APPLICATION NOTE

AT01295: Integration of QTouch Library with BitCloud ZigBee Light Link

Atmel MCU Wireless

Features

- Integration of Atmel® QTouch® Library with ZigBee® Light Link application
- Using AVR477 RF Touch Remote as ZLL Remote Controller

Introduction

Atmel BitCloud® ZigBee Light Link (ZLL) profile [5] provides a simple, robust, secure and low-cost communication network allowing wireless controlling of lighting devices like LED fixtures, lighting bulbs, timers, switches, and remotes. Add to this, the best-in-class capacitive touch sensing technology for intuitive user interface from Atmel, the end product is an ideal remote controlling solution for lighting control applications.

Touch technology from Atmel is supported by a QTouch Library [8] for Atmel microcontrollers. QTouch Library supports three patented capacitive touch acquisition methods: QTouch, QTouchADC and QMatrix™.

This application note describes how to integrate Atmel QTouch Library with the Atmel ZLL solution. The AVR477 RF Touch remote board ([1], [2]) is the reference hardware platform and uses QMatrix acquisition method.

This document is intended to be a guide to integrate QTouch Library specifically with ZLL profile applications, though this can also be used as a reference for integrating QTouch library with other BitCloud applications as well. For detailed information on Atmel capacitive touch (QTouch) technology and associated products, refer to [3].

Table of Contents

1. Overview

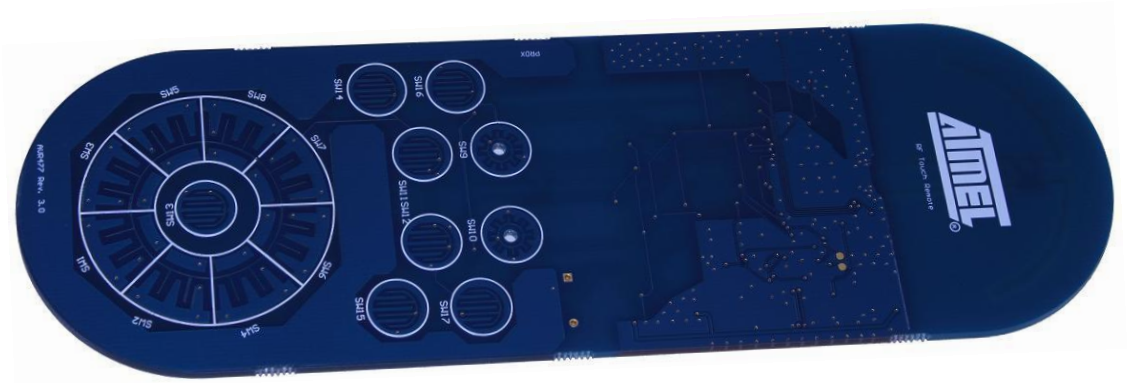
1.1 Hardware – RF Touch Remote Control Board

Atmel AVR477 RF Touch Remote Control Board is used as reference hardware for this application.

BitCloud ZLL software package supports only Atmel ATmega256RFR2 because the code size of ZLL stack is more than 128K. Hence device replacement in AVR477 hardware is required. i.e., ATmega128RFA1 is replaced with ATmega256RFR2. Since both these devices are pin-to-pin compatible, no other changes in the hardware are required for implementing this application.

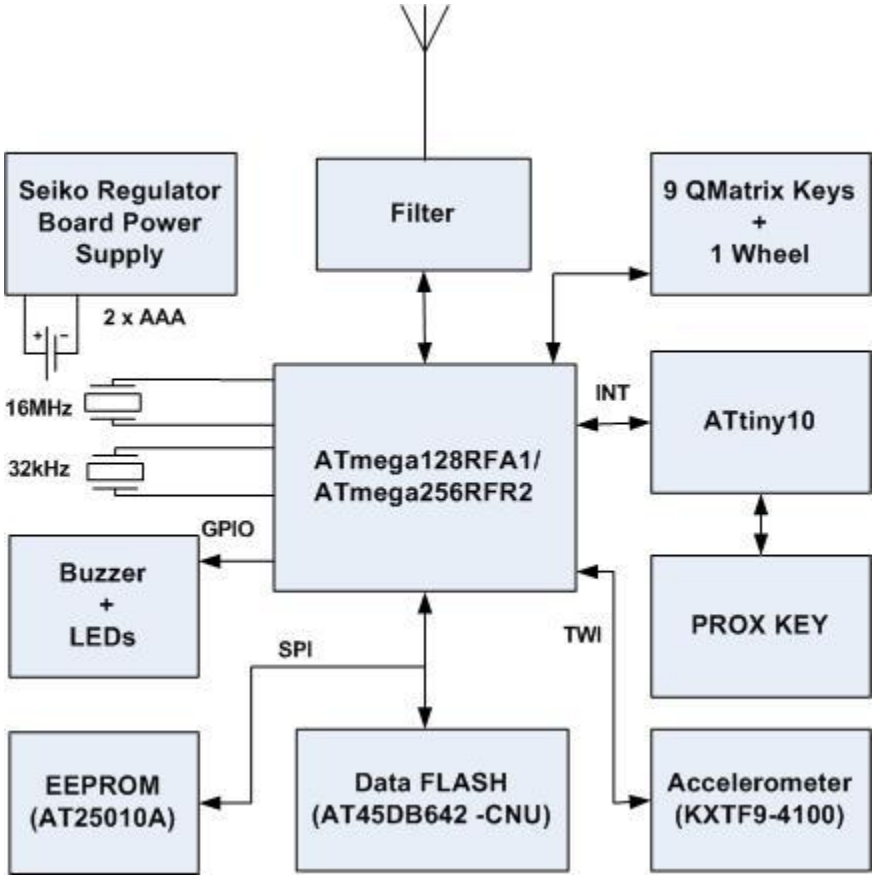
Figure 1-1 shows the AVR477 remote board. The board has nine touch keys and a wheel for user interface based on the QTouch technology patented by Atmel. It consists of an ATmega256RFR2 (changed from ATmega128RFA1 in actual AVR477 reference design for this application) and peripheral circuitry. An external EEPROM is used to store MAC address and additional board information to avoid accidental data erasure during microcontroller firmware development.

Figure 1-1. AVR477 RF Touch Remote Board



This board is used to detect the key touch or wheel position and communicate the corresponding key/wheel data to the remote device using RF Proximity. Accelerometer functionalities are also available by using Atmel ATtiny10 device and Kionix KXTF9 respectively. The backlight LEDs and buzzer can be controlled by the application for various user indications such as key touch, proximity, sleep, and fault indications.

Figure 1-2. System Block Diagram of AVR477 RF Touch Remote Control Board



Other features include; external EEPROM, external data Flash, 32KHz crystal, 16MHz transceiver crystal, JTAG programming header, and jumper provision for current measurement. Control for switching (ON and OFF) accelerometer and proximity controller for better power consumption. All components are placed on one side of the PCB to demonstrate a low-cost manufacturing solution. Atmel ATmega256RFR2 [4] is a System on Chip (SoC) that integrates a powerful 8-bit AVR® RISC microcontroller, an IEEE® 802.15.4-compliant transceiver, and additional peripheral features. The built-in radio transceiver supports the worldwide accessible 2.4GHz ISM band. The RF front-end implementation is kept minimal by using a PCB antenna. A dipole antenna designed to match ATmega256RFR2 is used.

RF Touch Remote Control Board hardware features:

- 2.4GHz RF hardware with PCB antenna
- Atmel ATmega256RFR2 centric design
- Nine capacitive touch (QTouch) keys
- One wheel with 8-bit resolution
- Proximity sensing
- 3D accelerometer
- External EEPROM and DataFlash
- TWI interface
- Buzzer control
- Backlight LED control
- Programming and debugging interfaces
- Battery powered (2 × AAA)

- Provision for current measurement
- Provision for external regulator (Seiko)

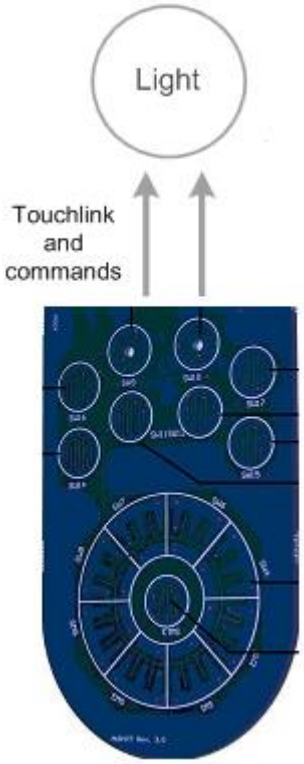
For detailed information on hardware features, refer to [1] and [2].

1.2 ZigBee Light Link Demo Application

The ZLLDemo reference application is part of the BitCloud SDK [5].

The reference application implements standard device types defined in the ZigBee Light Link Profile specification: Color scene controller, miscellaneous light devices, and bridge. Figure 1-3 shows a scheme of interactions between devices in a ZLL network. Pre-built firmware images are also provided for the Color light, Bridge, and Color Scene Controller.

Figure 1-3. Interactions Scheme in the ZLLDemo Application



However, the Color Scene Controller (Remote) device is built using Key Remote Control Board from RF4CE-EK [6] that has push buttons. Instead, using the AVR477 Remote as the ZLL controller offers the advantage of using the Atmel Capacitive Touch sensing technology for an intuitive user interface and to make the color scene controller an ideal end product for lighting control applications.

1.3 Package Content and Structure

The zip package (`AN1295_QTouch_BitCloud_Integration.zip`) provided with this application note contains:

- Application source files from standard ZLLDemo application that were modified with QTouch-application relevant changes
- BitCloud stack components – BSP and HAL files with changes suited for AVR477 remote control board
- Pre-compiled firmware image (`ZLLDemo_AVR477QTouchRemote_ATmega256RFR2_Remote.hex`)

Note: This package needs to be extracted and the files replaced into the BitCloud SDK [5] to get a fully functional project.

Table 1-1 describes the zip package directory structure with paths relative to the package's root directory.

Table 1-1. AN1295_QTouch_BitCloud_Integration.zip Files and Directories

Path	Description
\Evaluation Tools\ZLLDemo	Pre-compiled firmware image of the ZLL application with QTouch support
\Applications\ZLLDemo\	ZLL application files that are changed for QTouch support
\Applications\ZLLDemo\ColorSceneRemote	Color scene remote source code files with QTouch support
\BitCloud\Components\BSP	Board specific source code for the AVR477 remote control board
\BitCloud\Components\BSP\AVR477QTouch\src	QTouch specific sources: avr477QTouch.c memorymap.c qm_asm_tiny_mega.S Proximity sensor source: t10_prox.c
\BitCloud\Components\HAL	HAL files that required change for adding QTouch support, specific to the AVR477 remote control board
\BitCloud\lib	HAL library: libHAL_Avr477Atm256rfr2_Atmega256rfr2_8Mhz_Iar

Note: BitCloud stack libraries are not modified for this project and the default libraries provided in the BitCloud SDK shall be used.

2. Getting Started

2.1 Supported Platforms and Tools

Supported platforms and development tools are described in Table 2-1.

Table 2-1 System Requirements for BitCloud SDK

Parameter	Value	Note
MCU/RF	Atmel ATmega256RFR2 SoC	
Evaluation Board	AVR477 RF Touch Remote Control Board with ATmega256RFR2 SoC	The full reference design including the schematics and Gerbers are available in [2]
JTAG emulator	Atmel AVR JTAGICE mkII or Atmel JTAGICE 3	Required to upload firmware onto the boards and debug through JTAG
IDE	IAR Embedded Workbench® for AVR 6.40.3 (with C/C++ compiler 6.40.3)	Required to develop applications using API and to upload firmware images through JTAG

2.2 Running the Demo with the Pre-built Application Image

As this application note is an enhancement of the BitCloud ZLL(ZigBee Light Link)Demo, the content, structure, and recommended fuse settings are the same as in the BitCloud SDK package [5].

A major change would be the board support for the AVR477 Touch Remote instead of the default key remote controller board supported in the standard BitCloud ZLL reference application.

This application is typically used in the following way:

1. Download the BitCloud SDK package and unpack to a folder on the PC hard drive.

2. Copy the `Applications`, `BitCloud`, and `Evaluation Tools` folders from the zip package (AN1295_QTouch_BitCloud_Integration.zip) provided with this application note and paste them in the corresponding folders in the BitCloud SDK package [5]. Overwrite existing files if asked.

Now, the integration is complete and the changes relevant to this application note should be available in the BitCloud SDK.

3. Assemble the Color Light device hardware [9] and program it with the `Evaluation Tools\ZLLDemo\ ZLLDemo_KeyRemote_ATmega256RFR2_Light.hex` file.

4. Power up the AVR477 RF Touch Remote Controller board using two AAA batteries and a battery holder.

5. Program the AVR477 RF Touch Remote Controller board with the precompiled hex file `ZLLDemo_AVR477QTouchRemote_ATmega256RFR2_Remote.hex` that is present in the `Evaluation Tools\ZLLDemo` folder. The recommended fuse bits settings for the RF Touch remote control board is same as given in the BitCloud SDK Quick Start Guide (found in the BitCloud SDK \Documentation directory). As a quick reference, `0xFE 0x19 0x62` can be used as the resulting bytes for the fuse settings.

6. Switch on the Color light device and AVR477 RF Touch remote controller. Initially, FN will be displayed on the Color light, indicating that it is in factory new state.

7. Keep Color Light and AVR477 Remote Controller nearby (within 20cm) and touch the SW9 key on the AVR477 Remote controller to start the touchlinking procedure. The LCD on the color light will blink and status on the color light LCD will change to NFN (Non Factory New). If NFN is displayed on the color light then the touchlinking procedure is successful. If the touchlinking procedure is not successful, then perform power on reset on both the devices and repeat the touchlinking procedure.

8. Once the touchlinking has succeeded, touch the SW16 key on the AVR477 Remote to send the ON command to the paired color light. SW14 can be touched to switch off the color light. The On/Off status of the light is indicated on the top left corner of LCD in color light.

9. Other commands from the AVR477 touch key remote controller can also be sent to Color Light as described in Chapter 3.

10. There are two LEDs available on the AVR477 remote, placed behind SW9 and SW10. Both LEDs toggle everytime proximity detection happens. When a user touches a key or the rotor, LED0 (behind SW9) is toggled. When SW10 is touched in specific, LED 1 is toggled.

11. A sniffer tool [10] can be used to view the data exchange between the Color scene remote and the color light.

3. Using AVR477 Remote as ZLL Color Scene Controller

3.1 Color Scene Controller's Main Commands

Figure 3-1 and Table 3-1 lists Touch keys for the main color scene controller's commands. Since the number of touch keys on AVR477 board is less than the buttons on the red key remote control board, only selected functionalities are implemented on the AVR477 board. But, note that the touch keys can also be used to implement other commands present on Red Key Remote Control Board.

Figure 3-1. Interactions Scheme in the ZLLDemo Application

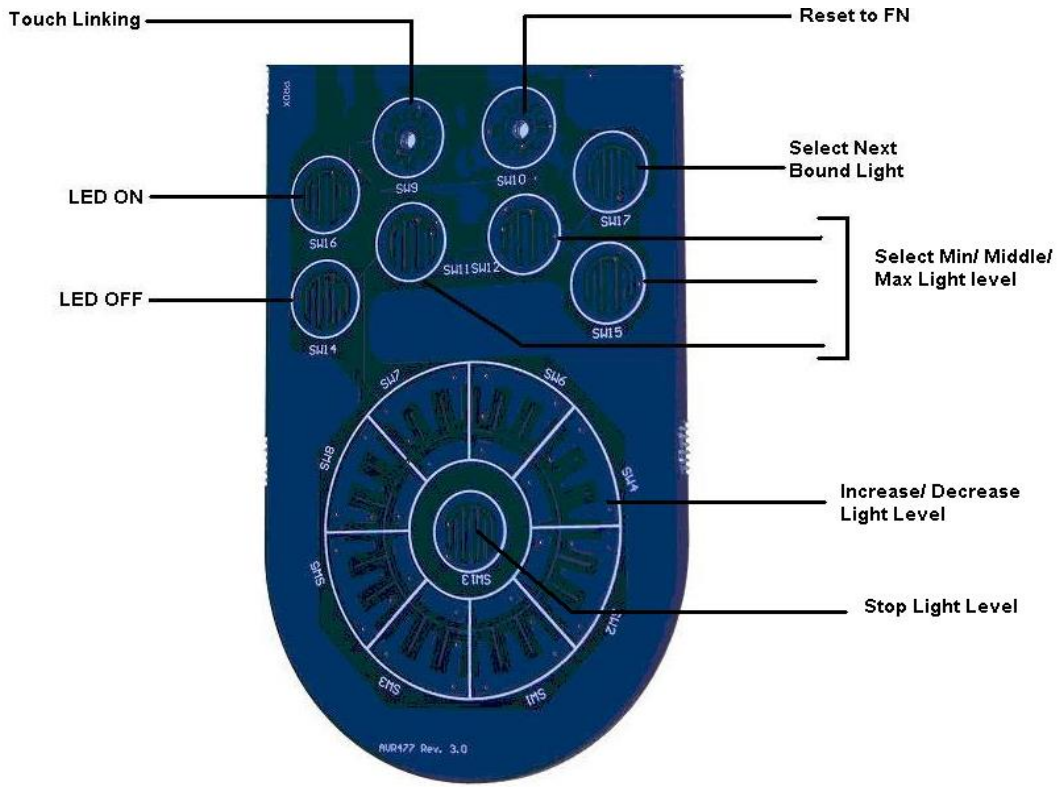


Table 3-1 Touch Keys for Executing ZLL Controller Commands

Key	Description
SW9	Initiate touchlinking procedure
SW10	Reset device to factory new state
SW16	On Command
SW14	Off Command
SW17	Select the next bound device and requests it to identify itself. This allows sending unicast commands to a single device. Groupcast mode will be entered automatically after five seconds of inactivity.
SW11	Set light level to minimum brightness (1)
SW12	Set light level to medium brightness (127)
SW15	Set light level to maximum brightness (255)
Rotor (SW1 – SW8)	Increase/decrease light level
SW13	Stop increasing/decreasing light level

3.1.1 Touch Link / Forming a Network

For touch linking, a color scene controller is brought close to a new light, which is not in the network yet. By pressing SW9 key on a color scene controller, the user initiates a commissioning procedure, whose goal is to transfer network parameters to the light. At this moment the devices are not yet in the network, but the communication is possible, because it happens on the MAC level without a need for routing.

The light receives network parameters and starts the network as a router. Once the network is started, the color scene controller joins this network as an end device.

When the first light is commissioned, the color scene controller may be brought near the second light, which, on receiving the network parameters, will not start the network once again, but will join the existing network as another router. The subsequent lights are commissioned in the same way.

3.1.2 Reset to Factory New State

To reset a color scene controller device to the factory new state, touch the SW10 key. Both LEDs will blink once to indicate that the device has been reset to the factory new state.

3.1.3 Selecting Next Bound Device

The SW17 button may be used to select a light. When this button is pressed and released, the next light becomes active, which means that all commands initiated by the user are sent to this light in a unicast manner. Groupcast mode will be entered automatically after five seconds of inactivity.

3.1.4 Controlling a Light Device – On/Off Commands

When key SW16 or SW14 is pressed, a command to switch on or off Color Light is sent from the controller. The On/Off status of the light is indicated on the top left corner of LCD in the color light.

3.1.5 Controlling a Light Device – Light Level Control Commands

The Color Light's intensity (brightness) level can also be controlled by the AVR477 Remote controller. The intensity can vary between 1 and 255. Touch key SW11 to set the intensity level to minimum (1). SW12 and SW15 can be touched to set the intensity level to middle (127) and maximum (255) respectively. The rotor can be used to increase or decrease the intensity level from the current level.

4. Integrating QTouch Library into BitCloud ZLLDemo Application

This chapter describes the general steps (each marked with **Step X:**) required to create a ZLL QTouch controller application based on the ZLLDemo application in the BitCloud SDK. As an example, it lists the changes done in the zip package (`AN1295_QTouch_BitCloud_Integration.zip`) provided with this application note to support the AVR477 controller board. Thus the first step is:

Step 1: Download BitCloud SDK package [5].

Unpack it to a folder on the PC hard drive. All further modifications and additions shall be done within this package.

4.1 Selecting the Correct QTouch Library

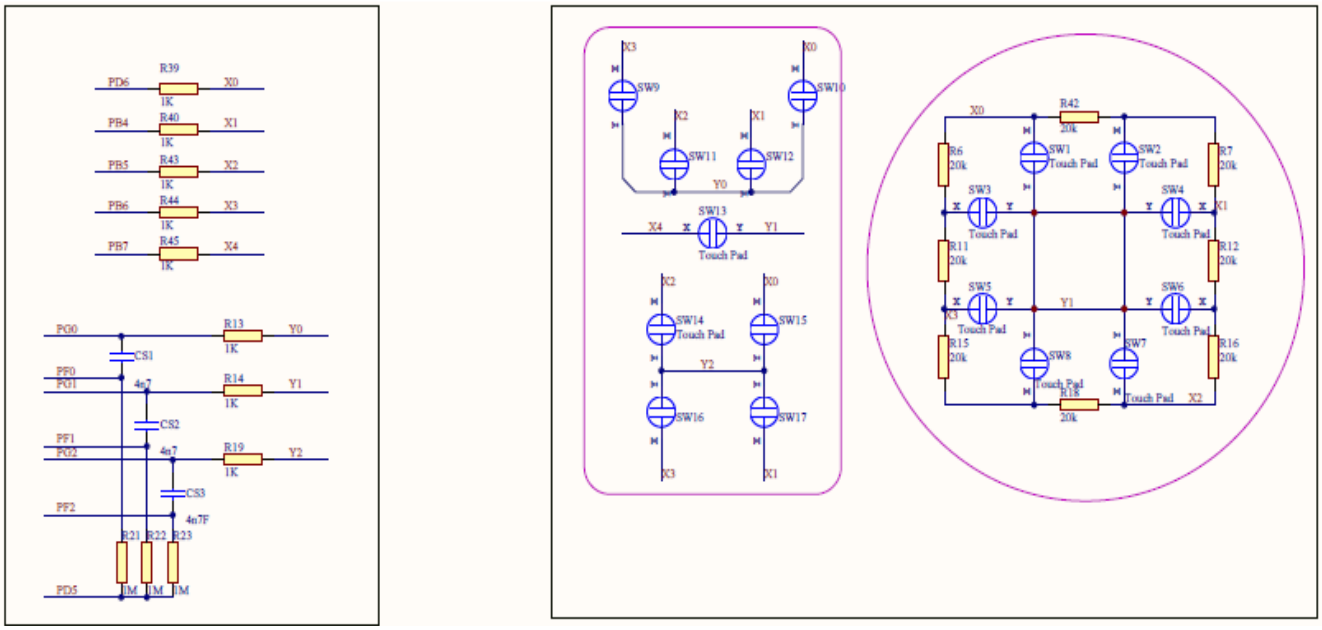
This section describes the methodology involved in choosing an appropriate QTouch Library based on the design.

Note that, for the AVR477 Remote Controller board, the QTouch Library is available in the zip package (`AN1295_QTouch_BitCloud_Integration.zip`) in **`\BitCloud\Components\BSP\AVR477QTouch\lib`**. Hence, the information below on choosing the QTouch Library is for reference purpose only.

The Atmel QTouch Library User Guide [7] is the starting point for selecting the QTouch Library most relevant to the application requirements such as the microcontroller used, the number of touch sensing channels needed, the acquisition method, pin availability and configuration, covered in the QTouch Library user guide.

This section illustrates the steps required to select the right QTouch acquisition method library variant and configuration for your application. Follow the steps below to integrate the QTouch library in the application. The AVR477 schematics [2] can be refered to for the port pin configuration. The various touch keys and their pin connections on the AVR477 board are given in Figure 4-1.

Figure 4-1. AVR477 Board Pin Mapping for Touch Keys



Step 2: Select the correct QTouch Library variant.

The Atmel QTouch Library can be downloaded from the Atmel website [8].There are specific QTouch Library variants distributed for each Atmel microcontroller. Once the QTouch Llibrary package is installed, the library files can be found in **`\Generic_QTouch_Libraries\AVR_Tiny_Mega_XMega\QMatrix\library_files.`**

For a specific touch design, one needs to configure or select the following parameters to finalise on the correct library variant:

- **Target MCU to be used for the design**

  AVR477 board design is based on ATmega128RFA1. However, since the ZLLDemo application is based on the ATmega256RFR2, a chip replacement can be done on the board. Both microcontrollers are pin to pin compatible and similar in functionality.

- **The number of QTouch sensing channels needed by the application**

  Refering to the AVR477 board schematic, there are 17 keys (SW1 to SW17); hence minimum a number of 17 channels are needed. For flexibility of usage, the accelerometer positions are mapped to certain channels. This application requirement is thus satisfied by a maximum of 32 channels.

- **Number of X-lines to be used in the design**

  Refering to the AVR477 board schematic, there are five port-pin lines (X0, X1, X2, X3, and X5) where X-lines are permitted.

- **Number of Y-lines to be used in the design**

  Refering to the AVR477 board schematic, there are a total of three port-pin lines (Y0, Y1, and Y2) where Y-lines are permitted.

- **Rotor Slider support**

  Refering to the AVR477 board schematic, rotor slider support is available and one rotor is supported.

- **Compiler platform**

  BitCloud ZLL Profile Suite Application Demo uses IAR™ compiler, therefore the QTouch libraries for IAR are needed.

Based on the above parameters, for the AVR477 board, the appropriate QTouch library with QMatrix acquisition method is: **`libm256rfr2_32qm_8x_4y_krs_4rs.r90`** that supports ATmega256RFR2 device with maximum of: 32 channels, eight X-lines, four Y-lines, and one key-rotor-slider with support for up to four rotor sliders. Selected library and associated files shall be added to the application project as described in Section 4.6.

4.2 Configuring the QTouch Interface

Once the QTouch library variant has been chosen, the next step is to define certain constants and symbols required in the host application files where the touch API is to be used. These values are derived from the parameters defined in **`step 2`**.

These definitions and configuration parameters need to be present in the project space to ensure that QTouch library operates correctly.

Step 3: Define QTouch constants/symbols in the application project space.

```
#define _QTOUCH_
#define QT_NUM_CHANNELS       32    /* Library with maximum 32 channel support */
#define NUM_X_LINES           5     /* X lines are X0, X1, X2, X3, X4 */
#define NUM_Y_LINES           4     /* Y lines are Y0, Y1, Y2 */
#define NUM_X_PORTS           2     /* Port D and Port B */
#define PORT_X_1              D     /* First X Port is connected to Port D */
#define PORT_NUM_1            1     /* First X Port is indexed as 1 */
#define PORT_X_2              B     /* 2nd X Port is connected to Port B */
#define PORT_NUM_2            2     /* 2nd X Port is indexed as 2 */
#define PORT_YA               G     /* Port YA is connected to Port G */
#define PORT_YB               F     /* Port YB is connected to Port F */
#define PORT_SMP              D     /* Port SMP is connected to Port D */
#define SMP_PIN               5     /* SMP Pin is connected to PD5 */
#define QT_DELAY_CYCLES       2
```

```
#define ROTOR_SLIDER_
#define SHARED_YAYB           0
#define QT_MAX_NUM_ROTORS_SLIDERS 4
```

Note: For AVR477 board the above definitions are available in `touch_config.h` file in `BitCloud\Components\BSP\AVR477QTouch\include\` folder.

Step 4: Fill the line masks.

Line masks need to be filled for the X- and Y-lines to provide a mapping between the sense lines and the port pins.

Fill in the `arrays x_line_info [NUM_X_LINES], ya_line_info[NUM_Y_LINES]` and `yb_line_info[NUM_Y_LINES]` according to the pin configuration in your design.

According to the pin availability for the touch sensing, the arrays for AVR477 board are initialized in `BitCloud\Components\BSP\AVR477QTouch\src\avr477QTouch.c` file as shown below:

```
X_line_info_t x_line_info[NUM_X_LINES]= {
FILL_OUT_X_LINE_INFO(1,6),          /* 1st X line is connected to PD6 */
FILL_OUT_X_LINE_INFO(2,4),          /* 2nd X line is connected to PB4 */
FILL_OUT_X_LINE_INFO(2,5),          /* 3rd X line is connected to PB5 */
FILL_OUT_X_LINE_INFO(2,6),          /* 4th X line is connected to PB6 */
FILL_OUT_X_LINE_INFO(2,7)           /* 5th X line is connected to PB7 */
};

y_line_info_t ya_line_info[NUM_Y_LINES]= {
FILL_OUT_YA_LINE_INFO(  0u ),       /* 1st Ya line is connected to PG0 */
FILL_OUT_YA_LINE_INFO(  1u ),       /* 2nd Ya line is connected to PG1 */
FILL_OUT_YA_LINE_INFO(  2u ),       /* 3rd Ya line is connected to PG2 */
FILL_OUT_YA_LINE_INFO(  3u ),       /* 4th Ya line is connected to PG3 */
};

y_line_info_t yb_line_info[NUM_Y_LINES]= {
FILL_OUT_YB_LINE_INFO(  0u ),        /* 1st Yb line is connected to PF0 */
FILL_OUT_YB_LINE_INFO(  1u ),        /* 2nd Yb line is connected to PF1 */
FILL_OUT_YB_LINE_INFO(  2u ),        /* 3rd Yb line is connected to PF2 */
FILL_OUT_YB_LINE_INFO(  3u ),        /* 4th Yb line is connected to PF3 */
};
```

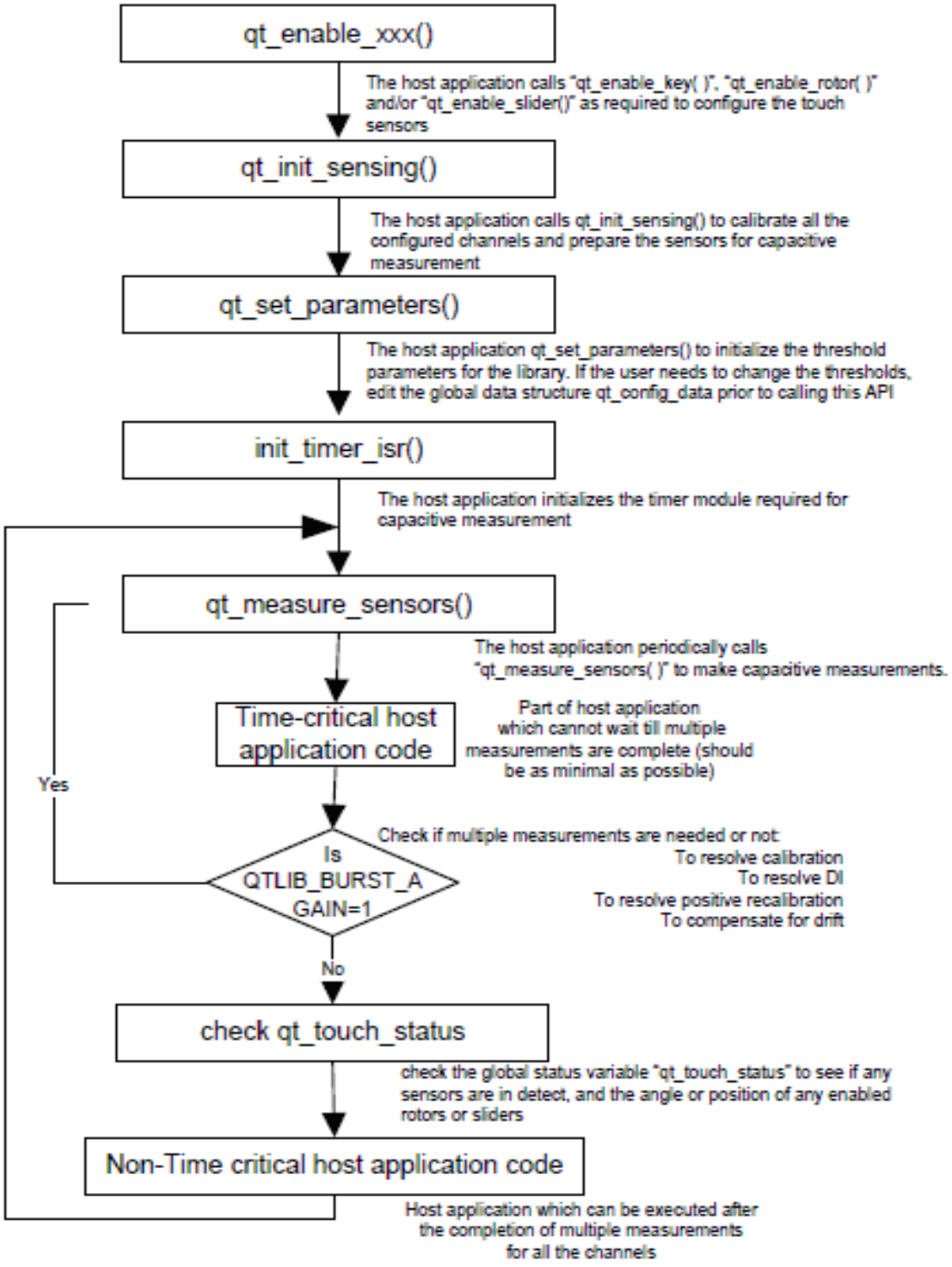
4.3 Using QTouch Library API in the Application

After the above steps are done, the QTouch Library APIs can be used to create, initialize, and perform touch sensing.

Step 5: Initialize and use QTouch sensing API.

Figure 4-2 shows the sequence of QTouch API calls for touch sensing control.

Figure 4-2. Sequence of QTouch API Calls to Support Touch Sensing



For the AVR477 board, all touch related functionality including the QTouch Library files are added as part of the BitCloud BSP component as described in Section 4.5.3. The AVR477 board specific behavior is implemented in the `BitCloud\Components\BSP\AVR477QTouch\src\avr477QTouch.c` file. Initialization is done in the `BSP_InitQTouch(BSP_TouchEventHandler_t handler)` function that gets called by the application code. The touch routine `qt_measure_sensors()` is called inside the `BSP_TaskHandler()` repeatedly every

qt_measurement_period_msec interval. To simplify integration into the ZLLDemo application, once the key touch is sensed it is mapped to a corresponding red key remote button [9] and this button value is passed on to the application appButtonsInd() function for further processing. Thus the code in the ZLL color scene remote application needs very minimal changes.

For example: AVR477 key SW9 can be mapped to the PWR button on the red key remote control board [9] to initiate the touchlinking procedure. A typical mapping will look like:

```
#define KEY_SW09    BUTTON_PWR     /*Key SW9 is mapped to PWR button on red remote. */

#define KEY_SW16    BUTTON_LEFT_P /* Key SW16 is mapped to L+ button on red remote */
```

Refer to the BitCloud\Components\BSP\AVR477QTouch\src\avr477QTouch.c file for the implementation details.

4.4 Proximity Sensing

The AVR477 has a proximity sensing feature controlled by the ATtiny10 device on the AVR477 Remote hardware. When a user approaches the remote control, proximity is detected and an external interrupt is issued from the ATtiny10 MCU to the ATmega256RFR2.

The build switch PROX_ENABLE needs to be added to the IAR pre-processor options to enable proximity sensing.

The ZLLDemo application uses Sleep When Idle feature and if proximity sensing is enabled, the ATmega256RFR2 sleeps when there are no pending tasks and wakes up on proximity interrupt. The system remains awake as long as the user holds the remote or performs touch activity.

4.5 Modifying BitCloud SDK

4.5.1 BitCloud Core Stack

No modifications are required in BitCloud core stack components that generate core BitCloud library (APS, ZDO, NWK, MAC_PHY, and Security components). Thus access to BitCloud source code is not needed.

4.5.2 BitCloud HAL Component

BitCloud HAL component include the MCU-specific code and generally no QTouch-specific changes are required to support QTouch. However, in some cases board-specific design might put special needs for modifications in HAL as well (for example, if MCU pins that are used by BitCloud are needed for application purposes). This shall be analyzed for each custom board and application.

For example, modifications in HAL done to support AVR477 board were put together into a new HAL platform configuration (PLATFORM_AVR477) that covers the following changes specific to this board:

- Setting HAL_RF_RX_TX_INDICATOR = FALSE in BitCloud\Components\HAL\Configuration file. This is done as external RF PA control is enabled in the default HAL configuration, but AVR477 board uses the same pins for configuring the touch keys.
- In halInit.c, the line halSetIrqConfig(IRQ_4,IRQ_LOW_LEVEL); is not compiled for this configuration as this interrupt line is connected to pin PE4 and the AVR477 board uses this pin to configure the LEDs.
- Rebuilding the HAL library for PLATFORM_AVR477 (by executing command **make clean all** in the command prompt from the BitCloud\Components\HAL directory) generates a new HAL library libHAL_Avr477Atm256rfr2_Atmega256rfr2_8Mhz_Iar.a. This is done to avoid the overwriting of the default HAL library used by non-QTouch based ZLL devices.

4.5.3 BitCloud BSP Component

BSP component includes API and code specific to a particular board (e.g. buttons, LEDs, LCD controls, etc.). If the board design is significantly different when compared to the supported Atmel kits, it is recommended to create a custom board configuration in the BSP component. Thus, for the AVR477 board, a new configuration `BOARD_AVR477_QTOUCH` was added with the corresponding code in `Components\BSP\AVR477QTouch` that includes QTouch Library, its header, and configuration files as well as application-specific handling of touch events (`avr477QTouch.c` file). For information on QTouch Library configuration and API usage, see Sections 4.2 and 4.3.

The QTouch-specific action required is:

Step 6: Add the QTouch Library and related code files to the `BitCloud\Components\BSP\<Board>\` folder.

4.6 Updating Application

Modifications in BitCloud ZLL application are required for including the QTouch-related code into the project space as well as initialization and handling of the touch events.

4.6.1 Application Code

Generally it is also possible to have QTouch Library initialization and sensing API directly in the application code. In such case this step can be skipped as the application would be already updated as part of Step 5 described in Section 4.3. However, to abstract from the board-specific configurations we recommend to keep it in BSP component and share common API to the application. Thus the QTouch-specifc action required is:

Step 7: Add QTouch initialization and event handling based on the API added to the BSP (see Section 4.5.3).

In the zip package (AN1295_QTouch_BitCloud_Integration.zip) provided with this application note, the ZLLDemo application was extended with `BOARD_AVR477_QTOUCH` define that covers QTouch initialization and handling for the AVR477 board.

Additionally some non-touch related modifications might be required depending on the board and application design. Thus since the AVR477 has no integrated UID chip as the RCB board has, the `CS_UID` parameter needs to be initialized to non-zero value in `configuration.h` file and need to be set unique for each AVR477 board present in the network.

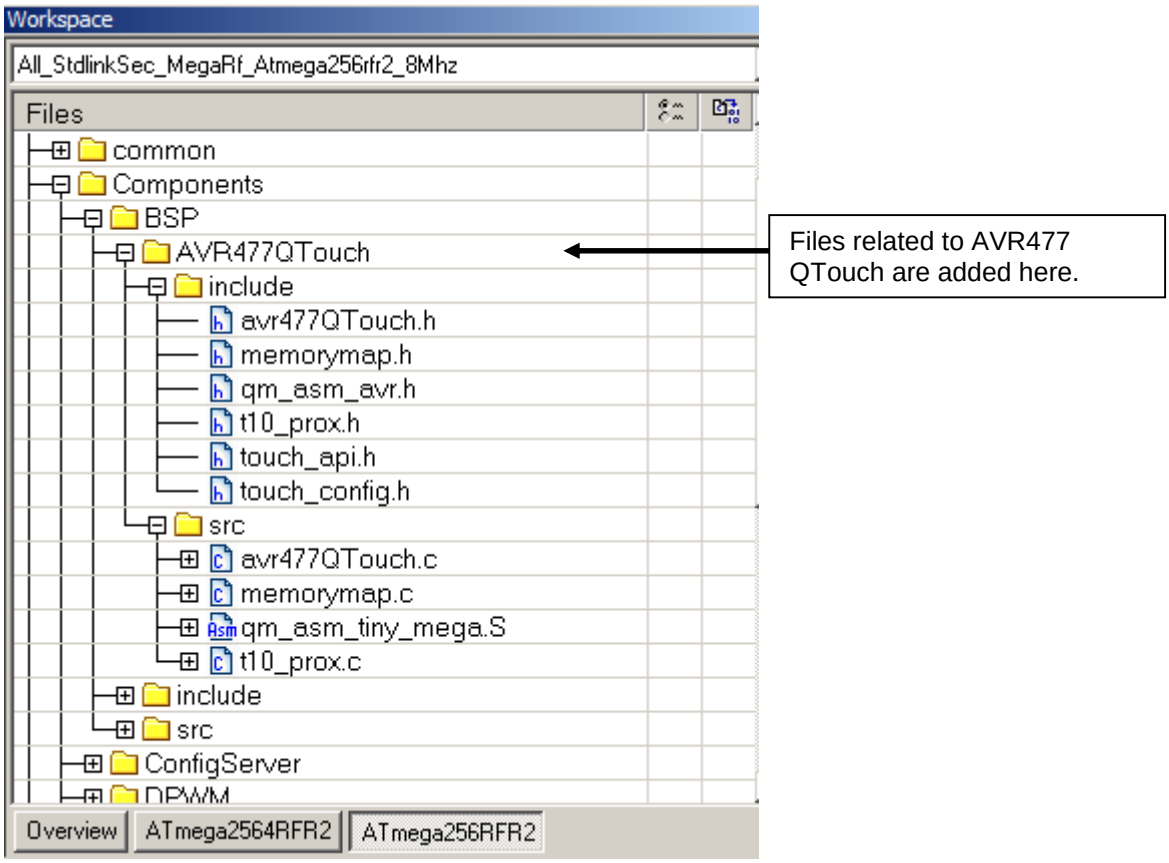
4.6.2 IAR Application Workspace

Application project configuration shall include the new QTouch-related code added to the BSP (see Section 4.5.3) for compilation as well as updated. Here are the QTouch-specifc actions required:

Step 8: Modify the IAR Workspace to add the QTouch functionality to ZLLDemo.

Open the ZLLDemo.eww IAR workspace and choose the `All_StdlinkSec_MegaRf_Atmega256rfr2_8Mhz` project configuration. In the project, go to `Components->BSP` and add a new group called `AVR477QTouch`. Create `include` and `src` folders in `AVR477QTouch` and include the header files and source from `Components\BSP\AVR477QTouch` provided in the zip package (AN1295_QTouch_BitCloud_Integration.zip) (see Figure 4-3).

Figure 4-3. Placement of AVR477 QTouch Code in the IAR Workspace



Step 9: Add include path for QTouch related header files in Compiler options.

Figure 4-4 shows how this is done for AVR477 in the zip package (`AN1295_QTouch_BitCloud_Integration.zip`) provided with this application note. Additionally the `BOARD_AVR477_QTOUCH` and `PROX_ENABLE` defines are added to the list of defines for this configuration.

Figure 4-4. Modifying the Compiler Options for the New Configuration

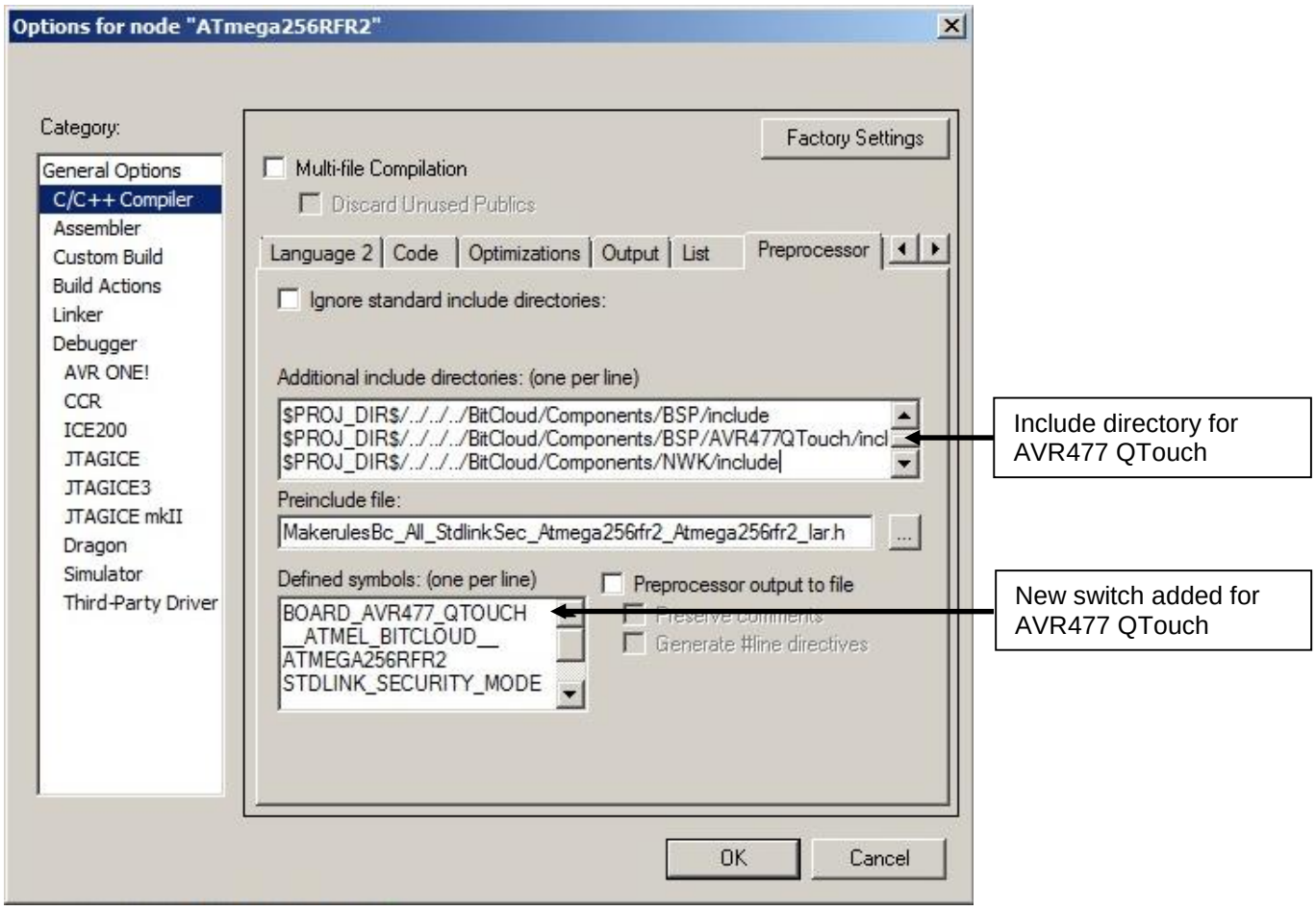
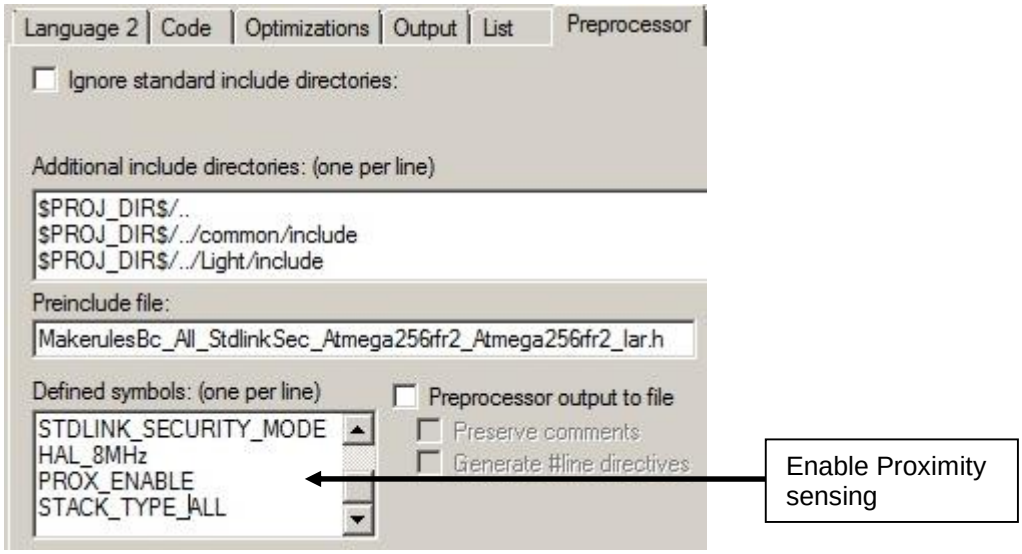


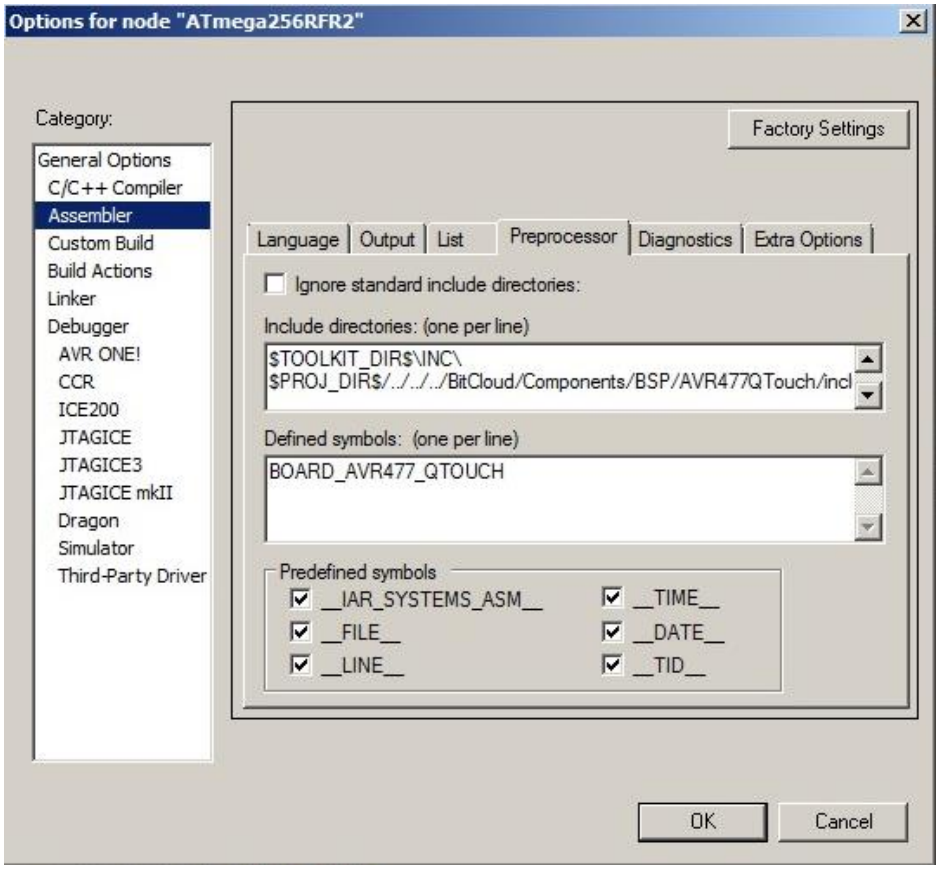
Figure 4-5. Enabling Proximity Sensing



Step 10: Update assembler options of the project configuration to contain the path to QTouch header files.

Figure 4-6 shows how this is done AVR477 in the zip package (`AN1295_QTouch_BitCloud_Integration.zip`) provided with this application note.

Figure 4-6. Modifying the Assembler Options for the New Configuration

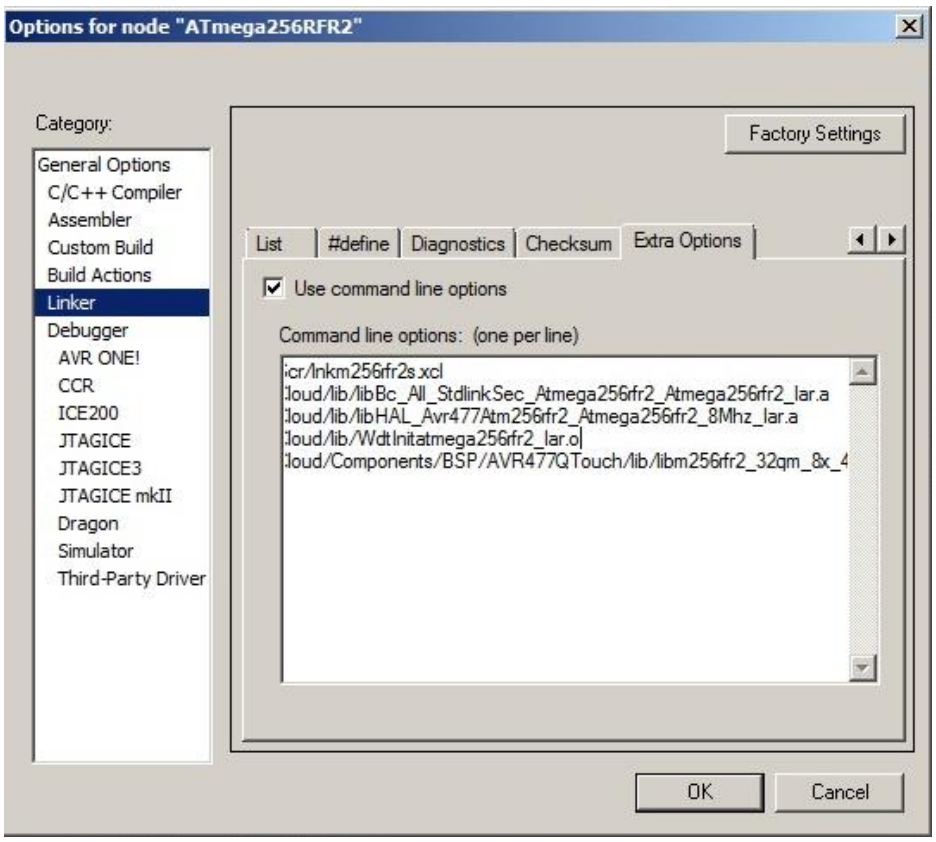


Step 11: Update Linker options.

Linker options need to be updated with the HAL library specific to the AVR477 hardware and to include the QTouch library.

Figure 4-7 shows how this is done AVR477 in the zip package (`AN1295_QTouch_BitCloud_Integration.zip`) provided with this application note.

Figure 4-7. Modifying the Linker Options for the New Configuration



Step 12: Ensure that the default BSP_TaskHandler is under the build switch BOARD_AVR477_QTOUCH.

As the project defines its own BSP_TaskHandler to handle the QTouch measurements, add conditional check `#ifndef BOARD_AVR477_QTOUCH` to the default BSP_TaskHandler in `Components\BSP\src\bspTaskManager.c`.

The modified BSP_TaskHandler() that handles touch measurements is available in `Components\BSP\AVR477QTouch\src\avr477QTouch.c`

Once all the above steps are done, rebuild the project.

The hex file `ZLLDemo_ATmega256RFR2_Controller_AVR477_QTouch.hex` will be created and can be programmed on the AVR477 RF Touch remote board.

4.7 Resources used by the QTouch Library

The QTouch Library use minimal resources of the microcontroller. The sampling of the sensors is controlled by the QTouch Library, while the sampling period is controlled by the application. For timing measurements, the application is using `TIMER0` as the interrupt source.

The QTouch acquisition method libraries internally use `TIMER1` for the operation; `TIMER1` will not be available for critical sections of the code where the library is called. This is available to the host application when the user's normal application is running.

QTouch Library disables interrupts for time-critical periods during touch sensing. These periods are generally only a few cycles long, and host application interrupts should remain responsive during touch sensing. However, any interrupt service routines (ISRs) during touch sensing should be as short as possible to avoid affecting the touch measurements or the application responsiveness. For more details, refer to the Atmel QTouch User Guide [7].

The time to execute a single measurement depends on various HW parameters such as sampling capacitor, operating voltage, and different software parameters such as `QT_DELAY_CYCLES`, CPU frequency, etc. But generally it is within 2 - 3ms. Since BitCloud system rule 3 says that 'All user callbacks should execute in 10ms or less' (as per BitCloud Developers Guide in the BitCloud SDK \Documentation directory), this is well within the range while also leaving enough timing for host application code to run.

5. References

[1] Atmel AVR477: RF4Control – Touch Remote Control (application note)

[2] Atmel AVR477 Software Package

[3] Touch Solutions

[4] Atmel ATmega256RFR2 Datasheet

[5] BitCloud – ZigBee PRO

[6] AVR2104: RF4CE-EK Remote Control Evaluation Kit - User Guide

[7] Atmel QTouch Library - User Guide

[8] Atmel QTouch Library

[9] ATmega256RFR2-EK

[10] Atmel AT02597: ZigBee PRO Packet Analysis with Sniffer (application note)

6. Revision History

Doc Rev.	Date	Comments
42173B	03/2015	Update proximity and LED behavior. Tested with BitCloud SDK version 3.2.
42173A	05/2014	Initial document release.

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**