Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

CE220974 - PSoC 6 MCU Multi-Config Scan_ADC Example

Objective

This example demonstrates a Scan_ADC Component with the DieTemp sensor using low-level Peripheral Driver Library (PDL) function calls in a PSoC® 6 MCU.

Overview

This example demonstrates a Scan_ADC with two conversion configurations implemented using direct PDL functions calls. The example selects between a fast free-running two-channel differential system and a firmware-triggered scan with averaging of the internal die temperature (DieTemp) sensor. The DieTemp sensor is scanned once per second using a watchdog timer. Results of all channel readings are transmitted over UART.

Requirements

Tool: PSoC Creator™ 4.2 with PDL 3.0.1

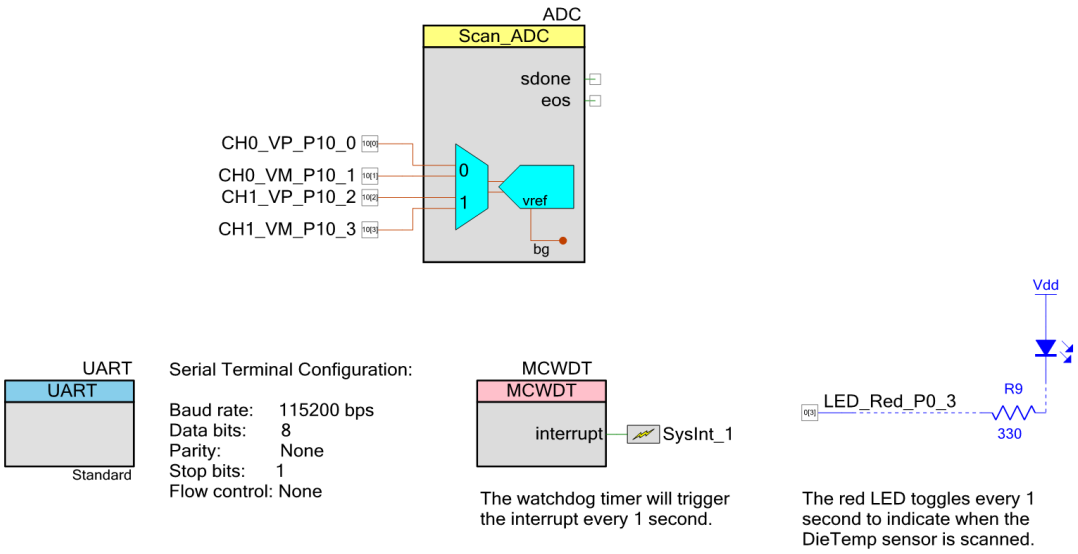
Programming Language: C (Arm® GCC 5.4-2016-q2-update)

Associated Parts: PSoC 6 MCU family of devices

Related Hardware: CY8CKIT-062-BLE PSoC 6 MCU BLE Pioneer Kit and CY8CKIT-062-WiFi-BT PSoC 6 MCU Pioneer Kit

Design

Figure 1. PSoC Creator Component Schematic



The code example configures the SAR ADC using PDL function calls. The Scan_ADC Component shown in the above schematic is for visualization purposes only.

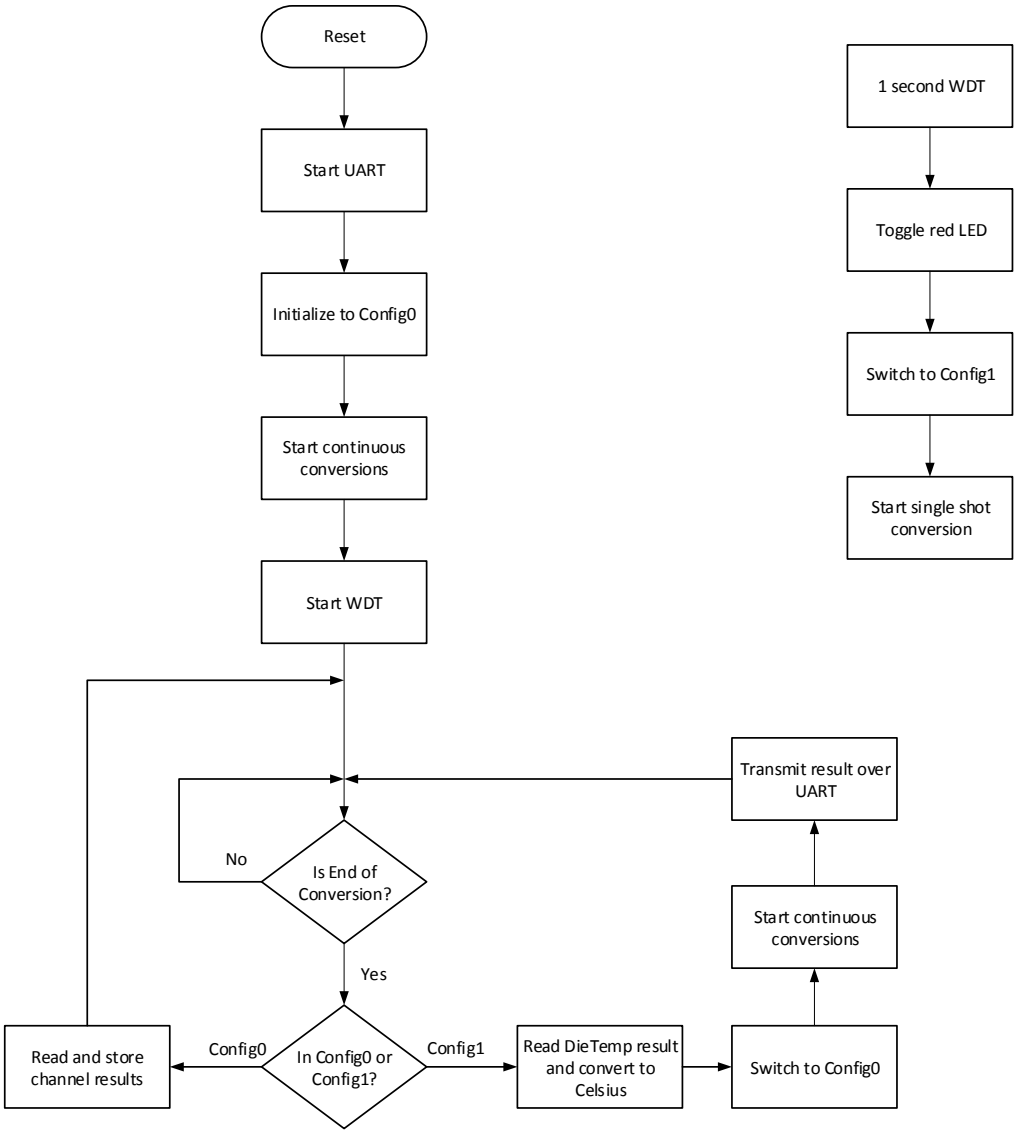
The SAR ADC continuosly scans the two differential channels in Config0 in free-running mode. Using the watchdog timer (WDT) at 1-second intervals, the firmware switches to Config1 where the SAR ADC does a single scan of the die temperature sensor. After the die temperature scan is complete, the firmware switches back to Config0 and begins continuous scans again.

The SAR ADC End of Scan (EOS) interrupt is handled by the CPU. Upon the end of scan, results for the channels in the active configuration are retrieved and stored. ADC results are transmitted over UART with an update rate of 1 Hz. The results include the most recent voltage readings of the two differential channels from Config0 and the die temperature sensor temperature reading in degrees Celsius from Config1.

A red LED (LED5 on the CY8CKIT-062-BLE kit) will toggle at 1 Hz to indicate when the die temperature sensor is scanned.

The internal die temperature sensor on the device is enabled once the connection to the SAR ADC is made. There is no PDL driver for the die temperature sensor.

Figure 2. Code Example Flowchart



SAR ADC PDL Configuration

Table 1 and Table 2 show how Config0 and Config1 are configured in PDL.

Table 1. Config0 PDL Settings

Parameter	Value
ADC clock rate	16.67 MHz
Number of channels	2
Free-running scan rate	49.9 ksps
Reference source	Internal 1.2-V bandgap with bypass cap enabled
Channel 0	Differential from paired pins Averaging disabled
Channel 1	Differential from paired pins Averaging disabled
Interrupts	End of Scan (EOS) interrupt enabled Range and saturation interrupts disabled

Table 2. Config1 PDL Settings

Parameter	Value
ADC clock rate	16.67 MHz
Number of channels	1
Scan time	34.56 µs (firmware-triggered)
Reference source	Internal 1.2-V bandgap with bypass cap enabled
Channel 0	Single-ended from die temperature Sensor Sequential-fixed averaging enabled with 32 samples Negative terminal connected to the VSSA pin
Interrupts	End of Scan (EOS) interrupt enabled Range and saturation interrupts disabled

Scan Rate

The ADC clock is an integer divider of PeriClk. The maximum supported frequency of the SAR ADC is 18 MHz. With a 50-MHz PeriClk, the code example configures the ADC clock rate to 16.67 MHz.

$$ADC\ Clock\ Rate = \frac{PeriClk\ Rate}{Integer\ Divider} = \frac{50\ MHz}{3} = 16.67\ MHz$$

Config0 Scan Rate

The scan rate is based on the ADC clock rate, number of channels, acquisition times, resolution, and averaging (if enabled). For Config0, the acquisition times of channels 0 and 1 are:

$$Minimum\ Aperature\ Cycles = ceiling(ADC\ Clock\ Rate * Mimum\ Sample\ Time) + 1$$

$$Minimum\ Aperature\ Cycles = \ ceiling(16.67\ MHz * 167\ ns) + 1 = ceiling(2.78) + 1 = 3 + 1 = 4$$

Additional channel aperture cycles can be added, thereby slowing conversions to a desired scan rate. For example, the minimum of 4 aperture cycles was increased to 302 for Ch0 to obtain a 50-kHz scan rate for Config0. Config0 requires a total of 306 aperture cycles to meet the desired scan rate. To calculate the total number of aperture cycles required to meet a desired scan rate, you must first add all the samples performed in a single scan of the configuration. The total number of samples is the sum of one for each unfiltered channel, and the total number of filter samples for each filtered channel. Once this value is obtained, the total number of aperture cycles can be spread across the channels as desired for the application as long as the minimum cycle count is met.

$$Total\ aperture\ cycles = \frac{ADC\ Clock\ Rate}{Scan\ Rate} - \big((Resolution + 2)\ \times\ Total\ Samples\big)$$

$$Total\ aperture\ cycles = \frac{16.67\ MHz}{50\ kHz} - \big((12\ bits + 2)\ \times\ 2\big) = 306$$

After the aperture cycles for each channel are known, you can calculate the actual acquisition time for each channel and therefore the total scan rate.

$$Ch0\ Acquisition\ Time = \frac{Ch0\ Aperature\ Cycles - 1}{ADC\ Clock\ Rate} = \frac{302 - 1}{16.67\ MHz} = 18.06\ us$$
$$Ch1\ Acquisition\ Time = \frac{Ch1\ Aperature\ Cycles - 1}{ADC\ Clock\ Rate} = \frac{4 - 1}{16.67\ MHz} = 180\ ns$$

The sample time for a channel is the time required to acquire the signal and convert it to a digital code. This is where the resolution of the ADC comes in. The sample times for channels 0 and 1 are:

$$Ch0\ Sample\ Time = Ch0\ Acquisition\ Time + \frac{Resolution + 3}{ADC\ Clock\ Rate} = 18.06\ us + \frac{12\ bits + 3}{16.67\ MHz} = 18.96\ us$$
$$Ch1\ Sample\ Time = Ch1\ Acquisition\ Time + \frac{Resolution + 3}{ADC\ Clock\ Rate} = 180\ ns + \frac{12\ bits + 3}{16.67\ MHz} = 1.08\ us$$

The total scan time is the sum of each channel's total sample time. If averaging is enabled, each channel's sample time is multiplied by the averaging count. Averaging is disabled for both channels in Config0.

$$Scan\ Time = Ch0\ Sample\ Time\frac{Ch0\ Samples}{Scan} + Ch1\ Sample\ Time\frac{Ch1\ Samples}{Scan} = 18.96\ us + 1.08\ us = 20.04\ us$$
$$Scan\ Rate = \frac{1}{Scan\ Time} = \frac{1}{20.04\ us} = 49.9\ kSPS$$

The free-running scan rate for Config0 is 49.9 ksps.

Config1 Scan Rate

Config1 has one channel with averaging enabled. Because scans of the die temperature sensor are firmware-triggered, the sample time is set to the fastest possible.

$$Ch0\ Acquisition\ Time = \frac{Ch0\ Aperature\ Cycles - 1}{ADC\ Clock\ Rate} = \frac{4 - 1}{16.67\ MHz} = 180\ ns$$
$$Scan\ Time = \left(Ch0\ Acquisition\ Time + \frac{Resolution + 3}{ADC\ Clock\ Rate}\right)\frac{Ch0\ Samples}{Scan} = \left(180\ ns + \frac{12\ bits + 3}{16.67\ MHz}\right)32 = 34.56\ us$$

Using the WDT, the effective scan rate of Config1 is 1 Hz.

Design Considerations

This code example is designed to run on the CY8CKIT-062-BLE Pioneer Kit with the CY8C6347BZI-BLD53 device. To port the design to a different PSoC 6 MCU device and/or kit, change the target device using **Device Selector** and update the pin assignments in the **Design Wide Resources Pins** settings as needed. For single-CPU PSoC 6 MCU devices, port the code from *main_cm4.c* to *main.c*.

Hardware Setup

This example uses the kit's default configuration. Refer to the kit guide to ensure that the kit is configured correctly.

Operation

1. Plug the CY8CKIT-062 BLE kit into your computer's USB port.
2. Build the project and program it into the PSoC 6 MCU device. Choose **Debug** > **Program**. For more information on device programming, see PSoC Creator Help. Flash for both CPUs are programmed in a single program operation.

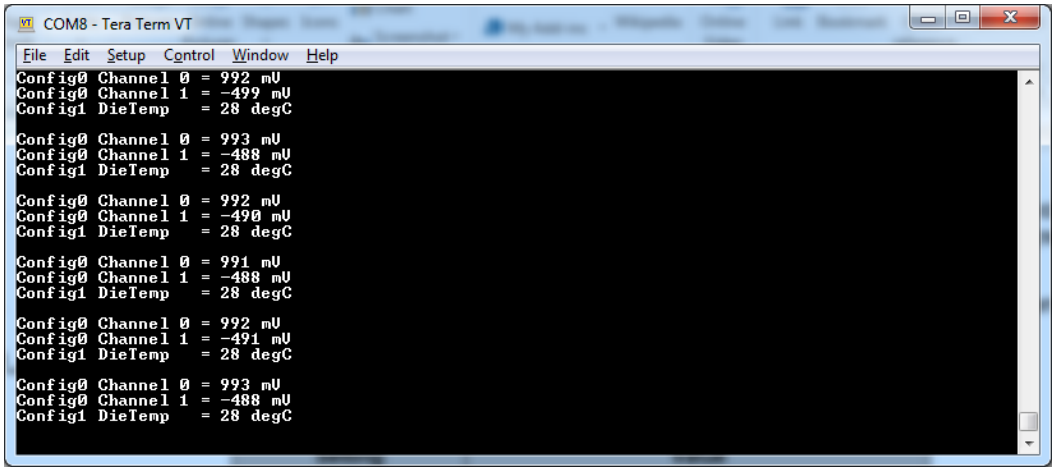
3. Apply a differential voltage between -1.2 V and 1.2 V across P10[0] and P10[1] for channel 0 of Config0. Pins are on the J2 header for the CY8CKIT-062 kit.
4. Apply a differential voltage between -1.2 V and 1.2 V across P10[2] and P10[3] for channel 1 of Config0. Pins are on the J2 header for the CY8CKIT-062 kit.
5. Open a serial port communication program such as Tera Term and select the corresponding COM port for the KitProg2 USB-UART. Configure the serial port to match the configuration of the PSoC Creator UART Component in the project.

Table 3. Serial Port Setup

Parameter	Value
Baud rate (bps)	115200
Data bits	8
Parity	None
Stop bits	1
Flow control	None

6. Observe the ADC results and die temperature reading printed in the terminal window.

Figure 3. Expected Serial Terminal Output Messages



7. Observe the red LED (LED5) on the kit toggling at 1 Hz.

Components

Table 4 lists the PSoC Creator Components used in this example, as well as the hardware resources used by each.

Table 4. PSoC Creator Components

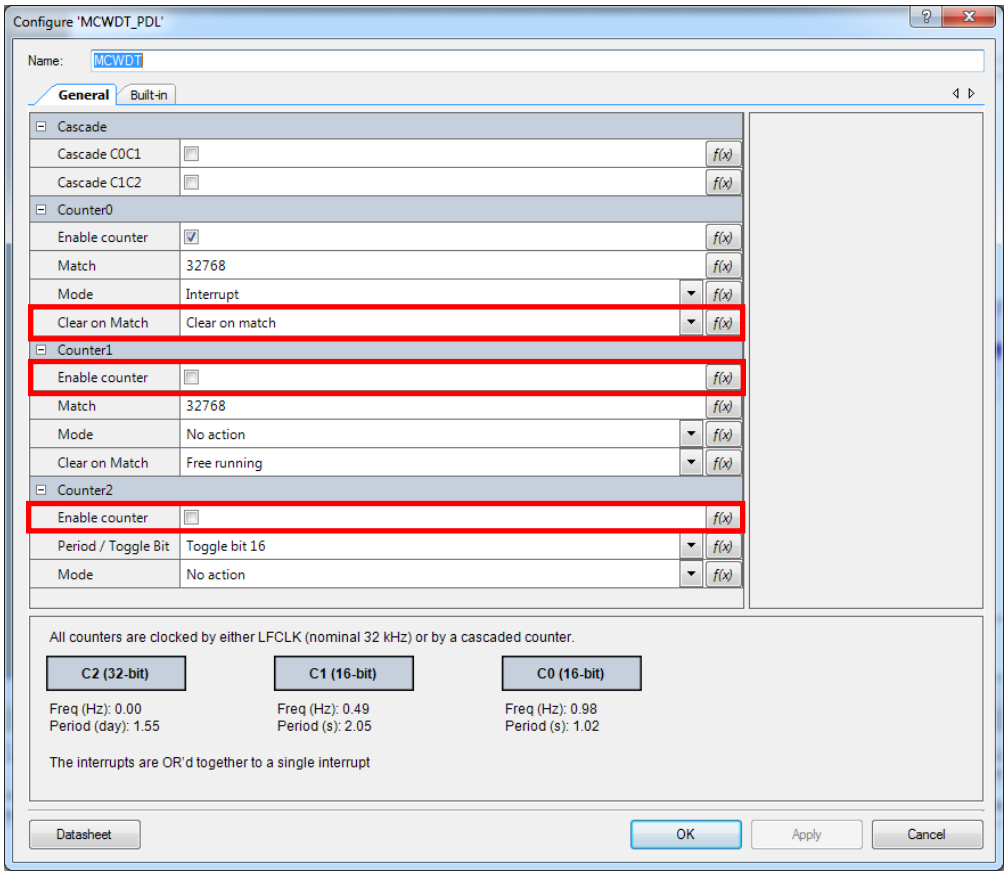
Component	Instance Name	Hardware Resources
Scanning SAR ADC	ADC	1 SAR ADC, 1 interrupt, 1 PeriClk divider
UART	UART	1 SCB, 1 interrupt, 1 PeriClk divider, 2 digital output pins
Multi-Counter Watchdog Timer	MCWDT	1 MCWDT
Interrupt	SysInt_1	1 Interrupt
Digital Pin	LED_Red_P0_3	1 Digital output pin
Analog Pin	CH0_VP_P10_0 CH0_VM_P10_1	4 Analog input pins

Component	Instance Name	Hardware Resources
	CH1_VP_P10_2 CH1_VM_P10_3	

Parameter Settings

The SAR ADC is configured entirely using PDL driver function calls. The UART Component uses default settings. Figure 4 highlights the non-default settings for the MCWDT Component.

Figure 4. MCWDT Component Parameter Settings



Only Counter0 is needed in this code example. Clear-on-match is set to achieve a 1-Hz timer.

Design-Wide Resources

Table 5 shows the pin assignments for the code example.

Table 5. Pin Names and Location

Pin Name	Pin Location
\UART:rx\	P5[0]
\UART:tx\	P5[1]
CH0_VP_P10_0	P10[0]
CH0_VM_P10_1	P10[1]
CH1_VP_P10_2	P10[2]

Pin Name	Pin Location
CH1_VM_P10_3	P10[3]
LED_Red_P0_3	P0[3]

Related Documents

Application Notes	
AN210781	Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity
PSoC Creator Component Datasheets	
Scan_ADC	Supports scanning SAR ADC functions
UART	Supports UART communication
Multi-Counter Watchdog	Supports free running counter, periodic interrupts, or watchdog reset
General Purpose Input / Output (GPIO)	Supports all GPIO pin features
Device Documentation	
PSoC 6 MCU: PSoC 63 with BLE Datasheet	
PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual	
PSoC 6 MCU: PSoC 63 with BLE Register Technical Reference Manual	
PSoC 6 MCU: PSoC 62 Datasheet	
Development Kit (DVK) Documentation	
PSoC 6 BLE Pioneer Kit	
PSoC 6 MCU WiFi BT Pioneer Kit	

Document History

Document Title: CE220974 – PSoC 6 MCU Multi-Config Scan_ADC Example

Document Number: 002-20974

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5947224	GJV	02/19/2018	New code example

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Arm® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

Cypress Developer Community

Community Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709