

Using Atmel START with the SAM D21 MCU ADC and I²C

🎯 Objective

This project shows you how to:

- Use **Atmel START** to add and configure an Analog-to-Digital Converter (ADC) and **I²C** driver for your project.
- Export your project for the **Atmel Studio 7 IDE**.
- Use **Advanced Software Framework 4 (ASF4)** example code to:
 - Measure the analog output of a light sensor (using a SAM D21 ADC)
 - Read the value from a temperature sensor over an I²C bus (using a SAM D21 SERCOM configured for I²C)
- Program the SAM D21 Xplained Pro Evaluation Kit to verify your code is working.

☑ Materials

Hardware Tools

Tool	🔗 About	🛒 Purchase
SAM D21 Xplained Pro Evaluation Kit	🔗 (/boards:sam-d21-xpro)	🛒 (https://www.microchipdirect.com/product/search/all/atsamd21-xpro)
I/O1 Xplained Pro Extension Kit	🔗 (http://www.microchip.com/Developmenttools/ProductDetails/AT101-XPRO)	🛒 (https://www.microchipdirect.com/product/search/all/AT101-XPRO)

► Show Hardware Setup

Software Tools

Tool	🔗 About	Installers			
		Windows	Linux	Mac OSX	Installation Instructions
Atmel® Studio Integrated Development Environment	🔗 (/atstudio:start)	📄 (http://studio.download.atmel.com/7.0.1931/as-installer-7.0.1931-full.exe)	📄	📄	🔗 (/install:atstudio)
Atmel® START (ASF4) Integrated Software Framework	🔗 (/atstart:start)	Web Based (http://start.atmel.com/)			

Optional Lab Manual

A hardcopy of this tutorial is available here:

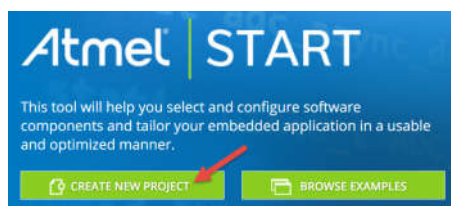
Using Atmel START with the SAM D21 MCU > (<https://microchip.box.com/s/6fjv4kskvu9yk04tfgoplrkg53zkis63>)

Procedure:

- Step 1: Create a new Atmel START project
- Step 2: Add and configure an ADC driver
- Step 3: Add and configure the temperature sensor middleware and its associated I²C driver
- Step 4: Add a Delay driver (sensor measurements every second)
- Step 5: Save and export the application for the Studio 7 IDE
- Step 6: Import the Atmel START project into the Studio 7 IDE
- Step 7: Modify ADC example code to read light sensor data
- Step 8: Program the board
- Step 9: Verify light sensor measurements
- Step 10: Modify I²C example code to read temp sensor data and verify measurements

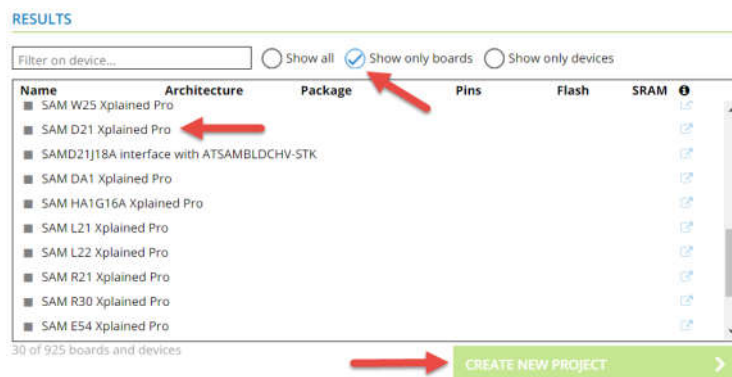
Step 1: Create a new Atmel START project

- 1 Open a web browser, go to <http://start.atmel.com> (<http://start.atmel.com>), and select **CREATE NEW PROJECT**.



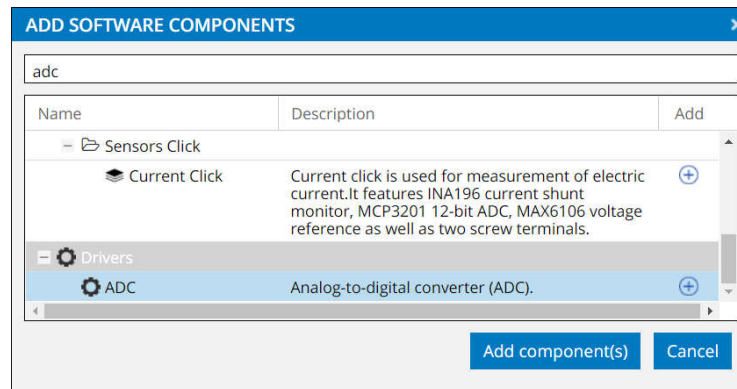
(<https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/create.png>)

- 2 Select a specific board for the project.
 - Click on 'Show only boards' from the "RESULTS" section then select the 'SAM D21 Xplained Pro' board
 - Click **CREATE NEW PROJECT** to complete the project creation.



(<https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/boards.png>)

The project is created in Atmel START and you now have access to the *DASHBOARD* view. Scroll to the bottom of this window to verify the ATSAMD21J18A is the selected device and explore its capabilities.



(https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/add_adc.png)

3 Configure the ADC driver:

- Click on the 'ADC_0' component block to start its configuration.



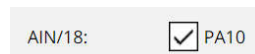
(https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/adc_config.png)

- Configure ADC Component Settings:
 - Driver: HAL:Driver:ADC_Async



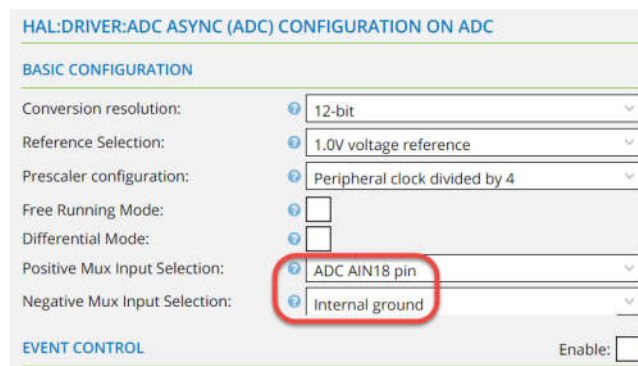
(https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/adc_component.png)

- Configure ADC Signals:
 - AIN/18 - PA10: Enabled



(https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/adc_io.png)

- Configure ADC Basic Configuration:
 - Positive Mux Input Selection: ADC AIN18 pin
 - Negative Mux Input Selection: Internal ground



(https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/adc_basic.png)

The ADC driver is now configured.

Step 3: Add and configure the temperature sensor middleware and its associated I²C driver

The I/O1 Xplained Pro extension board uses a Microchip AT30TSE758 temperature sensor chip with an 8 kb serial EEPROM inside. The sensor includes programmable high and low-temperature alarms, user selectable temperature resolution up to 12 bits and an I²C/SMBus™ compatible serial interface. The SAM D21 reads the temperature using its serial communication interface (SERCOM) peripheral configured for I²C.

- 1 Determine which pins from the SAM D21 Xplained Pro interface with the temperature sensor.
 - Determine which pins on the I/O1 Explained Pro board is used for the temperature sensor.
 - The I/O1 Xplained Pro Extension Kit User Guide (http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42078-IO1-Xplained-Pro_User-Guide.pdf) specifies this:

Table 4-8 Temperature Sensor Connections

Pin on EXT connector	Pin name	AT30TSE758 temperature sensor pin	Comment
11	SDA	1	Data line of serial interface
12	SCL	2	Clock line of serial interface

(https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/temp_pins.png)

- Determine which SAM D21 pins connect to pins 11 and 12 on the EXT2 interface.
 - The SAM D21 Xplained Pro Users Guide (http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42220-SAMD21-Xplained-Pro_User-Guide.pdf) specifies this:

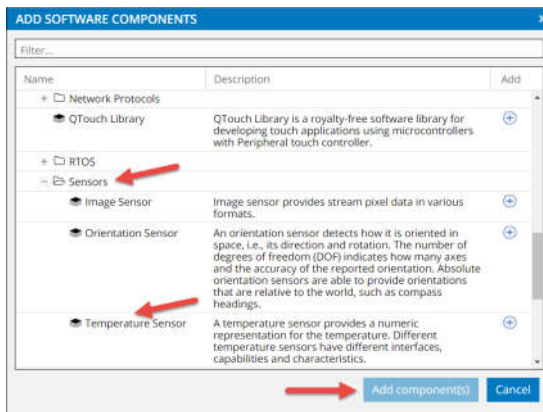
Table 4-2. Extension Header EXT2

Pin on EXT2	SAM D21 pin	Function	Shared functionality
1 [ID]	-	-	Communication line to ID chip on extension board.
2 [GND]	-	-	GND
3 [ADC(+)]	PA10	AIN[18]	
4 [ADC(-)]	PA11	AIN[19]	
5 [GPIO1]	PA20	GPIO	
6 [GPIO2]	PA21	GPIO	
7 [PWM(+)]	PB12	TC4/WO[0]	
8 [PWM(-)]	PB13	TC4/WO[1]	
9 [IRQ/GPIO]	PB14	EXTINT[14]	
10 [SPI_SS_B/GPIO]	PB15	GPIO	
11 [TWI_SDA]	PA08	SERCOM2 PAD[0] I ² C SDA	EXT1, EXT3, and EDBG
12 [TWI_SCL]	PA09	SERCOM2 PAD[1] I ² C SCL	EXT1, EXT3, and EDBG

(https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/ext2_i2c.png)

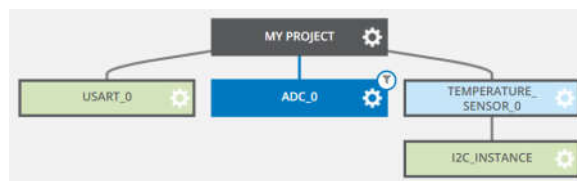
Notice SAM D21 pin numbers PA08 and PA09 (function SERCOM2 I²C SDA and SCL) are connected to the temperature sensor.

- 2 Add the Temperature Sensor Middleware.
 - Select 'DASHBOARD' and click 'Add software component'.
 - Expand **Middleware > Sensors**, find the Temperature Sensor middleware and click on it.
 - Click **Add component(s)**.



(https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/add_temp_sensor.png)

Notice that adding the temperature sensor middleware automatically adds an I²C driver which is required to interface to the sensor.



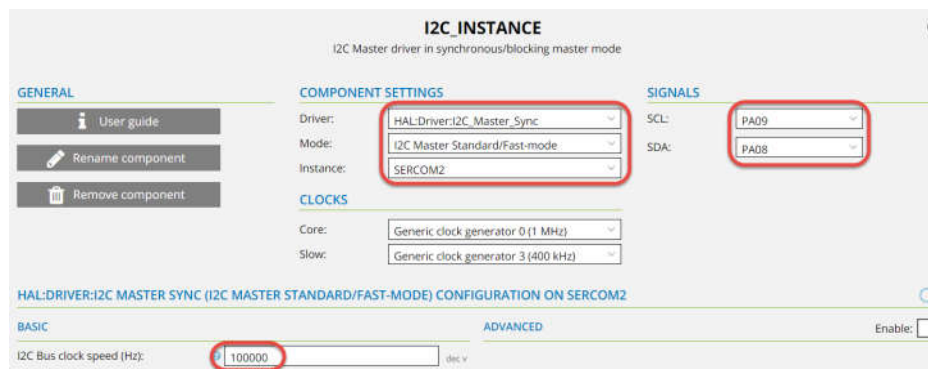
(https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/temp_i2c.png)

The Temperature Sensor middleware is now added to the application.

If you click on the Temperature Sensor component block, you will see the dependencies of that middleware.

3 Configure the I²C driver.

- Click on the 'I2C_INSTANCE' component block to start its configuration.
- Configure the I²C Component Settings:
 - Driver: HAL:Driver:I2C_Master_Sync
 - Mode: I²C Master Standard/Fast-mode
 - Instance: SERCOM2
- Configure I²C Signals:
 - SCL: PA09
 - SDA: PA08
- Configure I²C Basic Configuration:
 - I²C Bus Clock speed (Hz): 100 kHz



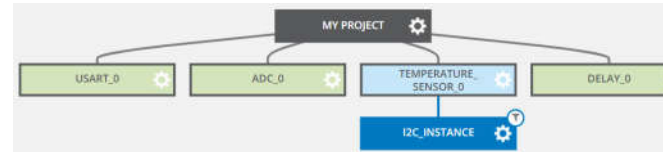
(https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/i2c_driver_config.png)

The I²C driver is now configured.

Step 4: Add a Delay driver (sensor measurements every second)

This driver provides functions which allow you to add delays in μ s or ms using the **Cortex[®]-M0+** SysTick timer.

- Add the Delay driver:
 - Select **DASHBOARD** and click 'Add software component'.
 - Expand 'Drivers', find the Delay driver and click on it.
 - Click **Add component(s)**.



(<https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/delay.png>)

The Delay driver is now added to the application.

Step 5: Save and export the project for the Studio 7 IDE

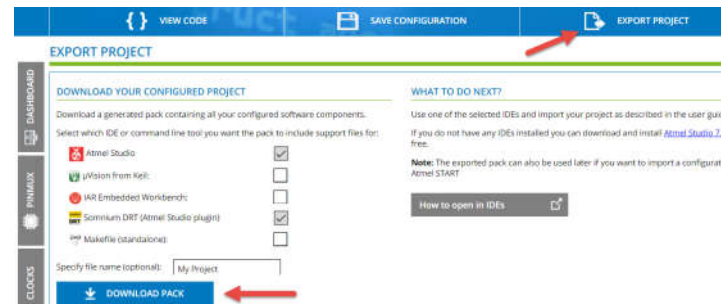
You have now created and configured your Atmel START based project. It's a good idea to save this configuration so you can make changes to it sometime in the future. Atmel START allows you to restore any project using its configuration file (*.atstart file format).

- 1 Save your Atmel START project.
 - Select **SAVE CONFIGURATION**.
 - Click on **DOWNLOAD CONFIGURATION**.
 - Provide a filename and location and then click **Save**.



(https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/save_config.png)

- 2 Export the project for the Studio 7 IDE.
 - Select **EXPORT PROJECT** then **DOWNLOAD PACK**.
 - Provide a filename and location, then click **Save**.



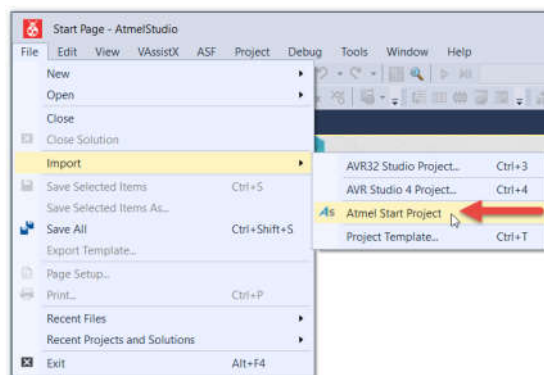
(<https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/export.png>)

Your Atmel START project has been exported for the Studio 7 IDE in a *.atzip file format (standard zip format automatically recognized by Studio 7).

Step 6: Import the Atmel START project into the Studio 7 IDE

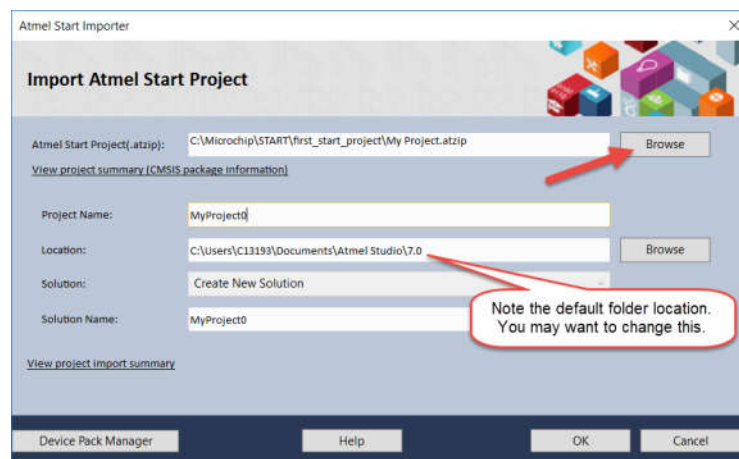
1 Open Studio 7 and select:

- **File > Import > Atmel Start Project**



(<https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/import.png>)

2 Browse to the project you exported from START (*.atzip) and click **OK**. Your project is now created.

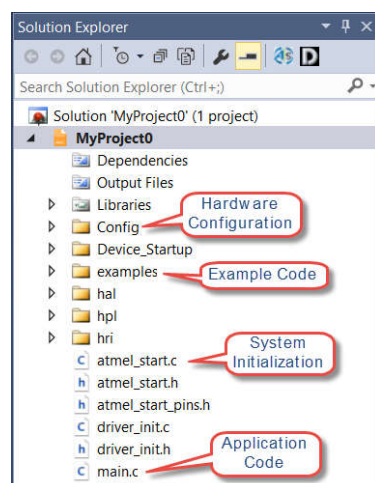


(<https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/import1.png>)



Note a default project name and location have been selected for you. Feel free to change these.

Atmel START projects come with some useful examples to help you get started on the different peripherals you're looking to use. You can find these in the `driver_examples.c` file in the examples folder.



(<https://microchip.wdfiles.com/local-->

files/atstart:sam-d21-adc-i2c/solution_explorer.png)

Open the `main.c` file. You will see the only function called in the current project is the `atmel_start_init()` function:

```
#include <atmel_start.h>

int main(void)
{
    /* Initializes MCU, drivers and middleware */
    atmel_start_init();

    /* Replace with your application code */
    while (1) {
    }
}

(https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/main1.png)
```

The `atmel_start_init()` function calls the `system_init()` and `temperature_sensors_init()` functions.

The `system_init()` function calls the following functions:

- `init_mcu()` : initialize oscillators, clocks, flash wait states...
- `ADC_0_init()`
- `I2C_INSTANCE_init()`
- `delay_driver_init()`



Want to see what these functions do? Right-click on the function name and select 'Go to Implementation' to see the function definition (C file) and description (header file). For more details, see the ASF4 API Reference Manual (<http://ww1.microchip.com/downloads/en/DeviceDoc/50002633A.pdf>).

Open the `driver_init.c` file to see these and other initialization functions generated in response to the selections you made in Atmel START. You may also want to check out the configuration header files (found in the `Config` folder).

Step 7: Modify ADC example code to read light sensor data

As mentioned above, the examples in the `driver_examples.c` file will be used to help you get started.

- 1 In step 2, you configured the ADC driver as 'HAL:Driver:ADC_Async'. This means that the ADC takes a measurement and then generates an interrupt. The ADC interrupt then calls a callback function that will have to be implemented.
 - Open `driver_examples.c` (in the `examples` folder) and copy the following functions:
 - `convert_cb_ADC_0()`
 - `ADC_0_example()`
 - Paste them above the `main()` function in the `main.c` file.
 - Rename the `ADC_0_example()` function as `ADC_light_init()`.
 - Rename the `convert_cb_ADC_0()` function as `convert_cb_ADC()`.
 - Update `convert_cb_ADC()` callback call in `ADC_light_init()`.

Your code should look like this:

```
static void convert_cb_ADC(const struct adc_async_descriptor *const descr, const uint8_t channel)
{
}

void ADC_light_init(void)
{
    adc_async_register_callback(&ADC_0, 0, ADC_ASYNC_CONVERT_CB, convert_cb_ADC);
    adc_async_enable_channel(&ADC_0, 0);
    adc_async_start_conversion(&ADC_0);
}

(https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/code4.png)
```

- 2 Initialize the ADC by calling this function after `atmel_start_init()` :

- `ADC_light_init();`

3 Start an ADC conversion every second.


First, use the Delay driver you added to implement a one second delay.

- Add this delay function inside the `while(1)` loop in `main()`.
 - `delay_ms(1000);`

Now you have to call a function to start the conversion. The `ADC_light_init()` function uses this `start conversion` function, but it doesn't need to be in there.

- Cut the following function from `ADC_light_init()` and paste it after the `delay(1000)` function:
 - `adc_async_start_conversion(&ADC_0)`

```
void ADC_light_init(void)
{
    adc_async_register_callback(&ADC_0, 0, ADC_ASYNC_CONVERT_CB, convert_cb_ADC);
    adc_async_enable_channel(&ADC_0, 0);
    adc_async_start_conversion(&ADC_0);
}
```



(<https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/code19.png>)

Your `main.c` should look like this:

```
#include <atmel_start.h>

static void convert_cb_ADC(const struct adc_async_descriptor *const descr, const uint8_t channel)
{
}

void ADC_light_init(void)
{
    adc_async_register_callback(&ADC_0, 0, ADC_ASYNC_CONVERT_CB, convert_cb_ADC);
    adc_async_enable_channel(&ADC_0, 0);
}

int main(void)
{
    /* Initializes MCU, drivers and middleware */
    atmel_start_init();

    ADC_light_init();

    /* Replace with your application code */
    while (1) {
        delay_ms(1000);
        adc_async_start_conversion(&ADC_0);
    }
}
```

(<https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/code20.png>)

4 Add code for ADC callback function.

- Add the following variable declaration after the include statement in `main.c`. This is used to indicate if the conversion is completed or not.
 - `volatile bool conversion_done = false;`


```
#include <atmel_start.h>

volatile bool conversion_done = false;
```

(<https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/code21.png>)

- Update the ADC callback to indicate the conversion is done:
 - `conversion_done = true;`

```
static void convert_cb_ADC(const struct adc_async_descriptor *const descr, const uint8_t channel)
{
    conversion_done = true;
}
```



(<https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/code9.png>)

- Wait for conversion done in the while loop:
 - `while(!conversion_done);`

- `conversion_done = false;`

```
while (1) {
    delay_ms(1000);
    adc_async_start_conversion(&ADC_0);

    while(!conversion_done);
    conversion_done = false;
}
```

(<https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/code10.png>)

5 Declare a two-byte buffer to store the 12-bit ADC value from the light sensor.

- Add the following code to the variable declarations at the top of `main.c`:
 - `uint8_t ADC_buffer[2];`

6 Read the light sensor data.

- Call the following function in the while loop, once data is converted to read the 12-bit value.
 - `adc_async_read_channel(&ADC_0, 0, ADC_buffer, 2);`



Note: The ADC driver functions can be found in the `hal_adc_async.c` file (`hal/src` folder).

Your `main.c` file should look like this:

```
#include <atmel_start.h>

volatile bool conversion_done = false;
uint8_t ADC_buffer[2];

static void convert_cb_ADC(const struct adc_async_descriptor *const descr, const uint8_t channel)
{
    conversion_done = true;
}

void ADC_light_init(void)
{
    adc_async_register_callback(&ADC_0, 0, ADC_ASYNC_CONVERT_CB, convert_cb_ADC);
    adc_async_enable_channel(&ADC_0, 0);
}

int main(void)
{
    /* Initializes MCU, drivers and middleware */
    atmel_start_init();

    ADC_light_init();

    /* Replace with your application code */
    while (1) {
        delay_ms(1000);
        adc_async_start_conversion(&ADC_0);

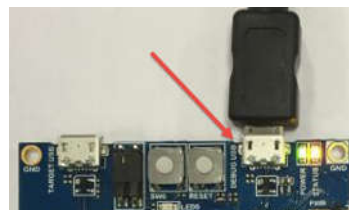
        while(!conversion_done);
        conversion_done = false;

        adc_async_read_channel(&ADC_0, 0, ADC_buffer, 2);
    }
}
```

(<https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/code22.png>)

Step 8: Program the board

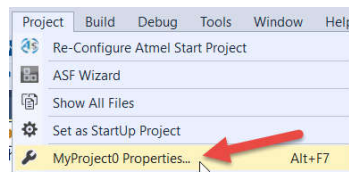
- 1 Connect the SAM D21 Xplained Pro Evaluation Kit to your computer.
 - Power-up your SAM D21 Xplained Pro board using DEBUG USB Connector.
 - Please be patient as the driver installs (it may take a minute or so).



(<https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/edb.png>)

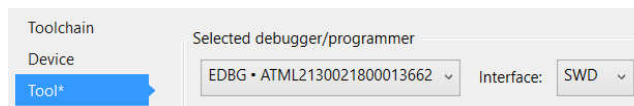
- 2 Select the embedded (on-board) debugger found on the SAM D21 XPRO board as the Debugger/Programmer for the project.

- Click on **Project > Properties**.



(https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/proj_prop.png)

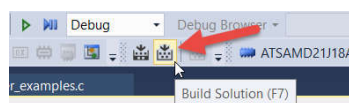
- Select **Tool > EDBG ...** as debugger/programmer.



(<https://microchip.wdfiles.com/local--files/atstart:sam-d21-uart/tool.png>)

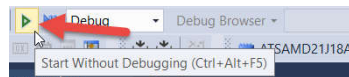
- 3 Compile the project and program the board.

- Compile the project by clicking on the 'Build Solution' icon or by typing 'F7', and verify that it builds successfully.



(<https://microchip.wdfiles.com/local--files/atstart:sam-d21-uart/build.png>)

- Program the application by clicking on the 'Start Without Debugging' icon.



(https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/start_without_debug.png)

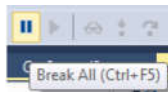


Note: If the firmware on the evaluation board is out of date, a *Firmware Upgrade* window will appear asking you if you want to upgrade the firmware. Select **Upgrade** and allow the process to complete.

Your application is now running on the evaluation board.

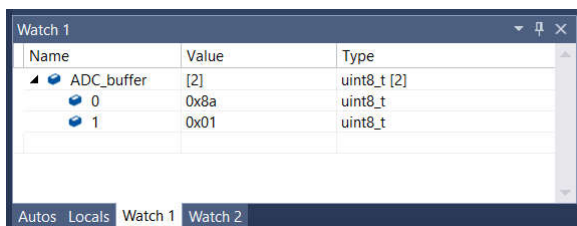
Step 9: Verify light sensor measurements

- Break the application by clicking the 'Break All' icon:



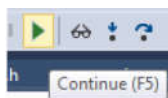
(<https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/break.png>)

- In the `main.c` file, right click on the 'ADC_buffer' variable then 'Add Watch':
- The *Watch 1* window will appear with your selected variable. Right-click on the *Watch* window value to select 'Hexadecimal Display'.



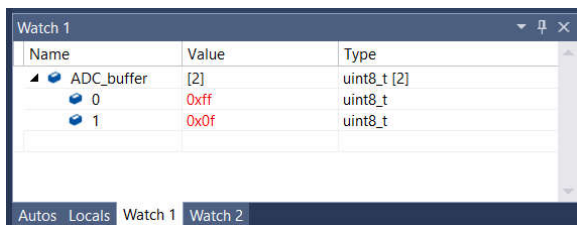
(<https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/watch1.png>)

- Click on the 'Continue' icon to execute the application.



(<https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/continue.png>)

If you hide the light sensor (i.e., put your finger on it) then click the 'Break All' icon to break the application, you can verify the buffer value should approach `0x0FFF`.



(<https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/watch2.png>)

Congratulations!

The SAM D21 ADC is measuring the output of the light sensor!

Step 10: Modify I²C example code to read temp sensor data and verify measurements

- 1 Add your application code to retrieve the Temperature Sensor data over I²C.
 - Call `temperature_sensors_init()` after `ADC_light_init()` as shown below:

```
int main(void)
{
    /* Initializes MCU, drivers and middleware */
    atmel_start_init();

    ADC_light_init();

    temperature_sensors_init();

    (https://microchip.wdfiles.com/local--
    files/atstart:sam-d21-adc-i2c/code23.png)
```



Note: The temperature sensor initialization source code can be found in the temperature_sensor_main.c file.

A global variable will be used to get temperature data.

- Add temperature global variable:

- float temperature;

```
#include <atmel_start.h>

volatile bool conversion_done = false;
uint8_t ADC_buffer[2];
float temperature;

(https://microchip.wdfiles.com/local--
files/atstart:sam-d21-adc-
i2c/code24.png)
```

- Get the temperature using the `at30tse75x_read()` function:

- temperature = at30tse75x_read(TEMPERATURE_SENSOR_0);

```
while (1) {
    delay_ms(1000);

    temperature = at30tse75x_read(TEMPERATURE_SENSOR_0);

    adc_async_start_conversion(&ADC_0);

    while(!conversion_done);
    conversion_done = false;

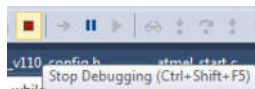
    adc_async_read_channel(&ADC_0, 0, ADC_buffer, 2);
}

(https://microchip.wdfiles.com/local--files/atstart:sam-d21-
adc-i2c/code16.png)
```

The I²C Temperature Sensor implementation is now completed.

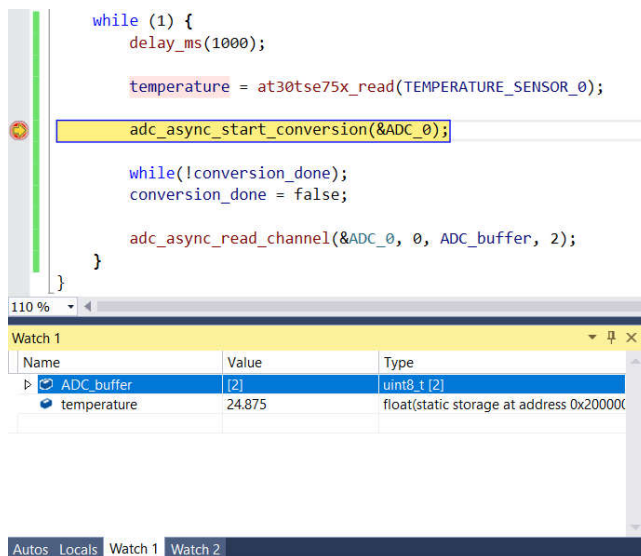
2 Use a *Watch* window to view the temperature sensor value.

- Stop the debugger.



(https://microchip.wdfiles.com
/local--files/atstart:sam-d21-
adc-i2c/stop.png)

- Compile the project by clicking the 'Build Solution' icon or by typing 'F7', and verify it builds successfully.
- Launch the debugger by clicking the 'Start Debugging' icon then break the application by clicking the 'Break All' icon.
- Right-click on the temperature variable then 'Add Watch' (Unselect Hexadecimal Display).
- Put a breakpoint after the `at30tse_read` call and execute the application to get the temperature value:



(<https://microchip.wdfiles.com/local--files/atstart:sam-d21-adc-i2c/watch3.png>)

Place your warm finger on the temperature sensor and click the debugger's 'Continue' icon to verify the temperature rises as expected.

Congratulations!

The SAM D21 SERCOM (configured for I²C) is receiving data from the temperature sensor over the I²C bus!