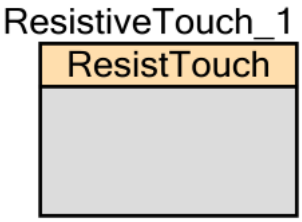
Resistive Touch (ResistiveTouch) 2.0



Features

- Supports 4-wire resistive touchscreen interface
- Supports the Delta Sigma Converter for both the PSoC 3 and PSoC 5LP devices
- Supports the ADC Successive Approximation Register for PSoC 5LP devices

General Description

This resistive touchscreen component is used to interface with a 4-wire resistive touch screen. The component provides a method to integrate and configure the resistive touch elements of a touchscreen with the emWin Graphics library. It integrates hardware-dependent functions that are called by the touchscreen driver supplied with emWin when polling the touch panel.

This component is designed to work with the SEGGER emWin graphics library. This graphics library is provided by Cypress to use with Cypress devices and is available on the Cypress website at www.cypress.com/go/comp_emWin. This graphics library provides a full-featured set of graphics functions for drawing and rendering text and images.

When to Use a ResistiveTouch

Use a ResistiveTouch component where low cost and simple interface electronics are required.

Input/Output Connections

This section describes the various input and output connections for the ResistiveTouch.

xm – Digital Input / Output

Signal x– from axis X of the resistive touch panel (active low).

xp – Analog / Digital Output

Signal x+ from axis X of the resistive touch panel (active high).

ym – Digital Input / Output

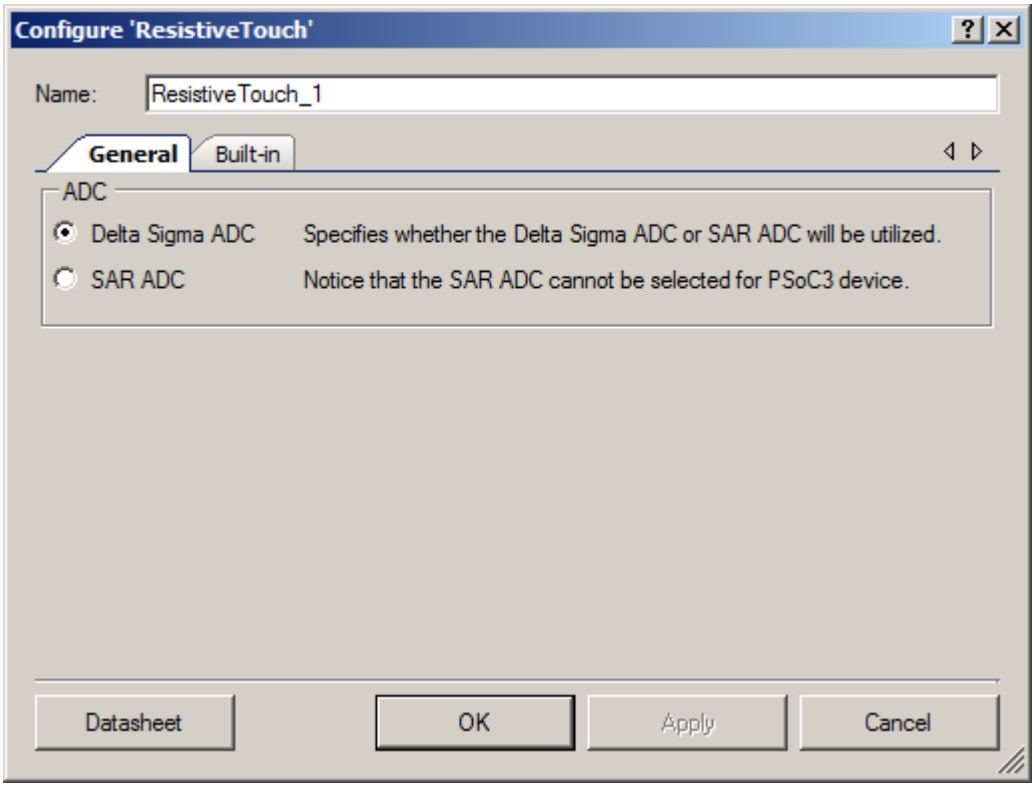
Signal y– from axis Y of the resistive touch panel (active low).

yp – Analog / Digital Output

Signal y+ from axis Y of the resistive touch panel (active high).

Component Parameters

Drag a ResistiveTouch component onto your design and double click it to open the **Configure** dialog.



ADC

The **ADC** parameter determines which ADC to use. For PSoC 3 devices, select the **Delta Sigma ADC** option.

Application Programming Interface

Application Programming Interface (API) routines allow you to configure the component using software. The following table lists and describes the interface to each function. The subsequent sections cover each function in more detail.

By default, PSoC Creator assigns the instance name “ResistiveTouch_1” to the first instance of a component in a given design. You can rename it to any unique value that follows the syntactic rules for identifiers. The instance name becomes the prefix of every global function name, variable, and constant symbol. For readability, the instance name used in the following table is “ResistiveTouch”.

Functions

Function	Description
ResistiveTouch_Start()	Calls the ResistiveTouch_Init() and ResistiveTouch_Enable() APIs.
ResistiveTouch_Stop()	Stops the ADC and the AMux component.
ResistiveTouch_Init()	Calls the Init() functions of the ADC and AMux components.
ResistiveTouch_Enable()	Enables the component.
ResistiveTouch_ActivateY()	Configures the pins to measure the Y-axis.
ResistiveTouch_ActivateX()	Configures the pins to measure the X-axis.
ResistiveTouch_TouchDetect()	Detects a touch on the screen.
ResistiveTouch_Measure()	Returns the result of the ADC conversion.
ResistiveTouch_SaveConfig()	Saves the configuration of the ADC.
ResistiveTouch_Sleep()	Prepares the component for entering the low power mode.
ResistiveTouch_RestoreConfig()	Restores the configuration of the ADC.
ResistiveTouch_Wakeup()	Restores the component after waking up from the low power mode.

void ResistiveTouch_Start(void)

Description: Sets the ResistiveTouch_initVar variable, calls the ResistiveTouch_Init() function, and then calls the ResistiveTouch_Enable() function.

Parameters: None

Return Value: None

Side Effects: None

void ResistiveTouch_Stop(void)

Description: Calls the Stop() functions of the ADC and the AMux components.

Parameters: None

Return Value: None

Side Effects: None

void ResistiveTouch_Init(void)

Description: Calls the Init() functions of the ADC and AMux components.

Parameters: None

Return Value: None

Side Effects: None

void ResistiveTouch_Enable(void)

Description: Calls the Enable() function of the ADC component.

Parameters: None

Return Value: None

Side Effects: None

void ResistiveTouch_ActivateX(void)

Description: Configures the pins to measure the X-axis.

Parameters: None

Return Value: None

Side Effects: None

void ResistiveTouch_ActivateY(void)

Description: Configures the pins to measure the Y-axis.

Parameters: None

Return Value: None

Side Effects: None

int16 ResistiveTouch_Measure(void)

Description: Returns the result of the ADC conversion.

Parameters: None

Return Value: int16: The result of the ADC conversion

Side Effects: None

uint8 ResistiveTouch_TouchDetect(void)

Description: Detects a touch on the screen.

Parameters: None

Return Value: uint8: The touch state
0 – untouched
1 – touched

Side Effects: None

void ResistiveTouch_SaveConfig(void)

Description: Saves the configuration of the ADC.

Parameters: None

Return Value: None

Side Effects: None

void ResistiveTouch_RestoreConfig(void)

Description: Restores the configuration of the ADC.

Parameters: None

Return Value: None

Side Effects: None

void ResistiveTouch_Sleep(void)

Description: Prepares the component for entering the low power mode.

Parameters: None

Return Value: None

Side Effects: None

void ResistiveTouch Wakeup(void)

Description: Restores the component after waking up from the low power mode.

Parameters: None

Return Value: None

Side Effects: None

Global Variables

Variable	Description
ResistiveTouch_initVar	Indicates whether the ResistiveTouch has been initialized. The variable is initialized to 0 and set to 1 the first time ResistiveTouch_Start() is called. This allows the component to restart without reinitialization after the first call to the ResistiveTouch_Start() routine. If reinitialization of the component is required, then the ResistiveTouch_Init() function can be called before the ResistiveTouch_Start() or ResistiveTouch_Enable() function.

Sample Firmware Source Code

PSoC Creator provides many example projects that include schematics and example code in the Find Example Project dialog. For component-specific examples, open the dialog from the Component Catalog or an instance of the component in a schematic. For general examples, open the dialog from the Start Page or **File** menu. As needed, use the **Filter Options** in the dialog to narrow the list of projects available to select.

Refer to the “Find Example Project” topic in the PSoC Creator Help for more information.

MISRA Compliance

This section describes the MISRA-C:2004 compliance and deviations for the component. There are two types of deviations defined:

- project deviations – deviations that are applicable for all PSoC Creator components
- specific deviations – deviations that are applicable only for this component

This section provides information on component-specific deviations. Project deviations are described in the MISRA Compliance section of the *System Reference Guide* along with information on the MISRA compliance verification environment.

The Resistive Touch component does not have any specific deviations.

This component has the following embedded components: Pins, Delta Sigma ADC, SAR ADC.

Refer to the corresponding component datasheets for information on their MISRA compliance and specific deviations.

API Memory Usage

The component memory usage varies significantly, depending on the compiler, device, number of APIs used and component configuration. The following table provides the memory usage for all APIs available in the given component configuration.

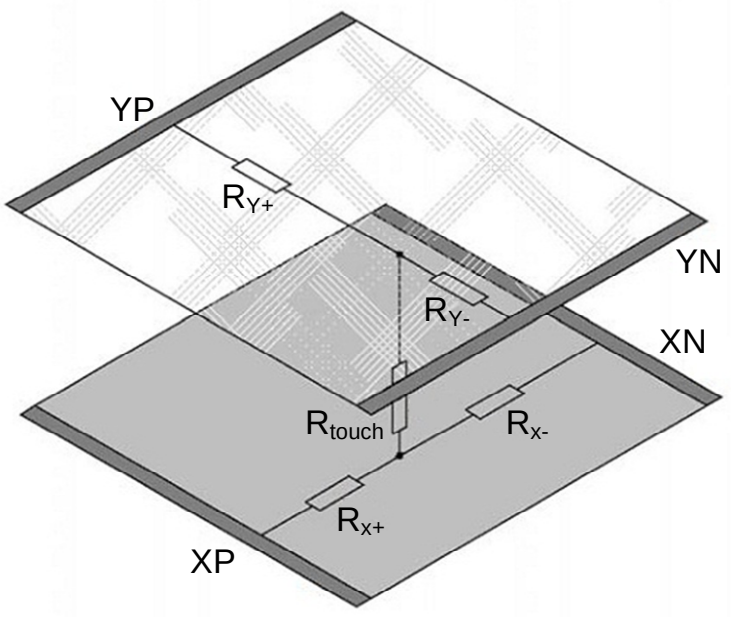
The measurements have been done with the associated compiler configured in Release mode with optimization set for Size. For a specific design the map file generated by the compiler can be analyzed to determine the memory usage.

Configuration	PSoC 3 (Keil_PK51)		PSoC 5LP (GCC)	
	Flash Bytes	SRAM Bytes	Flash Bytes	SRAM Bytes
Delta Sigma ADC	2341	22	2052	24
SAR ADC	N/A	N/A	982	15

Functional Description

This component provides a 4-wire resistive touch screen interface to read touchscreen coordinates and measure screen resistance. This component provides access to the touchscreen functionality of the SEGGER emWin graphics library for translation of resistance to screen coordinates.

The diagram that follows shows the schematic of a 4-wire touchscreen when pressure is applied.



The point of contact “divides” each layer in a series resistor network with two resistors (see the diagram), and a connecting resistor between the two layers. By measuring the voltage at this point, you get information about the position of the contact point orthogonal to the voltage gradient. To get a complete set of coordinates, you must apply the voltage gradient once in the vertical and then in the horizontal direction. First, you must apply a supply voltage applied to one layer and perform a measurement of the voltage across the other layer; next connect the supply to the other layer and measure the opposite layer voltage. When in touch mode, one of the lines is connected to detect touch activity. The following table defines the configuration of the pins while measuring the coordinates or touch.

	XP	XM	YP	YM
touch	Res Pullup	Analog Hi-Z	Analog Hi-Z	Strong Drive
X-Coordinate	Strong Drive	Strong Drive	Analog Hi-Z	Analog Hi-Z
Y-Coordinate	Analog Hi-Z	Analog Hi-Z	Strong Drive	Strong Drive

When a ResistiveTouch component is placed onto the project schematic, it is important that I/O port designations be set for xm, xp, ym, yp pins. Designation of these pins is not done on the symbol or schematic; instead, it is done in the Pins tab of the Design-Wide Resources window.

Resources

Depending on the configuration, the ResistiveTouch component uses the Delta Sigma ADC or SAR ADC component, and four pins.

DC and AC Electrical Characteristics

Depending on configuration the resistive touch component uses the Delta Sigma ADC or SAR ADC components. Therefore the resistive touch component is limited by the performance of these components. Refer to the Characteristics section of the associated ADC component datasheet for details.

References

Refer also to the Delta Sigma ADC and SAR ADC component datasheets.

Component Changes

This section lists the major changes in the component from the previous version.

Version	Description of Changes	Reason for Changes / Impact
2.0.a	Updated this datasheet change section to note that the internal clock configuration was modified in the Resistive Touch 2.0 component.	If the SAR ADC is chosen as ADC for the Resistive Touch component, any project that uses the maximum number of digital clocks will fail to build after updating the Resistive Touch component from v1.30 to v2.0. This is because the ResistiveTouch_ADC_SAR_theACLK internal clock uses the Digital domain by default. To resolve this problem, change the clock domain of the ResistiveTouch_ADC_SAR_theACLK internal clock from Digital to Analog in the Design-Wide Resources (*<project>.cydwr*) file **Clocks** tab.
2.0	Updated versions of the embedded ADC and pins components to the most current versions.	Out of date components may contain defects or incompatibilities.
	Updated MISRA compliance section.	The component was verified for MISRA compliance.
	Datasheet edits and corrections.	
1.30	Corrected the component changes made in PSoC Creator 3.0 SP1.	Correction of the Component Errata item – Cypress ID 191257.
1.20	Added MISRA Compliance section.	The component was not verified for MISRA compliance.
	Updated for compatibility with PSoC Creator v2.2	
1.10.a	Minor edits and updates to datasheet	
1.10	Added PSoC 5LP support	

© Cypress Semiconductor Corporation, 2015. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC® is a registered trademark, and PSoC Creator™ and Programmable System-on-Chip™ are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.