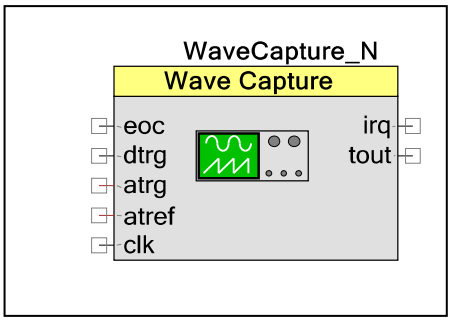
WaveCapture

1.0

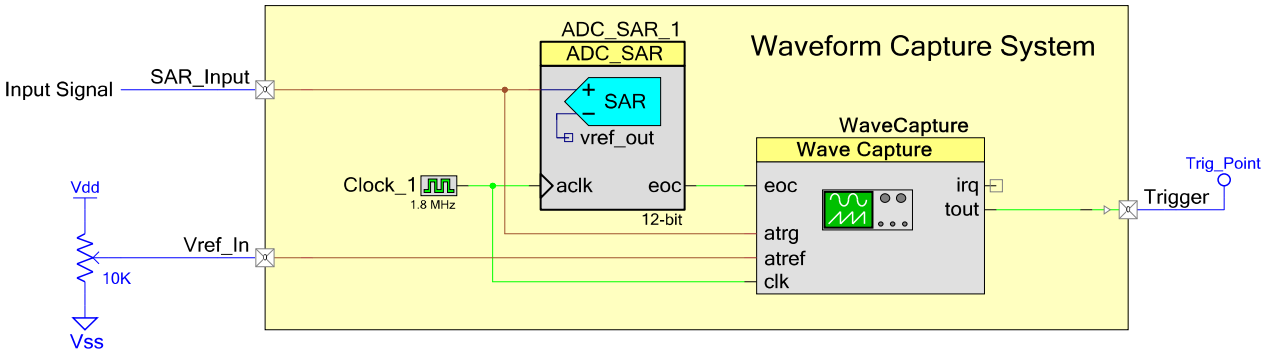
Features



- Using ADC SAR in PSoC 5LP to capture waveforms
- Controls SAR ADC for the user
- Analog and digital triggering
- Captures waveforms up to 4096 points.
- APIs automatically setup DMA and sort captured data
- Triggers and captures data with no CPU overhead
- Internal or external analog trigger reference

General Description

The WaveCapture component is a companion to the PSoC 5LP SAR ADC. It is used to capture a signal or waveform, triggered either by a digital signal or analog voltage level. Below is a typical example of how the WaveCapture component may be connected to the SAR ADC.



This simplifies the user code to just a few lines of code using the supplied relatively simple API. TheWaveCapture hardware which utilizes DMA, offloads all data collection from the CPU until the data is captured and ready to analyze.

Input/Output Connections

This section describes the various input and output connections for the VDAC8. An asterisk (*) in the list of I/Os indicates that the I/O may be hidden on the symbol under the conditions listed in the description of that I/O.

eoc – Digital Input

This input is connected to the SAR ADC's "eoc" signal to signal that a conversion has been completed.

Dtrg* – Digital Input

This terminal is present when the digital trigger option is selected. When a rising edge is sensed on the terminal, data will be collected to fill the predetermined buffer size.

atrg – Analog Input

This input signal is usually connected to the input of the SAR ADC. It is used to trigger the data capture sequence when a voltage becomes larger than the voltage reference signal.

Atref* – Analog Input

This analog input signal is present when the external analog reference option is selected. When the signal at "atrg" is larger than the value at this input, the data collection sequence will begin.

clk – Clock Input

The clock should be the same clock as the SAR ADC. If the same clock cannot be used, the clock should be twice that of the SAR ADC.

irq – Digital Output

This signal goes high when the data collection sequence is complete. It is meant to be connected to an interrupt component.

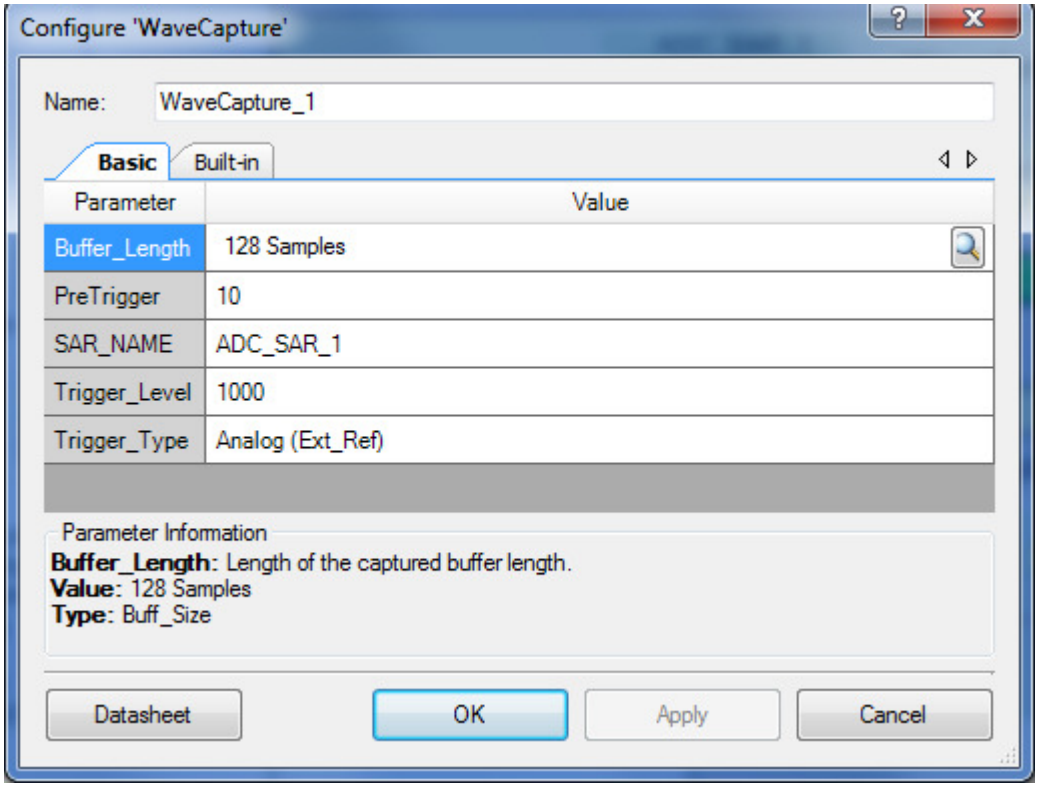
tout – Digital Output

This signal goes high when a trigger to begin the data collection sequence has been detected.

Parameters and Setup

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

Figure 1 Configure WaveCapture Dialog



The WaveCapture component provides the following parameters.

Buffer Length

This parameter selects the depth of the capture buffer. A buffer up to 4096 samples may be selected. The default sample size is 128 samples. Each sample requires 2 bytes of data in which it stores a 12-bit ADC result.

- 32 Samples
- 64 Samples
- **128 Samples**
- 256 Samples
- 512 Samples
- 1024 Samples
- 2048 Samples
- 4096 Samples

PreTrigger

This parameter determines how many samples before the capture are collected. It is often important to look at several samples prior to the trigger event. The samples taken after the trigger event will be the Buffer Length minus the PreTrigger count.

SAR_NAME

This parameter is the instance name of the SAR ADC that is used with the WaveCapture component. It is used so that the WaveCapture component can make the proper API calls to start the SAR ADC without the user having to write the SAR ADC code. Because of this, the user does not write any SAR ADC code, the component takes care of starting and stopping of the SAR ADC component. If this parameter does not exactly match that of the SAR ADC's instance name, errors will be generated during compile time.

Trigger_Level

This parameter is only valid when the Trigger_Type parameter is set to "Analog (Int Ref)". It sets the trigger level in mVolts up to a reference of 4095 mVolts.

Trigger_Type

This parameter determines what causes the data sequence to be begin. Below are the trigger type options.

- Digital
- Analog (Int Ref)
- **Analog (Ext_Ref)**

The "Digital" option waits for a digital signal to go high to start data collection. The Analog options wait for an analog signal to become higher than the reference. The reference can either be generated by an internal VDAC or provided by an external source.

Resources

The WaveCapture component requries 2 UDBs, a DMA channel and optionally an analog comparator and voltage DAC.

Application Programming Interface

Application Programming Interface (API) routines allow you to control and 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 "WaveCapture_1" to the first instance of a component in a given design. You can rename the instance 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 "WaveCapture".

Function	Description
void WaveCapture_Start(void)	Starts the component and the companion SAR ADC.
void WaveCapture_Stop(void)	Disables the SAR ADC and the WaveCapture component.
void WaveCapture_Init(void)	Sets up internal components. (Called by Start())
void WaveCapture_Enable(void)	Enables internal components. (Called by Start())
void WaveCapture_ArmTrigger(void)	Enables component so that it will automatically collect data.
void WaveCapture_DisArmTrigger(void)	Disables component so that it no longer will capture data.
Uint16 WaveCapture_GetStatus(void)	Returns status of capture process.
uint16 WaveCapture_GetDmaIndex(void)	Returns the position in the capture array of the last value collected.
void WaveCapture_GetArray(uint16 * Array)	Fills the “Array” with the collected data in the proper order.
void WaveCapture_SetTrigLevel(uint16 level)	Sets the analog value at which data collection will begin.

void WaveCapture_Start(void)

Description: Performs all of the required initialization for the component and enables power to the block. The first time the routine is executed, the input and feedback resistance values are configured for the operating mode selected in the design. When called to restart the WaveCapture component following a WaveCapture_Stop() call, the current component parameter settings are retained.

Parameters: None

Return Value: None

Side Effects: None

void WaveCapture_Stop(void)

Description: Turn off the WaveCapture block and the SAR ADC.

Parameters: None

Return Value: None

Side Effects: Disables the SAR ADC component.

void WaveCapture_Init(void)

Description: Initializes or restores the component according to the customizer Configure dialog settings. It is not necessary to call WaveCapture_Init() because the WaveCapture_Start() API calls this function and is the preferred method to begin component operation.

Parameters: None

Return Value: None

Side Effects: All registers will be set to values according to the customizer Configure dialog.

void WaveCapture_Enable(void)

Description: Activates the hardware and begins component operation. It is not necessary to call WaveCapture_Enable() because the WaveCapture_Start() API calls this function, which is the preferred method to begin component operation.

Parameters: None

Return Value: None

Side Effects: None

void WaveCapture_ArmTrigger(void)

Description: Arms the trigger hardware then waits for a trigger event, at which time it will collect the desired amount of samples.

Parameters: None

Return Value: None

Side Effects: None

void WaveCapture_DisArmTrigger(void)

Description: Disarms the trigger hardware so that a trigger even will be ignored and data will not be collected.

Parameters: None

Return Value: None

Side Effects: None

void WaveCapture_GetStatus(void)

Description: Returns the status of the WaveCapture component. This is used to determine if all the data has been collected.

Parameters: None

Return Value: uint16: Returns a zero if capture is not complete, and a non-zero value when capture is complete.

Side Effects: None

void WaveCapture_ GetDmaIndex (void)

Description: This function returns the index of the last value collected. This value is used to collect the data in the proper sequence.

Parameters: None

Return Value: uint16: The address in the TD buffer where the last sampled data is stored.

Side Effects: None

void WaveCapture_ GetArray(uint16 * Array) (void)

Description: This function is used to copy the captured data in the proper sequence from the TD array.

Parameters: (uint16 *) Pointer to array in which to copy the collected data.

Return Value: None

Side Effects: None

void WaveCapture_ SetTrigLevel(uint16 level)

Description: Sets the internal VDAC output to the proper trigger value in mVolts.

Parameters: (uint16) Level: Sets the internal trigger reference value between 0 and 4095 mVolts. When the input is greater than this value, the data collection sequence will begin.

Return Value: None

Side Effects: None

DC and AC Electrical Characteristics

For more specifications of the AC and DC performance of the WaveCapture see the SAR_ADC data sheet..

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
1.0		Initial document

© Cypress Semiconductor Corporation, 2012. 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.

