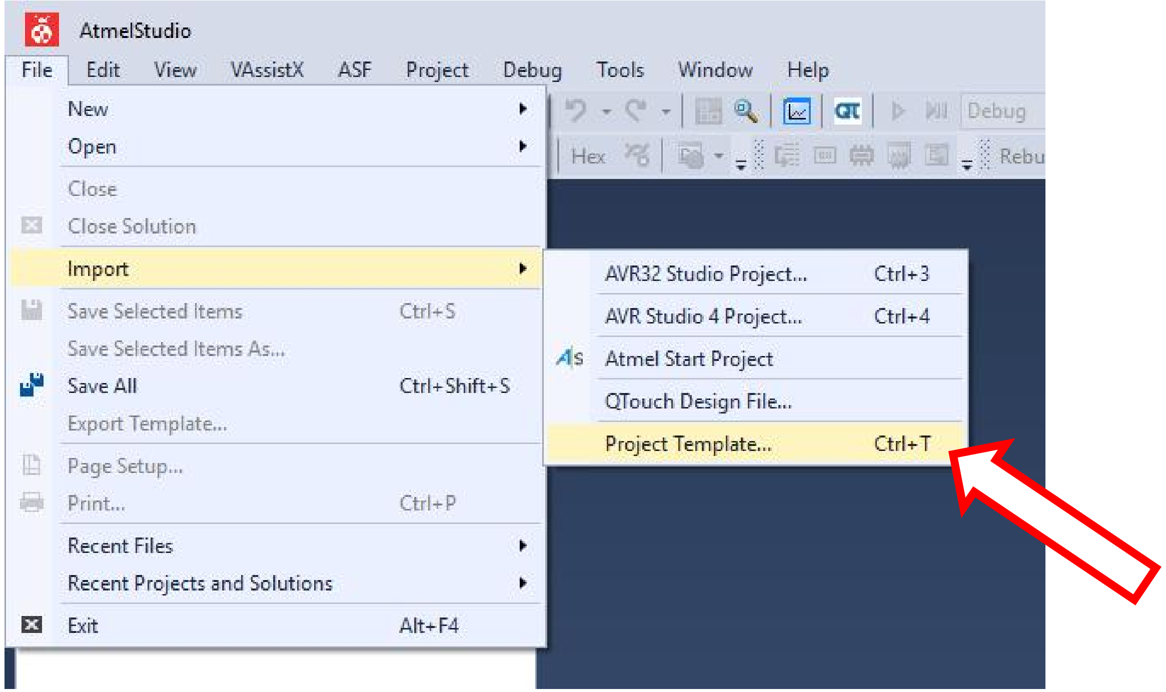
> ! You can verify if everything is correctly uploaded by using the “**Compare sent file with memory**” function, which compares the file in the “**Send File Name**” path with the *Flash Memory*, starting at the address indicated in the “**Address**” field.

9. Testing

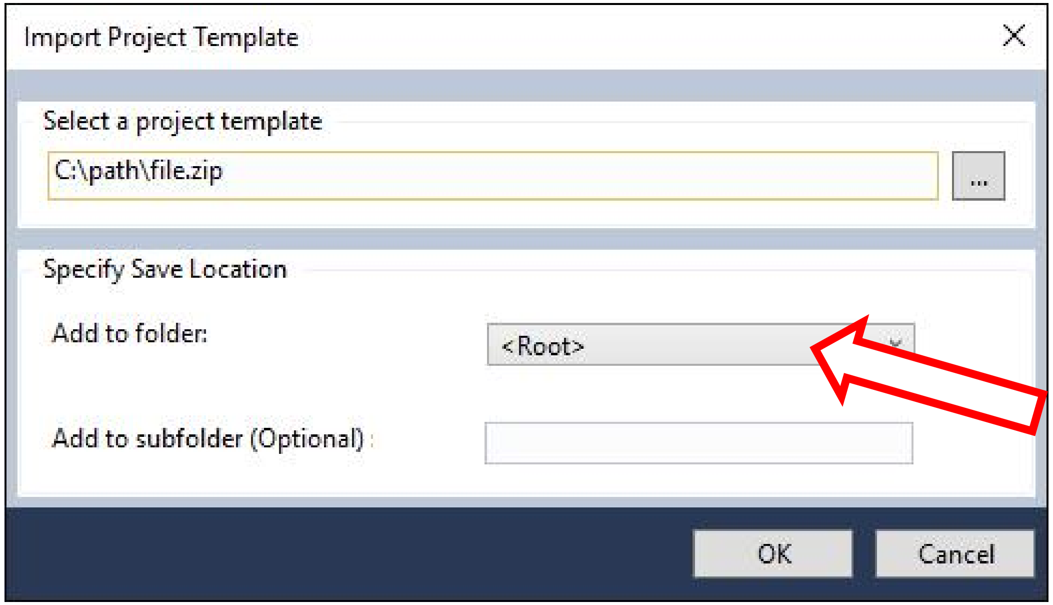
9.1. LED Project Template - Installing

We have prepared a project template for Atmel Studio 7.0 so that you can install them and easily start your own Projects, specifically made for your uDEV EDK. The LED component project template is available at our page at GitHub.

To import the template into your Atmel Studio application:
1. Go to **Import -> Project Template...**



2. Then select the correct .ZIP file that has the template you want to install, and set the template to whatever folder you want. You may also add a Subfolder if desired.



Import Project Template

Select a project template

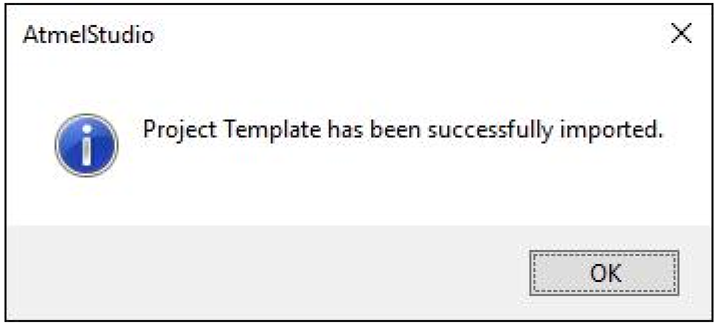
C:\path\file.zip

Specify Save Location

Add to folder: <Root>

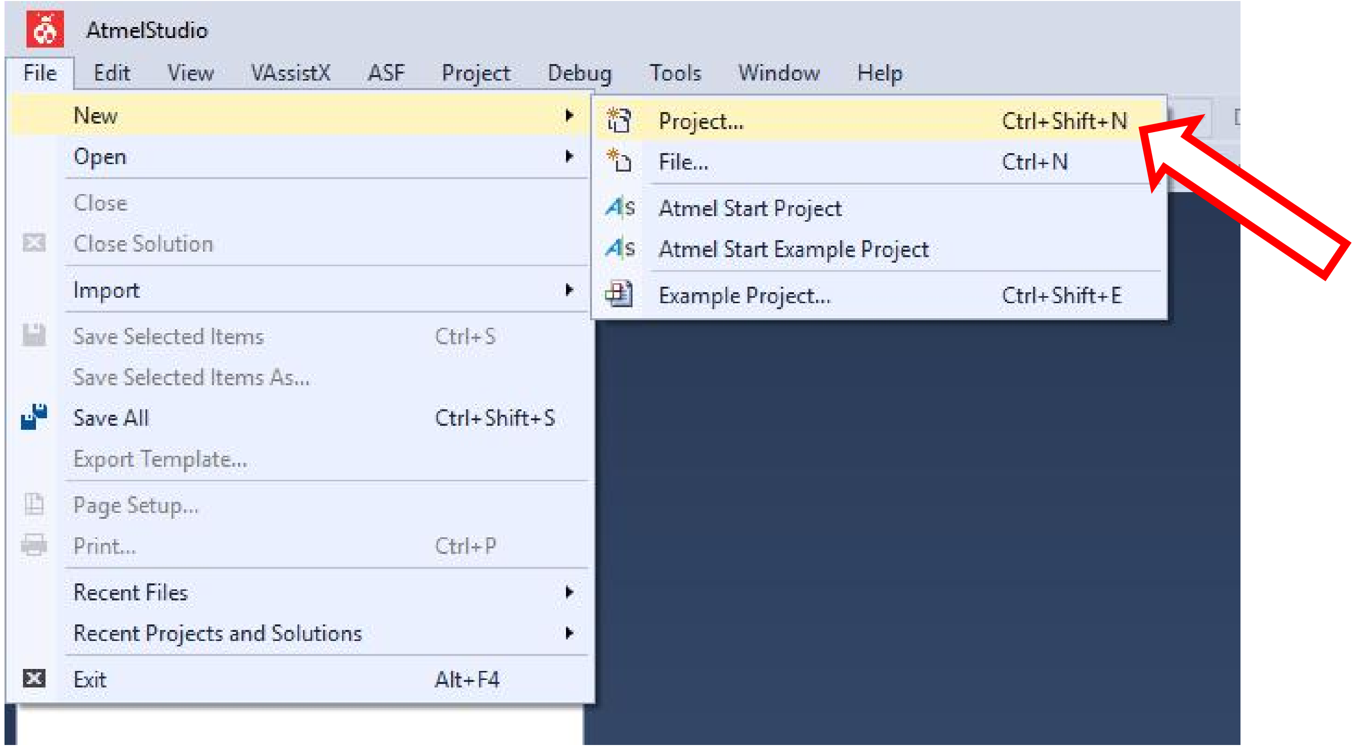
Add to subfolder (Optional):

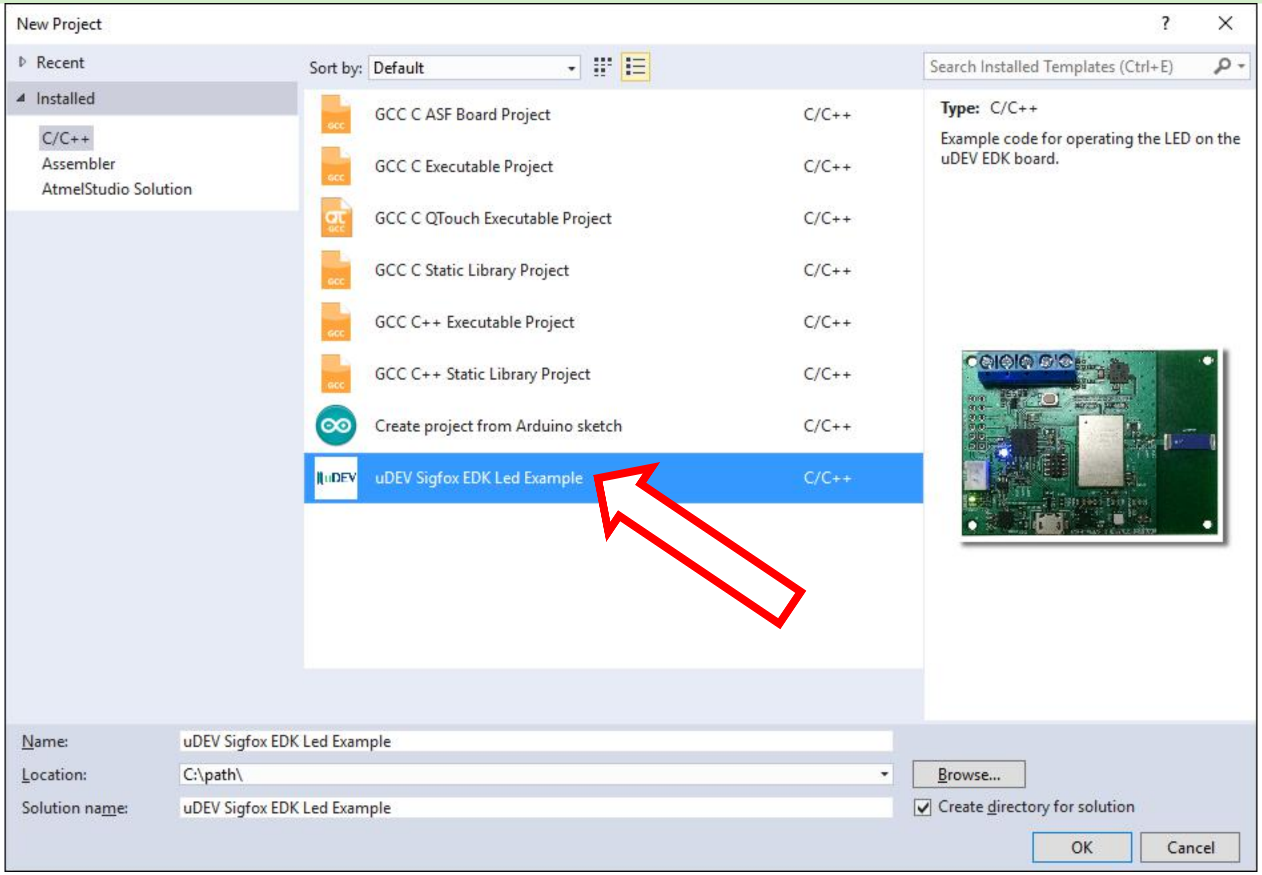
Atmel should display the following window:



Project Template has been successfully imported.

3. After that, the Template should be available in **New -> Projects…**





New Project

GCC C ASF Board Project	C/C++
GCC C Executable Project	C/C++
GCC C QTouch Executable Project	C/C++
GCC C Static Library Project	C/C++
GCC C++ Executable Project	C/C++
GCC C++ Static Library Project	C/C++
Create project from Arduino sketch	C/C++
uDEV Sigfox EDK Led Example	C/C++

Type: C/C++

Example code for operating the LED on the uDEV EDK board.

Name: uDEV Sigfox EDK Led Example

Location: C:\path\

Solution name: uDEV Sigfox EDK Led Example

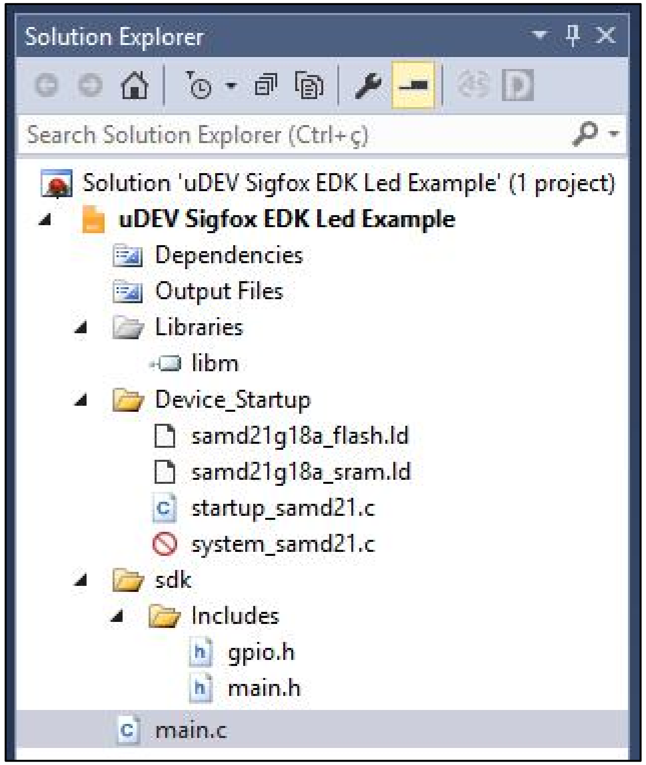
☑ Create directory for solution

If the template does not appear, try restarting the Atmel Studio application completely, as it may be trying to refresh the template list.

After creating your project using the uDEV template, you have to manually add the Library with the board configuration to the project parameters.

Luckily, this is a brief process.

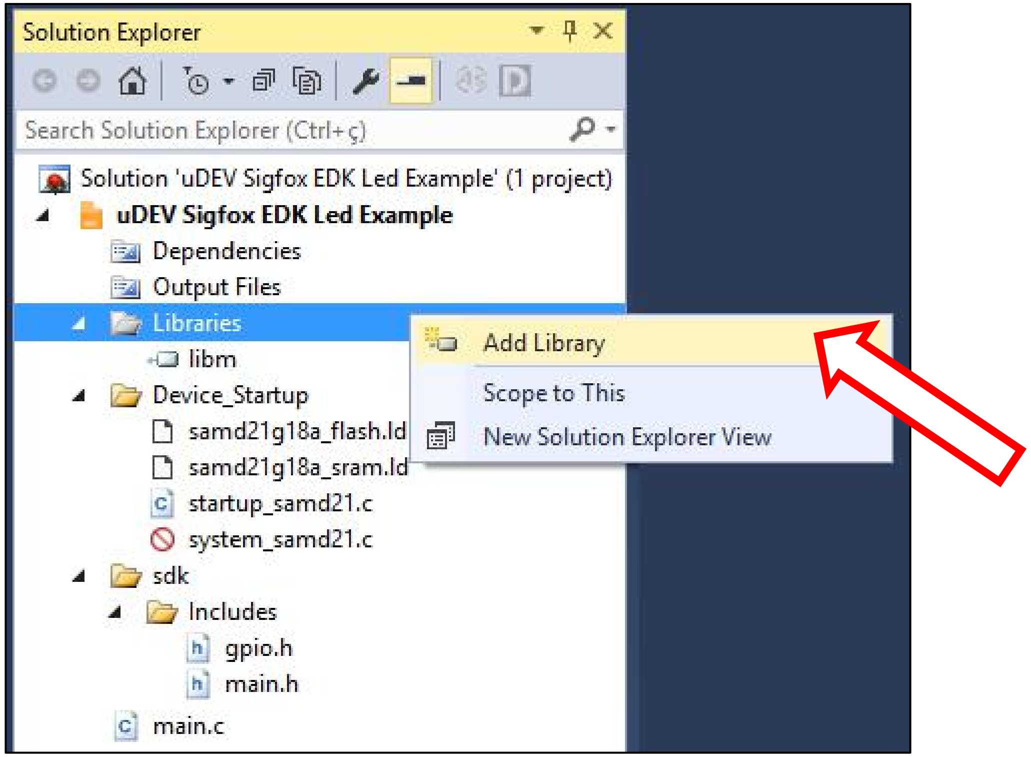
Your solution explorer should be exactly as below:



Search Solution Explorer (Ctrl+ç)

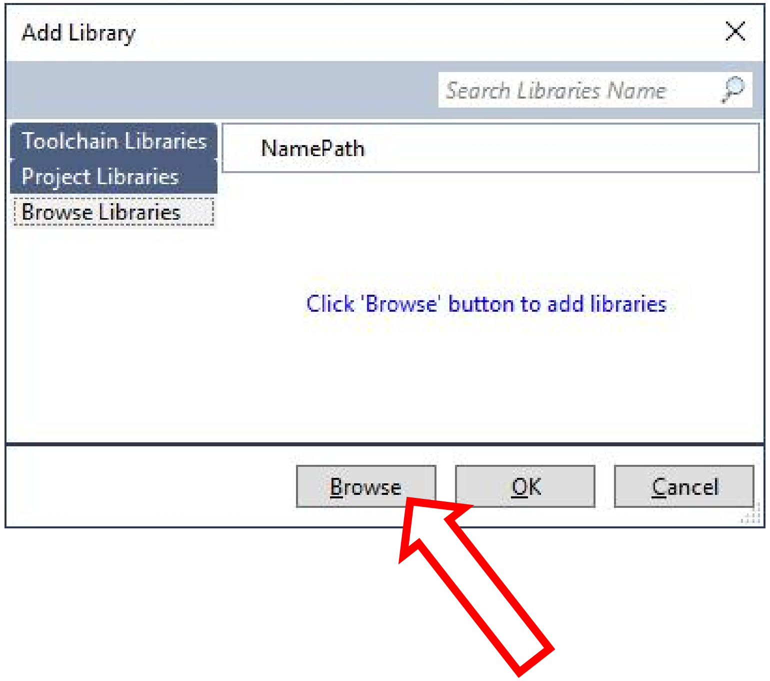
```
Solution 'uDEV Sigfox EDK Led Example' (1 project)
◢ uDEV Sigfox EDK Led Example
    Dependencies
    Output Files
  ◢ Libraries
      libm
  ◢ Device_Startup
      samd21g18a_flash.ld
      samd21g18a_sram.ld
      startup_samd21.c
      system_samd21.c
  ◢ sdk
    ◢ Includes
        gpio.h
        main.h
    main.c
```

4. Simply **right-click** the **"Libraries" folder**, and select "**Add Library**"

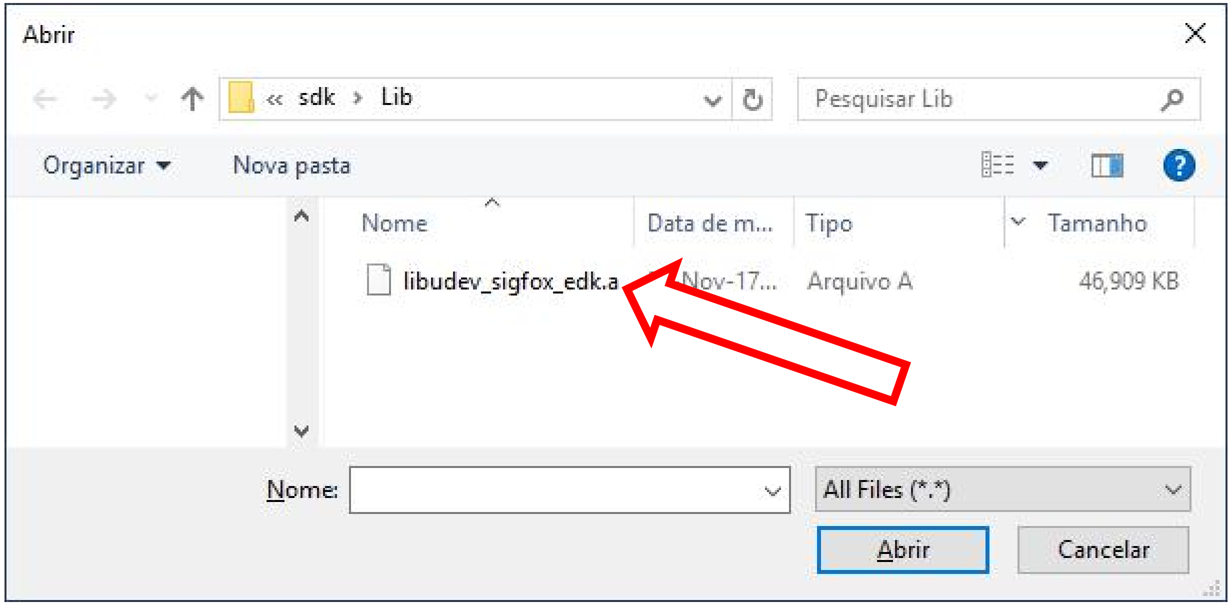


This will open a new window, so you can select which library to add to the project.

Choose “**browse library**” and navigate to wherever you saved the **udev_sigfox_edk.a** static library, and add that.

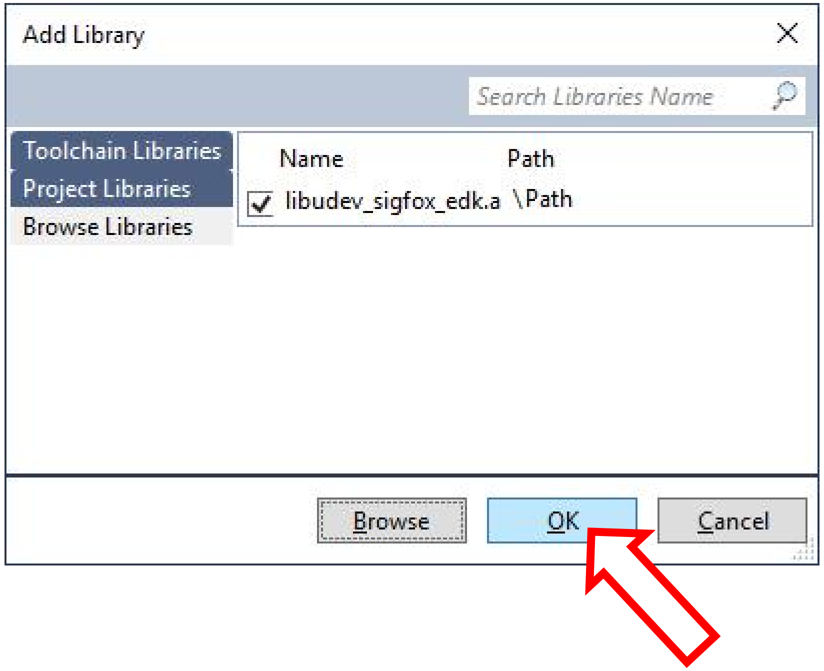


Click 'Browse' button to add libraries

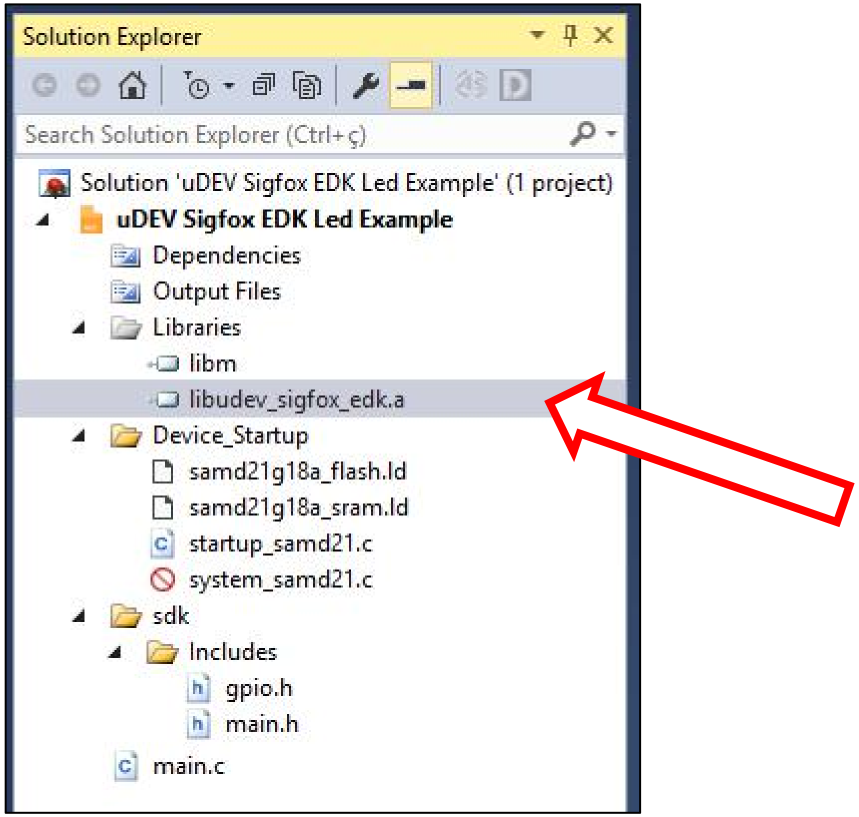


libudev_sigfox_edk.a Nov-17... Arquivo A 46,909 KB

Select and add the library indicated, and click **ok**.



5. Check if everything was correctly added by verifying that the library appears in the Solution Explorer:



The LED Example is now ready to be built and executed on your uDEV Sigfox EDK!

9.2 Generic Project Template

Alternatively, a generic project template can be installed, and headers can be added as needed to the project.

Simply repeat the steps described above, but with the **Generic Project Template** .ZIP archive that is available in our GitHub.

This template will not contain **gpio.h** or other headers (apart from **main.h**) in the Includes folder, and you will need to add the header files that you'll use in your project, at your discretion.

10. Registering your EDK in the Sigfox Backend

To see any data a device is sending in the **Sigfox Backend** - the name given to the Sigfox web interface, you must have that device registered beforehand.

To register a device, two pieces of information are necessary: the **Sigfox ID**, and the **Sigfox PAC**.

The **ID** is a static hex code used to **uniquely identify devices**, while the **PAC** is a **one-time only** code used to register the device.

Once you register a device using the **ID-PAC** code pair, the **PAC** is immediately changed. Should you want to register it again in the future, the new **PAC** code must be acquired.

The EDK comes with a sticker on the bottom side of the board, with its ID and initial PAC.
Be wary that after registration this PAC will no longer be valid.
It is possible to see the current device **PAC** in the **Sigfox Backend**, after registration.

To register your device:

- Access http://backend.sigfox.com/activate
- Find and click on the uDEV logo.
- Choose your country of operation under the tab “Pick your Country”
- Provide the device’s ID (hex) and PAC under the tab “Device Information”
- Provide your personal information on the “Account details” tab, this will be used to login to the SIGFOX Backend and see any messages associated with the device being registered.