AVR1511: QT600-ATxmega128A1 Training Guide

8-bit AVR® Microcontrollers

Application Note

Features

- **Knowledge level: Intermediate**
- **PC platform: Windows® 2000, XP, Vista®, Win7®**
- **Hardware requirements:**
  - **Atmel QT600 (entire kit)**
  - **USB cables**
- **Software requirements:**
  - **Atmel® QTouch Studio™ 4.3.0 or later**
  - **Atmel QTouch Library™ 4.3 or later**
  - **Atmel AVR Studio® 4 (latest version)**
  - **Latest version of WinAVR**
- **Estimated time to complete all tasks in this guide: 1 day**

1 Introduction

The purpose of this training is to get familiar with the Atmel QTouch Suite™ for developing and debugging any Atmel touch application. It includes four core solutions: Atmel QTouch Studio, Atmel QT600 development kit, Atmel QTouch Library, and Atmel AVR Studio.

Atmel QTouch Studio - Touch analyzer

QTouch Studio is the front-end software used to display and evaluate the data reported by the QT600 development kit.

Atmel QTouch Library - Touch software library for AVR

The QTouch Library is a software library for developing touch applications on standard Atmel AVR® microcontrollers. Customers can link the library into their firmware in order to integrate touch-sensing capability into their projects.

Atmel AVR Studio - Debugger and programmer

AVR Studio is a professional integrated development environment (IDE) for writing, simulating, and debugging applications for AVR microcontrollers. It also comprises the programming interface for all AVR tools.

Atmel QT600 - Touch hardware kit and example code

The QT600 is a complete touch development kit for buttons, sliders, and wheels. This advanced development platform allows designers to experiment with Atmel touch technology, and provides the easiest way to analyze and validate touch products. It supports both Atmel QTouch® and Atmel QMatrix acquisition methods. It comes with one USB-powered interface board, three MCU boards representing the Atmel tinyAVR®, Atmel megaAVR®, and Atmel AVR XMEGA® families of microcontrollers, and three touch sensor boards supporting up to 64 channels.

Rev. 8350A-AVR-01/11

The Atmel QT600 MCU boards can be connected to the Atmel STK600 for easy access to unused I/O pins. STK600, however, is not needed for the QT600 kit to function properly. QT600 is fully supported by Atmel QTouch Studio, Atmel QTouch Library, and Atmel AVR Studio, and together these tools form the Atmel QTouch Suite.

This hands-on training consists of a number of tasks:

1. Connect everything and assure that your hardware is working properly
2. Set up a project from scratch, select and include the library, add the touch files and assembler files
3. Create a virtual kit in design mode and use the pin configuration wizard
4. Configure the global options and set/enable the sensors
5. Add debug files and code to use the debug interface in order to take advantage of the live debugging features of QTouch Studio
6. Use the Analysis mode and a review brief description of the various parameters
7. Use the design validation wizard
8. Log data and use it for further analysis and debugging purposes

Basics for AVR Studio, WinAVR, or tools use are not covered.

2 Document overview

The hands-on training is split into several different assignments. Each assignment is further divided into individual sections to simplify understanding.

Throughout this document you will find some special icons. These icons are used to identify different sections of assignments and ease complexity.

Information
Delivers contextual information about a specific topic.

Tip
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.

3 Requirements

3.1 Software

Make sure you install the latest versions of:

- Atmel QTouch Studio 4.3.x (www.atmel.com/avrqtouchstudio)
- Atmel QTouch Library 4.3 (www.atmel.com/qtouchlib)
    - QTouch Library User Guide
    - QTouch Library Selection Guide
- Atmel AVR Studio 4 (www.atmel.com/avrstudio)
    - Install all matching AVR Studio service packs in sequential order (SP1, SP2, SP3…..):
        1. AVR Studio 4.18Build684
        2. AVR Studio 4.18Build692(SP1)
        3. AVR Studio 4.18Build700(SP2-Latest)
- WinAVR GCC

3.2 Hardware

In this hands-on training we are going to use:

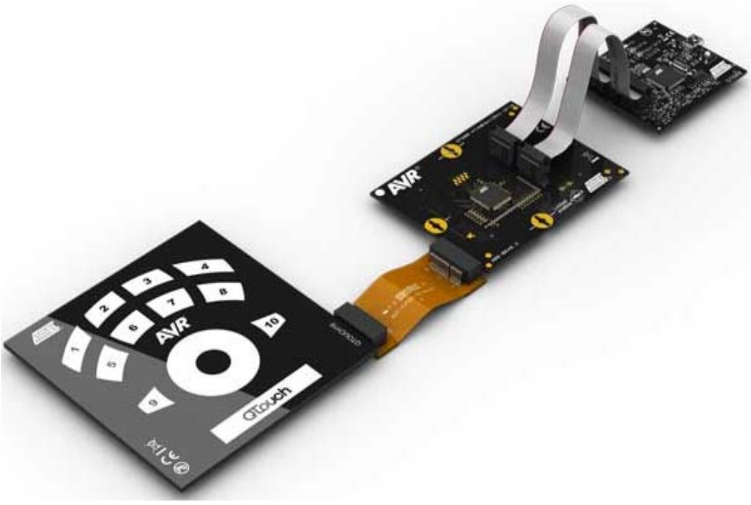
- QT600 interface board
- QT600-ATXMEGA128A1-QT16 MCU board
- QTOUCH16 16-channel touch sensor board

3.3 Atmel QT600 system description

The QT600 system is based on three boards connected together.

- QT600 interface board
- QT600 MCU board
- Touch sensor board

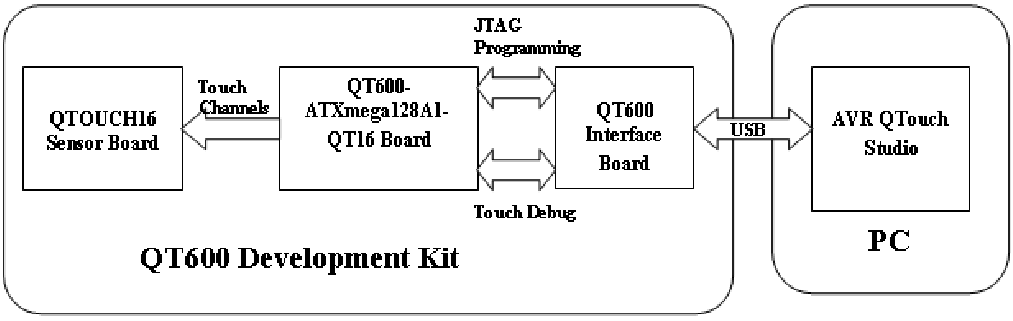
Figure 3-1. QT600 system.



The MCU board and touch sensor boards together comprise the user touch system. The QT600 interface board is used to stream live touch data from the AVR MCU mounted on the MCU board. QTouch Studio is used as the PC front end to visualize the touch data.

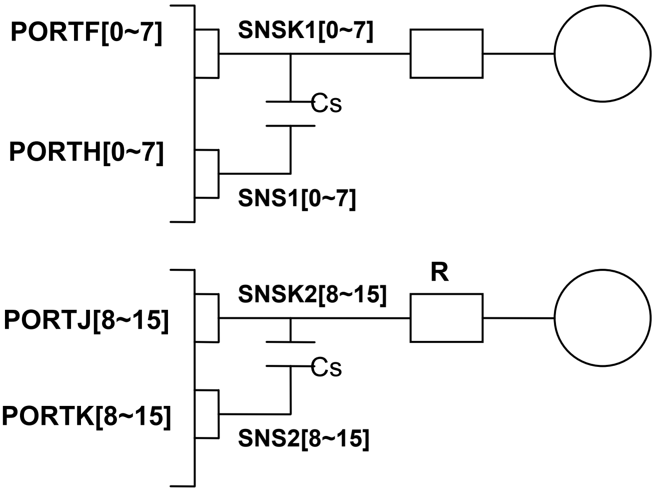
3.3.1 Block diagram

Figure 3-2. Block diagram.



3.3.2 Pin schematic

Figure 3-3. Pin schematic.



4 Assignment

For the following tasks, please note:

- Start with all the cables disconnected (PC to QT600, QT600 to MCU boards…)
- Ensure you have installed the latest versions of software listed in section 1
- From the same web page where you found this application note, download and expand the associated zip file (disk icon). Project folders for each task will be relative to where you expanded the zip file {AVR1511}. File paths will be relative to the expanded task folder where each project should be created. The zip includes all touch library files required for these tasks

4.1 Task 1: Connect the tools and run a test application

4.1.1 Introduction

This task is designed to make sure all the tools are connected and working as intended. No code writing is needed for this task. You can load the elf file QT600_ATxmega128a1_qt16_gnu.elf present at the location:

..\..\Exercises\QTouch\Tasks\Task1\

This can be programmed directly into the flash of the MCU. To use the Atmel QT600 as a programmer: Press the button on the interface board and keep it pressed until the status LED changes from orange to red (~5s). This will switch the QT600 from Touch Debug mode to Programmer mode.

The Touch Data LED on the QT600 indicates mode:

• Touch Data LED green:	Touch Debug mode (default mode after power-up)
• Touch Data LED off:	Programmer mode

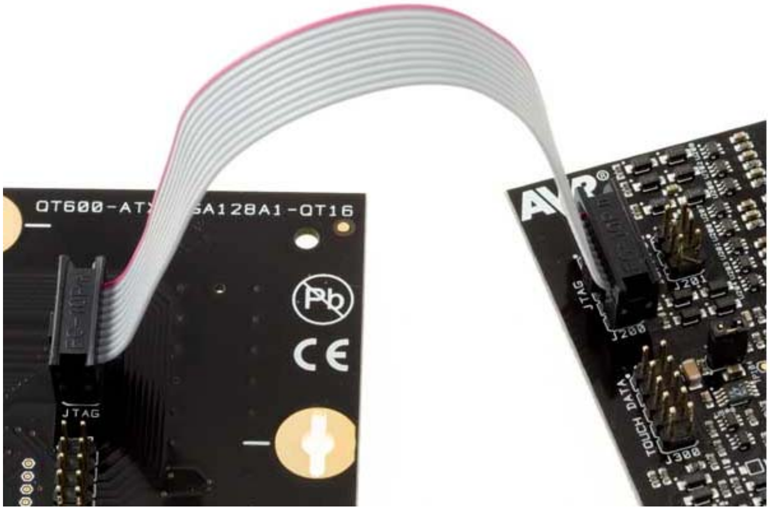
Make sure that the Touch Data LED is off before attempting to connect to the QT600 from the AVR Studio programming dialog. Use Atmel AVR Studio 4.18 or later.

Refer to the QT600 User Guide for more details. It can be accessed by launching QTouch Studio and then selecting Help→Contents→QTouch Studio→Supported Kits→QT600 User Guide.

4.1.2 Connecting the hardware

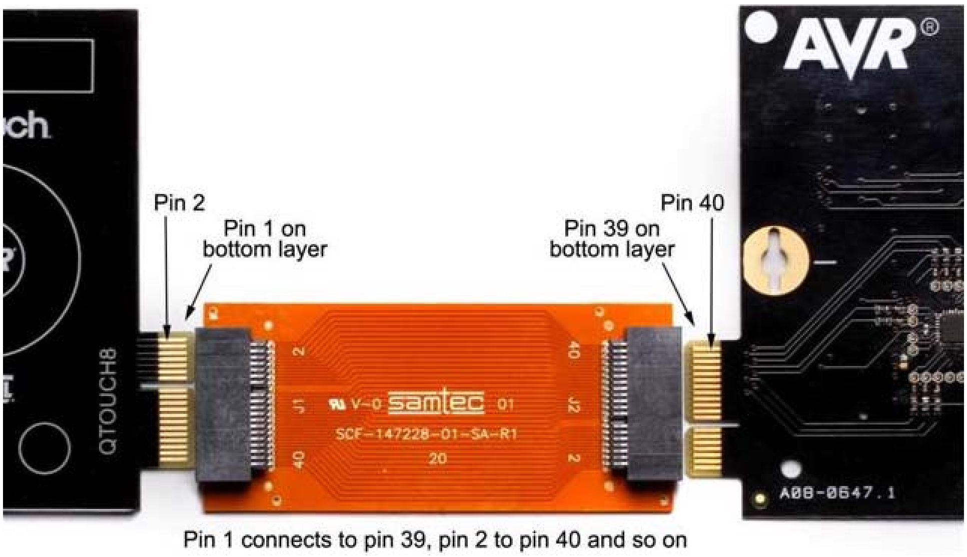
1. Connect the USB cable to the QT600 interface board
2. Make sure the VTG header is mounted on the QT600
3. Press the button on the QT600 to switch to Programmer mode (~5s)
4. From the connect dialog in AVR Studio, select QT600 and press connect
5. Go to the Hardware Settings tab and set the target voltage as per the device datasheet. Minimum voltage is 1.8V
6. Disconnect the USB plug
7. Reconnect the USB plug
8. Again press the button on the QT600 to switch to Programmer mode
9. Connect a 10-wire flat cable between the JTAG header on the QT600 and the QT600-ATXmega128A1-QT16 MCU board

Figure 4-1. JTAG connection.



10. Program the application code (QT600_ATxmega128a1_qt16_gnu.elf). For detailed instructions on how to do this, see the help section in AVR Studio
11. Disconnect the USB plug from the Atmel QT600
12. Remove the flat cable between the programming headers, and mount a 10-wire cable between the Touch Data headers
13. Connect the touch panel
14. For Atmel ATxmega128A1: Use the QTOUCH16 Panel

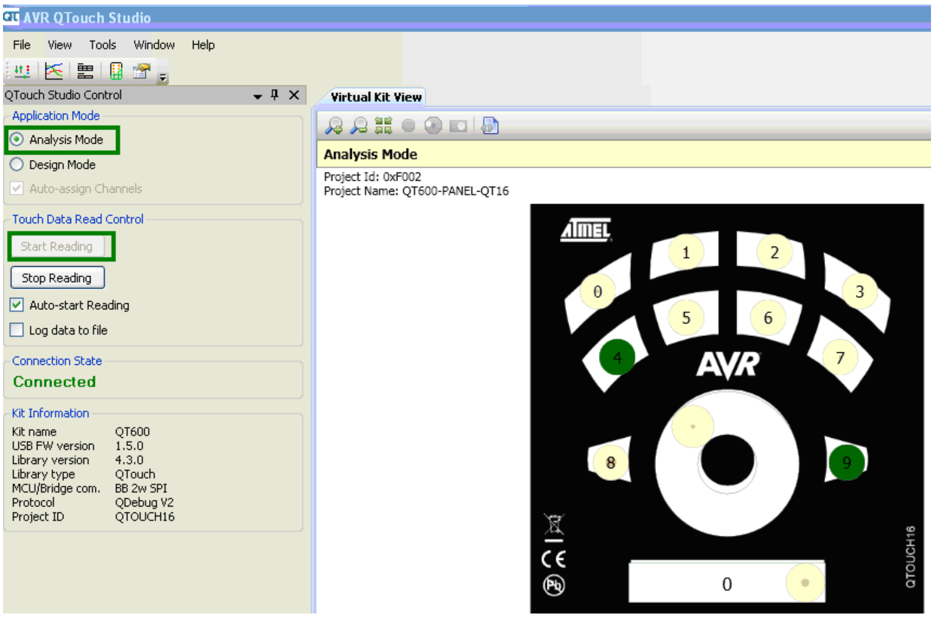
Figure 4-2. Sensor board connections.



15. Launch Atmel QTouch Studio
16. Plug the USB cable in to the QT600. QTouch Studio should now automatically connect to the kit
17. Select Analysis mode, and then press the Start Reading button

You should now be able to view the touch data signals and the state of each sensor. Refer the Figure 4-3.

Figure 4-3. Virtual kit view.



AVR QTouch Studio

File View Tools Window Help

QTouch Studio Control

Application Mode

● Analysis Mode

○ Design Mode

☑ Auto-assign Channels

Touch Data Read Control

Start Reading

Stop Reading

☑ Auto-start Reading

☐ Log data to file

Connection State

Connected

Kit Information

Kit name	QT600
USB FW version	1.5.0
Library version	4.3.0
Library type	QTouch
MCU/Bridge com.	BB 2w SPI
Protocol	QDebug V2
Project ID	QTOUCH16

Virtual Kit View

Analysis Mode

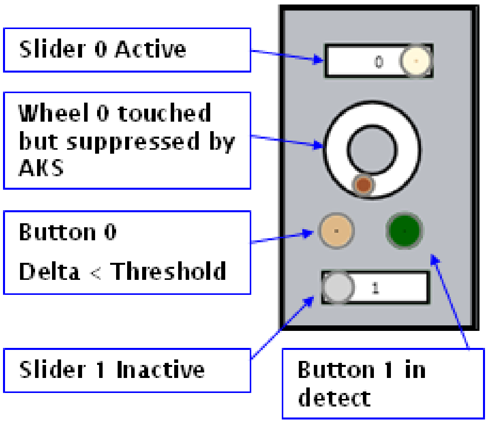
Project Id: 0xF002
Project Name: QT600-PANEL-QT16

In Analysis mode, sensors displayed in a light grey color are inactive, and the kit will not respond to touches to these sensors. Sensors displayed in a light brown color are active sensors, and the kit will respond to touches to these sensors.

Whenever a touch to an active sensor is detected, a green filled circle can be seen indicating where the user touches the sensor.

When a touch to the active sensor is no longer detected, the color of the sensor will go back to light brown again.

Figure 4-4. Sensor states.



4.2 Task 2: Set up a new project, select and include library, add touch files and assembler files

4.2.1 Introduction

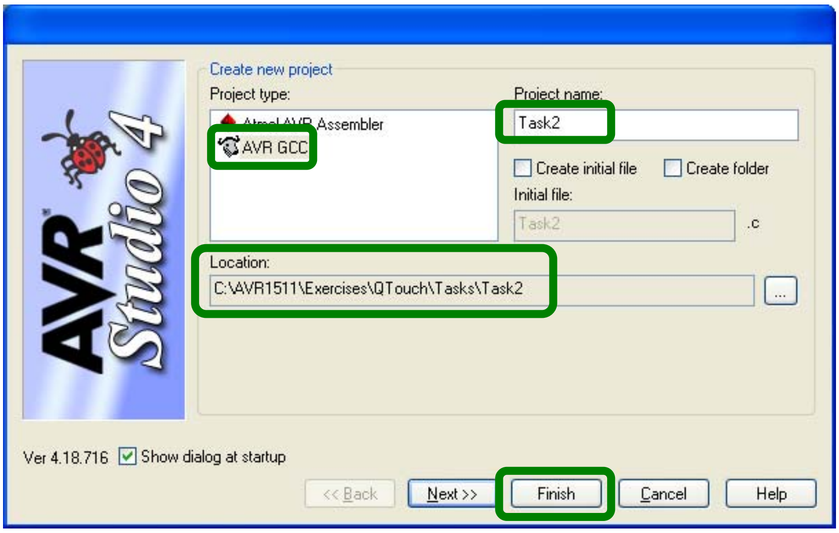
In this task, we will create a new project. We will add the necessary libraries, touch files and assembler files.

4.2.2 Setting up the code

Create a new project

1. Open AVR Studio and create a new project. (If AVR Studio is already running, then use Project→New project)
2. Select project type AVR GCC, set project name to Task2, uncheck Create initial file, uncheck Create folder, set Location, and then click Finish

Figure 4-5. New project.



Create new project

Project type: Atmel AVR Assembler / AVR GCC

Project name: Task2

□ Create initial file □ Create folder

Initial file: Task2 .c

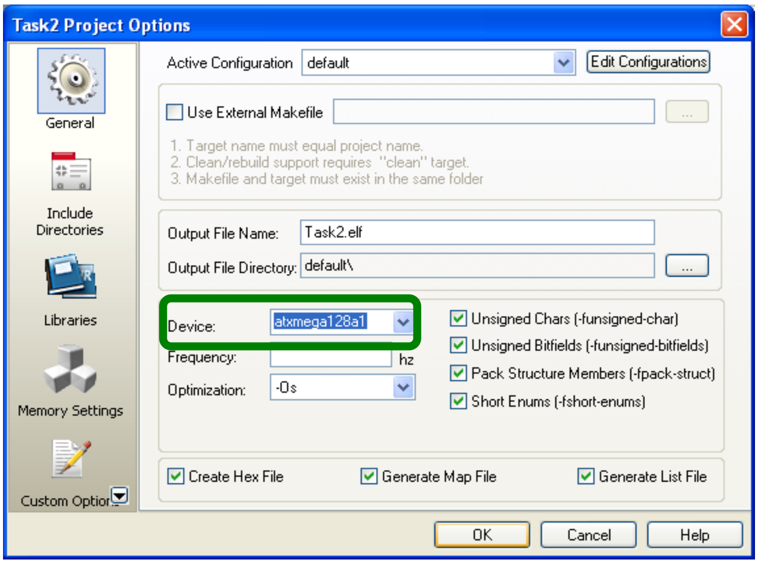
Location: C:\AVR1511\Exercises\QTouch\Tasks\Task2

Ver 4.18.716 ☑ Show dialog at startup

Set the MCU type

1. Select Project→Configuration options→General and set Device to ATxmega128A1, as shown in Figure 4-6

Figure 4-6. Selecting the device.



Task2 Project Options

Active Configuration: default

□ Use External Makefile

1. Target name must equal project name.
2. Clean/rebuild support requires "clean" target.
3. Makefile and target must exist in the same folder

Output File Name: Task2.elf

Output File Directory: default\

Device: atxmega128a1

Frequency: hz

Optimization: -Os

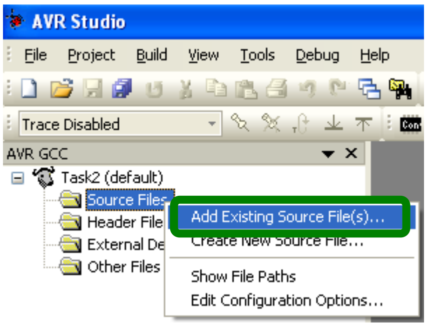
☑ Unsigned Chars (-funsigned-char)
☑ Unsigned Bitfields (-funsigned-bitfields)
☑ Pack Structure Members (-fpack-struct)
☑ Short Enums (-fshort-enums)

☑ Create Hex File ☑ Generate Map File ☑ Generate List File

Add the main.c file

1. Right click on Source Files and add the existing main.c source file to the project

Figure 4-7. Including the main.c.



AVR Studio
File Project Build View Tools Debug Help
Trace Disabled
AVR GCC
Task2 (default)
Source Files
Header Files
External De
Other Files
Add Existing Source File(s)...
Create New Source File...
Show File Paths
Edit Configuration Options...

Test compile to ensure all tools are installed OK

1. Select Build→Rebuild All, and ensure that the build completes successfully with no errors or warnings

Select the library file

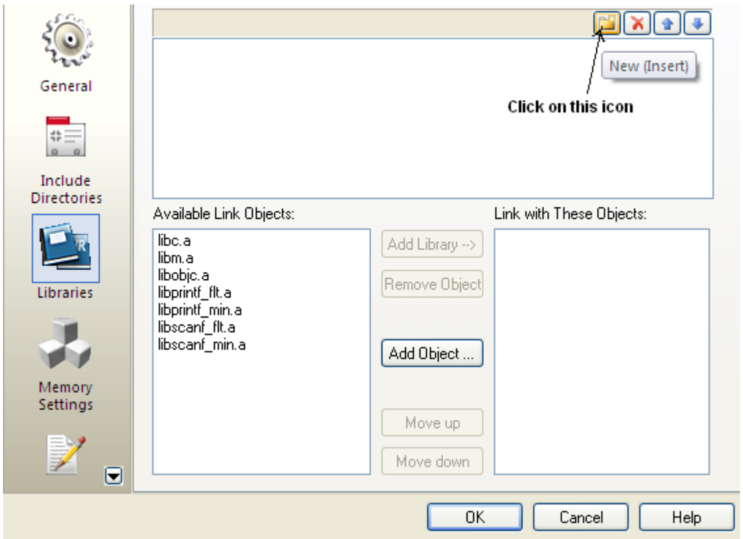
1. Open the Library_Selection_Guide.xls file in the Task2 folder and click on the QTouch Tab (we are using Atmel QTouch). Similarly, for Atmel QMatrix, click on the tab for QMatrix
2. Select the technology as QTouch, MCU family as Atmel XMEGA, MCU type as 8 bit, MCU as Atmel ATxmega128A1, the Tool chain as GCC, Ports available for QTouch as A,B,C,D,E,F, Maximum number of channels as 16, Maximum number of rotors or sliders as 2 and. As you can see, it shows the appropriate library file to be included and example project to be used
3. Note: Because we are building a new project, note down which library file is to be included

Here, the library file to be included is libavrxmega7g1-16qt-k-4rs.a.

Include the library file

1. Open the configuration tab and click on libraries
2. Click on the folder icon at the top right corner. Refer Figure 4-8

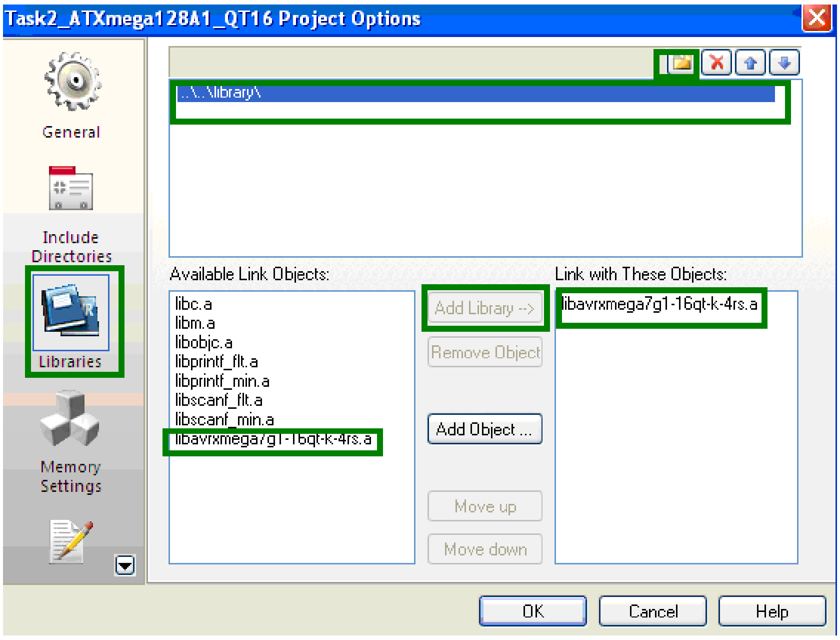
Figure 4-8. Including the library file.



libc.a
libm.a
libobjc.a
libprintf_flt.a
libprintf_min.a
libscanf_flt.a
libscanf_min.a

3. Enter the appropriate path and include this library
4. The library file can be found at ..\..\library_files\
5. Now select libavrxmega7g1-16qt-k-4rs.a and click on Add Library, as in Figure 4-9

Figure 4-9. Library file included.



Task2_ATXmega128A1_QT16 Project Options

..\..\library\

Available Link Objects:

libc.a
libm.a
libobjc.a
libprintf_flt.a
libprintf_min.a
libscanf_flt.a
libscanf_min.a
libavrxmega7g1-16qt-k-4rs.a

Link with These Objects:

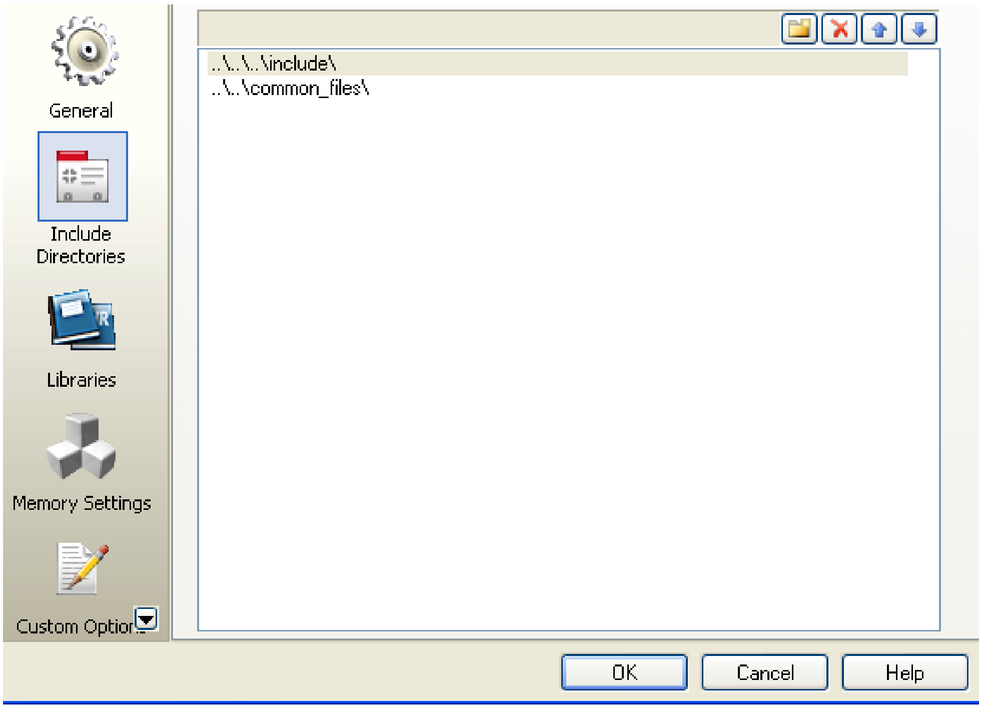
libavrxmega7g1-16qt-k-4rs.a

Include touch header files

1. To use the touch keys, first we need to add #include “touch_api.h” to our list of include files. This will make the Atmel QTouch Library API available
2. Add it under the following comment in the main.c file:
   /* now include touch api.h with the localization defined above */
3. Open Project Options, click on Include Directories, and enter the path where the touch.api file is found
4. This file is found in the include folder under Exercises
5. Similarly, click on Include Directories and add the path for the touch_qt_config.h file. This file is found in the common_files folder
6. The touch_qt_config.h file is where all the technology options are configured

Enter the relative path, as shown in the Figure 4-10 (similar to how you added the library).

Figure 4-10. Included directories.



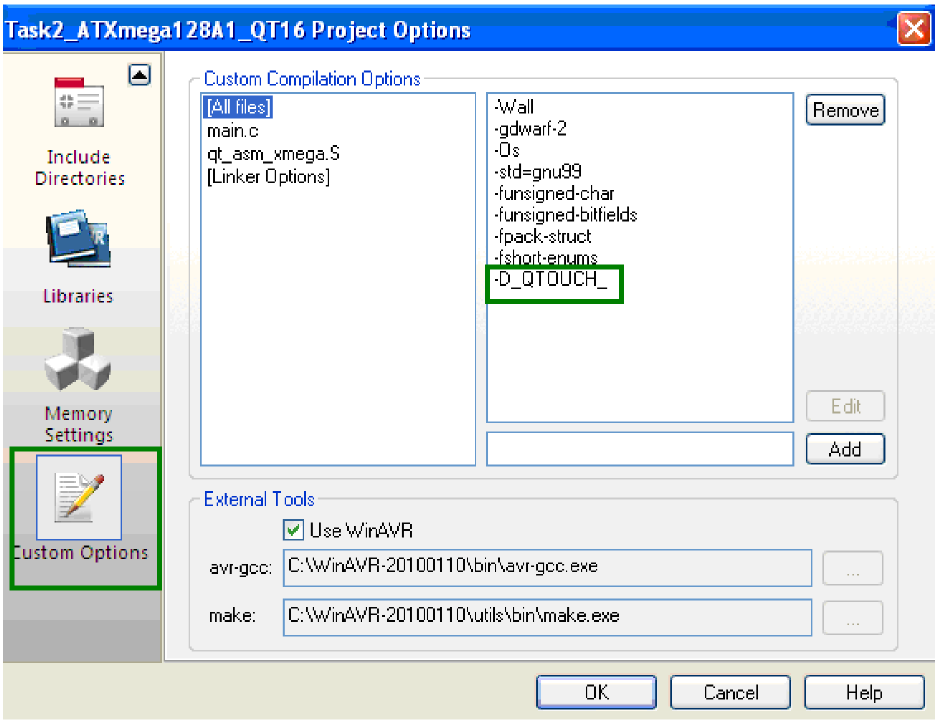
```
..\..\..\include\
..\..\common_files\
```

Custom Options

Open Project Options and click on Custom Options.

Add -D_QTOUCH_, as shown in Figure 4-11

Figure 4-11. Custom compilation options.



Task2_ATXmega128A1_QT16 Project Options

Custom Compilation Options

[All files]
main.c
qt_asm_xmega.S
[Linker Options]

```
-Wall
-gdwarf-2
-Os
-std=gnu99
-funsigned-char
-funsigned-bitfields
-fpack-struct
-fshort-enums
-D_QTOUCH_
```

External Tools

☑ Use WinAVR

avr-gcc: C:\WinAVR-20100110\bin\avr-gcc.exe

make: C:\WinAVR-20100110\utils\bin\make.exe

This tells the library that we are going to use Atmel QTouch measurement technology (as opposed to Atmel QMatrix).

Add the assembler file

1. Add the source file, qt_asm_xmega.S
2. The qt_asm_xmega.S file can be found in the common_files folder under the QTouch folder. qt_asm_xmega.S is a special assembly source file that contains optimization routines that need to be compiled with our project
3. In the AVR GCC tab in Atmel AVR Studio, right click on the project, select Add Existing File, and then select the assembly source file. Alternatively, you can drag and drop the source file onto the project name

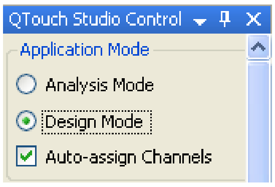
Now save and build the project, and if all the steps mentioned above have been followed, you should be able to build successfully with no errors.

4.3 Task 3: Create a virtual kit in design mode and use the pin configuration wizard

Create a virtual kit

1. Launch Atmel QTouch Studio. Select Design mode
2. Design mode shall provide editable virtual kit. It allows users to add buttons, wheels, sliders, and kit background image and select kit technology (QTouch or QMatrix)
3. Check the Auto-assign Channels option

Figure 4-12. Design mode.



QTouch Studio Control

Application Mode

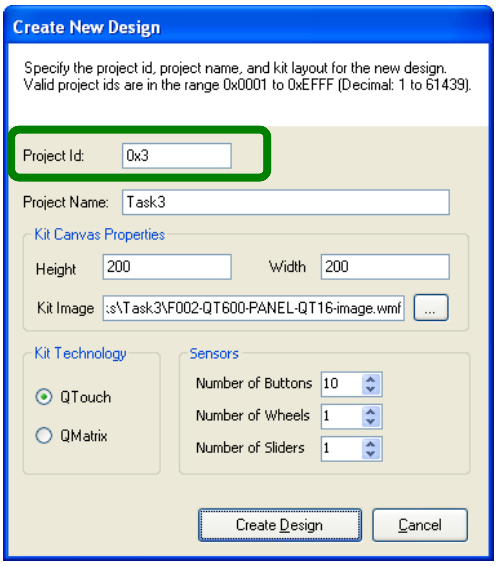
○ Analysis Mode

● Design Mode

☑ Auto-assign Channels

4. On the File menu, click New Design. In the Create New Design window, specify the project ID, project name, kit canvas properties like Height and Width (millimeters) and background (kit) image, and kit technology, along with a valid number of sensors for the new project
5. For this hands-on training, select QTouch as the kit technology. Under Sensors, select the number of buttons as 10, wheels as 1, and sliders as 1
6. For the background image, select the image provided in the Task3 folder named F002-QT600-PANEL-QT16-image
7. Assign channel numbers 0-9 to buttons 0-9, respectively. Assign channel numbers 10-12 to the slider, and channels 13-15 to the wheel

Figure 4-13. New design.



Create New Design

Specify the project id, project name, and kit layout for the new design.
Valid project ids are in the range 0x0001 to 0xEFFF (Decimal: 1 to 61439).

Project Id: 0x3

Project Name: Task3

Kit Canvas Properties

Height 200 Width 200

Kit Image :s\Task3\F002-QT600-PANEL-QT16-image.wmf

Kit Technology

● QTouch

○ QMatrix

Sensors

Number of Buttons 10

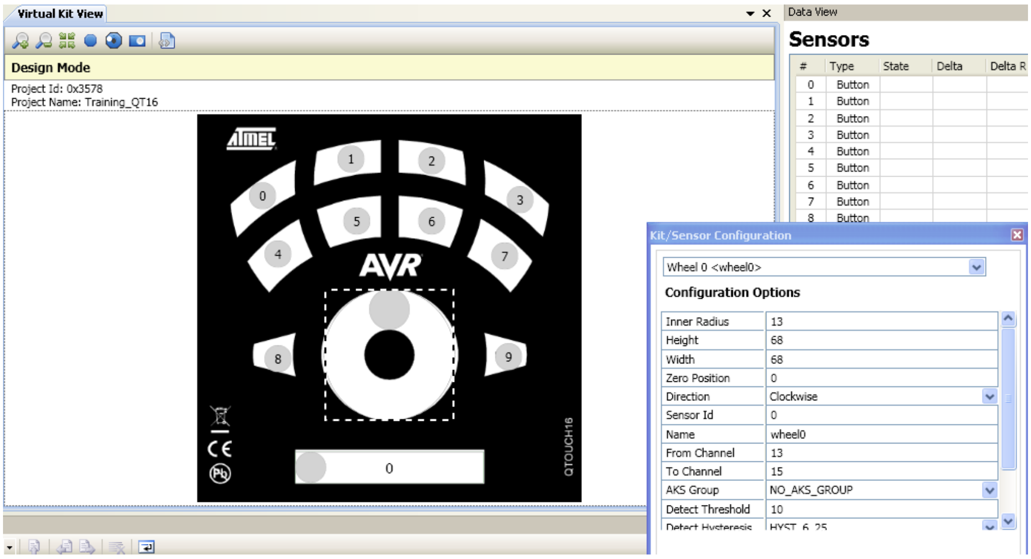
Number of Wheels 1

Number of Sliders 1

Note down the project ID

8. Click on each sensor and adjust the height and width accordingly in the kit/sensor configuration window. Refer to Figure 4-14

Figure 4-14. Design mode.



Virtual Kit View

Design Mode

Project Id: 0x3578
Project Name: Training_QT16

Data View

Sensors

#	Type	State	Delta	Delta R
0	Button			
1	Button			
2	Button			
3	Button			
4	Button			
5	Button			
6	Button			
7	Button			
8	Button			

Kit/Sensor Configuration

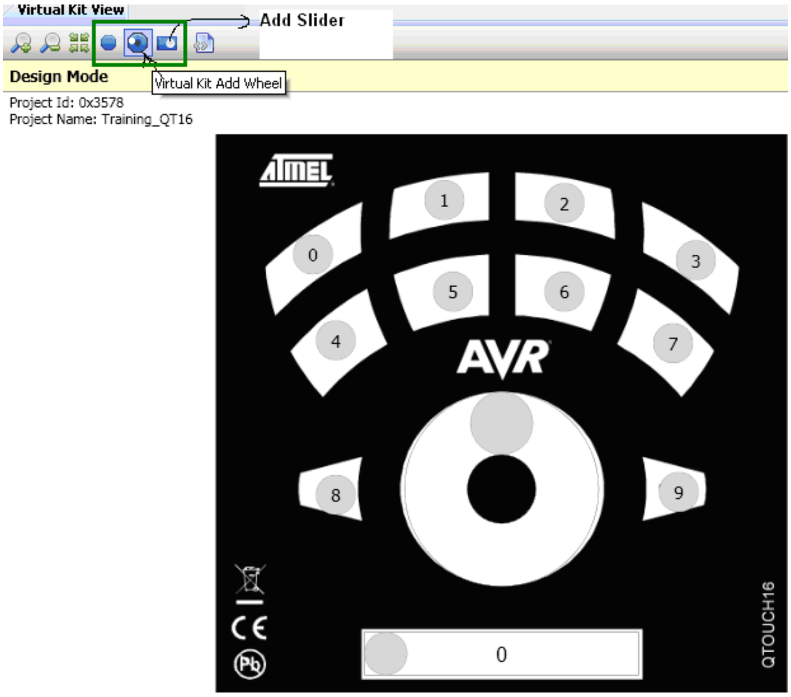
Wheel 0 <wheel0>

Configuration Options

Inner Radius	13
Height	68
Width	68
Zero Position	0
Direction	Clockwise
Sensor Id	0
Name	wheel0
From Channel	13
To Channel	15
AKS Group	NO_AKS_GROUP
Detect Threshold	10
Detect Hysteresis	HYST_6_25

9. Save the design in the same folder
10. Notice that the wheel has been assigned to channels 10-12 and the slider to 13-15
11. Delete the wheel by right clicking on the wheel and selecting the delete sensor option. In a similar manner, delete the slider
12. Notice the toolbar on the top left corner of the virtual kit window
13. Click on the slider icon and a slider gets added to the virtual kit. Similarly, click on the wheel icon to add a wheel to the virtual kit. Refer to Figure 4-15

Figure 4-15. Adding a slider and wheel.



Virtual Kit View

Add Slider

Design Mode

Virtual Kit Add Wheel

Project Id: 0x3578
Project Name: Training_QT16

14. Ensure that the channel numbers are assigned as follows:
    Slider: 10-12
    Wheel: 13-15
15. Now adjust the height and width of both the wheel and slider
16. Save the design (File → Save Design) and close it

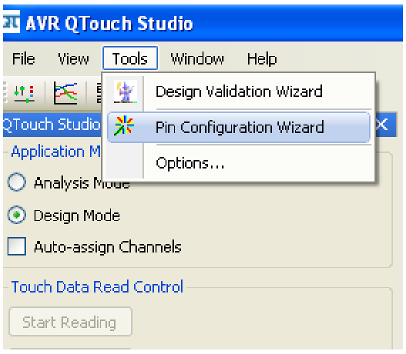
Make a note of the project ID

Make sure that the slider has been assigned channels 10-12 and the wheel channels 13-15.

Using the pin configuration wizard

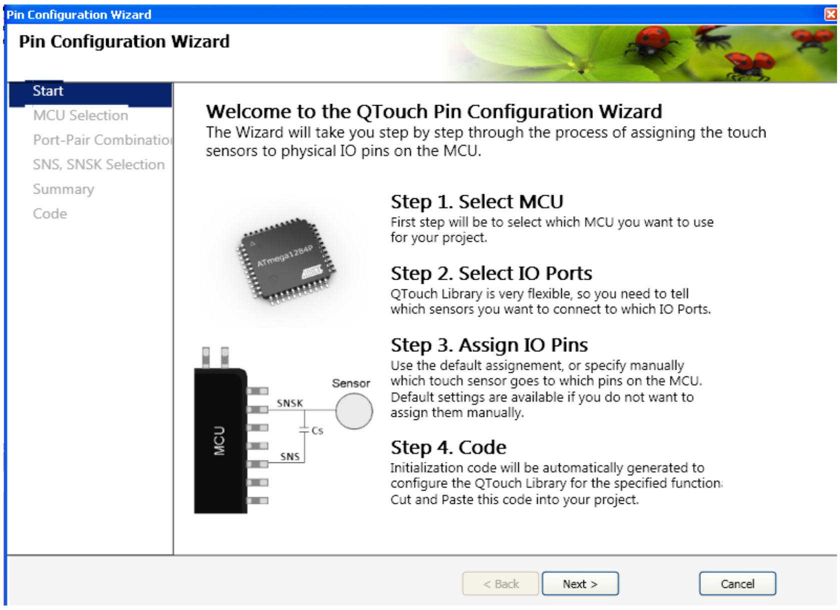
1. Now reopen the same design
2. Open the Tools menu, and click on Pin Configuration Wizard

Figure 4-16. Selecting pin configuration wizard.



3. The pin configuration wizard window opens

Figure 4-17. Pin configuration wizard.



Welcome to the QTouch Pin Configuration Wizard

The Wizard will take you step by step through the process of assigning the touch sensors to physical IO pins on the MCU.

Step 1. Select MCU

First step will be to select which MCU you want to use for your project.

Step 2. Select IO Ports

QTouch Library is very flexible, so you need to tell which sensors you want to connect to which IO Ports.

Step 3. Assign IO Pins

Use the default assignement, or specify manually which touch sensor goes to which pins on the MCU. Default settings are available if you do not want to assign them manually.

Step 4. Code

Initialization code will be automatically generated to configure the QTouch Library for the specified function Cut and Paste this code into your project.

4. Click on Next. Select the MCU family as Atmel XMEGA and the MCU as Atmel ATxmega128A1. Select the SNS and SNSK ports for pair1 as F and H (Button 0 – Button 7). Select the SNS and SNSK ports for pair2 as J and K (Button 8 – Button 9, Slider 0, Wheel 0). Click on Next
5. Select pins for the sensors. In this case, you can use auto assign
6. Click on Next and the Summary Page opens
7. Click on Next again. Copy the code onto a notepad
8. Add the following piece of code in the main.c file under the label

   ```
   /* Declare the SNS_array[][] and the SNSK_array[][] here ------------*/
   #ifdef QTOUCH_STUDIO_MASKS
           extern TOUCH_DATA_T SNS_array[2][2];
           extern TOUCH_DATA_T SNSK_array[2][2];
   #endif
   ```

   This is to hold the burst masks generated
9. Copy this part of the code in the main.c file in the main function under the comment

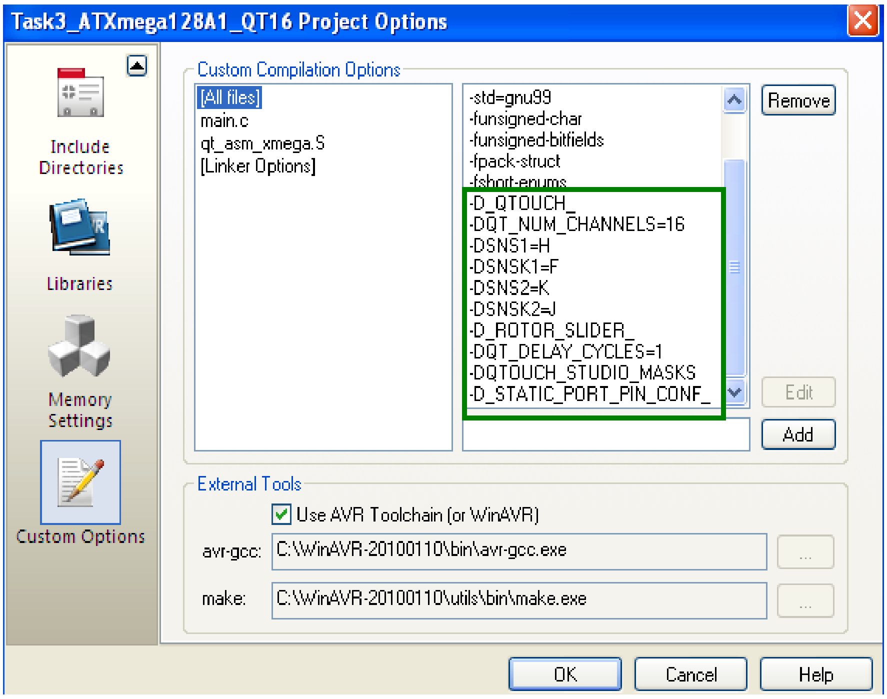
   ```
   /* The code generated by Pin Configuration Wizard */
   #ifdef QTOUCH_STUDIO_MASKS
                   SNS_array[0][0]= 0x55;
                   SNS_array[0][1]= 0xaa;
                   SNS_array[1][0]= 0x1d;
                   SNS_array[1][1]= 0xe2;


                   SNSK_array[0][0]= 0x55;
                   SNSK_array[0][1]= 0xaa;
                   SNSK_array[1][0]= 0x1d;
                   SNSK_array[1][1]= 0xe2;
       #endif
   ```

Compile options

1. Open Project Options and click on Custom Options
2. To use the pin configurability, enable the macro QTOUCH_STUDIO_MASKS in the project options
3. Add these options one at a time (refer to Figure 4-18)

   ```
   -DQTOUCH_STUDIO_MASKS
   -DQT_NUM_CHANNELS=16
   -D_ROTOR_SLIDER_
   -DSNS1=H
   -DSNSK1=F
   -DSNS2=K
   -DSNSK2=J
   -DQT_DELAY_CYCLES=1
   ```

Figure 4-18. Custom compile options.



[All files]
main.c
qt_asm_xmega.S
[Linker Options]

```
-std=gnu99
-funsigned-char
-funsigned-bitfields
-fpack-struct
-fshort-enums
-D_QTOUCH_
-DQT_NUM_CHANNELS=16
-DSNS1=H
-DSNSK1=F
-DSNS2=K
-DSNSK2=J
-D_ROTOR_SLIDER_
-DQT_DELAY_CYCLES=1
-DQTOUCH_STUDIO_MASKS
-D_STATIC_PORT_PIN_CONF_
```

☑ Use AVR Toolchain (or WinAVR)

avr-gcc: C:\WinAVR-20100110\bin\avr-gcc.exe

make: C:\WinAVR-20100110\utils\bin\make.exe

This tells the library which ports the measurements are done on, that a rotor and slider are defined, and that the maximum number of channels is 16.

–DQT_DELAY_CYCLES=1 tells the library there will only be one delay cycle.

The exact meanings of some of these parameters are not too important right now, but if you want further information, please refer to QTouch Library User Guide. These settings need to be entered into any project using the QTouch Library.

If not enabled in the project options, you can define the above macros in the touch_qt_config.h file.

Now build the project, and if everything has been done as per the guidelines given, you should be able to build successfully with no errors.

4.4 Task 4: Configure the global options and enable/set the sensors

4.4.1 Setting up the code

Initialize the library

After this initial configuration, we need to initialize the QTouch Library. This is done by simply calling qt_init_sensing(); Add a call to this function under the label:

```
/* Initialise and set touch parameters */
```

After this, add qt_set_parameters();

Adding more QTouch Library configuration

Next, we need to set up some runtime global QTouch parameters. We have provided a function, qt_set_parameters();, in which this should be done. Find this function and add the following code to it:

```
qt_config_data.qt_di                    = DEF_QT_DI;
qt_config_data.qt_neg_drift_rate        = DEF_QT_NEG_DRIFT_RATE;
qt_config_data.qt_pos_drift_rate        = DEF_QT_POS_DRIFT_RATE;
qt_config_data.qt_max_on_duration       = DEF_QT_MAX_ON_DURATION;
qt_config_data.qt_drift_hold_time       = DEF_QT_DRIFT_HOLD_TIME;
qt_config_data.qt_recal_threshold       = DEF_QT_RECAL_THRESHOLD;
qt_config_data.qt_pos_recal_delay       = DEF_QT_POS_RECAL_DELAY;
```

Exactly what these parameters specify isn't important right now; we will go into further detail later. A tip here would of course be to copy paste the code from the block above into your project.

Adding the timer ISR

Call init_timer_isr(); to configure the timer ISR to fire regularly. Add this before the label
/*Pin configuration code as generated by Pin Configuration Wizard */

Find this function and add the following code to it:

```
/*  Set timer period    */
TCC0.PER = TICKS_PER_MS * qt_measurement_period_msec;
/*  select clock source */
TCC0.CTRLA = (TOUCH_DATA_T)4;
/*  Set Compare A interrupt to low level   */
TCC0.INTCTRLB = 1u;
/*  enable low lever interrupts in power manager interrupt control  */
PMIC.CTRL |= 1u;
```

Find the function ISR(TCC0_CCA_vect), and add the following code to it:

```
/*  set flag: it's time to measure touch    */
time_to_measure_touch = 1u;
/*  update the current time  */
current_time_ms_touch += qt_measurement_period_msec;
```

Now it's time to configure our first button

1. We use the following API call to configure a button:

   qt_enable_key(CHANNEL_0, AKS_GROUP_1, 10u, HYST_6_25);

This tells Atmel QTouch that channel 0 should be treated as a key. We will look at the last three parameters a little later.

2. Add the following piece of code after the init_system( ); call, after the label

   /* Config Sensors */:

   qt_enable_key(CHANNEL_0, NO_AKS_GROUP, 10u, HYST_6_25);

   Similarly, configure the other keys.

3. To enable a slider using channels 10, 11, and 12, add the following line:

   qt_enable_slider (CHANNEL_10, CHANNEL_12, NO_AKS_GROUP, 10u, HYST_6_25, RES_8_BIT, 0u);

   Note: the RES_8_BIT argument is the slider resolution.

4. Similarly, enable the rotor using channels 13, 14, and 15 by adding the following line:

   qt_enable_rotor(CHANNEL_13, CHANNEL_15, NO_AKS_GROUP, 10u, HYST_50, RES_8_BIT, 0u);

The other arguments are not too important right now.

Somewhere to store time

We are now almost ready to start taking measurements. A single measurement is taken by calling qt_measure_sensors (uint16_t current_time_ms), but as you can see, it expects an argument. This argument is used to keep track of time to allow compensating for over-time drifting in the measured values. *If you are not concerned with drifting, you can pass this argument as zero.*

For our application, we are using current_time_ms_touch as a simple counter to keep track of time

Filter functions

This function is called after the library has made capacitive measurements, but before it has processed them. The user can use this hook to apply filter functions to the measured signal values (possibly to fix sensor layout faults).

Add this code line,

qt_filter_callback = 0;

in the main function after the label /* Filter callback function */.

We are now ready to perform a touch measurement

Call enable_interrupt(); just before the for loop beings to enable interrupts.

Inside the main loop, under /* loop Forever */, add the following piece of code below the label /* Measure touch once */,

status_flag = qt_measure_sensors(current_time_ms_touch);

burst_flag = status_flag & QTLIB_BURST_AGAIN;

Build the code

It should build successfully with no errors or warnings.

4.4.2 Further explanation

Most of the function calls used in this task are quite simple and self explainatory. The most complex line is the qt_enable_key() function, so let's look closer at this:

```
void qt_enable_key(channel_t channel,
                   aks_group_t aks_group,
                   threshold_t detect_threshold,
                   hysteresis_t detect_hysteresis);
```

The parameters are as follows:

channel	= which touch channel the key sensor uses
aks_group	= which AKS group (if any) the sensor is in
detect_threshold	= the sensor detection threshold
detect_hysteresis	= the sensor detection hysteresis value

The slider and wheel declarations have similar parameters:

```
void qt_enable_slider(
         channel_t from_channel,
         channel_t to_channel,
         aks_group_t aks_group,
         threshold_t detect_threshold,
         hysteresis_t detect_hysteresis,
         resolution_t position_resolution,
         uint8_t position_hysteresis);
```

```
void qt_enable_rotor(
         channel_t from_channel,
         channel_t to_channel,
         aks_group_t aks_group,
         threshold_t detect_threshold,
         hysteresis_t detect_hysteresis,
         resolution_t angle_resolution,
         uint8_t angle_hysteresis);
```

The slider and wheel declarations have a few more parameters:

from_channel	- the first channel in the rotor sensor
to_channel	- the last channel in the rotor sensor
aks_group	- which AKS group (if any) the sensor is in
detect_threshold	- the sensor detection threshold
detect_hysteresis	- the sensor detection hysteresis value
position_resolution	- the resolution of the reported position value
angle_resolution	- the resolution of the reported angle value
position_hysteresis	- the hysteresis of the reported position value
angle_hysteresis	- the hysteresis of the reported angle value

4.5 Task 5: Use the debug interface

4.5.1 Introduction

The debug interface is used to communicate various touch sensor information to the computer running QTouch Studio. The debug interface as seen from the XMEGA is implemented completely in firmware. This is to be added only for debugging purposes. Here, Bit Bang SPI protocol has been implemented to transfer touch data.

4.5.2 Setting up the code

Add the source files

- QDebug.c
- QDebugTransport.c
- eepromaccess.c
- BitBangSPI_Master.c

1. QDebug.c, QDebugTransport.c, and eeprom_access.c can be found in the debug folder. BitBangSPI_Master.c can be found in the Task5 folder
2. The debug interface is used to communicate various touch sensor information to a computer running Atmel QTouch Studio
3. In the AVR GCC tab in AVR Studio, right click the project and select Add Existing File, and then select the BitBangSPI_Master.c source file. Alternatively, you can drag and drop the source file onto the project name
4. The Atmel QT600 interface board and QTouch Studio can be used to read touch data from any application based on the QTouch Library. Here we are using Bit Banged SPI as the communication interface
5. Open the Project Options window, click on Include Directories, and specify the right paths for all four files (enter the relative path)

Add the Include files

Include the following header files in the main.c file below the label
/* now include the debug files--------------------*/:

```
#include "QDebug.h"
#include "QDebugTransport.h"
```

Custom Options

Open the Project Options window, switch to the Custom Options tab, and add the following flag:

-D_DEBUG_INTERFACE_

Adding code

1. In the QDebugSettings.h file, add the project ID (same project ID given to the virtual kit image in Task 3) under the label
/* Define Custom project as the project id of the virtual kit */
2. Under the label // Set up project info, assign the project ID
`#define PROJECT_ID CUSTOM_PROJECT`

3. Define the communications protocol. In this hands-on training, we are using the Bit-Bang SPI protocol.
   ```
   #define QDEBUG_SPI_BB
   ```
4. In the BitBangSPI_Master.h file, verify that the Bit-Bang SPI pins, port, and clock frequency are defined correctly
5. Add this piece of code under the label /* Initialize debug protocol */
   ```
   #ifdef _DEBUG_INTERFACE_
           QDebug_Init();
   #endif
   ```
6. Add this piece of code under the label /* send debug data */
   ```
   #ifdef _DEBUG_INTERFACE_
           QDebug_SendData(status_flag);
   #endif
   ```
7. Add this piece of code under the label /* Process commands from PC */
   ```
   #ifdef _DEBUG_INTERFACE_
           QDebug_ProcessCommands();
   #endif
   ```

Tip **QDebug_ProcessCommands(); is called twice in the code.**

Build the code

1. It should build successfully with no errors or warnings
2. Now connect the kit as shown in Task 1
3. Program the code
4. Launch Atmel QTouch Studio and switch to Analysis mode
5. Check if your kit gets connected
6. Switch to Analysis mode and click on the Start Reading button
7. Test whether you are able to detect touch and see it on QTouch Studio or not. In the Data view you can see the values of reference, signal, and delta and the state of each sensor

4.6 Task 6: Analysis mode

4.6.1 Introduction

Analysis mode has a scalable image of the connected kit. It will show touch events such as button presses, wheel use, and slider use. The toolbar buttons in the virtual kit view window shall be used to perform zoom-in, zoom-out, auto-fit and auto code generation operations.

The Analysis mode can be selected by using the radio buttons in the Control view. Atmel QTouch Studio reads out the channel to sensor mapping for an AVR QTouch kit when it is connected to the PC. Using this information, the image of the kit will be updated.

As explained during Task 1, sensors drawn in a light grey color are inactive, and sensors drawn in a light brown color are active sensors. Whenever a touch to an active sensor is detected, a filled circle indicating where the user touches the sensor is drawn. The filled circle increases when the delta increases, and changes color when the threshold is reached.

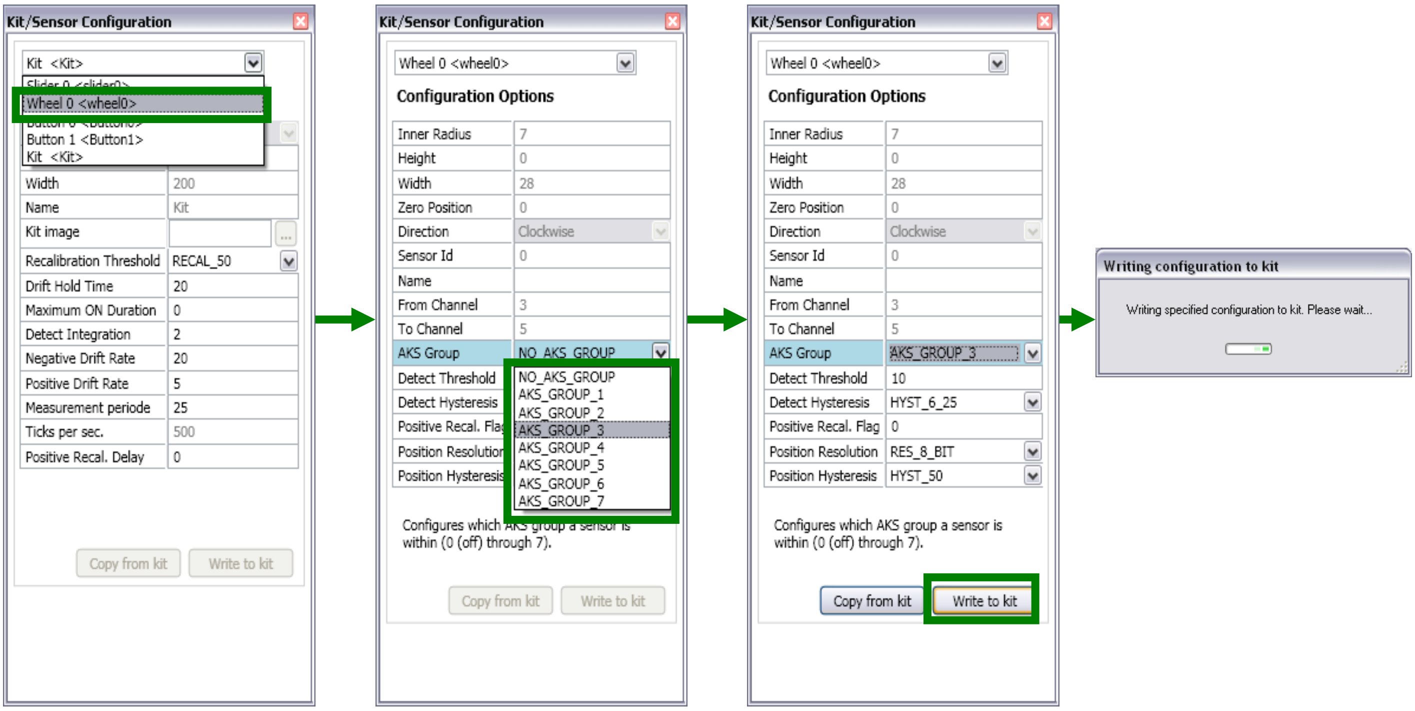
When a touch to the active sensor is no longer detected, the color of the sensor will go back to light brown again. The user shall be able to edit sensor configuration options (non-design options) and kit configuration options (non-design options) in Analysis mode.

4.6.2 Configuring the kit/sensors

Adjacent Key Suppression (AKS)

1. Open the Kit/Sensor Configuration window, select Button 0, and configure the AKS Group as AKS_GROUP_1
2. Similarly, configure Button 5 and Button 9 as AKS_GROUP_1
3. Now, if you touch all three buttons at the same time, only one is detected

Figure 4-19. Modifying kit/sensor configuration.



Kit/Sensor Configuration

Kit <Kit>

Slider 0 <slider0>
Wheel 0 <wheel0>
Button 0 <Button0>
Button 1 <Button1>
Kit <Kit>

Width	200
Name	Kit
Kit image	
Recalibration Threshold	RECAL_50
Drift Hold Time	20
Maximum ON Duration	0
Detect Integration	2
Negative Drift Rate	20
Positive Drift Rate	5
Measurement periode	25
Ticks per sec.	500
Positive Recal. Delay	0

Kit/Sensor Configuration

Wheel 0 <wheel0>

Configuration Options

Inner Radius	7
Height	0
Width	28
Zero Position	0
Direction	Clockwise
Sensor Id	0
Name	
From Channel	3
To Channel	5
AKS Group	NO_AKS_GROUP
Detect Threshold	
Detect Hysteresis	
Positive Recal. Flag	
Position Resolution	
Position Hysteresis	

NO_AKS_GROUP
AKS_GROUP_1
AKS_GROUP_2
AKS_GROUP_3
AKS_GROUP_4
AKS_GROUP_5
AKS_GROUP_6
AKS_GROUP_7

Configures which AKS group a sensor is within (0 (off) through 7).

Kit/Sensor Configuration

Wheel 0 <wheel0>

Configuration Options

Inner Radius	7
Height	0
Width	28
Zero Position	0
Direction	Clockwise
Sensor Id	0
Name	
From Channel	3
To Channel	5
AKS Group	AKS_GROUP_3
Detect Threshold	10
Detect Hysteresis	HYST_6_25
Positive Recal. Flag	0
Position Resolution	RES_8_BIT
Position Hysteresis	HYST_50

Configures which AKS group a sensor is within (0 (off) through 7).

Writing configuration to kit

Writing specified configuration to kit. Please wait...

In designs where the sensors are close together or set for high sensitivity, multiple sensors might report detect simultaneously if touch is near them. To allow applications to determine the intended single touch, the touch library provides the user the ability to configure a certain number of sensors in an AKS group.

When a group of sensors are in the same AKS group, only the first, strongest sensor will report detection. The sensor reporting detection will continue to report detection even if another sensor’s delta becomes stronger. The sensor stays in detect until its delta falls below its detection threshold, and then, if any more sensors in the AKS group are still in detect, the strongest will report detection.

And so at any given time, only one sensor from each AKS group will be reported to be in detect.

Modify all Global and Sensor parameters runtime as shown in Figure 4-19

Detect Threshold

1. Open the Kit/Sensor Configuration window, select any sensor, and change the value of the detect threshold. Click on Write to kit
2. Observe how sensitivity of that sensor changes with different values
3. The lower the value, the more sensitive is the sensor

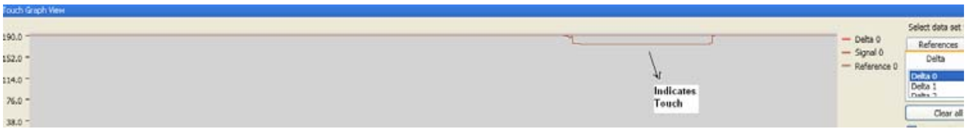
A sensor's detect threshold defines how much its signal must drop below its reference level to qualify as a potential touch detect. It has a range of between 3 and 255.

For more details on all the configuration parameters, please refer to the QTouch Library User Guide.

Maximum On Duration

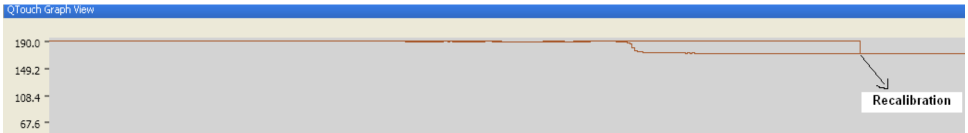
1. Click on the View menu and click on Graph View. The QTouch Graph view window opens
2. Select data set for channel 0 and observe what happens on touch

Figure 4-20. Maximum ON duration = 0.



3. If a touch happens for a prolonged duration when the Maximum ON Duration is zero, the sensor remains in detect
4. Open the Kit/Sensor Configuration window
5. Change the Maximum ON Duration to 30, and the sensor automatically recalibrates after six seconds

Figure 4-21. Maximum ON duration = 30.



If an object unintentionally contacts a sensor and results in a touch detection for a prolonged interval, it is usually desirable to recalibrate the sensor in order to restore its function, perhaps after a time delay of some seconds. The Maximum ON Duration timer monitors such detections.

4.7 Task 7: Design validation

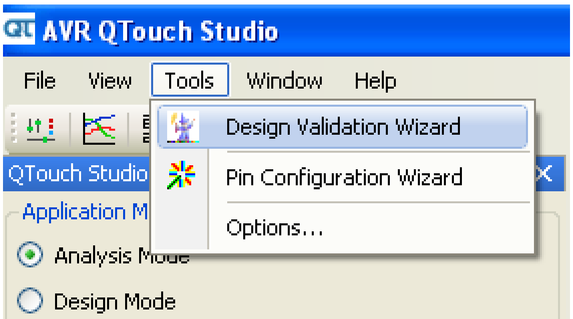
4.7.1 Introduction

The design validation wizard is a tool used to validate the design by measuring and logging values received from the kit.

4.7.2 Using the design validation wizard

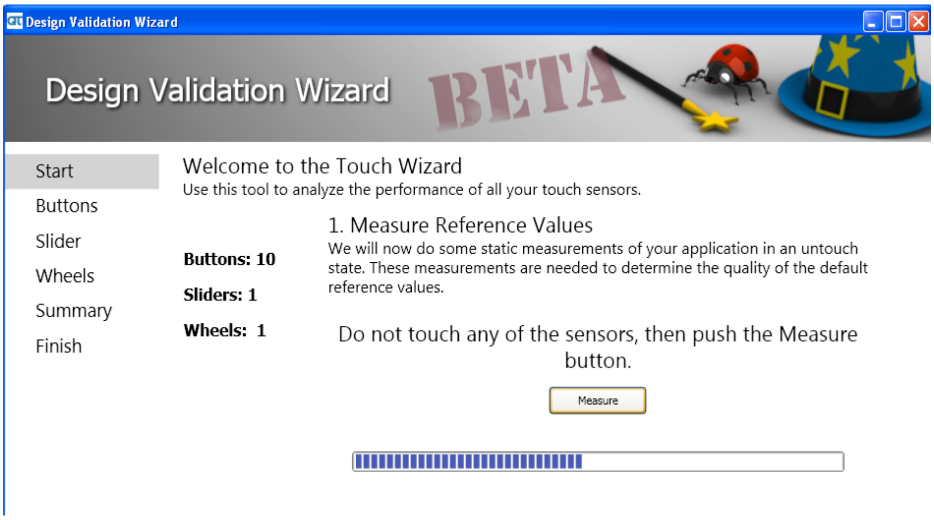
1. Connect the kit and start a touch debug session
2. Launch Atmel QTouch Studio and start the design validation wizard from the Tools menu

Figure 4-22. Selecting design validation wizard.



3. The design validation wizard opens

Figure 4-23. Design validation wizard.



Welcome to the Touch Wizard

Use this tool to analyze the performance of all your touch sensors.

Buttons: 10

Sliders: 1

Wheels: 1

1. Measure Reference Values

We will now do some static measurements of your application in an untouch state. These measurements are needed to determine the quality of the default reference values.

Do not touch any of the sensors, then push the Measure button.

4. On the first page, it will ask you to click the Measure button without touching any of the sensors. Follow the instructions
5. Click Next and follow subsequent instructions
6. At the end, you will get a summary with details about each sensor and recommended settings
7. You can either write these recommended settings to the kit or project file
8. Save the design and project file and reopen them again to see if the changes have been made

4.8 Task 8: Log and analyze data

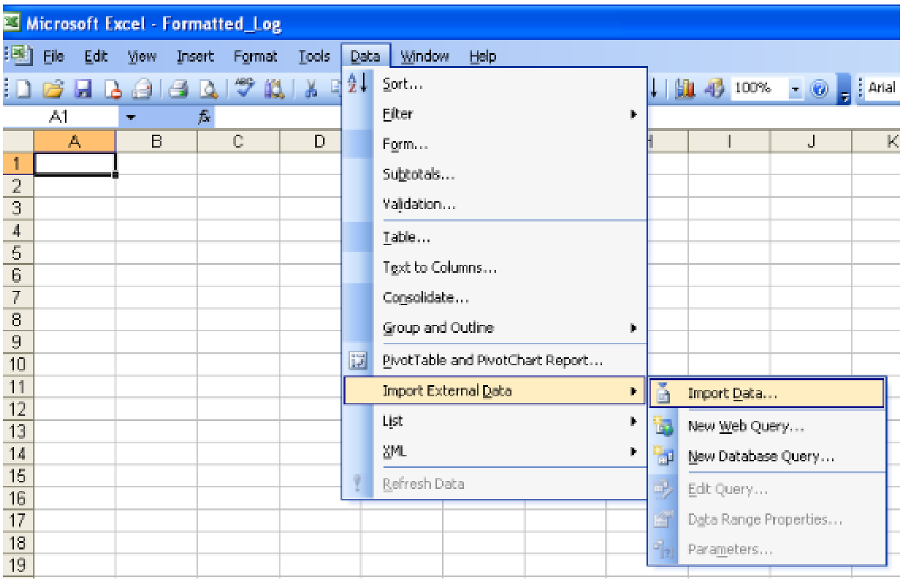
Logging the data

1. If the Auto-start Reading option is checked, then Atmel QTouch Studio will automatically start reading touch data when a kit is connected
2. If unchecked, you must click the Start Reading button to make QTouch Studio start reading touch data. Checking the Log data to file checkbox will make QTouch Studio log data read from the kit
3. As soon as reading has stopped, either because the user has clicked on the Stop Reading button or the kit has been disconnected from the PC, a dialog will pop up to let the user specify a location and file name to save the logged data
4. Note that clicking the Cancel button will simply discard all data logged and close the dialog

Analysis and debugging

1. Open a new Excel spreadsheet and import data from the logged data
2. Select tab and semicolon as delimiters and save the file

Figure 4-24. Importing data.



3. You can also create charts, which help in analyzing the logged data and observing variations. In the Excel spreadsheet named Formatted_Log_Chart, you can observe the delta variations for channel 4 in the form of a chart as depicted

5 Summary

The main goal with this hands-on training was to learn how to set up and use the Atmel QTouch Suite and the Atmel QT600 kit:

- Set up the hardware
- Create a project from scratch and include libraries and other header files
- Create a virtual kit, use the pin configuration wizard, and configure technology options
- Configure global options and enable sensors
- Add buttons, sliders, and wheels
- Use Analysis mode
- Use the design validation wizard
- Log and analyze data

In addition, you have learned how to use the Atmel QTouch Studio GUI, to write C code, debug your code, and program the device.

NOTE

All the include files and other source files in this hands-on training have been taken from Atmel QTouch Library version 4.3. If a new library version is released, please use the latest files from it.

6 Atmel technical support center

Atmel has several support channels available:

• Web portal:	http://support.atmel.no/	All Atmel microcontrollers
• Email:	touch@atmel.com	All touch products
• Email:	avr@atmel.com	All AVR products
• Email:	avr32@atmel.com	All AVR32 products

Please register on the web portal to gain access to the following services:

- A rich FAQ database
- Easy submission of technical support requests
- A history of all your past support requests
- Atmel microcontrollers' newsletters
- Information about available trainings and training material

7 Table of Contents

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Milennium 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
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chou-ku, Tokyo 104-0033
JAPAN
Tel: (+81) 3523-3551
Fax: (+81) 3523-7581

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.