TRAINING MANUAL

Single-slope PWM Implementation using SAM D21 TCC in a Touch-based Application

AN-9232

Prerequisites

- **Hardware Prerequisites**
  - Atmel® SAM D21 Xplained Pro Evaluation Kit
  - Atmel QT1 Xplained Pro Kit
  - USB Micro Cable (TypeA/Micro-B)
- **Software Prerequisites**
  - Atmel Studio 6.2 or higher
  - Atmel Software Framework 3.17.0 or higher
  - Atmel QTouch® Composer 5.3 or higher
  - Atmel QTouch Library 5.3 or higher
- **Documentation Prerequisites**
  - *Atmel-42271-Introduction-to-QTouch-Design-Parameters-using-SAM-D21-Xplained-Pro_Training-Manual_AN-7846* Training Manual
- **Estimated completion time:** 60min.

Introduction

The goal of this hands-on is to:

- Create a Touch project using QTouch Project Builder
- Use QTouch Analyzer to test the project
- Implement a PWM signal using the SAM D21 Timer/Counters for Control peripheral (TCC) to drive the brightness of a LED
- Modify the LED brightness in relation to the Touch Delta value of a Button sensor by updating the PWM duty cycle

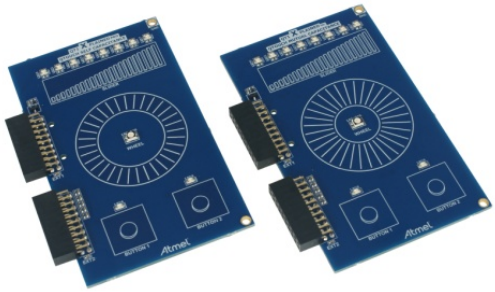


Table of Contents

Icon Key Identifiers

INFO	Delivers contextual information about a specific topic.
TIPS	Highlights useful tips and techniques.
TO DO	Highlights objectives to be completed.
RESULT	Highlights the expected result of an assignment step.
WARNING	Indicates important information.
EXECUTE	Highlights actions to be executed out of the target when necessary.

1. Training Module Architecture

This training material can be retrieved through different Atmel deliveries:

- As an Atmel Studio Extension (**.vsix** file) usually found on the Atmel Gallery web site (http://gallery.atmel.com/) or using the Atmel Studio Extension manager
- As an Atmel Training Executable (**.exe** file) usually provided during Atmel Training sessions

Depending on the delivery type, the different resources needed by this training material (hands-on documentation, datasheets, application notes, software, and tools) will be found on different locations.

1.1 Atmel Studio Extension (.vsix)

Once the extension installed, you can open and create the different projects using *"New Example Project from ASF..."* in Atmel Studio.

INFO The projects installed from an extension are usually under "*Atmel Training > Atmel Corp. Extension Name*".

There are different projects which can be available depending on the extension:

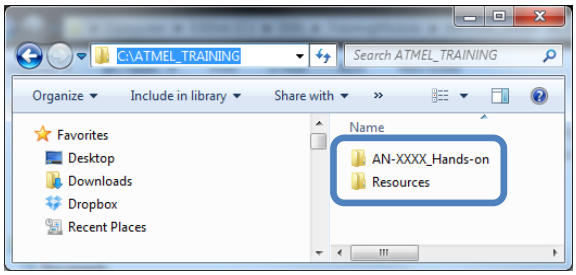
- **Hands-on Documentation**: contains the documentation as required resources
- **Hands-on Assignment**: contains the initial project that may be required to start
- **Hands-on Solution**: contains the final application which is a solution for this hands-on

INFO Each time a reference is made to some resources in the following pages, the user must refer to the **Hands-on Documentation** project folder.

1.2 Atmel Training Executable (.exe)

Depending where the executable has been installed, you will find the following architecture which is composed by two main folders:

- AN-XXXX_Hands-on: contains the initial project that may be required to start and a solution
- **Resources**: contains required resources (datasheets, software, and tools…)



INFO Unless a specific location is specified, each time a reference is made to some resources in the following pages, the user must refer to this **Resources** folder.

2. Introduction

This hands-on will focus on the development of a simple Touch application that uses a Button sensor from the Atmel QT1 Xplained Pro Self Capacitance extension board to modify the brightness of a LED.

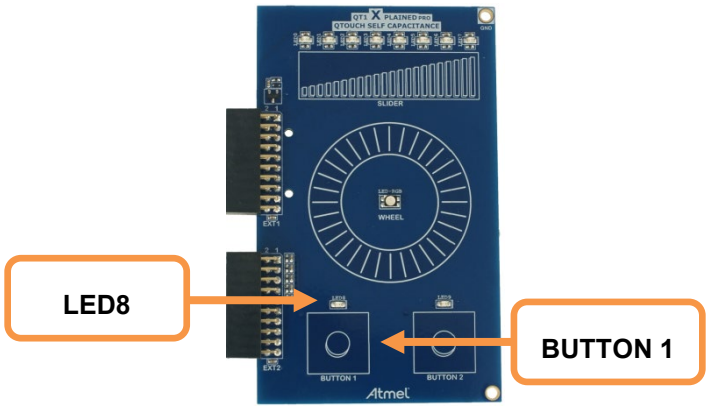
The LED brightness is controlled using a PWM signal that is generated by one of the Atmel SAM D21 Timer/Counters for Control (TCC) peripherals.

The SAM D21 embeds three different 24-bit Timer/Counters for Control peripherals which provide extended functions compared to the standard Timer/Counters (TC):

- Up to four compare channels with optional complementary outputs
- Advanced waveform signal generation with dead time management
- Deterministic fault protection, fast decay, and configurable dead-time between complementary outputs
- Dithering that increase resolution with up to five bits and reduce quantization error

The hands-on will be divided in the following assignments:

- Create a new Touch project based on BUTTON 1 Touch sensor using QTouch Project Builder from Atmel QTouch Composer:



- Use QTouch Analyzer to provide graphical representation and demonstration of real time touch data. This will allow testing the newly-created Touch project.
- Implement a PWM signal using the TCC0 peripheral to control the LED8 brightness
  - Atmel Software Framework (ASF) TCC Quick Start Guide will be used to implement it easily
  - A Waveform Output pin (WO6) from the SAM D21 MCU will be used to output the PWM signal directly to the LED8
- Modify the LED brightness in relation to the Touch Delta value of BUTTON 1 by updating the PWM duty cycle
  - Atmel QTouch library API will be used to get the delta value of BUTTON 1 sensor

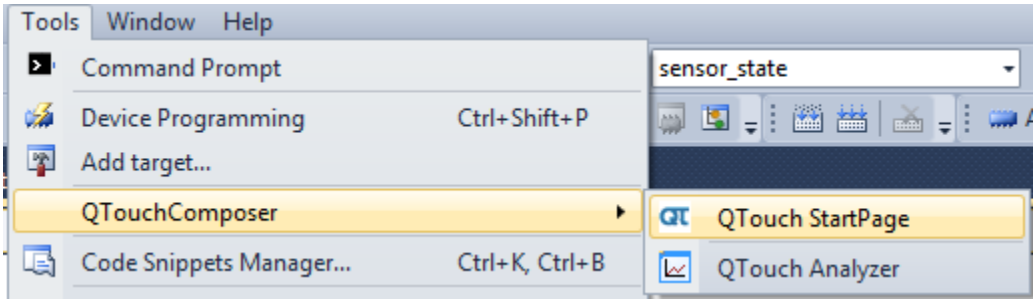
3. Assignment 1: Create a Touch project using QTouch Project Builder

3.1 Project Creation

TO DO Create a new QTouch Project.

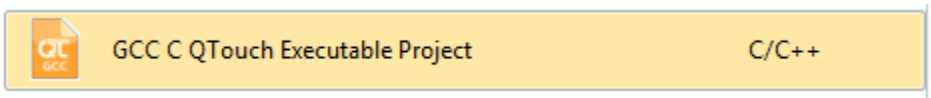
- Open Atmel Studio 6.2
- Click on the "QT" icon to open the QTouch Start Page: 

INFO QTouch StartPage is also accessible through the Tools menu:



- Select "New QTouch UserBoard Project…"

- Ensure "GCC QTouch Executable Project" is selected (default)*



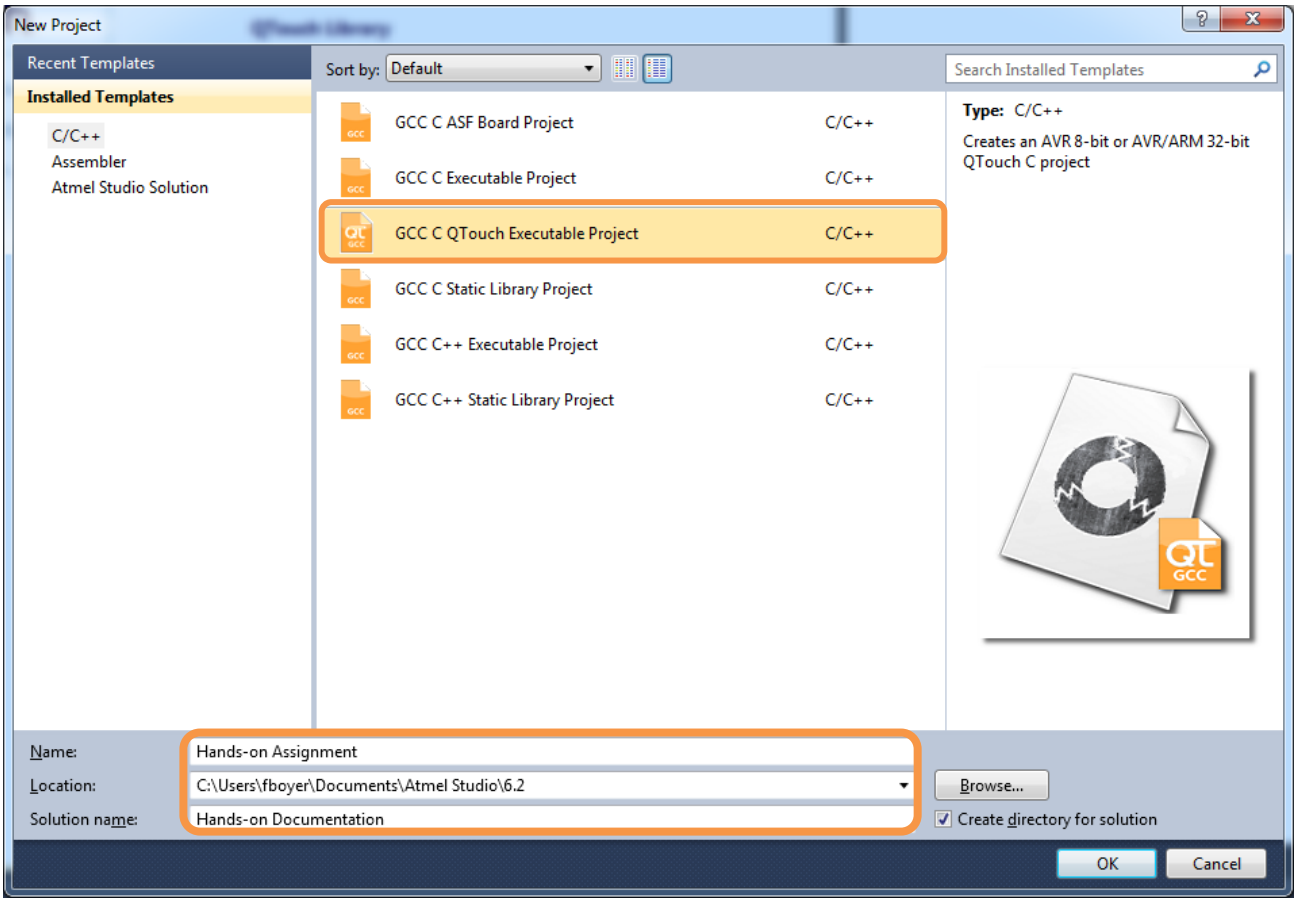
- Fill-in New Project fields according to following use cases:

Atmel Training Executable Case

- **Name**: Hands-On Assignment
- **Location**: “AN-9232_SAMD21-XPRO_QTouch_TCC\assignments” (relative path in the ATMEL_TRAINING installation folder)
- **Solution name**: Hands-On Assignment
- Click OK

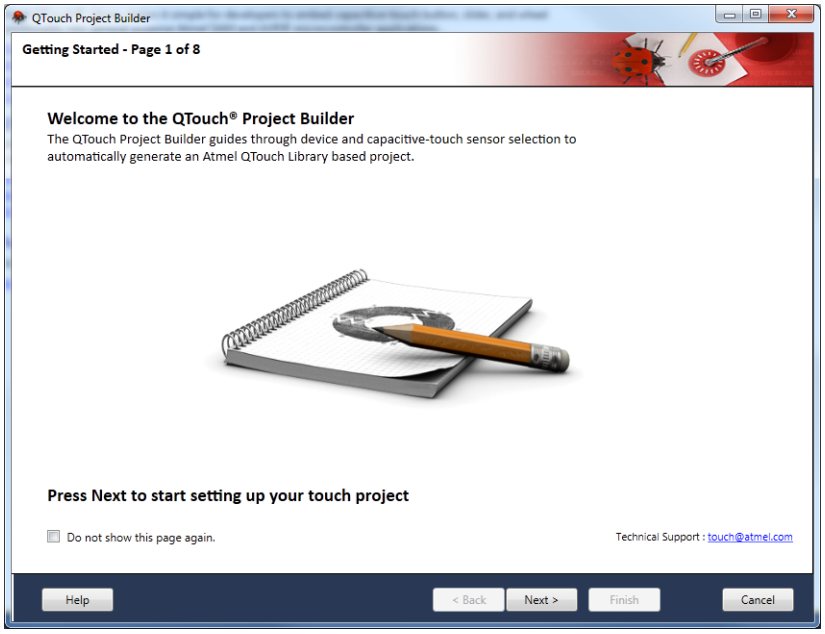
Atmel Extension Case (downloaded from Atmel Gallery or Studio Extension Manager)

- **Name**: Hands-On Assignment
- **Location**: existing Hands-on Documentation solution path
- **Solution name**: Hands-On Documentation
- Click OK



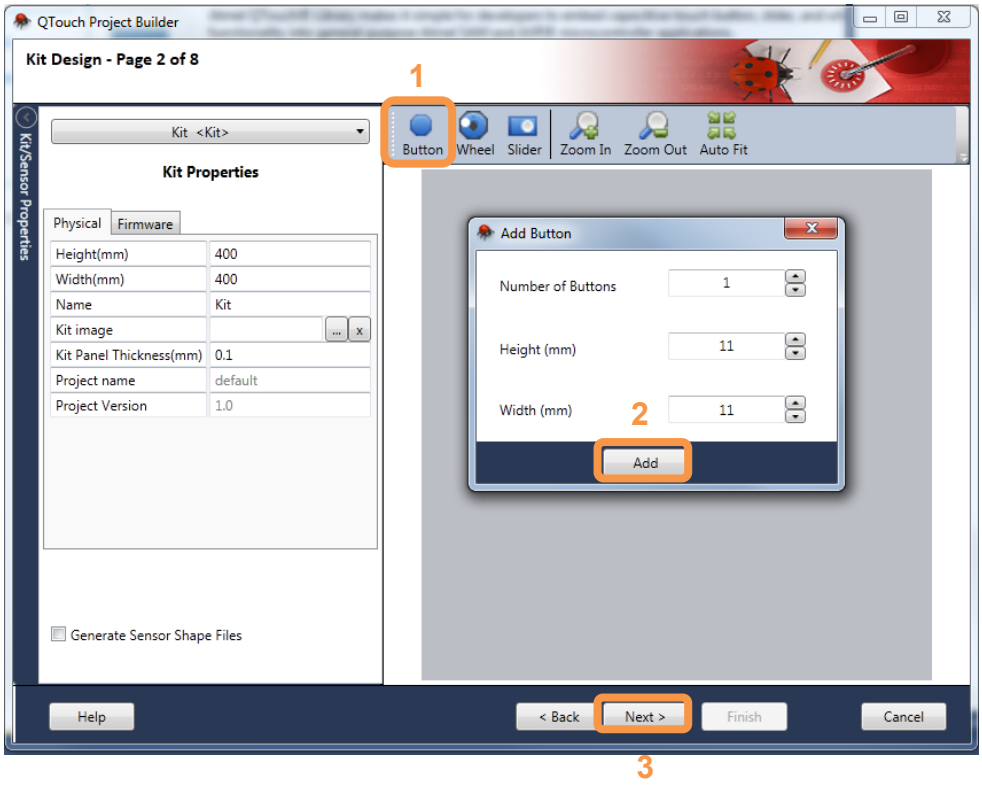
The QTouch Project Builder wizard will now guide through the steps involved in creating a QTouch project.

- Click “Next” in the *Getting Started* page (if shown):

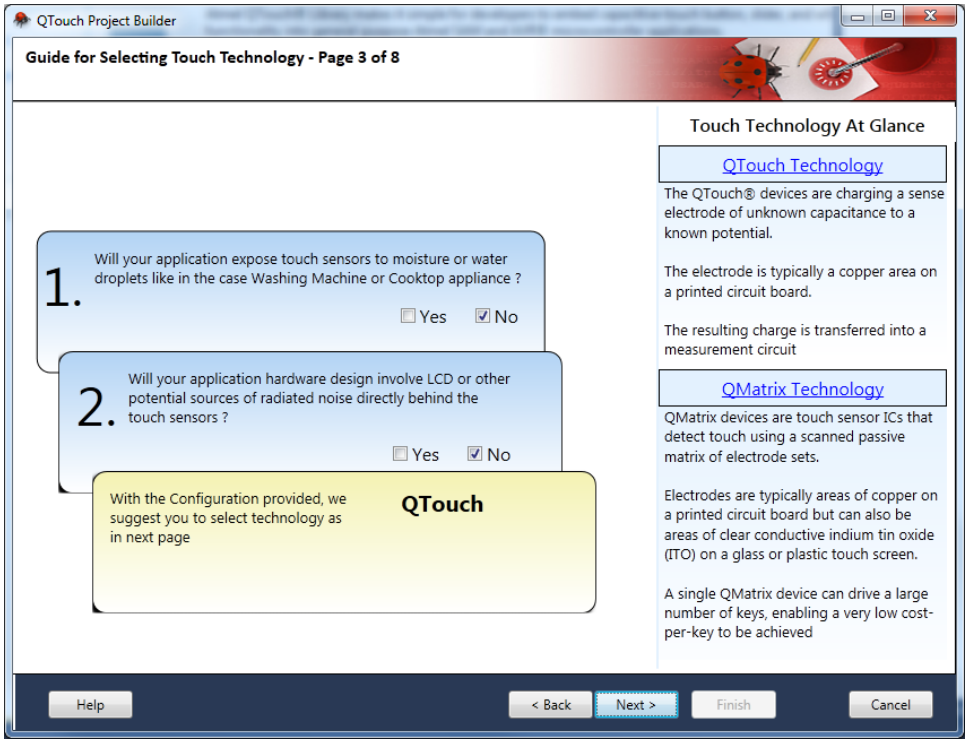


The Kit design Page provides an option to add, delete or move QTouch Sensors such as button, wheel and slider. It also provides options to setup physical and firmware properties of the board and sensors.

- Add a button in the *Kit Design* page by clicking on the “Button” icon, click “Add” and “Next”

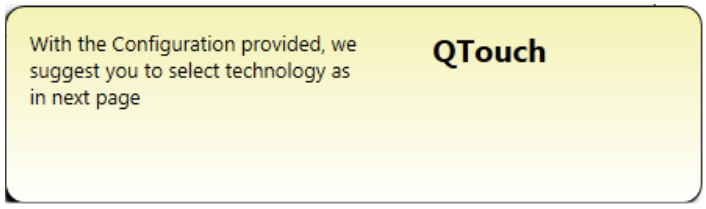


The Technology Selection Guide recommends which acquisition technology should be used based on the number of sensors previously selected in Kit Design page and answers from the below questionnaire.



- We will consider for this hands-on that our application hardware design is NOT exposed to moisture or water droplets or any specific sources of radiated noises

  So, select “No” as the answer to Question 1 and 2. You will see that the proposed technology will become QTouch (**i.e. Self Capacitance Technology**):



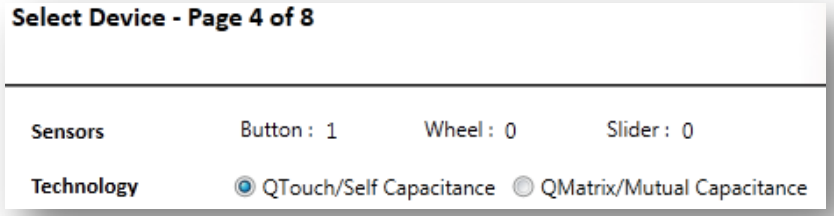
INFO For more information about designing the touch sensor, refer to *Buttons, Sliders and Wheels Touch Sensor Design Guide* available at www.atmel.com/images/doc10752.pdf.

- Click “Next” in the *Guide for Selecting Touch Technology* page

Device Selection Page provides the option to view the number of QTouch sensors such as Button, Wheel and Slider that was selected in the Kit Design Page.

Touch sensors can be based on two sub-technologies, **QTouch/Self Capacitance** and **QMatrix/Mutual Capacitance**.

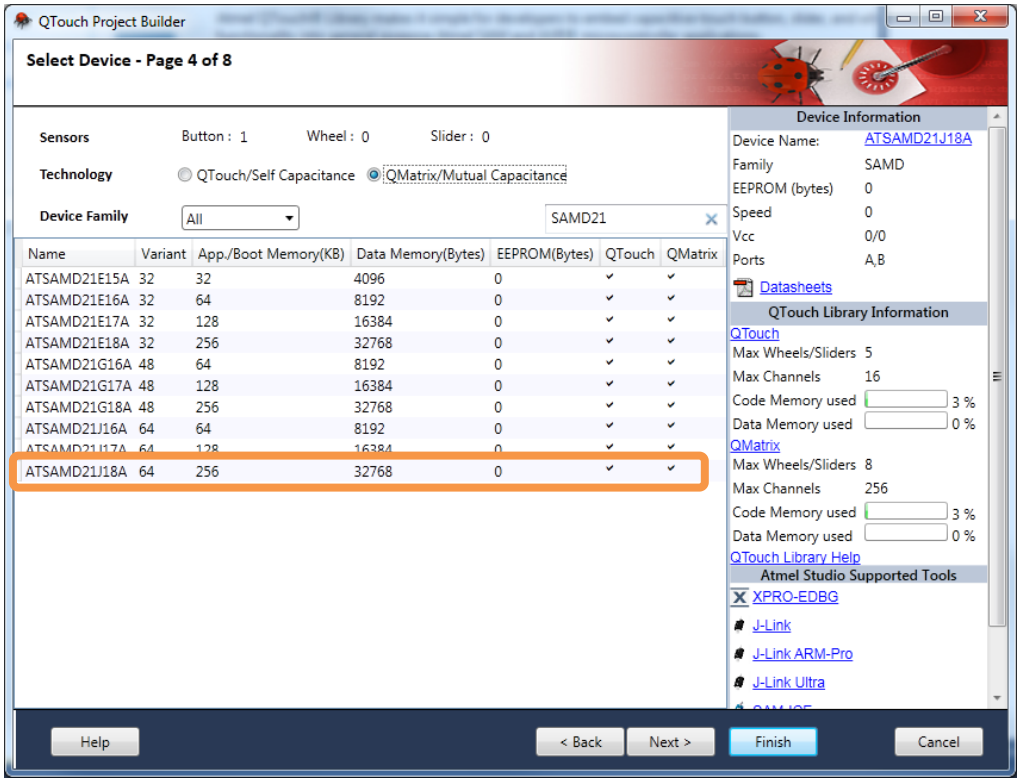
- We can verify that **QTouch/Self Capacitance** has been correctly selected following our previous choice in the *Guide for Selecting Touch Technology* page



The device list displays device details such as Device Name, Variant, Application/Boot Memory, Data Memory, EEPROM and supported technology. Data listing will vary based on device family, number of selected sensors and technology as only devices supporting all choices made will be shown. Additionally user has the option to search for a particular device.

- Type “SAMD21” in the search box to make a first filtering

- Select ATSAMD21J18A in the *Select Device* page and click “Next”. This is the device on the SAM D21 Xplained Pro kit.



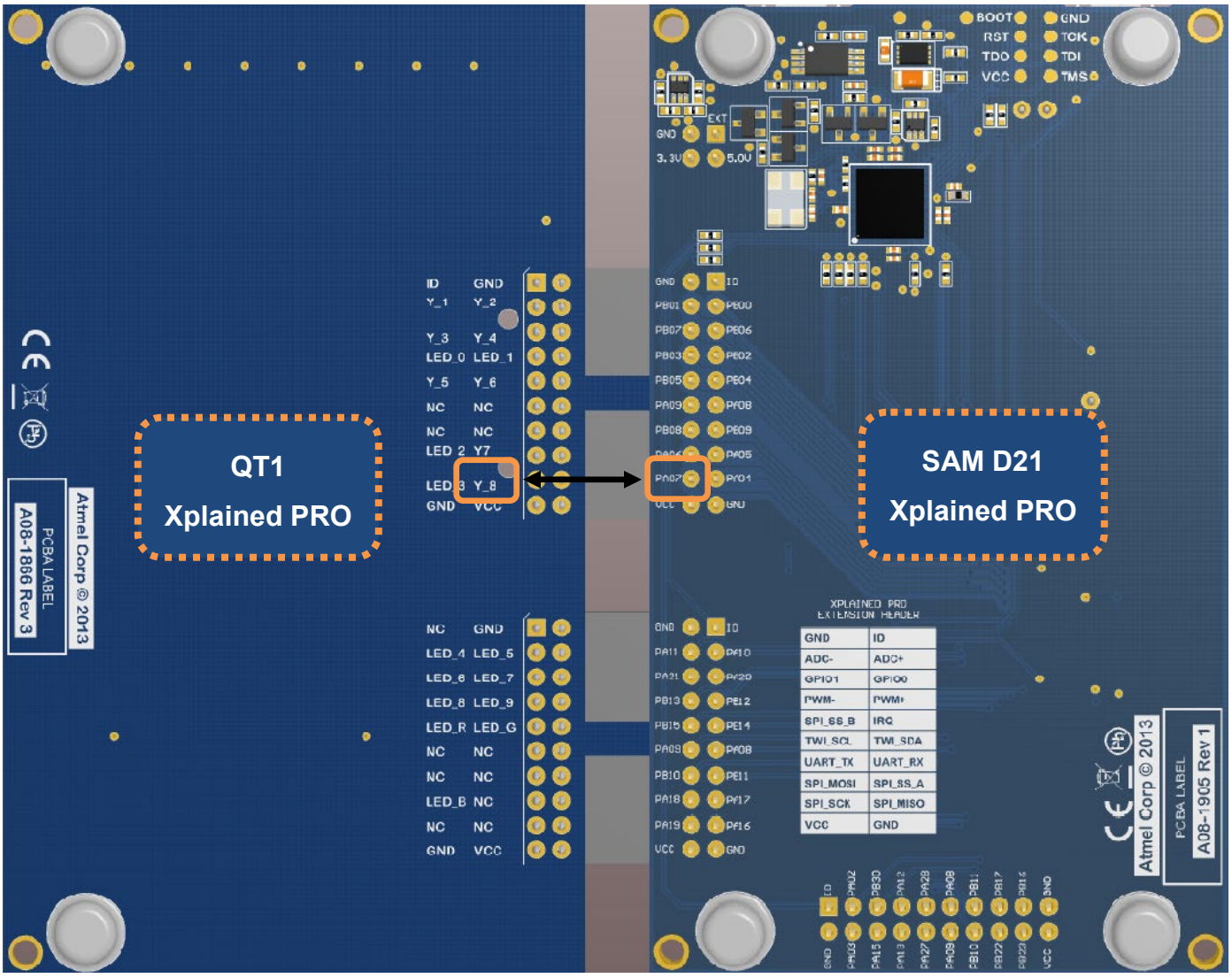
Self capacitance method uses a single sense electrode, denoted by a Y line.

Self capacitance touch button sensor is formed using a single Y line channel, while a touch rotor or slider sensor can be formed using three Y line channels.

WARNING The Y-line numbering in the QT1 Xplained Pro User Guide does not correspond to the Y-line numbering in the SAM D21 PTC module as the QT1 Xplained Pro is a generic touch board (not dedicated to the SAM D21).

- Flip the SAM D21 Xplained Pro and QT1 Xplained Pro boards and determine which SAM D21 I/O pins that are connected to BUTTON 1

TIPS Look at next page to determine the right Y-line from QT1 Xplained PRO extension board.



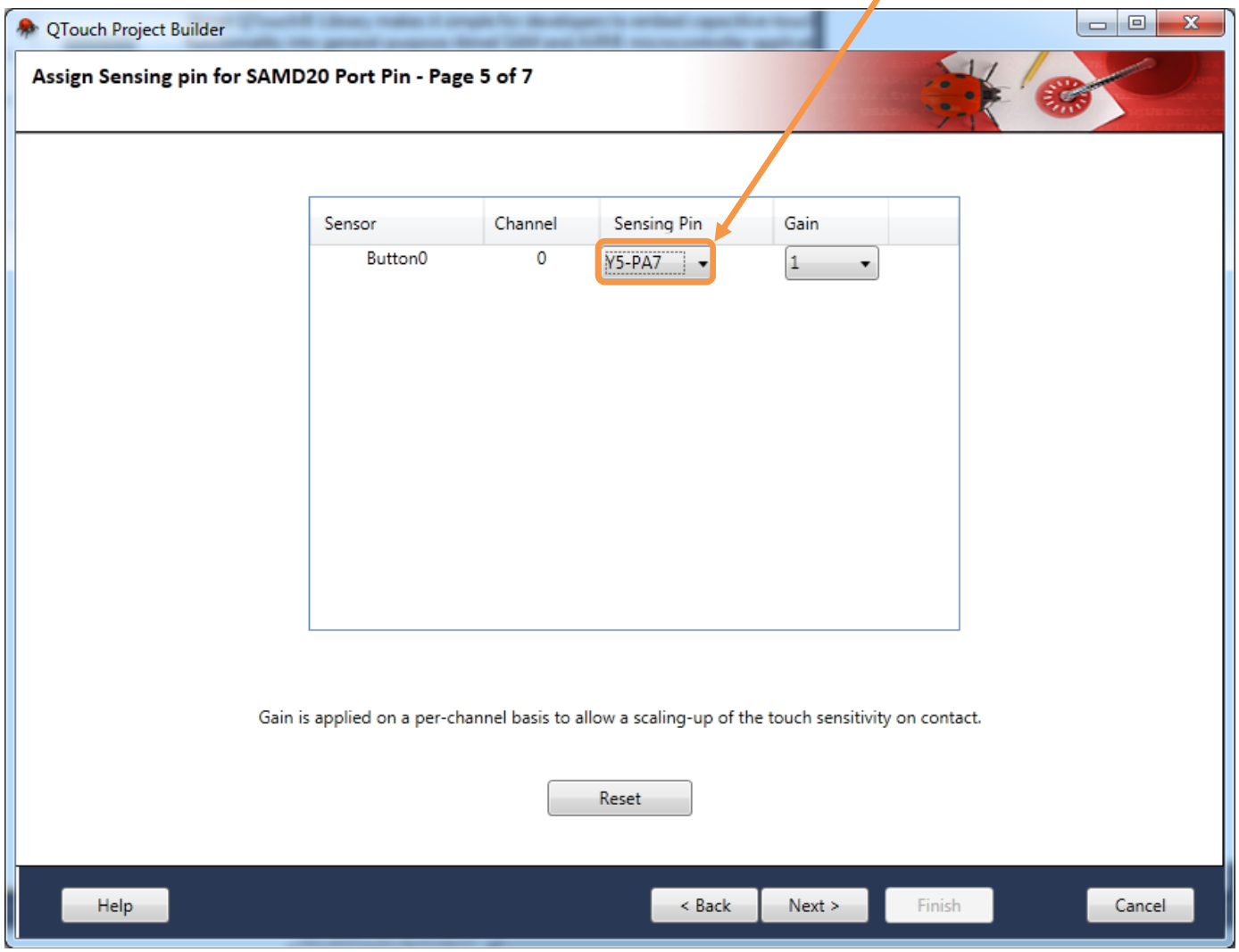
INFO Y-line for BUTTON 1 can be retrieved from the Atmel QT1 Xplained Pro User Guide (http://www.atmel.com/Images/Atmel-42193-QT1-Xplained-Pro_User-Guide.pdf):

Table 3-1. QT1 Xplained Pro SC Extension Header 1

Pin on EXT	Function	Description
1	ID	Communication line to ID chip.
2	GND	Ground
3	Y_1	Y-line 1 for Slider
4	Y_2	Y-line 2 for Slider
5	Y_3	Y-line 3 for Slider
6	Y_4	Y-line 4 for Wheel
7	LED_0	Slider, LED 0 (Yellow)
8	LED_1	Slider, LED 1 (Yellow)
9	Y_5	Y-line 5 for Wheel
10	Y_6	Y-line 6 for Wheel
11	Not Connected	
12	Not Connected	
13	Not Connected	
14	Not Connected	
15	LED_2	Slider, LED 2 (Yellow)
16	Y_7	Y-line 7 for Button 2
17	LED_3	Slider, LED 3 (Yellow)
18	Y_8	Y-line 8 for Button 1
19	GND	Ground
20	VCC	Target supply voltage

- Use the *Assign X-Y Line and Gain to the Channel* page from QTouch Project Builder to select the correct Y Line:

Y Line	QT1 Xplained PRO Y Line	SAM D21 Xplained PRO Y Line – I/O
Y-line	Y_8	Y[5] – PA07



We also need to select on this page the **Gain** for each channel.

INFO

Gain setting is applied on a per-channel basis to allow superior sensitivity upon contact. Recommended gain settings depend on the sensor design and touch panel thickness.

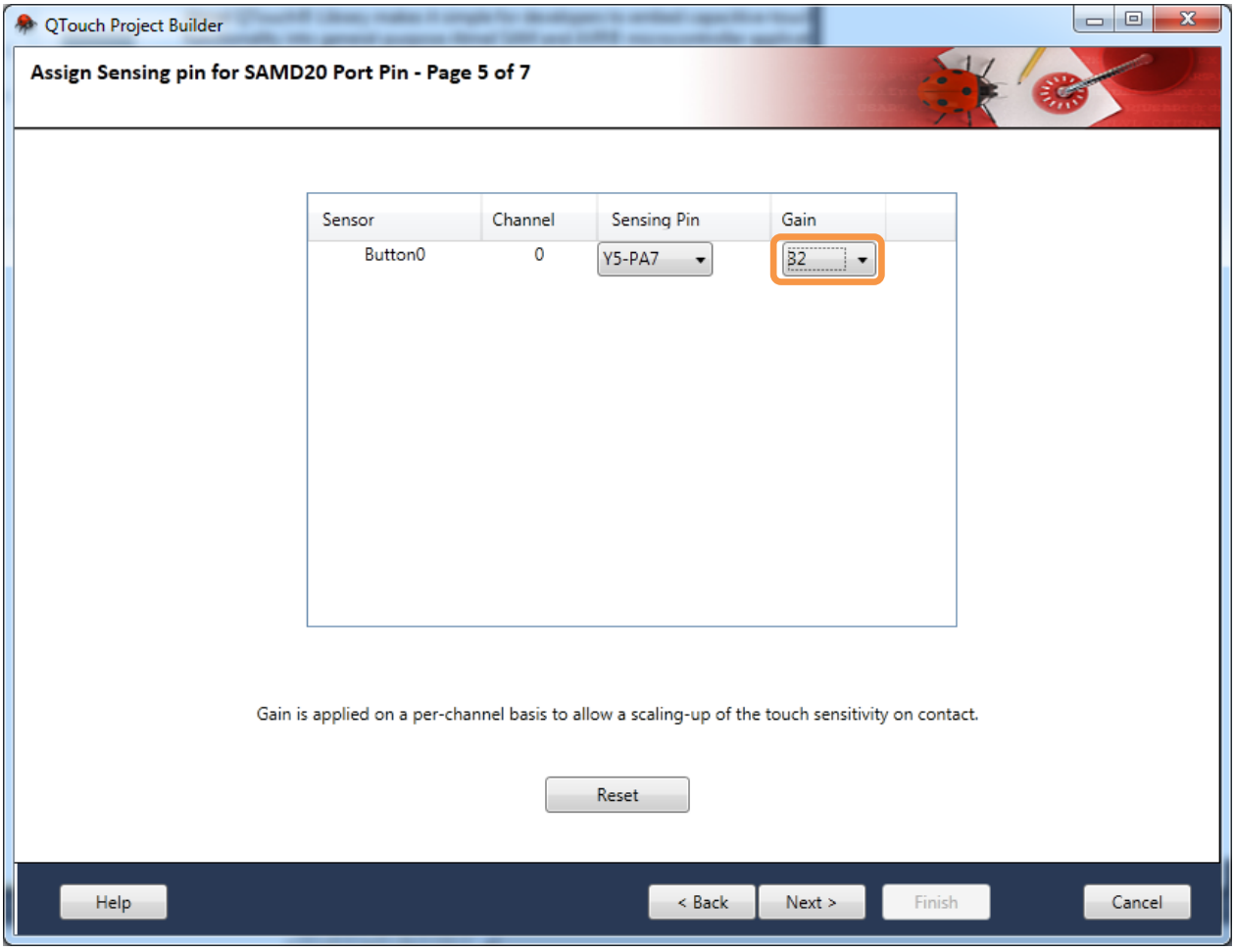
TIPS

The increase in the gain value is usually required by a thick front panel but also depends on the sensor design (size of the sensor and coupling between X and Y for mutual capacitance sensors).

Thicker front panels (3mm and more) can result in reduced sensitivity upon touch. Likewise, smaller size sensors or sensors surrounded by ground layer or sensors with ground on the bottom side typically require an increased gain setting.

- In this application, we need to increase the sensor sensitivity to get the highest proximity effect possible

  As a consequence, set Gain to 32 and click “Next”.



- Check the “Enable QDebug Interface” checkbox, which allows Live streaming of Touch data with QTouch Analyzer

INFO QDebug is the Touch data protocol used by QTouch Analyzer to communicate with the SAM D21 through the Atmel **D**ata **G**ateway **I**nterface (**DGI**).

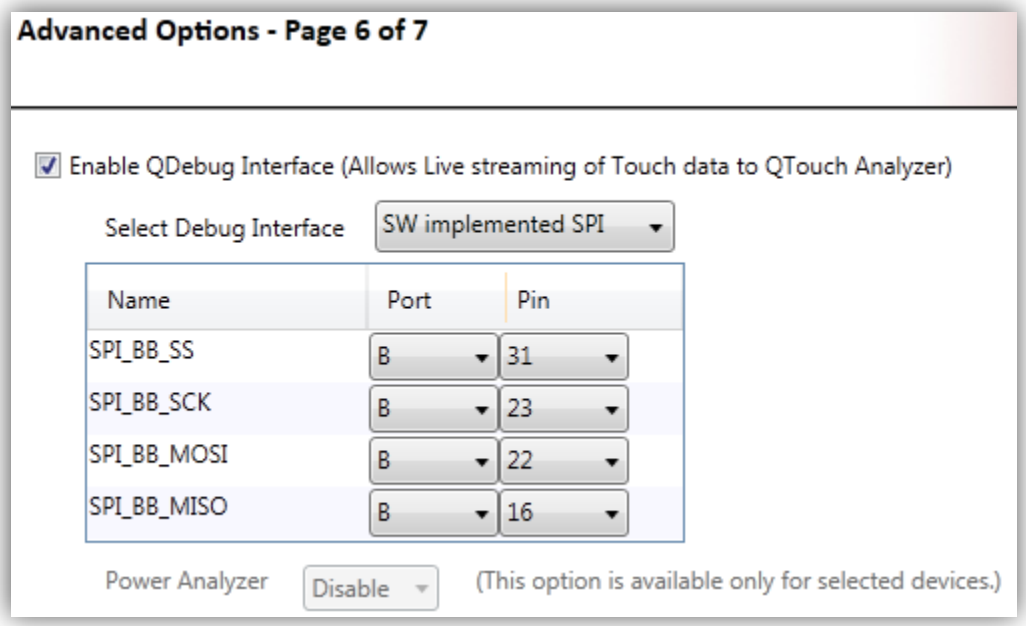
INFO The Atmel Embedded Debugger (**EDBG**) offers a Data Gateway Interface (**DGI**) for streaming data to a host PC. This is meant as an aid in debugging and demonstration of features in the application running on the target device. DGI consists of multiple interfaces for data streaming. The supported interfaces are SPI Interface, USART Interface, TWI Interface, and GPIO Interface.

TIPS SAM D21 XPRO Kit Data Gateway Interface supports SPI or I²C.

- Assign the right port(s) and pins to the SPI interface, these can be found in the SAM D21 Xplained Pro User Guide on the *Advanced Options* page (http://www.atmel.com/Images/Atmel-42220-SAMD21-Xplained-Pro_User-Guide.pdf):

Table 3-2. DGI Interface Connections when Using SPI

Pin on SAM D21	Function
PB31	SERCOM5 PAD[1] SPI SS (Slave select) (SAM D21 is Master)
PB16	SERCOM5 PAD[0] SPI MISO (Master In, Slave Out)
PB22	SERCOM5 PAD[2] SPI MOSI (Master Out, Slave In)
PB23	SERCOM5 PAD[3] SPI SCK (Clock Out)



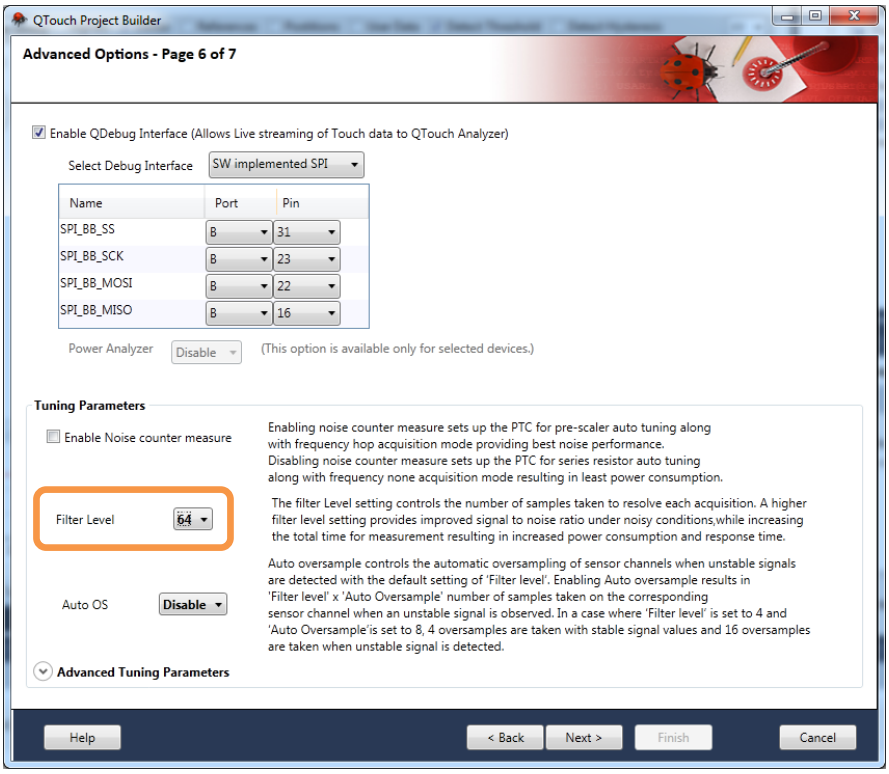
The following table extracted from the *QTouch Library PTC User Guide* (*http://www.atmel.com/Images/Atmel-42195-Qtouch-Library-Peripheral-Touch-Controller_User-Guide.pdf*), indicates the expected **Resting Signal** value (also called **Reference**) for a given combination of gain setting and filter level setting.

Expected Resting Signal Value for FILTER LEVEL and GAIN Combination	GAIN_1=0	GAIN_2=1	GAIN_4=2	GAIN_8=3	GAIN_16=4	GAIN_32=5
FILER_LEVEL_1 = 0	512	512	512	512	512	512
FILER_LEVEL_2 = 1	512	1024	1024	1024	1024	1024
FILER_LEVEL_4 = 2	512	1024	2048	2048	2048	2048
FILER_LEVEL_8 = 3	512	1024	2048	4096	4096	4096
FILER_LEVEL_16 = 4	512	1024	2048	4096	8192	8192
FILER_LEVEL_32 = 5	512	1024	2048	4096	8192	16384
FILER_LEVEL_64 = 6	512	1024	2048	4096	8192	16384

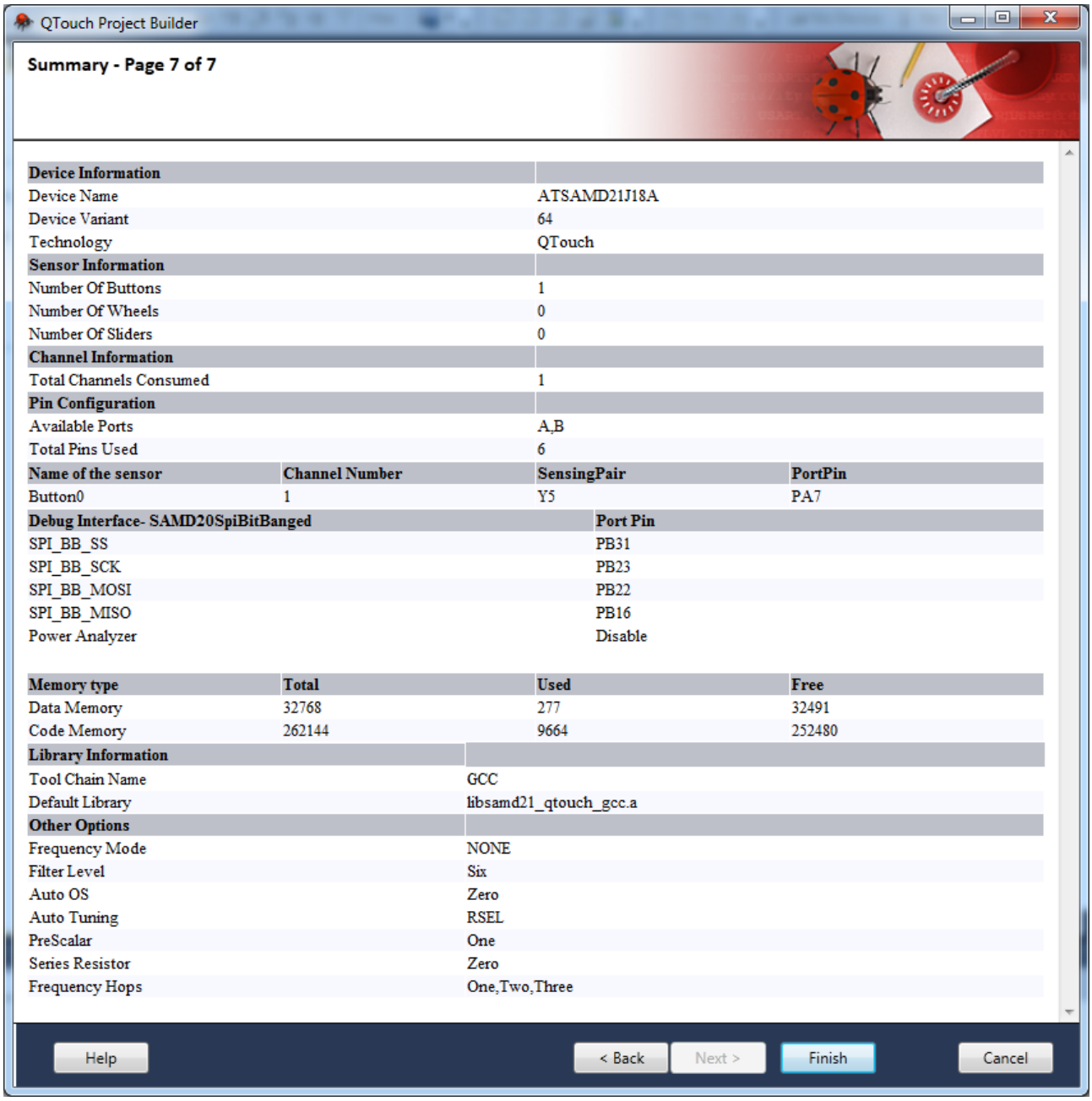
By increasing both the Gain and the Filter Level, we will increase the Reference value and so scale up the Touch Delta value, which will allow improving the proximity effect.

INFO (Touch) Delta represents the difference between the Signal and the Reference.

- On the on the *Advanced Options* page, update Filter Level to 64:



- Review the Summary page which provides a summary of the settings provided by the previous set of wizard pages. Then, click on Finish and wait for your Touch project to be generated.



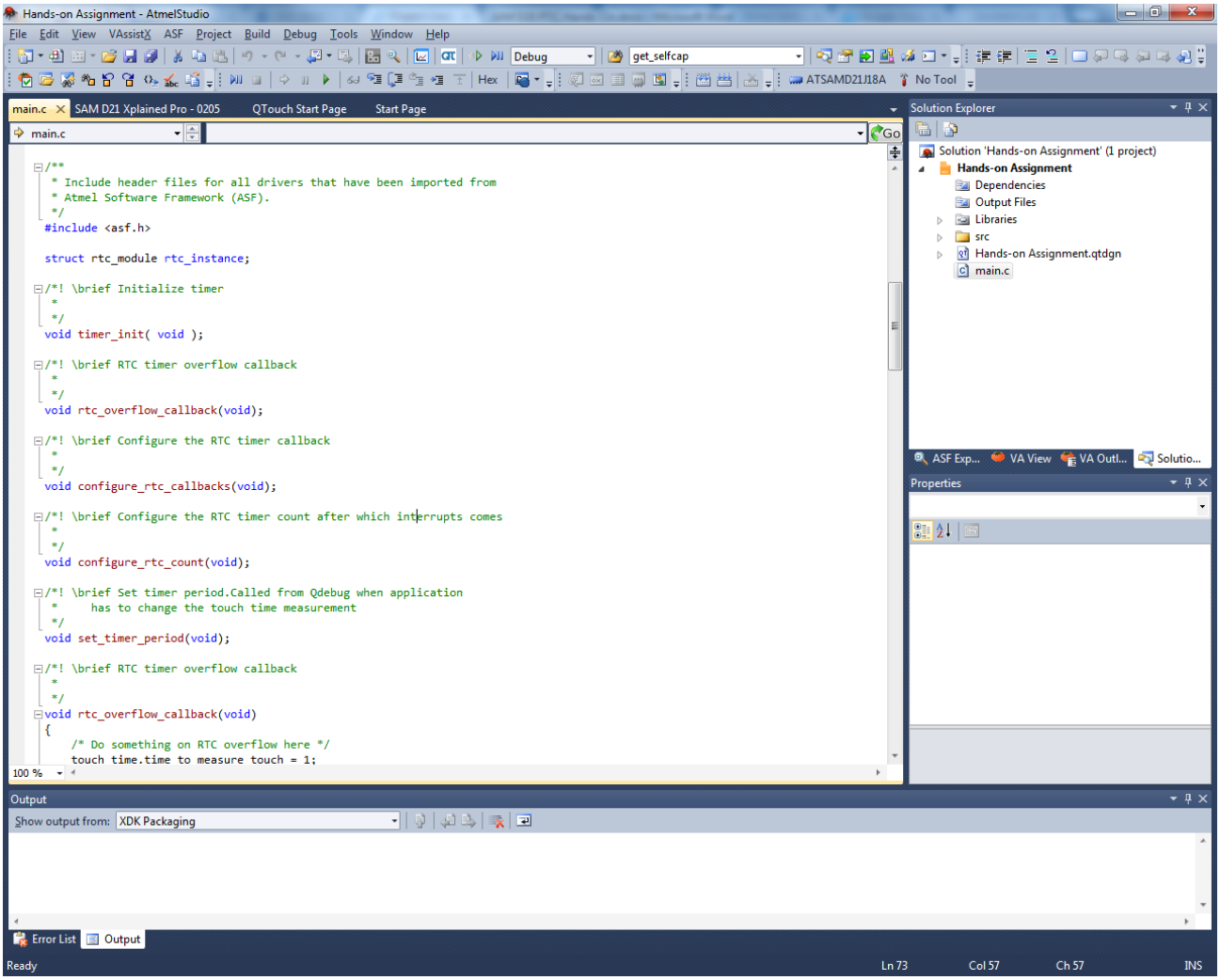
Device Information			
Device Name		ATSAMD21J18A	
Device Variant		64	
Technology		QTouch	
Sensor Information			
Number Of Buttons		1	
Number Of Wheels		0	
Number Of Sliders		0	
Channel Information			
Total Channels Consumed		1	
Pin Configuration			
Available Ports		A,B	
Total Pins Used		6	
Name of the sensor	**Channel Number**	**SensingPair**	**PortPin**
Button0	1	Y5	PA7

Debug Interface- SAMD20SpiBitBanged	Port Pin
SPI_BB_SS	PB31
SPI_BB_SCK	PB23
SPI_BB_MOSI	PB22
SPI_BB_MISO	PB16
Power Analyzer	Disable

Memory type	Total	Used	Free
Data Memory	32768	277	32491
Code Memory	262144	9664	252480

Library Information	
Tool Chain Name	GCC
Default Library	libsamd21_qtouch_gcc.a
Other Options	
Frequency Mode	NONE
Filter Level	Six
Auto OS	Zero
Auto Tuning	RSEL
PreScalar	One
Series Resistor	Zero
Frequency Hops	One,Two,Three

RESULT You have created a Touch Project with a Self Capacitance technology Touch Button.



3.2 Conclusion

In the first assignment, you have learnt:

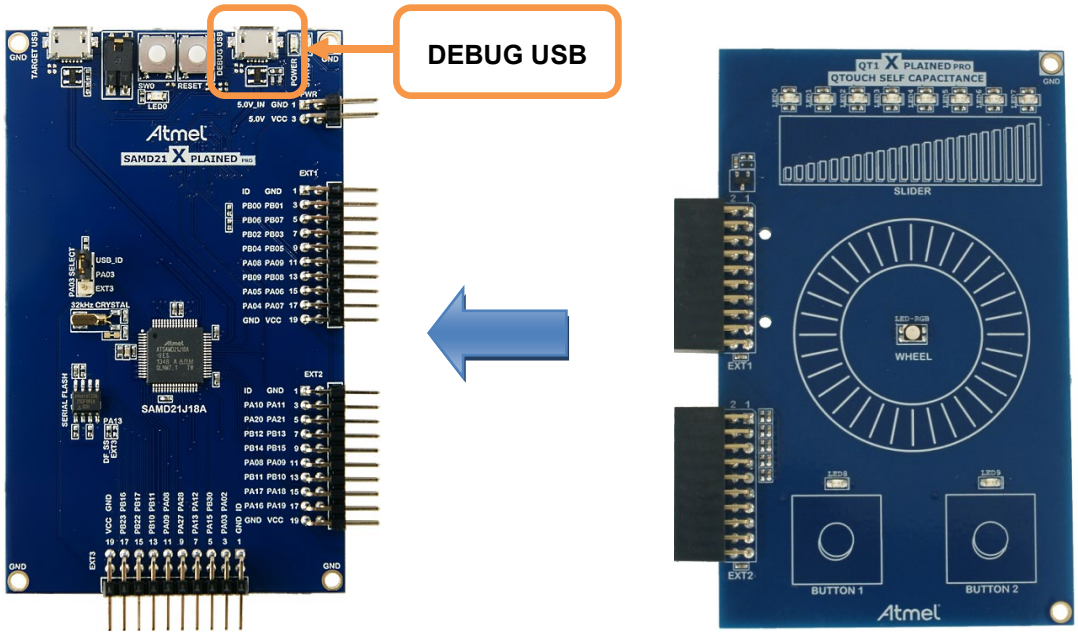
- How to create a Touch Project using QTouch Composer Project Builder
- How easy it is to use QTouch Composer to design a Touch project on the Atmel SAM D21 MCUs

4. Assignment 2: Use QTouch Analyzer to Test the Project

4.1 Connect the Kit to QTouch Analyzer

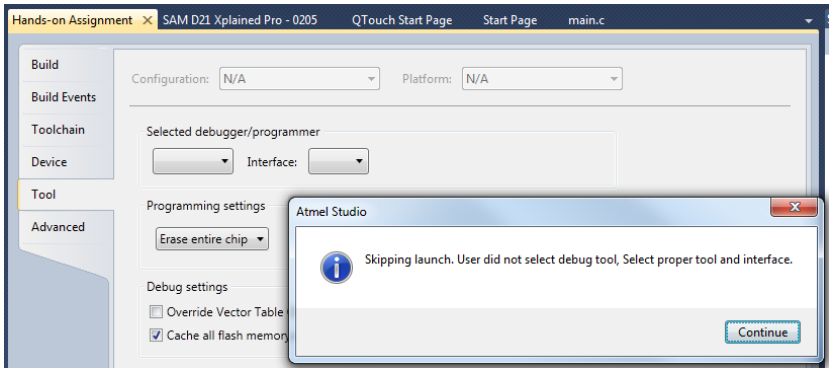
TO DO Build the QTouch project and Program the SAM D21 Xplained Pro.

- Connect the QT1 Xplained Pro Self Capacitance board to the SAM D21 Xplained Pro board:

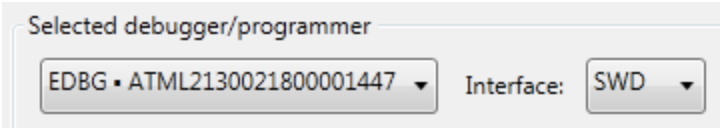


- Connect the SAM D21 Xplained Pro board to your PC using DEBUG USB connector
- Build the solution and ensure you get no errors: 
- Program the application by clicking on the Start Without Debugging icon: 

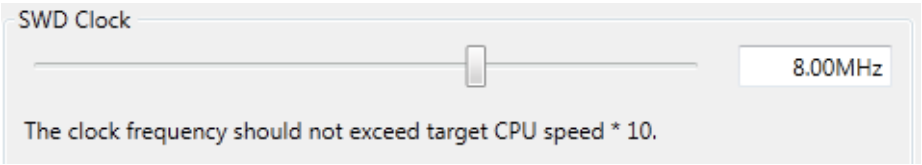
- You will be asked the first time to select your debug tool:



- Select your EDBG and SWD (Serial Wire Debug) as Interface:

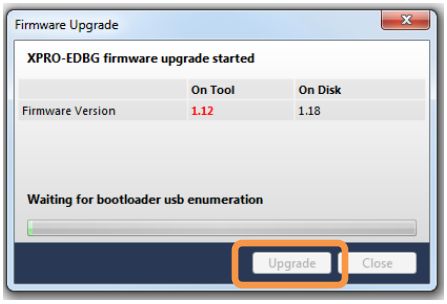


- Set SWD clock to 8MHz to speed up programming:



- Click again on the Start Without Debugging icon: 

INFO You may be asked to upgrade your EDBG firmware. If so, click on Upgrade:



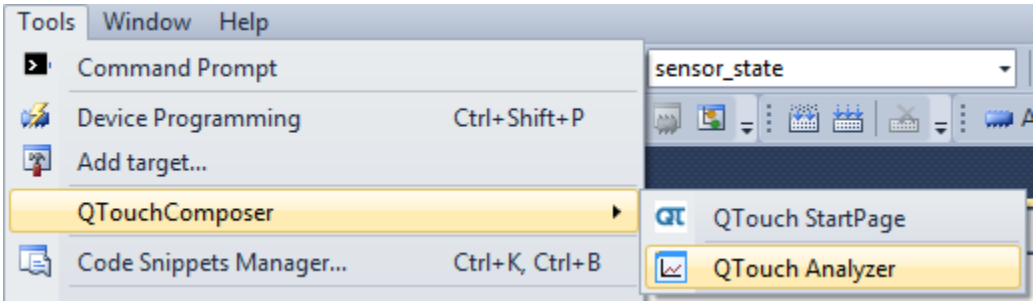
WARNING Upgrade operation may take a few minutes, please **wait** for the operation to complete.

RESULT Your QTouch project is now updated and programmed on the SAM D21 Xplained Pro.

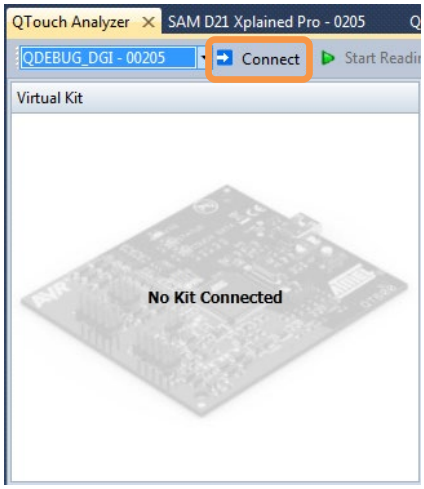
TO DO Connect your kit to QTouch Analyzer.

- Press RESET button on the SAM D21 Xplained Pro
- Launch QTouch Analyzer by clicking on the following icon: 

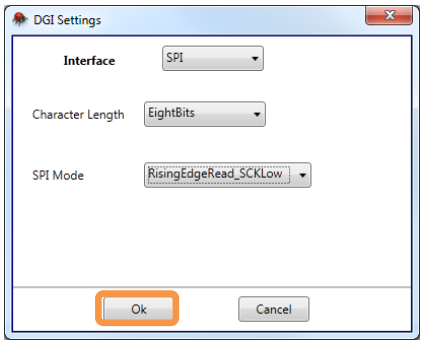
INFO QTouch Analyzer is also accessible through the Tools menu:



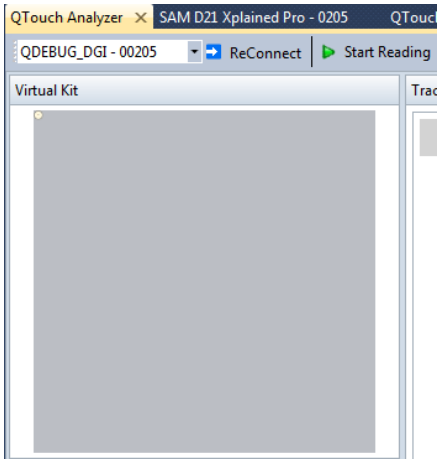
- A kit named QDEBUG_DGI will be listed in the combo box.
  Select the kit and click on Connect:

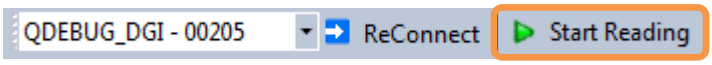


- DGI settings dialog box will be displayed as shown in the figure, click OK:



The Virtual kit will be displayed, which shows it has been correctly connected:



- Click on Start Reading: 

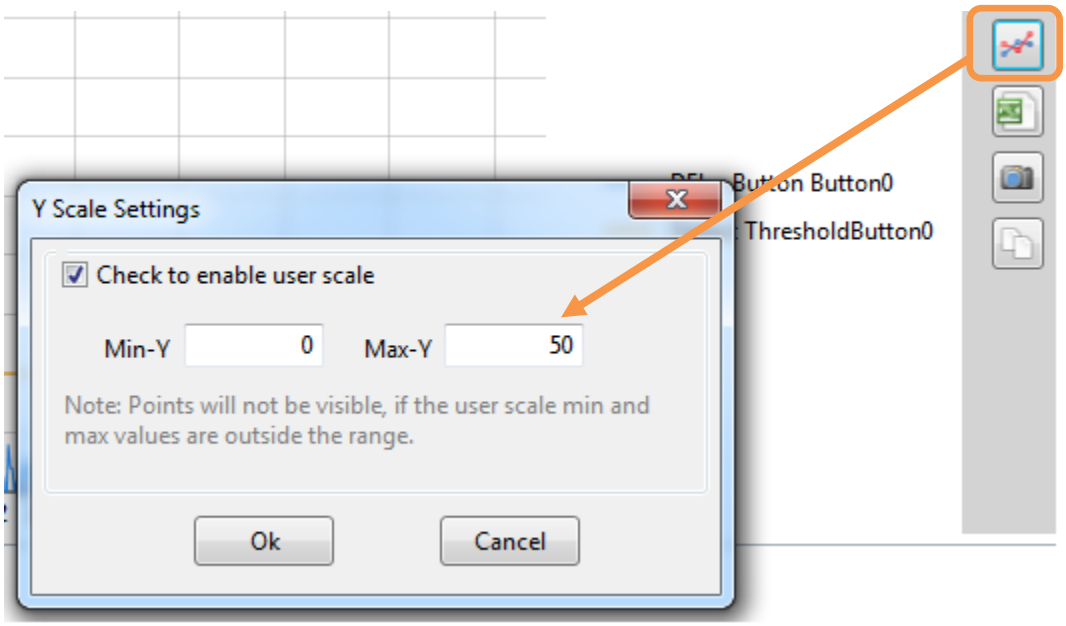
RESULT QTouch Analyzer is now connected to your kit and displays both the sensor signal and reference values.

TIPS Click on Stop Reading to prevent QTouch Analyzer from collecting touch data over a long period as this might slow down the computer.

4.2 Test the Proximity Effect of the QTouch Application

TO DO Check proximity effect.

- Click on checkbox “Check to enable user scale” and set Y scale from 0 to 50:

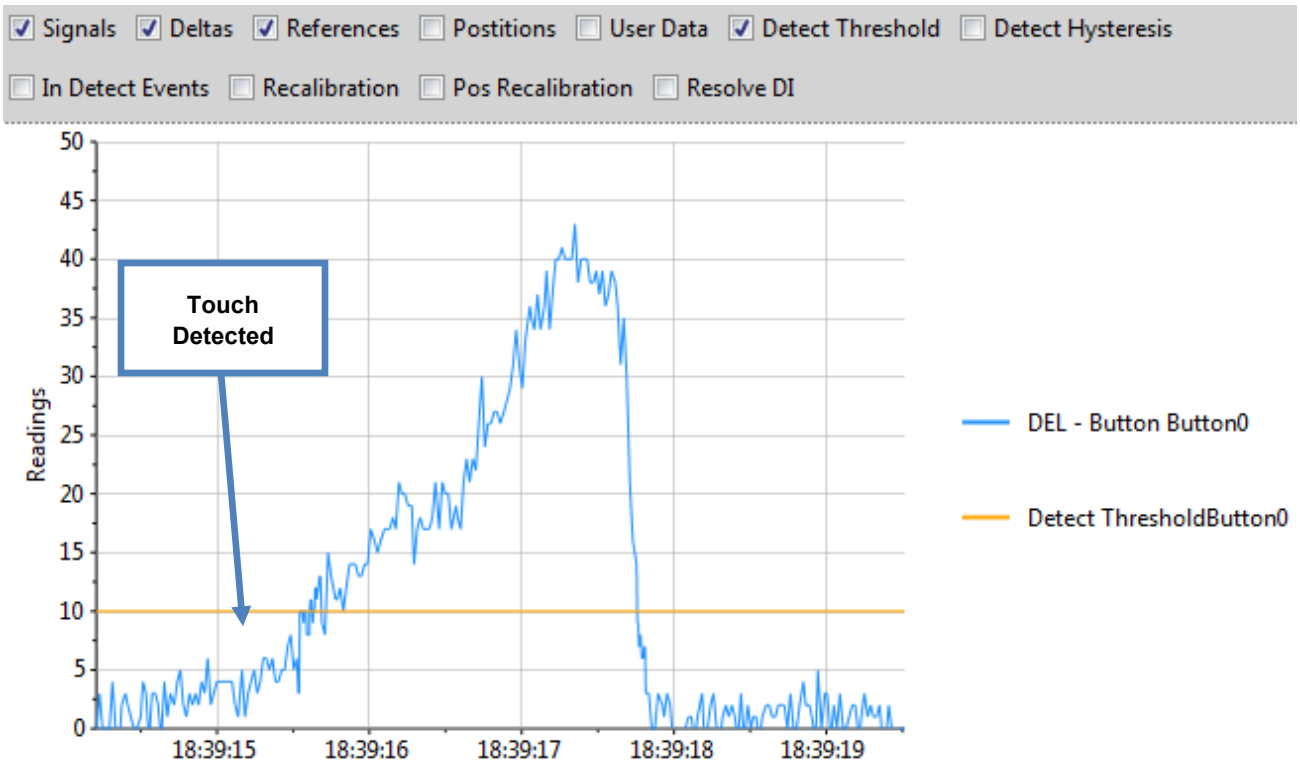


- Click on Start Reading: 

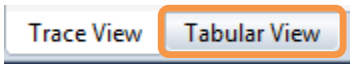
- Unselect Signals and References and Deltas in the Trace View

- Select Deltas and Detect Threshold in the Trace View

- Approach BUTTON 1 on your kit, which is displayed as Button0 on QTouch Analyzer and check that a detect touch can occur BEFORE touching the sensor (i.e. the sensor Touch Delta value becomes higher than its Detect Threshold).



TIPS You can also check signal, reference and delta values as well as Touch Button state by using the Tabular View instead of the Trace View:



INFO You may in some cases, observe a detect touch even without approaching the finger. This is due to the high gain value which has been configured for that sensor. The highest the gain is, the highest sensitivity the sensor is.

RESULT The proximity effect of the Touch project behaves as expected.

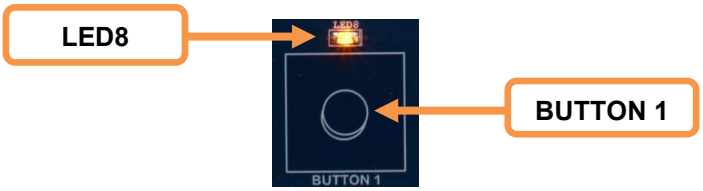
4.3 Conclusion

In this assignment, you have learnt:

- How to use QTouch Analyzer to test a Touch application

5. Assignment 3: Implement a PWM Signal using the SAM D21 Timer/Counters for Control Application Peripheral (TCC)

In this assignment, a Pulse-Width Modulation (PWM) signal will be generated using the SAM D21 Timer/Counters for Control Application (TCC) to light the LED8 of QT1 Xplained Pro extension board:



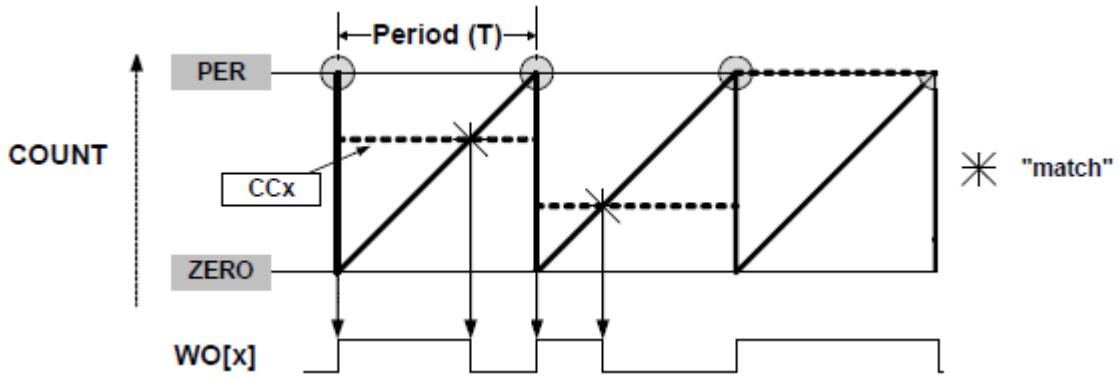
5.1 Introduction to Timer/Counters for Control Application (TCC)

The TCC will be used in Single-Slope PWM mode to generate the required PWM signal and output it on a dedicated Waveform Output pin (WO[x]).

For single-slope PWM generation mode, the period time is controlled by the Period register (PER), while the Compare/Capture registers (CCx) control the duty cycle of the waveform.

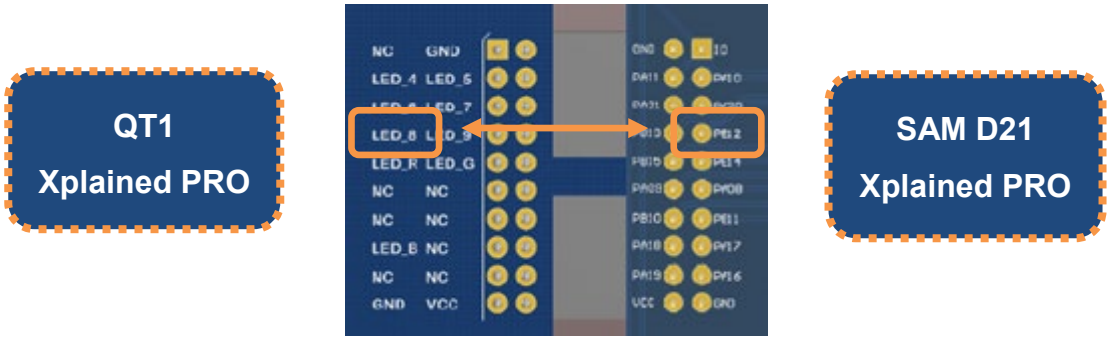
When up-counting (default configuration), WO[x] pin is:

- Set at start or compare match between the counter (COUNT) and period (PER) registers' values
- Cleared on compare match between the counter (COUNT) and Compare/Capture (CCx) registers' values



TO DO Determine which Waveform Output pin to use.

- By flipping the boards, you can check that the LED8 of the QT1 Xplained Pro extension board is connected to the SAM D21 Xplained Pro board pin PB12:



- By looking now at the SAM D21 product datasheet, you can verify that this is the Waveform Output pin 6 (WO[6]) from TCC0 which is multiplexed with PB12

Table 5-1. PORT Function Multiplexing (Continued)

Pin			I/O Pin	Supply	Type	A	B					C	D	E	F	G	H
SAMD21E	SAMD21G	SAMD21J				EIC	REF	ADC	AC	PTC	DAC	SERCOM	SERCOM-ALT	TC/TCC	TCC	COM	AC/GCLK
	19	23	PB10	VDDIO		EXTINT[10]							SERCOM4 /PAD[2]	TC5/WO[0]	TCC0/ WO[4]	I2S/ MCK[1]	GCLK_IO[4]
	20	24	PB11	VDDIO		EXTINT[11]							SERCOM4 /PAD[3]	TC5/WO[1]	TCC0/ WO[5]	I2S/ SCK[1]	GCLK_IO[5]
		25	PB12	VDDIO	I^2C	EXTINT[12]				X[12]		SERCOM4/ PAD[0]		TC4/WO[0]	TCC0/ WO[6]	I2S/ FS[1]	GCLK_IO[6]

INFO SAM D21 Product Datasheet can be retrieved at the following link: http://www.atmel.com/Images/Atmel-42181-SAM-D21_Datasheet.pdf.

RESULT WO[6] pin must be used to output the PWM signal directly to LED8 using TCC0.

5.2 TCC Peripheral Initialization Using ASF Quick Start Guide

The drivers in ASF have quick start guides as part of their API documentation.

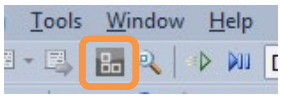
Quick Start guides show and explain, in a step-by-step process, the code and actions needed to set up and use a driver in one or more use cases.

Once a driver such as TCC is added using the ASF Wizard, its Quick Start Guide becomes available in the ASF Explorer view.

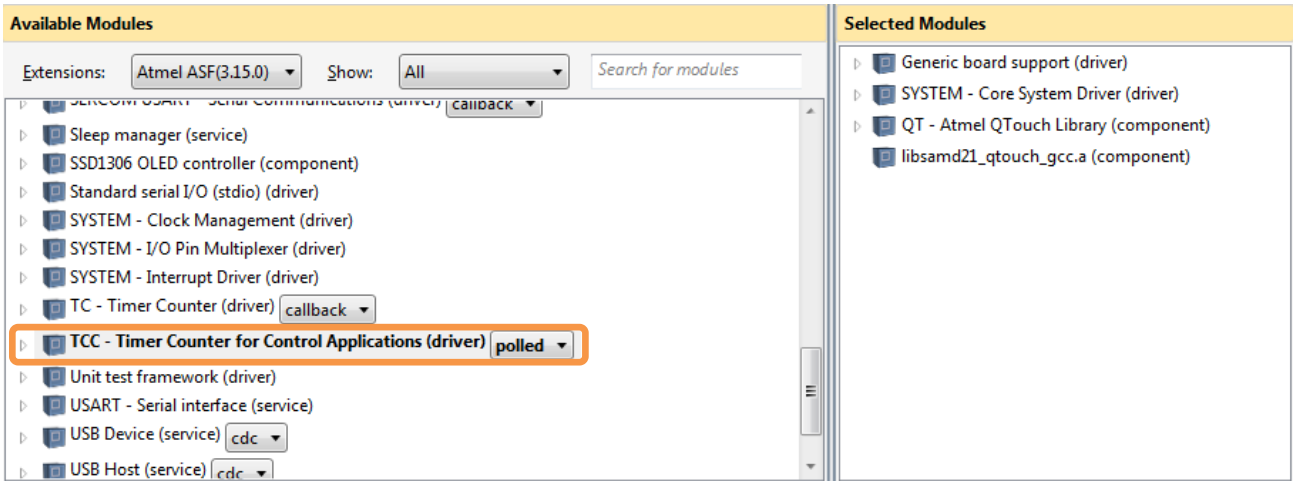
INFO To get started on the ASF API documentation as the Quick Start Guides, please go to: http://asf.atmel.com/docs/latest/get_started.html.

TO DO Add TCC Support using ASF Wizard.

- Click on the ASF Wizard icon or right click on the Hands-on Assignment project > ASF Wizard



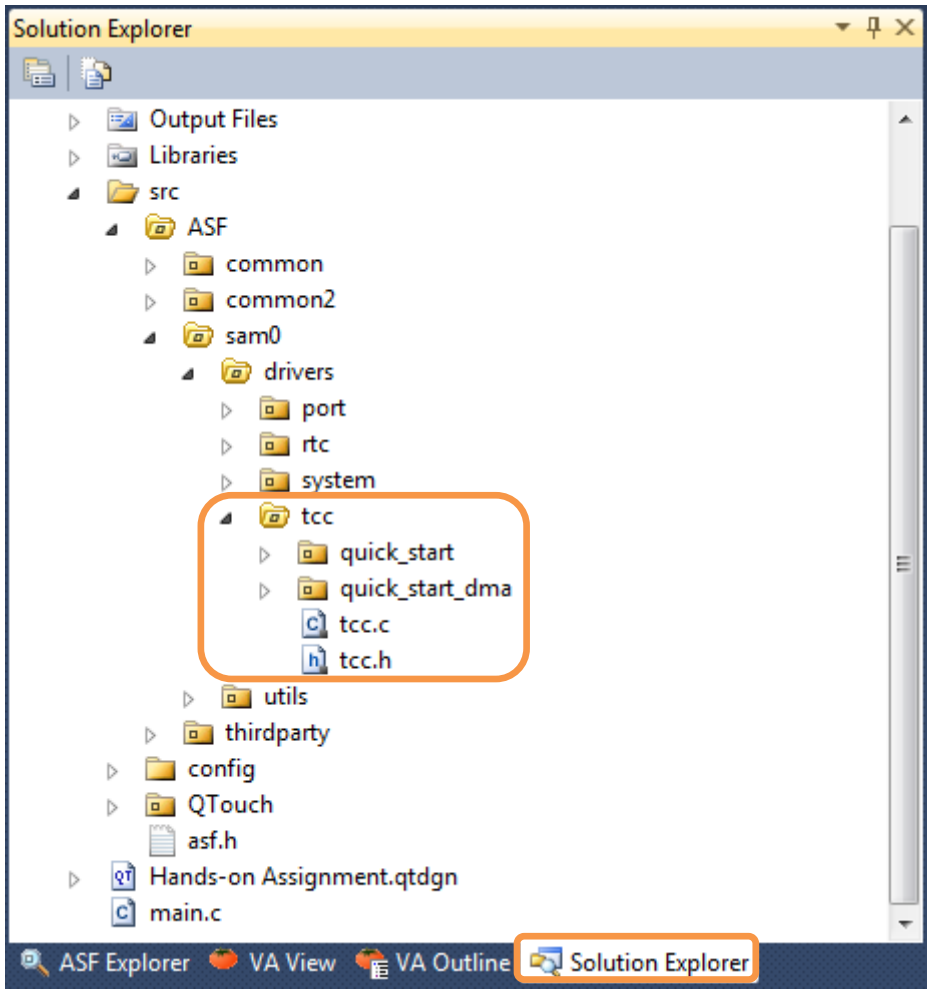
- From latest ASF version available, select the following component and add it to the Selected Modules (Add >> button):
  - TCC – Timer Counter for Control Applications (driver) – **polled**



INFO In this application, TCC polled driver version can be used as no interrupt(s) management is required to generate the PWM signal on the waveform output WO[x] pin.

- Click on Apply button

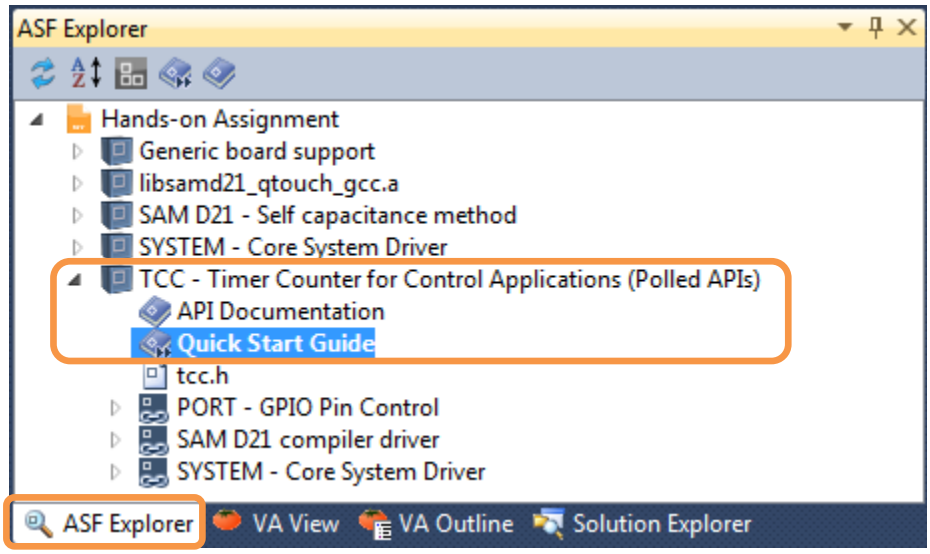
- Click on Solution Explorer tab and check that the TCC driver has been successfully added as shown below:



RESULT TCC driver is added to the project.

We will now implement the TCC initialization by using an example code available in the TCC Quick Start Guide.

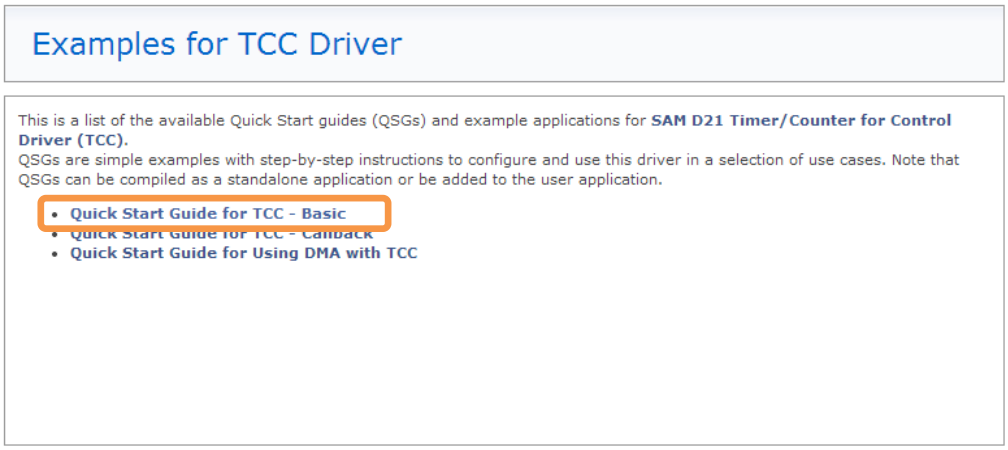
INFO TCC Quick Start Guide can be accessed by selecting the ASF Explorer tab then double clicking on TCC > Quick Start Guide:



WARNING The ASF API documentation is exclusively available on the web.

Several Quick Start Guides are available depending on the TCC driver characteristics: polled, callback, with DMA support...

INFO The TCC Quick Start Guide which relates to the ASF TCC driver in polled mode is called "*Quick Start Guide for TCC – Basic*":



The following code that will be implemented can be found in the sub section of the Quick Start Guide called Workflow.

That sub section provides a detailed step by step guide to initialize the TCC in a “polled mode” use:

Workflow

1. Create a module software instance structure for the TCC module to store the TCC driver state while it is in use.

```
struct tcc_module tcc_instance;
```

Note
This should never go out of scope as long as the module is in use. In most cases, this should be global.

2. Configure the TCC module.
   a. Create a TCC module configuration struct, which can be filled out to adjust the configuration of a physical TCC peripheral.

```
struct tcc_config config_tcc;
```

   b. Initialize the TCC configuration struct with the module's default values.

```
tcc_get_config_defaults(&config_tcc, CONF_PWM_MODULE);
```

Note
This should always be performed before using the configuration struct to ensure that all values are initialized to known default settings.

   c. Alter the TCC settings to configure the counter width, wave generation mode and the compare channel 0 value.

```
config_tcc.counter.period = 0xFFFF;
config_tcc.compare.wave_generation = TCC_WAVE_GENERATION_SINGLE_SLOPE_PWM;
config_tcc.compare.match[CONF_PWM_CHANNEL] = (0xFFFF / 4);
```

   d. Alter the TCC settings to configure the PWM output on a physical device pin.

```
config_tcc.pins.enable_wave_out_pin[CONF_PWM_OUTPUT] = true;
config_tcc.pins.wave_out_pin[CONF_PWM_OUTPUT]        = CONF_PWM_OUT_PIN;
config_tcc.pins.wave_out_pin_mux[CONF_PWM_OUTPUT]    = CONF_PWM_OUT_MUX;
```

   e. Configure the TCC module with the desired settings.

```
tcc_init(&tcc_instance, CONF_PWM_MODULE, &config_tcc);
```

   f. Enable the TCC module to start the timer and begin PWM signal generation.

```
tcc_enable(&tcc_instance);
```

TO DO TCC Initialization Implementation.

- In the main.c file, create a new function above main function (in red):

```
static void configure_tcc(void)
{
}

int main (void)
{
      system_init();
```

- Create a TCC module instance:

```
struct tcc_module tcc_instance;

static void configure_tcc(void)
{
}
```

- In the `configure_tcc` function, create a TCC module configuration instance and initialize it with the module's default values using `tcc_get_config_defaults` function:

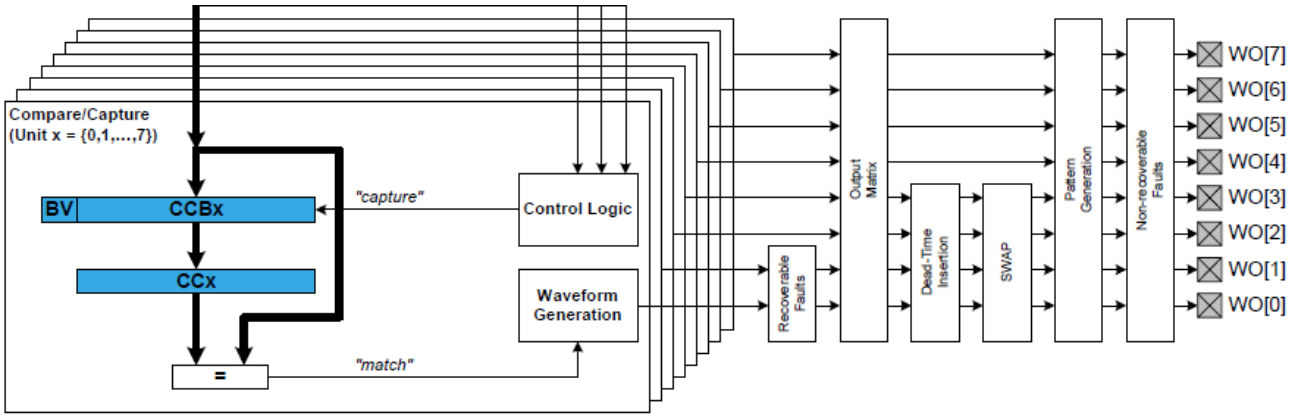
```
struct tcc_config config_tcc;

static void configure_tcc(void)
{
      struct tcc_config config_tcc;
      tcc_get_config_defaults(&config_tcc, TCC0);
}
```

INFO `TCC0` definition corresponds to the TCC0 peripheral base address.

The TCC0 peripheral has four different compare/capture channels. Each channel has a dedicated register (CC0 to CC3).

TCC0 has also an output matrix which can distribute and route out the TCC waveform outputs across the port pins in different configurations, each optimized for different application types.



By default, the Output Matrix Channel Pin Routing Configuration is the following:

Compare and Capture registers	Waveform Output pins
CC0	WO[0], WO[4]
CC1	WO[1], WO[5]
CC2	WO[2], WO[6]
CC3	WO[3], WO[7]

TIPS As a consequence, the Channel 2 Compare and Capture register (CC2) must be used to output the PWM signal on WO[6].

- Update default TCC settings to configure:
    - The Period register (PER) value to get a resolution of 16-bit for the PWM waveform
    - The waveform generation mode to Single-Slope PWM
    - The Channel 2 Compare and Capture register (CC2) value to half the period register value to get a duty cycle of 50%

```
struct tcc_config config_tcc;

static void configure_tcc(void)
{
struct tcc_config config_tcc;
tcc_get_config_defaults(&config_tcc, CONF_PWM_MODULE);

     config_tcc.counter.period = 0xFFFF;
     config_tcc.compare.wave_generation =
TCC_WAVE_GENERATION_SINGLE_SLOPE_PWM;
     config_tcc.compare.match[2] = 0x7FFF;
}
```

INFO The following equation calculates the exact resolution for a single-slope PWM waveform:

$$Resolution = \frac{\log(PER + 1)}{\log(2)}$$

To get a 16-bit resolution, PER must be equal to 65535 ⇔ 0xFFFF.

- Update default TCC settings to:
    - Enable the PWM waveform output pin WO[6]
    - Specify the WO[6] pin output which is PIN_PB12F_TCC0_WO6
    - Specify the WO[6] peripheral multiplexing port which is MUX_PB12F_TCC0_WO6

```
config_tcc.pins.enable_wave_out_pin[6] = true;
config_tcc.pins.wave_out_pin[6]        = PIN_PB12F_TCC0_WO6;
config_tcc.pins.wave_out_pin_mux[6]    = MUX_PB12F_TCC0_WO6;
```

- Configure the TCC module with the desired settings:

```
tcc_init(&tcc_instance, TCC0, &config_tcc);
```

- Enable the TCC module to start the timer and begin PWM signal generation:

```
tcc_enable(&tcc_instance);
```

RESULT TCC0 initialization is completed.

```
struct tcc_module tcc_instance;

static void configure_tcc(void)
{
      struct tcc_config config_tcc;
      tcc_get_config_defaults(&config_tcc, TCC0);

      config_tcc.counter.period = 0xFFFF;
      config_tcc.compare.wave_generation =
TCC_WAVE_GENERATION_SINGLE_SLOPE_PWM;
      config_tcc.compare.match[2] = 0x7FFF;

      config_tcc.pins.enable_wave_out_pin[6] = true;
      config_tcc.pins.wave_out_pin[6]        = PIN_PB12F_TCC0_WO6;
      config_tcc.pins.wave_out_pin_mux[6]    = MUX_PB12F_TCC0_WO6;

      tcc_init(&tcc_instance, TCC0, &config_tcc);
      tcc_enable(&tcc_instance);
}
```

TO DO Test PWM Signal Implementation.

- Call the `configure_tcc` function in your main function:

```
int main (void)
{
      system_init();

      configure_tcc();
```

INFO In STANDBY sleep mode, all clocks and functions are stopped expect those selected to continue running.

- Disable the entry in STANDBY sleep mode by commenting `system_sleep` function so that the TCC0 clock and so the waveform output WO[6] are not disabled

```
while (1) {
      /**
       * Goto STANDBY sleep mode, unless woken by timer or PTC
interrupt.
       */
      //system_sleep();
```

- Build the solution and ensure you get no errors: 
- Program the application by clicking on the Start Without Debugging icon: 
- Press RESET button on the SAM D21 Xplained Pro

RESULT You can see the LED8 lit.

5.3 Conclusion

In this assignment, you have learnt:

- How to configure the TCC to generate a PWM signal
- How to configure the TCC to output a PWM signal on a dedicated Waveform Output pin WO[x]

6. Assignment 4: Use QTouch Button to Control LED Brightness

In this assignment, the PWM duty cycle of waveform output WO[6] signal will be configured to control the LED8 brightness in relation to the BUTTON 1 Touch Delta value.

Here are the steps to follow in order to complete this assignment:

- Use QTouch Library API to read BUTTON 1 Touch Delta value
- Update Waveform Output WO[6] duty cycle in relation to BUTTON 1 Touch Delta value

6.1 Reading and using Button Touch Delta Value

We will use the API function called `touch_selfcap_sensor_get_delta` to retrieve the delta value corresponding to BUTTON 1.

Here is the API function definition for Self Capacitance technology:

```
touch_ret_t touch_selfcap_sensor_get_delta(
            sensor_id_t sensor_id,
            touch_delta_t *p_delta);
```

Where:

- `sensor_id` is the sensor ID for which delta value is being retrieved
- `p_delta` is the pointer to the delta variable to be updated by the Touch Library

INFO This API function is defined in `touch_api_SAMD.h` from QTouch folder.

The ON/OFF state of a button as the slider/rotor position, are touch status parameters.

These parameters must be read by the application only after the `measurement_done_touch` flag is set.

This flag is part of the touch measure data structure called `touch_measure_data_t`.

INFO A pointer to this data structure is already declared in `touch.c` and initialized by the QTouch library:

```
/* ! Self capacitance method measured data pointer. */
touch_measure_data_t *p_selfcap_measure_data = NULL;
```

So, the following code can be used to quickly implement a read of the Touch Delta value of a specific sensor:

```
/* ! Start Touch Sensor Measurement. */
touch_sensors_measure();

/* Update touch status once measurement complete flag is set. */
if ((p_selfcap_measure_data->measurement_done_touch == 1u))
{
    p_selfcap_measure_data->measurement_done_touch = 0;
    touch_selfcap_sensor_get_delta(sensor_id, p_delta);
}
```

TO DO Implement Button Touch Delta Reading using QTouch Library API.

- Create in main function a `touch_delta_t` instance:

```
int main(void)
{

      touch_delta_t button1_delta;
```

- Under `touch_sensors_measure` function, add the following code to implement the read of the Touch Delta value for BUTTON 1:

```
/* ! Start Touch Sensor Measurement. */
touch_sensors_measure();

/* Update touch status once measurement complete flag is set. */
if ((p_selfcap_measure_data->measurement_done_touch == 1u))
{
      p_selfcap_measure_data->measurement_done_touch = 0;
      touch_selfcap_sensor_get_delta(0, &button1_delta);
}
```

INFO `sensor_id` = 0 as we only have one sensor (BUTTON 1).

RESULT The application is now able to get BUTTON 1 Touch Delta value.

6.2 Using PWM to Control LED Brightness

To control the brightness of an LED, you have to vary the power which is sent to the LED. The more power the LED receives, the brighter it is.

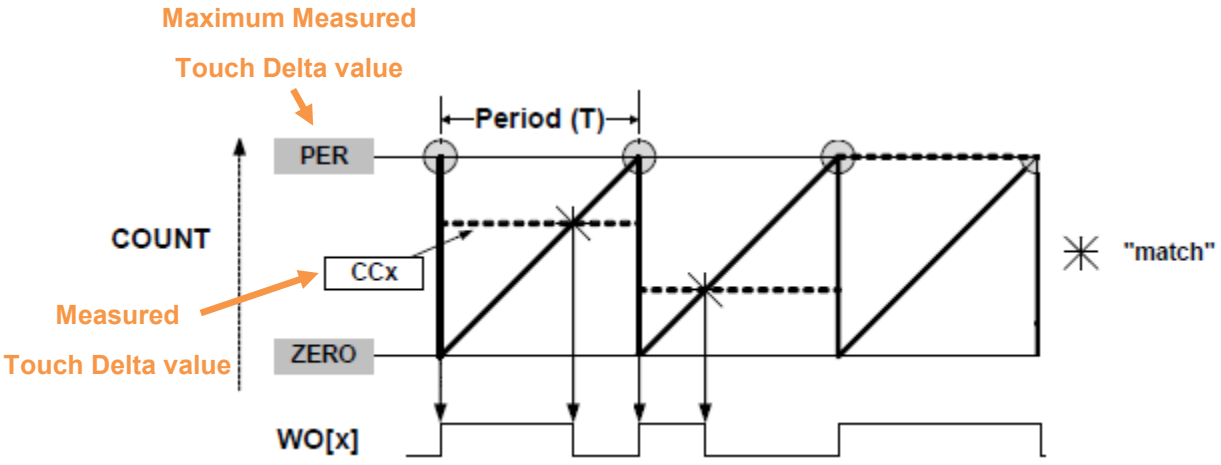
A PWM signal provides the ability to 'simulate' varying levels of power by oscillating the output from the microcontroller in relation to its duty-cycle.

INFO The PWM duty-cycle refers to the total amount of time a pulse is 'on' over the duration of the cycle.

So, LED8 brightness will be controlled by modifying the PWM duty cycle of the Waveform Output pin WO[6].

To do that, the TCC0 period value (PER register) must be modified and aligned to the maximum measured Touch Delta value. This will allow writing directly the measured Touch Delta value in the Channel 2 Compare and Capture register (CC2).

Any Touch Delta value change on the PTC will be then directly seen as a duty-cycle update on the PWM signal:



TO DO Get Maximum Measured Touch Delta value.

- Launch QTouch Analyzer by clicking on the following icon: 
- Press BUTTON 1 on your kit and check using the Tabular View the maximum Touch Delta value

INFO It may happen that the delta value goes slightly below 0 (negative values). This behavior has to be considered to correctly implement the duty cycle update later.

RESULT The maximum measured Touch Delta value will vary depending on how the button is pressed but should be in the range of the one measured below:

Tabular View

Buttons

Button Id	Name	State	Delta	Delta RMS	Channel Id	Signal
0	Button0	ON	11387	0	0	27286

We will now implement the update of the PWM duty cycle in relation to BUTTON 1 Touch Delta value.

The ASF API function called `tcc_set_compare_value` will be used to update the TCC0 Channel 2 Compare and Capture value.

Here is the ASF API function definition:

```
tcc_set_compare_value(
            const struct tcc_module *const module_inst,
            const enum tcc_match_capture_channel channel_index,
            const uint32_t compare)
```

Where:

- `module_inst` is the pointer to the software module instance
- `channel_index` is the index of the compare channel to write to
- `compare` is the new compare value to set

TO DO Update PWM duty cycle in relation to BUTTON 1 Touch Delta value.

- In the `configure_tcc` function, update PER and CC2 registers to the maximum measured Touch Delta value:

```
static void configure_tcc(void)
{
	struct tcc_config config_tcc;
	tcc_get_config_defaults(&config_tcc, CONF_PWM_MODULE);

	config_tcc.counter.period = 12000;
	config_tcc.compare.wave_generation =
TCC_WAVE_GENERATION_SINGLE_SLOPE_PWM;
	config_tcc.compare.match[2] = 12000;
```

INFO The Maximum Touch Delta value is rounded up as different maximum values may be observed depending on the user.

- In the `main` function, update the following code to have the LED8 brightness controlled by the measured Touch Delta value:

```
/* ! Start Touch Sensor Measurement. */
touch_sensors_measure();

/* Update touch status once measurement complete flag is set. */
if ((p_selfcap_measure_data->measurement_done_touch == 1u))
{
	p_selfcap_measure_data->measurement_done_touch = 0;
	touch_selfcap_sensor_get_delta(0, &button1_delta);

	if ((button1_delta < 0) || (button1_delta > 12000))
		button1_delta = 0;
	tcc_set_compare_value(&tcc_instance, 2, button1_delta);
}
```

TIPS `channel_index` = 2 as CC2 register is used to generate the PWM signal.

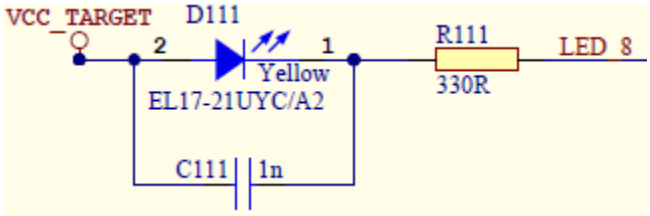
INFO Out-of-bounds values are handled.

- Build the solution and ensure you get no errors: 
- Program the application by clicking on the Start Without Debugging icon: 
- Press RESET button on the SAM D21 Xplained Pro

RESULT LED8 brightness varies when approaching the finger to BUTTON 1 but the proximity effect is not in line with the application expectation as LED8 is ON when no touch is detected and its brightness decreases when approaching the finger.

The application needs to have LED8 switched off when there is no touch or proximity touch.

On the QT1 Xplained extension board, LED8 is turned off when PIN_PB12 is set:



So, the solution is to invert the PWM waveform output signal thanks to the waveform output polarity feature of the SAM D21 TCC (POLx bit):

POLx	Waveform Generation Output Update	
	Set	Clear
0	Timer/counter matches TOP	Timer/counter matches CCx
1	Timer/counter matches CCx	Timer/counter matches TOP

INFO

POLx = 0 by default where x is the TCC Channel Number.

If POLx = '1', waveform output pin becomes:

- Cleared at start or on compare match between the counter (COUNT) and period (PER) registers' values (TOP)
- Set on compare match between the counter (COUNT) and Compare/Capture (CCx) registers' values

TO DO Invert PWM signal by modifying the Waveform Output Polarity.

- In the `configure_tcc` function, update TCC settings to:
  - Have LED8 switched off when there is no touch or proximity touch
  - Invert the Waveform Output WO[6] polarity (TCC0 Channel 2)

```
static void configure_tcc(void)
{
	struct tcc_config config_tcc;
	tcc_get_config_defaults(&config_tcc, CONF_PWM_MODULE);

	config_tcc.counter.period = 12000;
	config_tcc.compare.wave_generation =
TCC_WAVE_GENERATION_SINGLE_SLOPE_PWM;
	config_tcc.compare.match[2] = 0;
	config_tcc.wave.wave_polarity[2] = TCC_WAVE_POLARITY_1;
```

- Build the solution and ensure you get no errors: 
- Program the application by clicking on the Start Without Debugging icon: 
- Press RESET button on the SAM D21 Xplained Pro

RESULT LED8 brightness finally increases when approaching the finger to BUTTON 1.

6.3 Conclusion

In this assignment, you have learnt:

- How to use the QTouch Library API to read a Button Touch Delta value
- How to update a TCC Waveform Output PWM duty cycle in relation to a Button Touch Delta value

7. Conclusion

This hands-on demonstrated the ease of use of the different Atmel QTouch technology tools as well as the Atmel Software Framework APIs and their full integration in the Atmel Studio IDE.

The following topics have been covered:

- Touch project creation using Atmel QTouch Composer
- Touch project analysis using Atmel QTouch Analyzer
- Touch project implementation using the Atmel QTouch Library and the Atmel Software Framework APIs

You have seen how Atmel Studio, the Atmel Software Framework API and the Atmel QTouch tools make it easy to add capacitive touch sensing to your project.

8. Revision History

Doc. Rev.	Date	Comments
42327A	07/2014	Initial document release.

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Bldg.
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN
Tel: (+81)(3) 6417-0300
Fax: (+81)(3) 6417-0370