Atmel AVR262: Atmel QTouch with USB HID

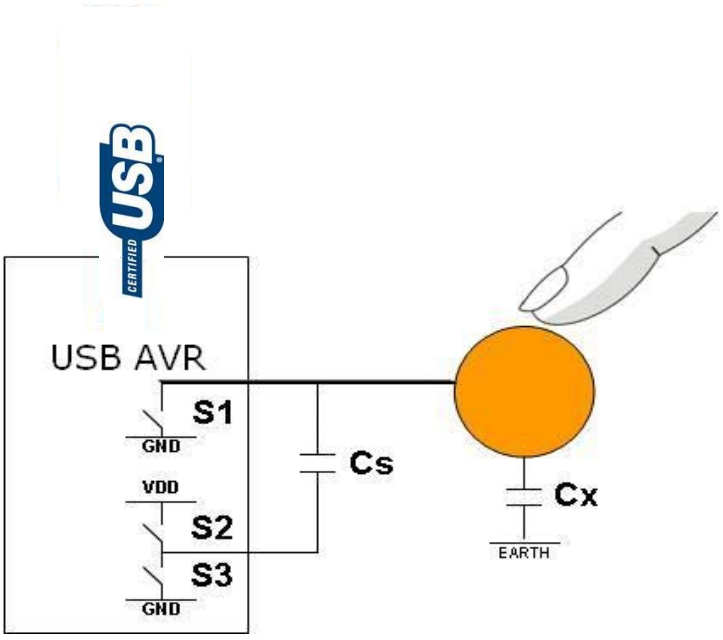
Features

- **Single-chip solution**
  - **Atmel® QTouch® and USB HID on the Atmel AVR® AT90USB646**
- **Using QTouch Library**
  - **Eight QTouch keys**
  - **Customized firmware using the Atmel QTouch Library**
- **Easy portability for different AVR USB devices**
- **Supports IAR™ and GCC**

1 Introduction

This application note illustrates how to use the QTouch Library on the AT90USB646 microcontroller to provide a single-chip solution for QTouch sensing through a USB interface. Touch sensor status is acquired and reported over the USB HID class to a host PC.

Figure 1-1. Block diagram.



8-bit Atmel Microcontrollers

Application Note

Rev. 8375A-AVR-03/11

2 Hardware requirements

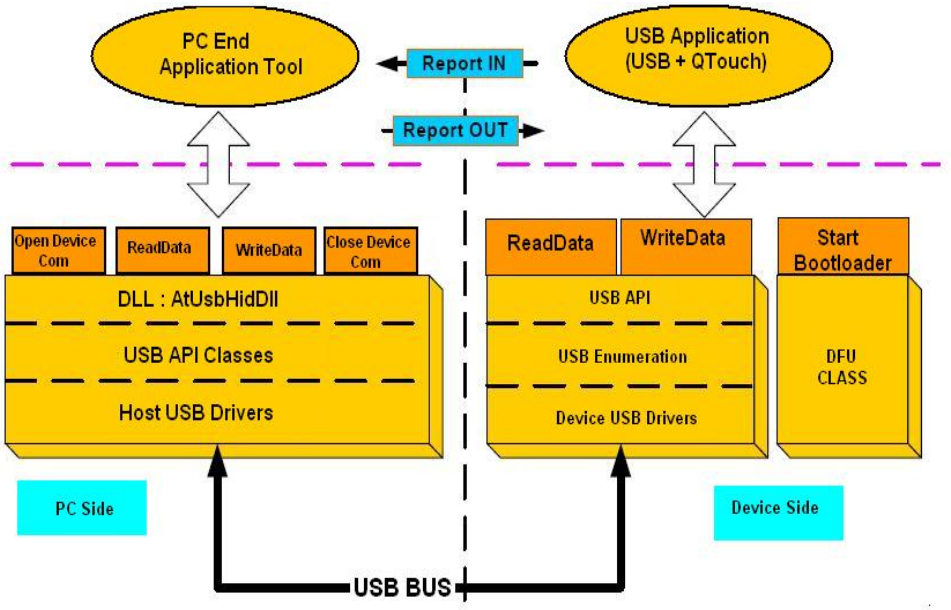
- Atmel STK®600, Atmel STK525, or custom board
- Atmel QTouch sensor board
- USB host

3 Application overview

This demo application acquires the QTouch key sensed, generates the report, and sends it to a PC (host) on request. The application is built over the generic USB HID application, which allows a simple data exchange between the PC (host) and the device. The PC requests new, available QTouch data from the device on a regular polling interval. The device will send data if available. Otherwise, a NAK (no acknowledge) is sent to inform the PC that no data is available. The data package sent from the device to the PC is referred to as Report IN.

Figure 3-1 gives the generic overview of the application.

Figure 3-1. Application overview.



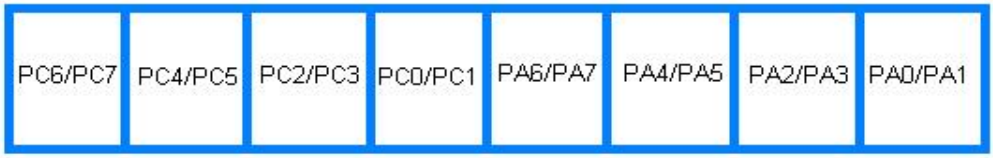
The reports sent to the host contain a set of bytes that can be used by the user application. Because this application is using just eight QTouch keys, one byte is sufficient to transfer the touch status, though this can be modified depending on the requirement.

In this application, port A and port C are used for touch sensing. When the SNS and SNSK pins are connected to the same port, the even pin numbers will be used as SNS pins and the odd pin numbers will be used as SNSK pins.

- PA0, PA2, PA4, PA6 → SNS0, SNS1, SNS2, SNS3
- PA1, PA3, PA5, PA7 → SNSK0, SNSK1, SNSK2, SNSK3

The least-significant bit (LSB) of the report byte has the value of the sensor connected to PA0 and PA1. Similarly the next higher bit of the report byte has the value of the sensor connected to PA2 and PA3. The positions of the sensors in the report byte are shown in Figure 3-2.

Figure 3-2. Touch data in report byte.



When a touch is sensed on a sensor connected to PA0/PA1, the value of the report byte is 0x01, and when it is sensed at PC4/PC5 the value is 0x40. The report byte values for the remaining sensors are treated similarly.

4 Touch library integration

To build the application, the appropriate touch header files, assembler files and library files (which are device and compiler specific) need to be integrated with other source files. To select the correct device and application-specific library files, refer to the Library_Selection_Guide.xls file in the Atmel QTouch library root folder. The library selection guide is also available on the Atmel website.

Select the technology as “QTouch,” the MCU family as “USB AVR,” the MCU type as “8-bit,” the MCU as “Atmel AT90USB646,” the tool chain as “GCC/IAR,” the ports available for QTouch as “A, B, C, D, E, F,” and the maximum number of channels as “8.” By selecting these parameters, the appropriate library files to be included can be seen.

- libv3g1-8qt-k-0rs.r90 – IAR library file for the respective device
- libavr5g1-8qt-k-0rs.a – GCC library file for the respective device
- qt_asm_tiny_mega.S – assembler file
- touch_qt_config.h
- touch_api.h

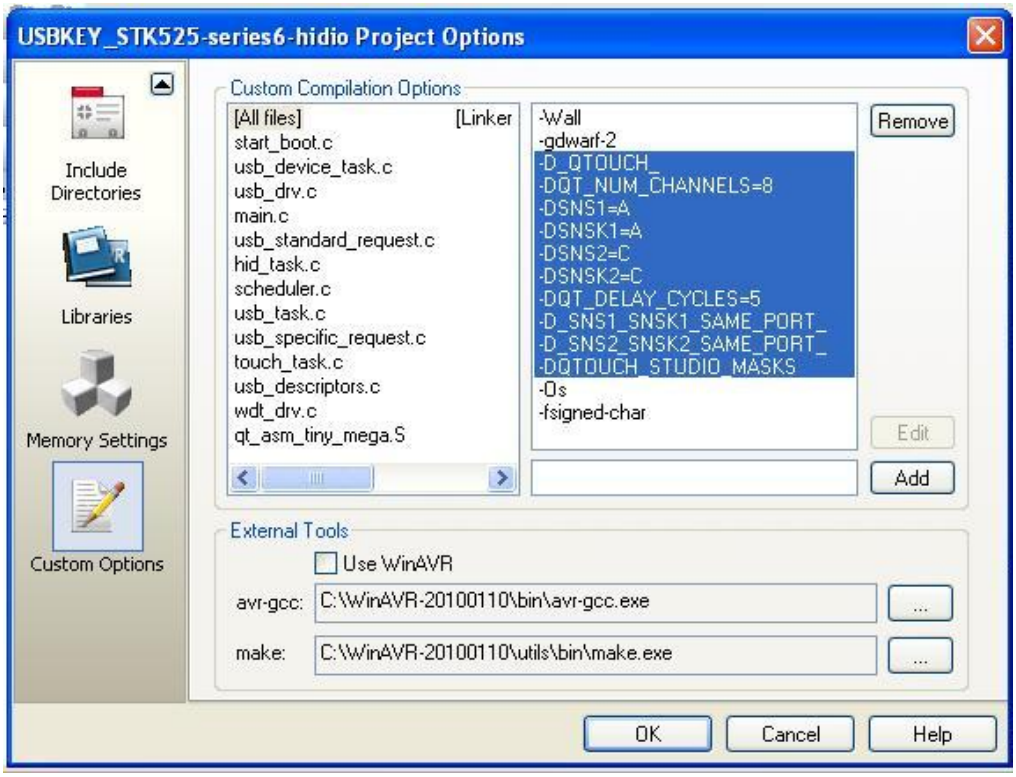
5 Compiler and assembler options

Building the application in IAR and GCC requires certain options to be set. As both the compilers are unique in their own way, minor differences are required for each. This section explains the necessary option settings required for each compiler.

In GCC, open Project Options, and then click on Custom Options. Add these custom compilation options one by one (refer to Figure 5-1).

```
-D_QTOUCH_
-DQT_NUM_CHANNELS=8
-DSNS1=A
-DSNSK1=A
-DSNS2=C
-DSNSK2=C
-DQT_DELAY_CYCLES=5
-D_SNS1_SNSK1_SAME_PORT_
-D_SNS2_SNSK2_SAME_PORT_
-DQTOUCH_STUDIO_MASKS
```

Figure 5-1. GCC compiler options.



For IAR, the options have to be set in the compiler as well as the assembler. In IAR Project Options, select the C/C++ Compiler category, and on the Preprocessor tab, the following options should be provided, as shown in Figure 5-2. The same options should be added in the Assembler category on the Preprocessor tab, as shown in Figure 5-3.

Figure 5-2. IAR compiler options.

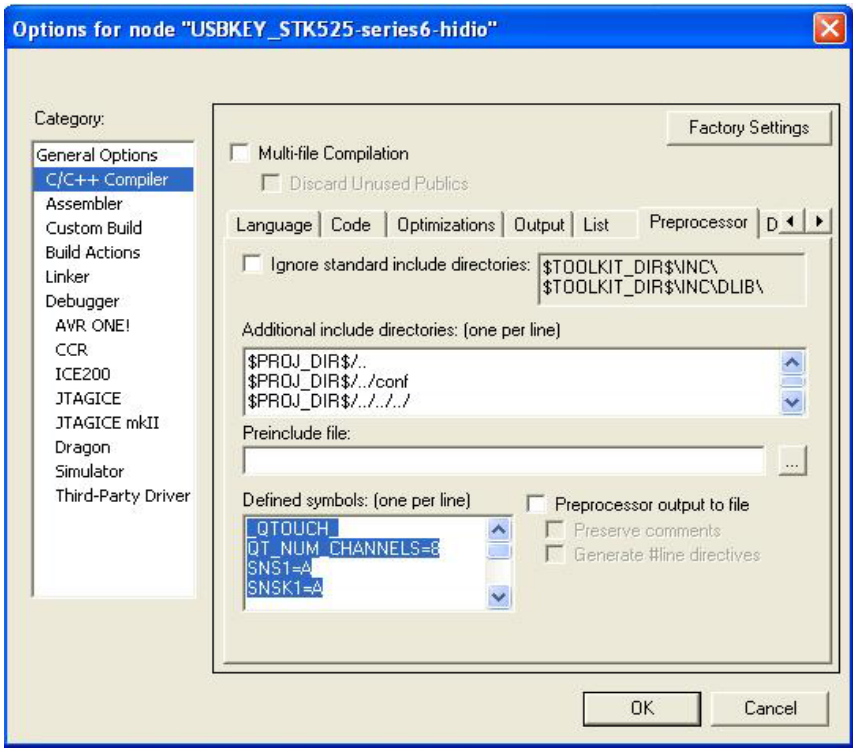
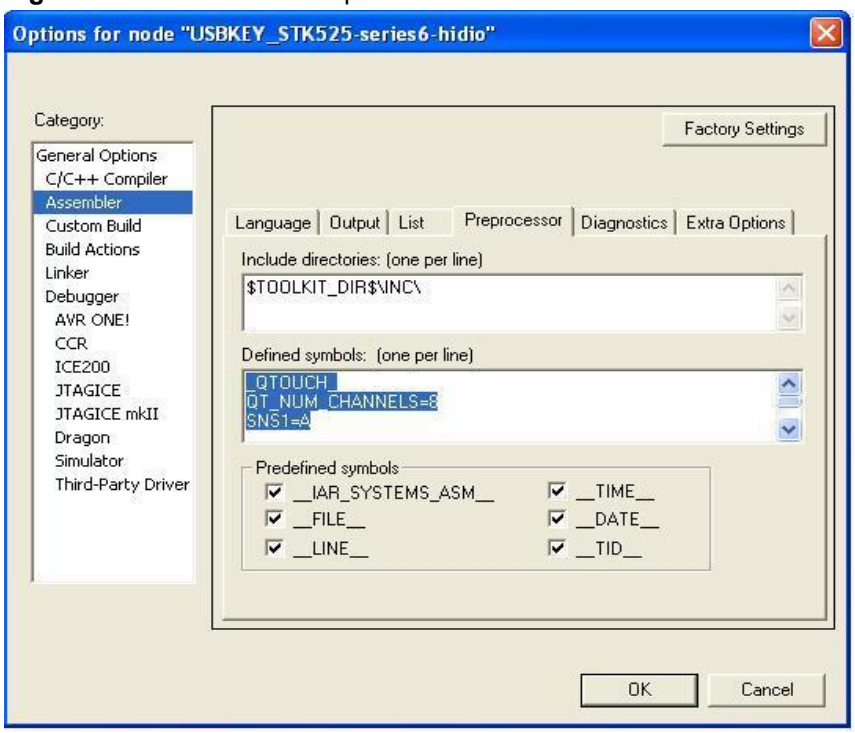


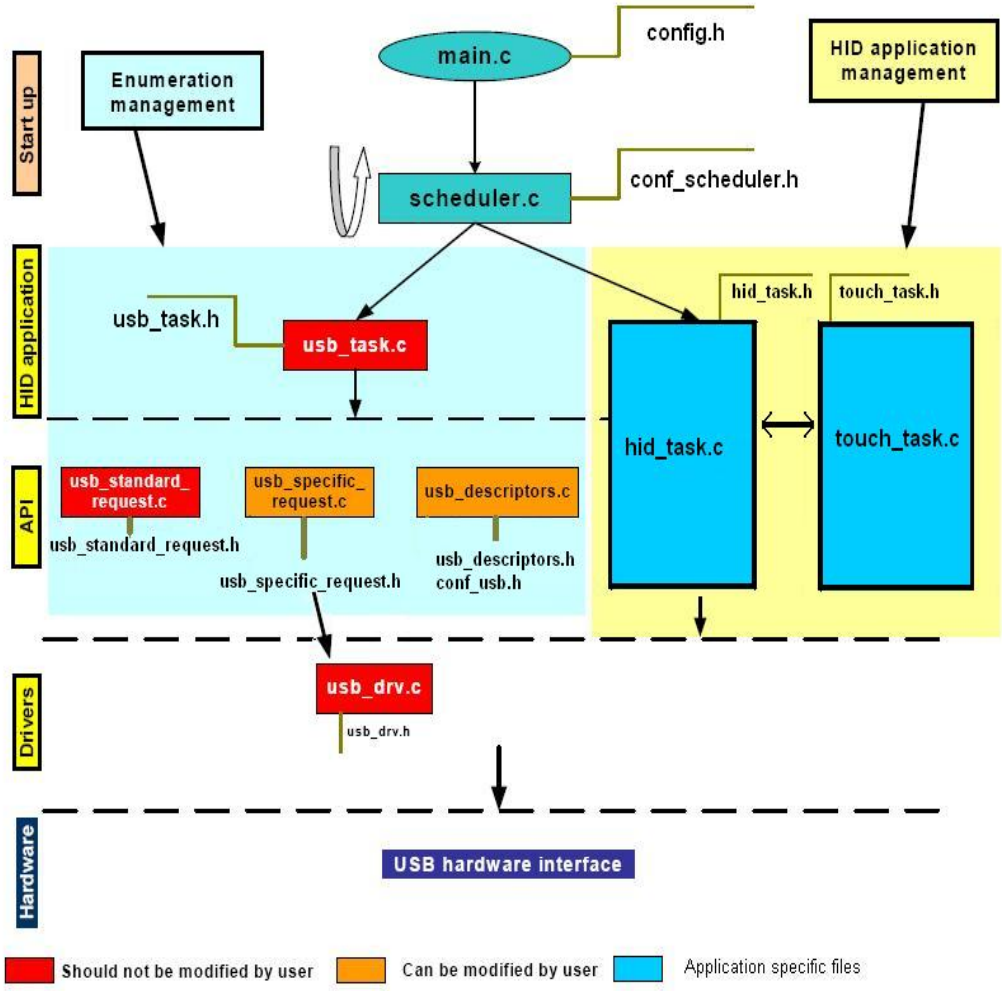
Figure 5-3. IAR assembler options.



6 Firmware

This application employs an HID application manager, which includes touch_task.c and hid_task.c. Figure 6-1 shows the architecture of the firmware with the help of a block diagram.

Figure 6-1. Firmware architecture.



The touch sensing and report generation are handled in hid_task.c and touch_task.c and their respective header files.

The various related functions can be grouped into three main categories. The necessary functions are explained under each relevant category.

- Functions related to initialization
- Touch library APIs
- HID report generation

6.1 Functions related to initialization

6.1.1 config_keys

The config_keys function is responsible for configuring the sensors as keys to the respective channels.

6.1.2 CHANNEL_t

The CHANNEL_t function specifies the channel number to be configured for a key.

6.1.3 AKS_GROUP

The AKS_GROUP function specifies the adjacent key suppression (AKS®) group associated with the sensor being configured as “key.”

6.1.4 10u

The 10u function specifies the detect threshold for the sensor.

6.1.5 HYST_6_25

The HYST_6_25 function specifies the detection hysteresis for the sensor.

```
qt_enable_key( CHANNEL_0, AKS_GROUP_1, 10u, HYST_6_25 );
qt_enable_key( CHANNEL_1, AKS_GROUP_1, 10u, HYST_6_25 );
qt_enable_key( CHANNEL_2, AKS_GROUP_1, 10u, HYST_6_25 );
qt_enable_key( CHANNEL_3, AKS_GROUP_1, 10u, HYST_6_25 );
qt_enable_key( CHANNEL_4, AKS_GROUP_1, 10u, HYST_6_25 );
qt_enable_key( CHANNEL_5, AKS_GROUP_1, 10u, HYST_6_25 );
qt_enable_key( CHANNEL_6, AKS_GROUP_1, 10u, HYST_6_25 );
qt_enable_key( CHANNEL_7, AKS_GROUP_1, 10u, HYST_6_25 );
```

Refer to the Atmel QTouch user guide for more information on this function.

6.1.6 qt_init_sensing

The qt_init_sensing function is used to initialize the touch sensing for all enabled channels. All required sensors should be configured before calling this function.

```
#define qt_init_sensing( ) qt_init_sensing_with_burst(
BURST_FUNC_NAME ,CALCULATE_MASKS)
```

6.1.7 qt_set_parameters

The qt_set_parameters function is used to initialize the global configuration settings in the variable qt_config_data of the QTouch acquisition method libraries. In other words, this will fill the default threshold values in the configuration data structure.

6.1.8 touch_init_HID

The touch_init_HID function is responsible for calling the initialization routines related to touch sensing.

```
/* Configure the Sensors as keys  */
   config_keys();

/* initialise touch sensing */
   qt_init_sensing();

/*  Set the parameters like recalibration threshold,
Max_On_Duration etc in this function by the user */
   qt_set_parameters( );

/* configure timer ISR to fire regularly */
   init_timer_isr();
```

6.2 Touch library APIs

6.2.1 touch_measure

The touch_measure function is responsible for measuring the touch sensing.

```
/*status flags to indicate the re-burst for library*/
    uint16_t status_flag = 0u;
    uint16_t burst_flag = 0u;
    if( time_to_measure_touch )
   {

      /*  clear flag: it's time to measure touch  */
      time_to_measure_touch = 0u;
      do
      {
        /*  one time measure touch sensors    */
        status_flag = qt_measure_sensors( current_time_ms_touch );

        burst_flag = status_flag & QTLIB_BURST_AGAIN;
      }while (  burst_flag) ;
   }
```

6.3 HID report generation

6.3.1 hid_report_in

Whenever the host requests data, the device responds accordingly. If there is no new data, the device sends a NAK response. When new data is available, the device needs to send the information to the host. This function is responsible for generating the report to be sent via USB.

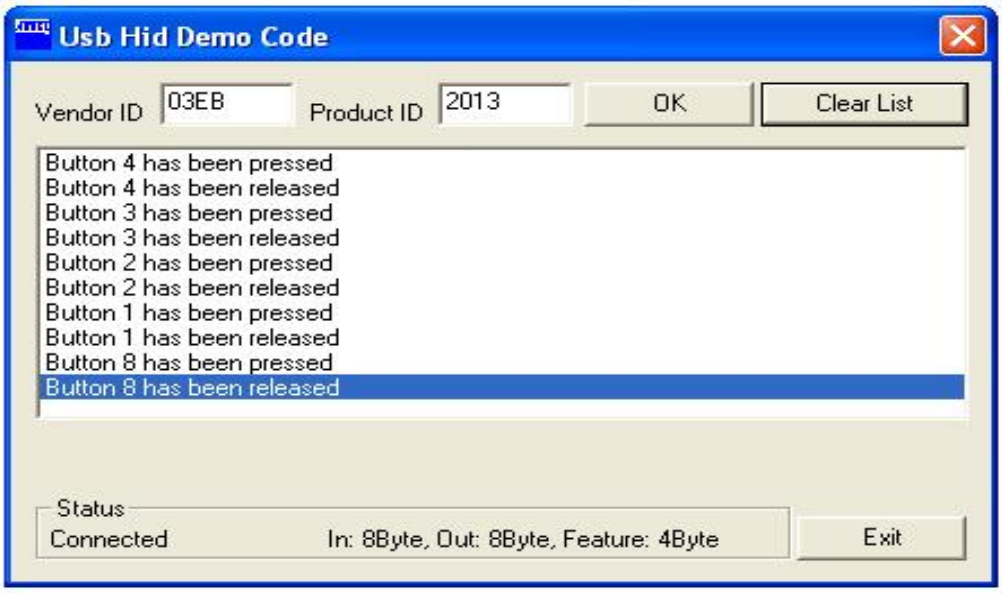
Whenever a touch or release is sensed, the value of the touch_data variable changes and the host is updated with the new status.

```
Usb_select_endpoint(EP_HID_IN);
   if(!Is_usb_write_enabled())
      return;                                  // Not ready to send
report


   if (touch_data == old_touch_data)
     return;
   old_touch_data = touch_data;


  // Send report
   Usb_write_byte(touch_data);                 // Touch Information
   Usb_write_byte(GPIOR1);                     // Dummy (not used)
   Usb_write_byte(GPIOR1);                     // Dummy (not used)
   Usb_write_byte(GPIOR1);                     // Dummy (not used)
   Usb_write_byte(GPIOR1);                     // Dummy (not used)
   Usb_write_byte(GPIOR1);                     // Dummy (not used)
   Usb_write_byte(GPIOR1);                     // Dummy (not used)
   Usb_write_byte(GPIOR1);                     // Dummy (not used)
   Usb_ack_in_ready();                         // Send data over the
USB
```

7 Host application

This application note includes a GUI-based PC host application that can be used to display which key is pressed or released. The PC executable is available in the associated deliverables. The respective key press or key release is displayed as shown in Figure 7-1.

Figure 7-1. Key status.



8 Related items

- Atmel AT90USB646 series datasheet:
  http://www.atmel.com/dyn/products/product_card.asp?part_id=3877
- Atmel AVR Studio®:
  http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725
- Atmel AVR QTouch Library 4.3:
  http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4627
- Atmel AVR QTouch Library User Guide:
  http://www.atmel.com/dyn/resources/prod_documents/doc8207.pdf
- AVR328: USB Generic HID Implementation on megaAVR® devices
  http://www.atmel.com/dyn/resources/prod_documents/doc7599.pdf
- AVR1511: QT600-ATxmega128A1 Training Guide
  http://www.atmel.com/dyn/resources/prod_documents/doc8350.pdf

9 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

8375A-AVR-03/11