Capacitive-Touch Remote Control Application Note

JN-AN-1170

Revision 1.1

8-Jan-2013

Contents

About this Document

This document describes a software algorithm that can be used in conjunction with the capacitive-touch keypad on NXP's RF Remote Control Unit (DR1159). The document is part of the Application Note JN-AN-1170, which also includes an example application based on the described algorithm.

Organisation

This document consists of 5 chapters and an appendix, as follows:

- Chapter 1 introduces the Remote Control Unit and the Application Note
- Chapter 2 details the hardware features of the Remote Control Unit
- Chapter 3 describes the operation of the capacitive-touch keypad and algorithm
- Chapter 4 describes the supplied example application which is based on the capacitive-touch algorithm
- Chapter 5 describes the firmware programming procedure for the Remote Control Unit
- The appendix describes the PCB/filter performance of the capacitive-touch keypad

Conventions

Files, folders, functions and parameter types are represented in **bold** type.

Function parameters are represented in *italics* type.

Code fragments are represented in the Courier typeface.

Acronyms and Abbreviations

API	Application Programming Interface
LED	Light Emitting Diode
PCB	Printed Circuit Board
RF	Radio Frequency
SDK	Software Developer's Kit
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

Related Documents

JN-RM-2056	USB Programming Dongle (DR1128) Reference Manual
JN-UG-3007	JN51xx Flash Programmer User Guide
JN-UG-3089	JenNet-IP EK040 Evaluation Kit User Guide
JN-UG-3093	JN516x-EK001 Evaluation Kit User Guide

Trademarks

1 Introduction

NXP's RF Remote Control Unit (DR1159) features a 16-key capacitive-touch keypad. The Application Note JN-AN-1170 (of which this document is a part) contains an example C application that can be used to facilitate the operation of the capacitive-touch keypad.

1.1 RF Remote Control Unit

The RF Remote Control Unit (DR1159) can be used as a node in a wireless network (e.g. in a JenNet-IP system), allowing other member nodes to be controlled from within the network. The design is based around an NXP JN51xx wireless microcontroller, mounted on a module.

The unit is supplied in the JenNet-IP EK040 and JN516x-EK001 Evaluation Kits, which each provides a stable platform for rapidly developing and testing applications that run on the JN51xx microcontrollers. The Remote Control Unit contains a JN5148-J01 or JN5168-001 device, depending on the evaluation kit.

The Remote Control Unit has the following features:

- 16 capacitive-touch buttons in a 4x4 keypad with configurable functions
- Separate 'Wake' button (below keypad) to bring unit out of sleep mode
- 2 green ultra low-power LEDs
- JN51xx module site
- Powered by 2 AAA batteries
- Firmware programming and debugging through header accessed via the battery compartment
- Designed to fit inside enclosure with part number 1397-120649 from AML Industrial Electronics Ltd

The battery compartment is accessed via a slide-cover on the rear of the unit. The required battery polarities are indicated on the inside of the compartment.

1.2 Application Note

The Application Note (JN-AN-1170) comprises a ZIP file containing the following:

- This document
- The schematic diagram for the board
- The BOM (Bill Of Materials) for the board
- Gerbers for the board
- Source file (**AN1170-Remote-Control.c**) of the capacitive-touch application, as well as a makefile and JN51xx binary executables

You must extract the contents of the ZIP file to your working area on your PC – for example, **<SDK_ROOT>\Application** where is **<SDK_ROOT>** is the installation directory of the relevant NXP Software Developer's Kit (SDK).

Details of the supplied application are provided in Chapter 3 and Chapter 4.

2 Hardware Details

2.1 Board layout

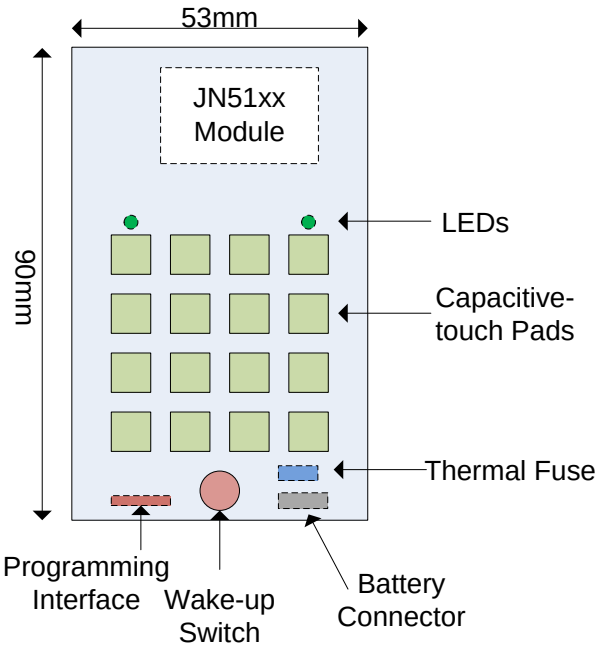

Figure 1: DR1159 Remote Control Board Layout

In the above diagram, components with a solid outline are on the front of the board and components with a broken outline are on the back of the board.

The board is designed to fit inside the case with part number 1397-120649 from AML Industrial Electronics Ltd, as depicted below.

Figure 2: Remote Control Unit in Case

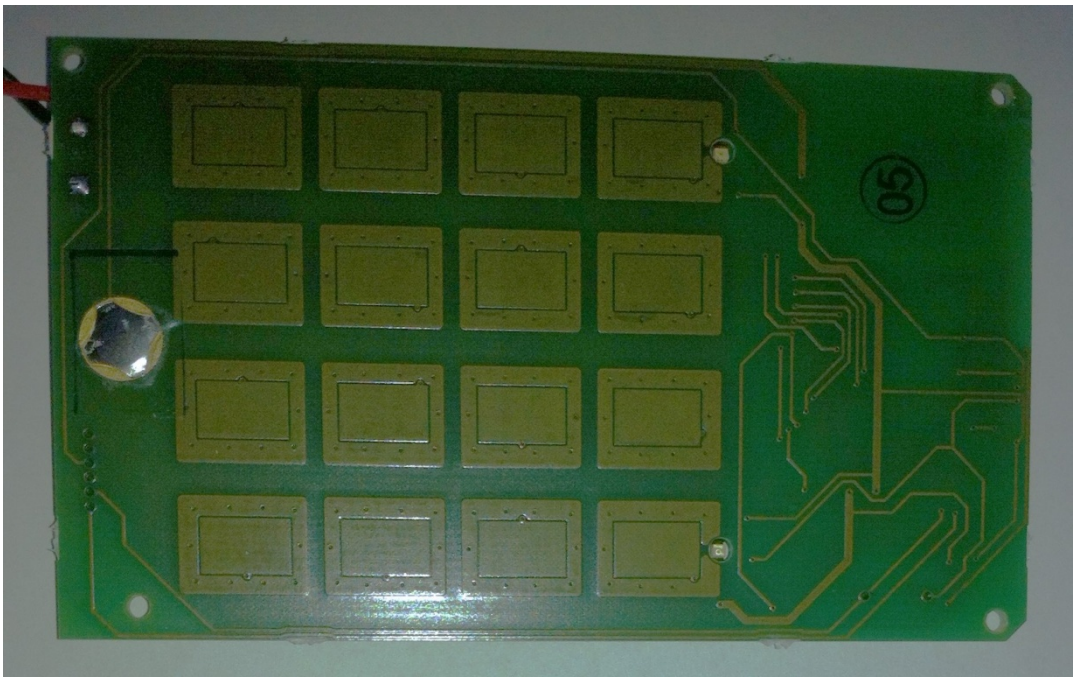

Figure 3: Front of Remote Control Board

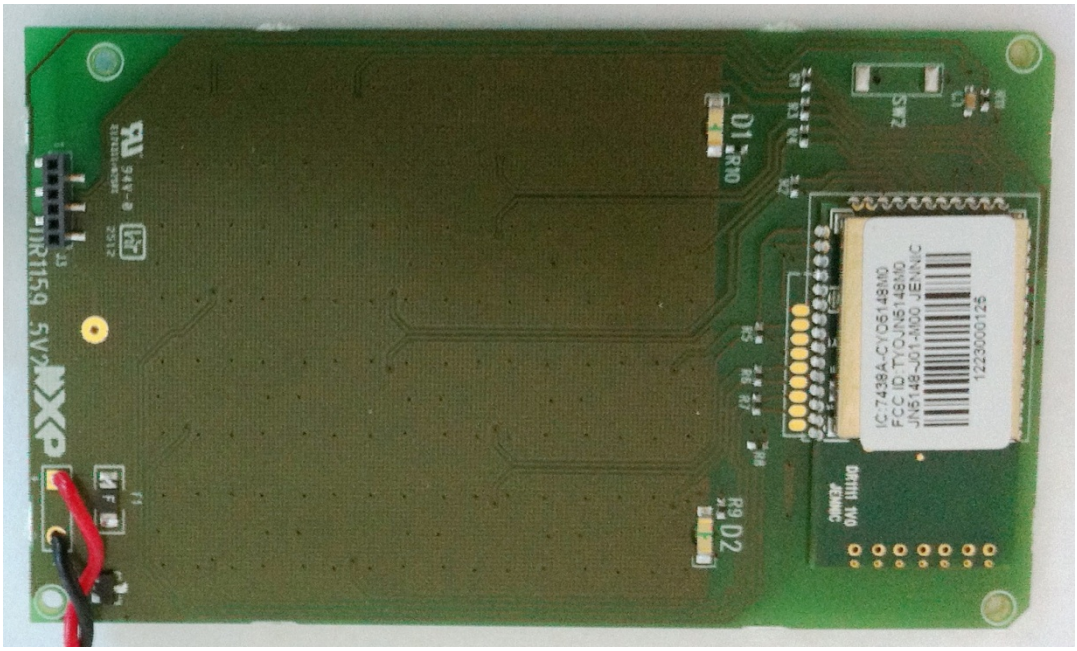

Figure 4: Back of Remote Control Board

2.2 JN51xx Module

The Remote Control PCB is designed to accept a standard JN51xx module. The unit supplied in the JenNet-IP EK040 Evaluation Kit contains a JN5148-J01 module and the unit supplied in the JN516x-EK001 Evaluation Kit contains a JN5168-001 module. However, a custom build of the Remote Control Unit can use any JN51xx module.

A JN51xx module has a small footprint and low component count. All RF layout and decoupling issues are handled by the design of the module, allowing it to be easily incorporated into systems by simply mounting the module onto a board.

The Remote Control Unit board provides a standard JN51xx module footprint, which conforms to normal well-understood low-speed digital design guidelines.

2.3 Batteries and Fuse

There are two connectors on the PCB for connecting the batteries. Please observe the polarity of these connectors. The batteries are connected through a thermal resettable fuse to protect against incorrect insertion of batteries.

2.4 Programming Interface

The connector J3 is the programming interface and employs 3.3V signals. This 6-pin connector is compatible with the USB Programming Dongle (DR1128), described in the Reference Design JN-RD-6021.

The pinout of this interface is detailed in the table below.

Pin	Name	Use
1	DIO6 (input)	Receive UART data, to connect to UART0TXD output from remote JN51xx, i.e. DIO6.
2	DIO7 (output)	Transmit UART data, to connect to UART0 RXD input of the JN51xx, i.e. DIO7.
3	MISO (output)	Control line to connect to SPIMISO of the JN51xx – used in conjunction with RESETN to put remote JN51xx into programming mode. It is only possible to cause this pin to go low, due to diode D5.
4	3V30	3.3V supply output from programming interface. This is capable of providing up to 40 mA.
5	RESETN (output)	Control line to connect to RESETN of the JN51xx - used in conjunction with MISO to put remote JN51xx into programming mode. Also used to reset the JN51xx after a programming cycle. It is only possible to cause this pin to go low, due to the diode D4.
6	GND	Ground

Table 1: Programming Interface Pins

> **!** ***Caution****: The programming interface connector is not keyed, so make sure that the USB Programming Dongle is oriented correctly. Incorrect insertion of the dongle may cause damage to the Remote Control Unit.*

2.5 Capacitive-Touch Keypad

The capacitive-touch mechanism and algorithm for the keypad are described in Chapter 3.

2.6 LEDs and Wake-up Switch

LEDs D1 and D2 are connected to DIOs 1 and 0 respectively. An LED can be illuminated by driving the relevant DIO low.

The wake-up switch is also connected to DIO0. When the Remote Control Unit is going to sleep, DIO0 should be set to an input with its pull-up on, ready for a wake-up event on DIO0. This also means that when the switch is pressed, LED D2 will illuminate.

3 Capacitive-Touch Keypad and Algorithm

3.1 Theory of Operation

The basic construction of the interface consists of a series of pads on the PCB which are connected to JN51xx DIO pins via a shared resistor per pad-pair. To determine the capacitance of a pad (both background and when touched), the associated DIO pin is configured as an input and the other DIO in the pad-pair used to charge and discharge the capacitive pad through the shared resistor.

Due to the rise/fall time of the RC circuit, there is a measurable delay between driving out a hi/lo and the JN51xx 'seeing' this as the voltage passes the relevant logic threshold on the DIO pin logic cell. This delay is directly proportional to the capacitance.

To determine the capacitance of the other pad, the DIO roles are reversed and the charge/discharge process repeated.

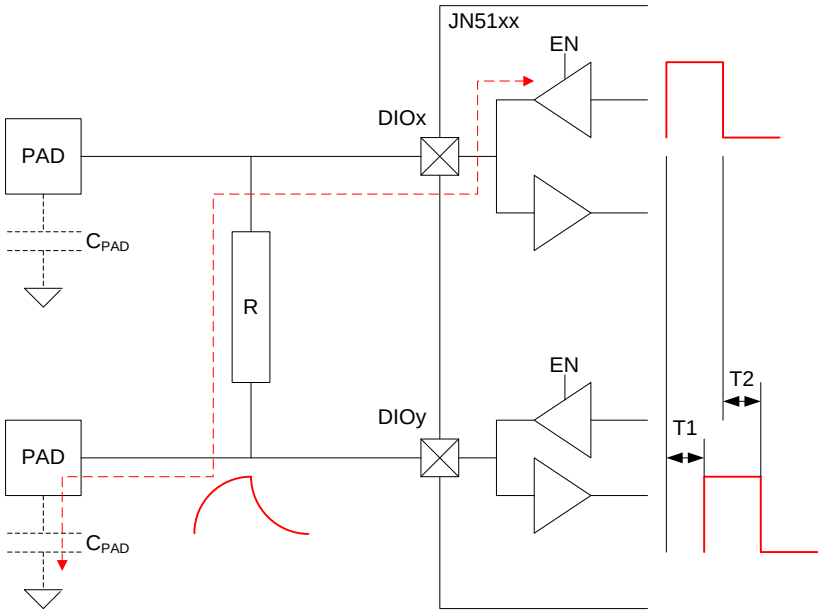

Figure 5: Capacitive-Touch Operation

3.2 Capacitive-Touch Software Architecture

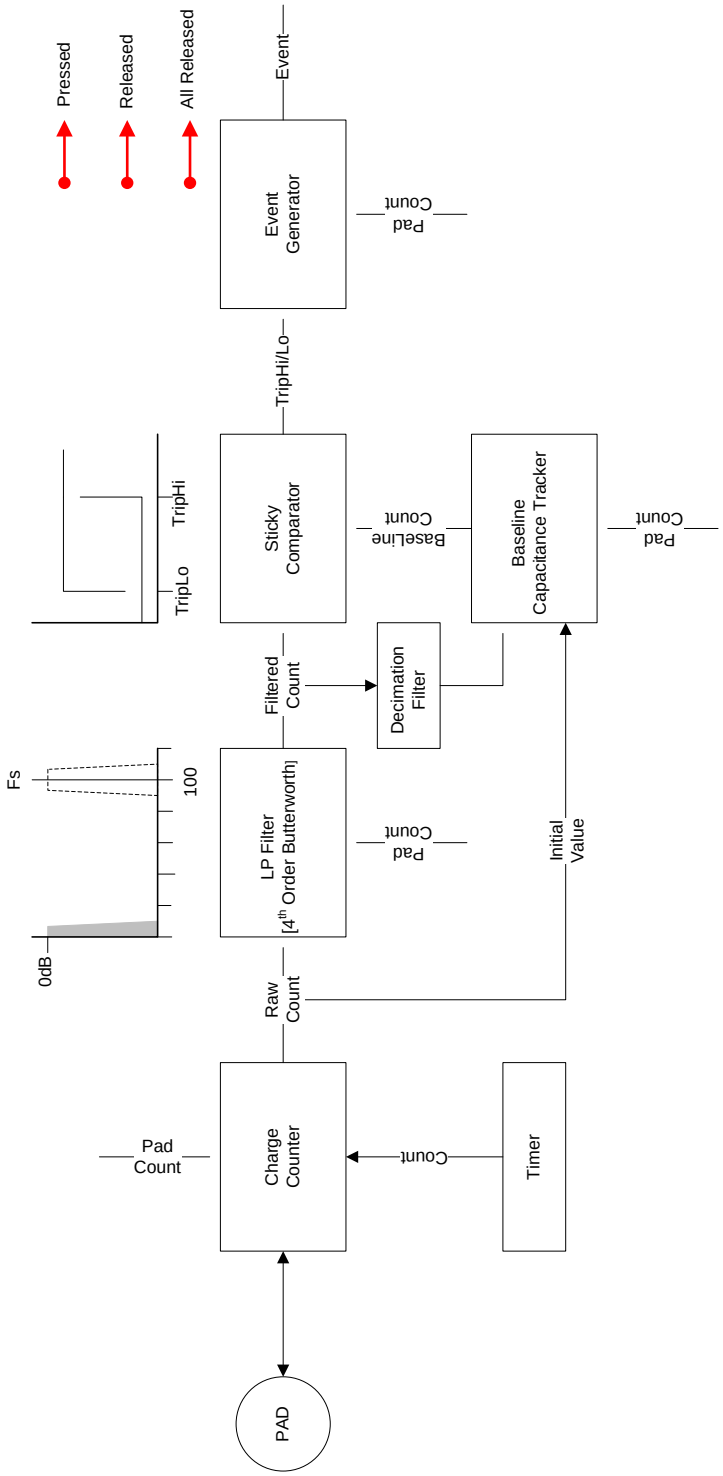

Figure 6: Capacitive-Touch Interface Block Diagram

3.3 Charge Counter

This process manages the charge/discharge of the capacitive pad and measurement of the time taken. A free-running 16-MHz counter is sampled when hi and lo are driven out, and when hi and lo are read back in. The two differences are summed giving a count proportional to the time and hence pad capacitance.

A bitmask is configured that defines which DIOs are used for touch-pad sensors. The number of bits must be even and the paired bits are adjacent to each other. Gaps between pairs in the bitmask are permitted.

For example:

```
#define TOUCH_DIO_MASK_BUTTONS 0x0f1bUL
#define TOUCH_SENSOR_COUNT     8
```

The above would be used if we had a remote control unit with 8 pads connected to DIOs 0-1, 3-4, 12-13 and 14-15, using 4 resistors between the pad-pairs.

3.4 Low-Pass Filter

To remove interference, the sampled counts for each pad are passed into a digital low-pass filter with a sampling frequency of 100Hz and a cut-off of 5Hz. The filters are a fixed-point, 4th-order Butterworth IIR type.

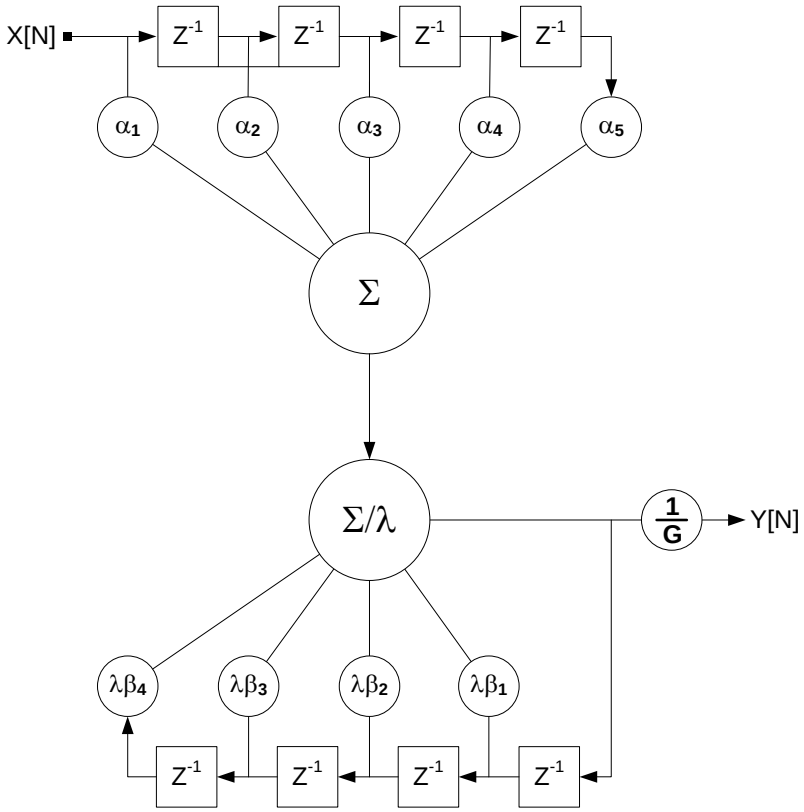

Figure 7: IIR Filter Architecture

3.5 Baseline Capacitance Tracker

Each pad has particular dynamic baseline capacitance which needs to be established and tracked in order to account for layout (PCB track lengths) and temperature/humidity variations over the operating lifetime of the remote control unit.

Upon initialisation, the charge counter reads and stores the first baseline count associated with each pad. As this is the first read, the filter is bypassed during this process. Subsequent reads of the pad are via the low-pass filter.

The tracking algorithm compares the measured count with the baseline count every ten samples of each pad. This decimation process is used to allow detection of sudden changes which occur at pad press or release events. The algorithm 'drifts' the baseline capacitance slowly up or down by a single count depending on the sign of Measured[Pad]-Baseline[Pad].

3.6 Sticky Comparator

The baseline counts and the filtered measured counts are fed into a comparator function with spatial and temporal hysteresis. Here, the baseline count is scaled by +65 to give a trip-hi threshold and by +25 to give a trip-lo threshold. Fixed offset values are used since the change in capacitance for each pad in the presence of a 'touch' is also fixed (approx 3pF change with a membrane) irrespective of what the background capacitance is. After each trip, the comparator is disabled for 200ms to 'stick' the trip and prevent excessive event generation to the rest of the system.

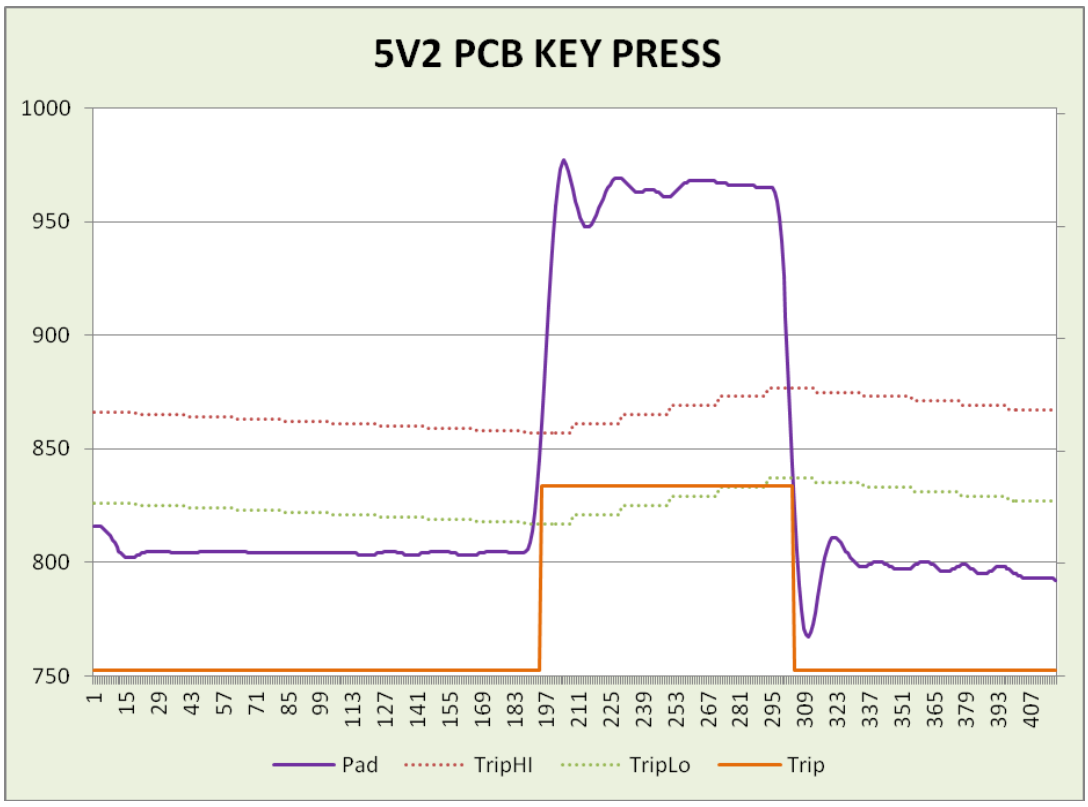

Figure 8: Press-Release Detection

3.7 Event Generator

The trip type along with the pad currently being checked is passed into an event generation process which executes a callback function to rest of the software, of the form:

vCbTouchEventButton(eTouchButtonEvent *eEvent*, **uint8** *u8ButtonNumber*);

where the event can be one of the following:

```
typedef enum
{
    TOUCH_BUTTON_EVENT_RELEASED,
    TOUCH_BUTTON_EVENT_PRESSED,
    TOUCH_BUTTON_EVENT_ALL_RELEASED,
} eTouchButtonEvent;
```

The interface supports multiple key presses and releases. A single call to the main function **eTouchProcess()** can result in none, one or many callbacks to the main program, depending on what is happening. For example, the last key released will cause two callbacks:

- `vCbTouchEventButton(TOUCH_BUTTON_EVENT_RELEASED,buttonId)`
- `vCbTouchEventButton(TOUCH_BUTTON_EVENT_ALL_RELEASED,0)`

3.8 Capacitive-Touch Interface Usage

This section describes the implementation of the capacitive-touch algorithm in C software (a wrapper application for this algorithm is described in Chapter 4).

3.8.1 Initialisation

There are two stages to initialisation.

First, the resources needed to run the capacitive-touch interface are initialised by calling:

eTouchInit();

Secondly, the interface needs to acquire a stable baseline capacitance. This is done by repeatedly calling the main function until the function indicates a stable capacitance.

```
while(eTouchProcess() ==  TOUCH_STATUS_DONT_SLEEP);
```

3.8.2 Running

The implementation contains a digital IIR filter that has been designed with a sampling frequency of 100Hz. It is the application's responsibility to ensure that this condition is satisfied by calling the following function one hundred times per second:

eTouchProcess();

Any key presses are fed back to the application by the callback mechanism as described in Section 3.7.

3.8.3 Sleeping

The implementation optionally allows for sleeping using the following two functions:

eTouchSleep() and **eTouchWake()**

These functions should be used for long sleep durations, just before the application goes to sleep and in the application wake-up (usually **AppWarmStart**) respectively.

4 Example Application Software Description

This chapter details the supplied C application, which illustrates the software wrapper needed for the capacitive-touch algorithm described in Chapter 3.

The description below includes the option of connecting a display terminal (e.g. on a PC) to the Remote Control Unit to allow status messages from the unit to be displayed on the terminal. If required, the terminal's USB port must be connected to the unit via the J3 programming header in the battery compartment of the unit (refer to the photo in Step 1 in Chapter 5).

From version 1.1 onwards the application can be targeted to three different module types (The module type is on the rear of the remote).

If you wish to build or modify the software ensure the following components have been installed.

NXP Module	SDK Installers required for build
JN5148-M0	JN-SW-4041-SDK-Toolchain-vX.Y.exe JN-SW-4051-JenNet-IP-SDK-Installer-vXYZ.exe
JN5148-J01-M0	
JN5168-M0	JN-SW-4041-SDK-Toolchain-vX.Y.exe JN-SW-4065-JN516x-JenNet-IP-SDK-vXYZ.exe

Table 2: SDK Combinations for Different Module Types

4.1 Overview

The example application supplied in the source file **AN1170-Remote-Control.c** illustrates how to drive the capacitive-touch interface, handle the responses to key presses and implement sleep functionality after a period of inactivity. Additionally, the application code shows how to dual-purpose one of the microcontroller DIO lines to drive an LED while the microcontroller is awake and to provide the wake-up event when the device is asleep.

The code comprises an initialisation function and a main process loop which calls the capacitive-touch interface and the inactivity timer manager a hundred times per second.

A callback function is also provided for handling key press/release events.

4.2 Function Descriptions

The example application defines and uses a number of functions, which are described below.

4.2.1 AppColdStart

AppColdStart() is the main program. It first calls the system initialisation function (see Section 4.2.2) and then executes an infinite loop waiting for timer events to schedule the regular calls to the capacitive-touch interface and the inactivity timer.

4.2.2 vInitSystem

The function **vInitSystem()** first initialises the System Controller callback function. This must be done before calling **u32AHI_Init()** (from the JN51xx Integrated Peripherals API) to allow the application to determine if the system has woken up from a DIO change of state.

Next, the Tick Timer is configured to produce a 100-Hz interrupt to ensure the correct timing operation of the capacitive-touch algorithm and the inactivity timers.

If a terminal is connected to the Remote Control Unit, the UART is configured (115200-8-N-1) to allow status messages to be displayed. To allow all 16 keys to be used by the capacitive-touch interface, the UART RTS/CTS lines are reclaimed for standard DIO usage.

The wake-up flag is consumed (if set) and a message is sent to the terminal (if connected).

Finally, the capacitive-touch algorithm is initialised and repeatedly called to acquire the background capacitance.

4.2.3 vCbTouchEventButton

The function **vCbTouchEventButton()** handles the three event types (see Section 3.7), and displays the event and the key (that was pressed) on the terminal (if connected).

4.2.4 vManageInactivityTimer

The function **vManageInactivityTimer()** times a period of inactivity after last the key-press. If the timer expires then the function changes the DIO on one of the LEDs to an input, configures the device to wake up on a falling edge on that pin and finally puts the device to sleep.

The function is called regularly by the main process loop for the timing and also from the capacitive-touch callback function to reset the inactivity timer.

The inactivity time is set to 10 seconds by the macro

```
#define INACTIVITY_TIME_S     10
```

4.2.5 vTickTimerISR and vSysConISR

vTickTimerISR() and **vSysConISR()** are interrupt service routines used to handle the Tick Timer and System Controller callback events respectively. In each case, an event flag is set which is used by the foreground process.

In the case of the Tick Timer, the event is a time 'tick' that is consumed by the main process loop, which simply puts the CPU in doze mode (stalls the clock) and waits for the tick flag. Any interrupt will bring the CPU out of doze mode.

The System Controller interrupt also sets an event as well as waking the device from sleep. The event is used after the system wakes in order to display a status message on the terminal (if connected).

4.3 Operation

When the application is running, both the LEDs on the unit light when any key is pressed. If a terminal is connected to the Remote Control Unit then a status message corresponding to the generated event is displayed.

If no key is pressed for ten seconds, the right LED will flash briefly to indicate that the device has entered sleep mode. Any subsequent key presses will then have no effect. The user must then press the 'Wake' button under the keypad. Both LEDs will then flash and the unit will again function in normal capacitive-touch mode.

5 Firmware Programming

The Remote Control Unit firmware (contained in a Flash memory device on the JN51xx module) can be programmed from a PC using the DR1128 USB Programming Dongle, as follows:

1. Remove the battery compartment cover on the rear of the unit to reveal the J3 header for the programming dongle (if there are batteries in the compartment, you can leave them or remove them).

2. Insert the programming dongle onto the header in the battery compartment, as illustrated below.

3. Use a 'USB A to Mini-B' cable to connect the programming dongle to a USB port of your PC.
4. Use the JN51xx Flash Programmer application on your PC to load the new firmware image into the Remote Control Unit. This application is provided in the SDK Toolchain (JN-SW-4041) and as a standalone utility (JN-SW-4007) – be sure to use a version of the Flash programmer which is compatible with the target JN51xx chip type. The programming procedure is described in the *JN51xx Flash Programmer User Guide (JN-UG-3007)*, but ignore references to resetting the device and putting it into programming mode - for the dongle, this is done automatically.

5. Once programming has completed, remove the programming dongle and replace the battery compartment cover.

Note: The programming dongle can also be used as a UART port on the Remote Control Unit for debug purposes.

.

Appendices

A PCB/Filter Performance

The DR1159 PCB is designed with guard rings around each pad and has a 75% flood fill ground plane underneath to create a 2-D “Flat” capacitor structure.

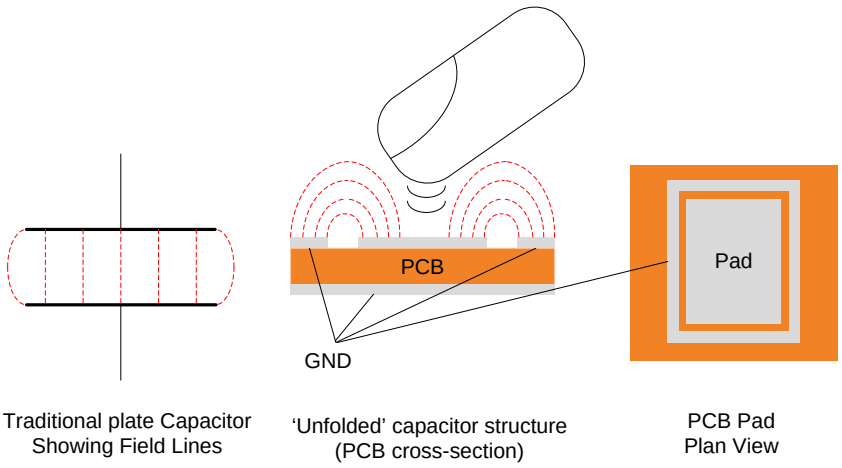

Figure 9: New Capacitor PCB Structure

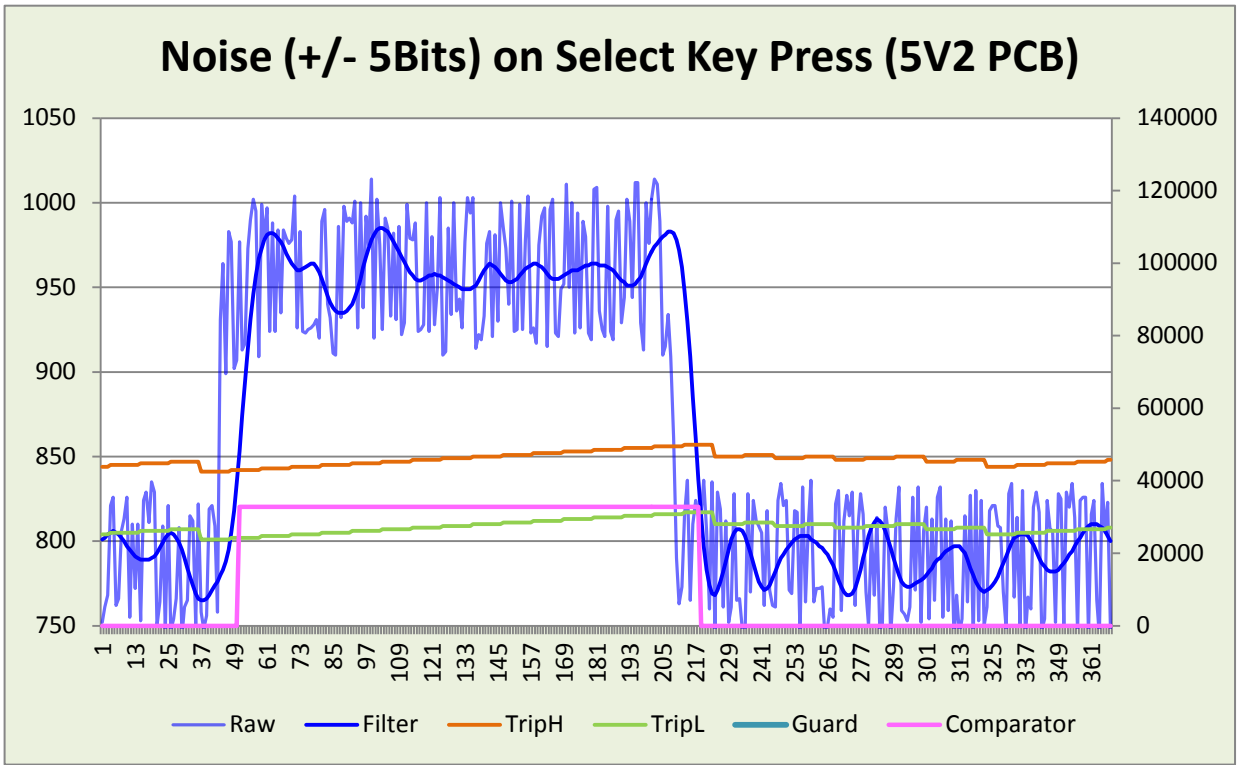

Figure 10: Capacitor PCB Performance

Revision History

Version	Date	Description
1.0	20-Sep-2012	First release
1.1	08-Jan-2013	Added SDK/Build matrix for various Microcontroller modules supported

Important Notice

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

NXP Laboratories UK Ltd
(Formerly Jennic Ltd)
Furnival Street
Sheffield
S1 4QT
United Kingdom

Tel: +44 (0)114 281 2655
Fax: +44 (0)114 281 2951

For the contact details of your local NXP office or distributor, refer to:

www.nxp.com/jennic

