AN47215

PSoC® RC Oscillator to Accurately Time Sleep Cycles

Author: Ben Kropf

Associated Project: Yes

Associated Part Family: CY8C20x34, CY8C21x23, CY8C21x34, CY8C23x33, CY8C24x23A, CY8C24x94, CY8C27x43, CY8C29x66, CYWUSB6953

Software Version: PSoC® Designer 5.4

Related Application Notes: None

To get the latest version of this application note, or the associated project file, please visit http://www.cypress.com/go/AN47215.

Many PSoC® applications require the use of sleep mode for low power operation. This application note describes a method of using an external RC oscillator to create sleep cycles with an accurate period. This method uses external components that are cheaper than an external crystal.

Contents

Introduction

Sleep mode is often used in PSoC applications. One common way of waking up a PSoC from sleep mode is by using the interrupt of the sleep timer circuit. This timer is clocked by the internal low speed oscillator (ILO) or the external crystal oscillator (ECO) of a PSoC. The ILO is the only internal oscillator that a PSoC can use to wake itself up. This is because all other internal oscillators are shut down during sleep. The ILO has a nominal frequency of 32 kHz. However, this frequency can deviate by –50% to +100% from the nominal frequency [1]. The ILO can be bypassed to use an ECO instead, to achieve an accurate 32.768 kHz clock frequency. However, some PSoC device families do not have this option. There are two primary options for timing sleep cycles:

- No ECO - very low timing accuracy
- Use ECO - higher system cost

This application note describes an external resistor-capacitor (RC) oscillator solution that is a good compromise between the two options. This method improves sleep cycle timing accuracy when compared to the first option. And the added system costs of this method are lower than those of the second option.

[1] See the F_{32K1} specification in each specific PSoC device data sheet.

Circuit

Figure 1 shows a PSoC interfaced to an external RC oscillator circuit. Before going to sleep, the firmware code sets an output pin to drive HIGH. The voltage at the $v_{OSC}(t)$ node is seen in Equation 1. The input pin of Figure 1 has a threshold voltage at which it switches from LOW to HIGH. This threshold voltage is referred to as V_{TH}. When $v_{OSC}(t)$ in Equation 1 reaches V_{TH}, the input pin changes from LOW to HIGH and triggers an interrupt. This GPIO interrupt brings the PSoC out of sleep mode to resume normal operation.

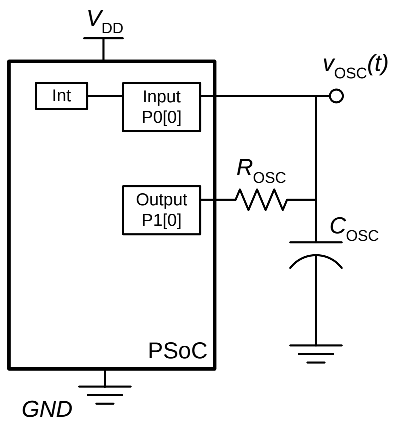
Equation 2 is a solution for the time it takes for the $v_{OSC}(t)$ signal to reach V_{TH} and wake up the PSoC. T_{SLEEP} is the period of the sleep cycle.

$$v_{OSC}(t) = V_{DD}(1 - e^{-t/R_{OSC}C_{OSC}}) \qquad \text{Equation 1}$$

$$T_{SLEEP} = -R_{OSC}C_{OSC}\ln(1 - \tfrac{V_{TH}}{V_{DD}}) \qquad \text{Equation 2}$$

From Equation 2, it is clear that R_{OSC} and C_{OSC} can be selected to choose a suitable T_{SLEEP} for an application. Assume V_{DD} is 5 V, V_{TH} is 2.5V, and the desired T_{SLEEP} is 1s. When these values are used in Equation 2, the values of R_{OSC} and C_{OSC} must be chosen to have a product of approximately 1.44.

Figure 1. External RC Circuit



This circuit needs some method to quickly discharge C_{OSC}. This is because the RC circuit must be reset to start a new sleep cycle. Figure 2 shows the same circuit with the addition of a switch that can connect to *GND* to discharge the capacitor. During sleep when C_{OSC} is charging, P0[0] is configured as a high impedance input. This is equivalent to the switch in Figure 2 being open. After the P0[0] GPIO interrupt wakes the chip from sleep mode, P0[0] is reconfigured as an output pin driving LOW. This is equivalent to the switch in Figure 2 being closed. This allows all charge to flow out of the capacitor into *GND*, which resets the C_{OSC} voltage at 0 V. In Figure 2, P0[0] functions as both an input and a switch that resets the $v_{OSC}(t)$ signal to 0 V by discharging the capacitor.

Figure 2. External RC Circuit with Discharging Pin

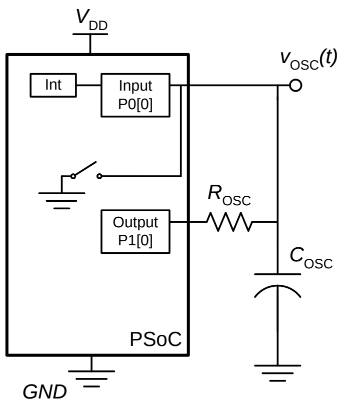


Figure 3. Waveform of $v_{osc}(t)$ Signal

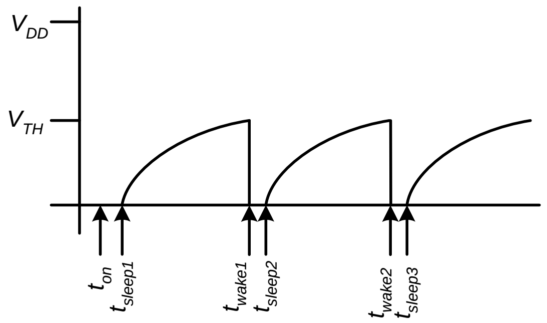


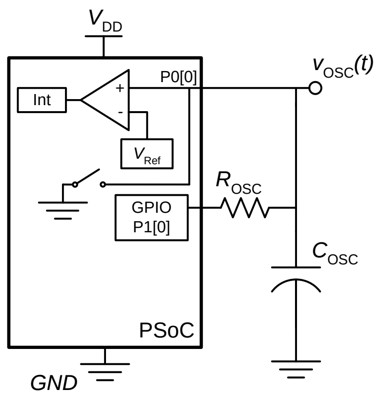
Figure 3 shows the $v_{OSC}(t)$ waveform seen at the capacitor. The device continually goes into sleep mode, wakes up for a short period of time to do its intended operation, and then goes back to sleep. The t_{on} time is when the device first boots up and starts executing initialization code. The t_{wakex} times are when the V_{TH} voltage is reached. This wakes the device up with an interrupt. Just after exiting sleep mode, P1[0] is set LOW and P0[0] is put in a driving LOW state to discharge the capacitor, resetting $v_{OSC}(t)$ at 0 V. The t_{sleepx} times are when the device enters into sleep mode. Just before doing so, P0[0] is put back in a high impedance state and P1[0] is set HIGH.

There is a major drawback with this method. V_{TH} is a value between the V_{IH} and the V_{IL} specifications of a PSoC device. Typically, this is a value between 0.8 V and 2.1 V [2]. This possible variance in V_{TH} causes a variance in T_{SLEEP} from Equation 2. For a V_{DD} of 5 V, this variance in T_{SLEEP} is approximately –49% to +59% from the nominal. This accuracy does not account for variances in the value of C_{OSC} and it is still almost as inaccurate as the ILO.

Using a More Accurate Reference

Figure 4 shows the circuit from Figure 2 with a comparator instead of a GPIO input to detect the $v_{OSC}(t)$ signal voltage. The reference for this comparator, V_{Ref}, is much more accurate than V_{TH}. This allows T_{SLEEP} to also be much more accurate. The following sections describe different ways of implementing the comparator seen in Figure 4.

Figure 4. RC Oscillator Using Comparator



Fixed Reference Implementation

All PSoC device families can implement a comparator with a fixed reference. For this application, it is desirable that the comparator consumes as little current as possible when operating. This is because the comparator must be active while the device is in sleep mode when low power operation is essential. The CY8C21x23 and CY8C21x34 device families should use the CMP User Module. It is the only comparator user module available for these devices and it typically consumes 10 µA of current when operating. The CY8C23x33, CY8C24x23A, CY8C24x94, CY8C27x43, and CY8C29x66 device families should use the CmpLP User Module for this application. This is because it is the comparator user module that uses the least amount of current. The typical current consumption of this user module is also 10 µA when operating.

For the CMP User Module, V_{Ref} is a fixed internal reference, an external reference, or a programmable reference created with other analog circuitry in the device. For this application, it is recommended to use the fixed reference. This is because this requires no external components and consumes less power when compared with the other two options. The fixed reference in these devices is the internal bandgap voltage (V_{BG}) and typically ranges between 1.28 V and 1.32 V [3]. Equation 3 is the new solution for the sleep cycle time when using the CMP User Module along with the V_{BG} internal fixed reference. The accuracy of T_{SLEEP} when using V_{BG} as the reference is ±1.8% (does not factor in C_{OSC} or V_{DD} variance). This is a good improvement over the previous method. It is also more accurate than the ILO.

$$T_{SLEEP} = -R_{OSC}C_{OSC}\ln(1-\frac{V_{BG}}{V_{DD}}) \qquad \text{Equation 3}$$

For the CmpLP User Module, the V_{Ref} voltage is always an internal reference. This reference is a function of the device's power supply voltage. It has 14 possible values between $0.021V_{DD}$ and $0.75V_{DD}$. Equation 4 gives the new T_{SLEEP} solution when using the CmpLP User Module. The k term is the V_{DD} multiplier selected for the CmpLP User Module. Equation 5 is a simplified version of Equation 4. The V_{DD} term is eliminated. The solution of Equation 5 is advantageous because it does not depend on the power supply voltage. Therefore, the accuracy of T_{SLEEP} ideally depends only on variances in C_{OSC}.

$$T_{SLEEP} = -R_{OSC}C_{OSC}\ln(1-\frac{kV_{DD}}{V_{DD}}) \qquad \text{Equation 4}$$

$$T_{SLEEP} = -R_{OSC}C_{OSC}\ln(1-k) \qquad \text{Equation 5}$$

[2] See the V_{IL} and V_{IH} specifications in each specific PSoC device data sheet.

[3] See the BG specification in each specific PSoC device data sheet.

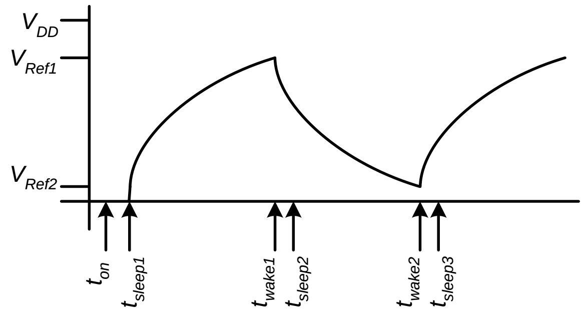
For the fixed reference implementations described in this section, the $V_{OSC}(t)$ waveform looks like that shown in Figure 3. V_{TH} in Figure 3 is either kV_{DD} or V_{BG}, depending on which comparator user module is used.

For a description of the accompanying example project that implements this method, see the Fixed Reference Project section in this document. The section contains more details about actually implementing this in the PSoC.

Dynamic Reference Implementation

The CY8C23x33, CY8C24x23A, CY8C24x94, CY8C27x43, and CY8C29x66 device families use the CmpLP User Module for the comparator implementation of Figure 4. The reference for this comparator user module is dynamically programmable. Therefore, it is possible to change the reference each time the chip exits sleep mode. This allows a $v_{OSC}(t)$ waveform seen in Figure 5. The software flowchart to implement this is shown in Figure 6. The time labels next to the flowchart show which processes correspond to the waveform of Figure 5.

Figure 5. $v_{OSC}(t)$ Waveform when Changing the Reference



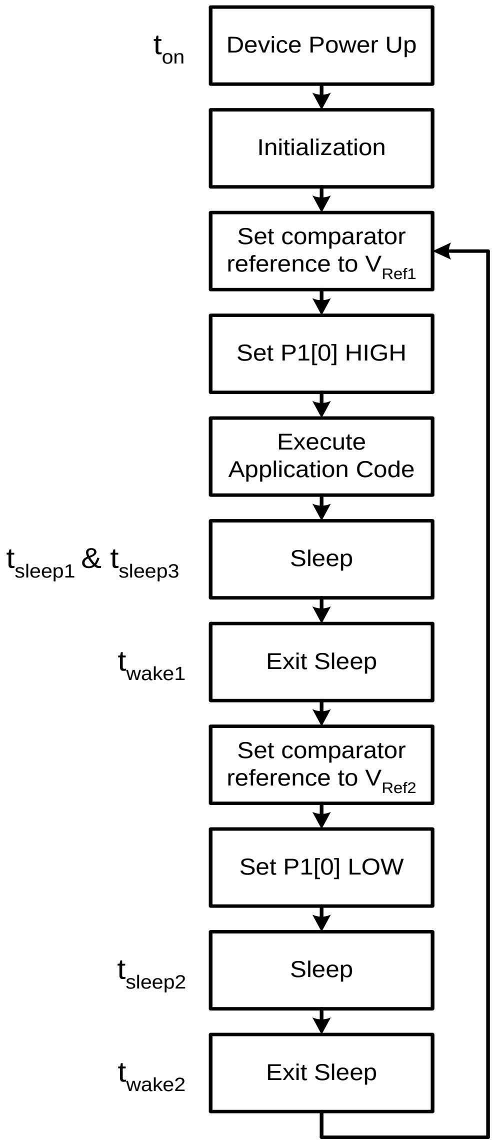
Equation 6 gives the period of the sleep cycle during the rising edge of $v_{OSC}(t)$. Equation 7 gives the period of the sleep cycle during the falling edge of $v_{OSC}(t)$. A small portion of each sleep cycle is used by normal program execution. Equation 6 is the solution for $v_{OSC}(t)$ to rise from V_{Ref2} up to V_{Ref1}. Equation 7 is the solution for $v_{OSC}(t)$ to decay from V_{Ref1} down to V_{Ref2}.

$$T_{SLEEP1} = -R_{OSC}C_{OSC}\ln\left(1-\frac{V_{Ref1}-V_{Ref2}}{V_{DD}}\right) \quad \text{Equation 6}$$

$$T_{SLEEP2} = -R_{OSC}C_{OSC}\ln\left(\frac{V_{Ref2}}{V_{Ref1}}\right) \quad \text{Equation 7}$$

The advantage of dynamically changing the comparator reference is that the RC circuit can achieve longer sleep cycles of T_{SLEEP1} added with T_{SLEEP2} while using the same amount of current as before. The power consumption of the RC circuit is discussed in the following section.

Figure 6. Flowchart for Sleep Cycles Using Two References



For a description of the accompanying example project that implements this method, see the Dynamic Reference Project section in this document. The section contains more details about actually implementing this in the PSoC.

RC Circuit Power Consumption

The RC circuit consumes some amount of power because the capacitor is being charged and discharged repeatedly. It is desirable to minimize this power as much as possible. When C_{OSC} is fully charged up to the upper reference, it has the stored charge seen in Equation 8. The V_{Ref} term in the equation is the upper reference being used.

$$Q_C = V_{Ref} C_{OSC} \qquad \text{Equation 8}$$

If C_{OSC} is discharged completely at the end of each sleep cycle as seen in Figure 3, the current used to charge C_{OSC} is discarded without getting any value from it. The average current consumption of the RC circuit is the amount of charge stored in the capacitor divided by the length of the charging time. This average current consumption is given by Equation 9.

$$\bar{I}_{RC} = \frac{Q_C}{T_{SLEEP}} \qquad \text{Equation 9}$$

Equation 10 uses substitution to combine Equations 8 and 9. It shows that an easy way to minimize the RC current is to minimize C_{OSC}. However, decreasing C_{OSC} also decreases T_{SLEEP} as seen in Equation 3. Therefore, R_{OSC} must be increased to maintain the desired T_{SLEEP}. Unfortunately, it is not practical to choose an arbitrarily large value for R_{OSC}. As the resistance of R_{OSC} increases beyond approximately 10 MΩ, resistors are more difficult to obtain and the prices are marginally more expensive than smaller value resistors. In addition, very large resistances can create situations where contaminants on the PCB start having a noticeable effect on the circuit.

$$\bar{I}_{RC} = \frac{V_{Ref} C_{OSC}}{T_{SLEEP}} \qquad \text{Equation 10}$$

Dynamically changing the reference (as seen in Figure 5) further reduces the current consumption of the RC circuit. This is because C_{OSC} is discharged slowly. The slow discharge of C_{OSC} is used to time out part of each sleep cycle. Therefore, C_{OSC} is only charged up during part of each sleep cycle. This reduces the average current used by the RC circuit.

Generally, both solutions discussed in this application note are most effective when T_{SLEEP} is 100 ms to 1000 ms, C_{OSC} is 0.1 μF to 1.0 μF, and R_{OSC} is 100 kΩ to 10 MΩ. These target ranges provide a good balance of achieving a long T_{SLEEP}, a low average I_{RC}, and a low cost for the passive components.

When a very long period of time is required between each application code execution, it is better for most applications to use multiple sleep cycles instead of longer sleep cycles. For example, if one minute is needed between each time the processor wakes up to execute its primary functionality, then it is better to use 600 sleep cycles with a period of 100 ms instead of creating an RC circuit to give a one minute sleep cycle.

Choosing the Right Capacitor

All the methods described above depend heavily on the variance of the C_{OSC} capacitor used for the system. Because it is easy to get 1% accurate resistors, the capacitor variance primarily determines the variance in T_{SLEEP}. Therefore, it is possible to determine the accuracy of the sleep cycle by choosing a particular capacitor. Good capacitors with less variance tend to cost more. In each case they should still be less expensive than crystals. This is an advantage as the designer only needs to pay for as much accuracy as needed.

The data sheet for the capacitor manufacturer must be consulted for the accuracy of the capacitor used. The table below shows the typical accuracy of capacitance values for common dielectric types.

Table 1. Typical Accuracies for Capacitor Types [4]

Capacitor Type	Capacitance Accuracy	Typical Maximum Capacitance
C0G ceramic	5%	0.047 μF
NPO ceramic	5%	0.047 μF
X7R ceramic	10%	0.33 μF
Y5V ceramic	+20/–80%	22 μF
Z5U ceramic	20%	2.2 μF
2E6 ceramic	20%	2.2 μF
PZT ceramic	1%	1 μF
Polycarbonate	1%	10 μF
Polyester	10%	10 μF
Electrolytic	+80/–20%	>1,000 μF
Tantalum	5% to 20%	>100 μF
Polystyrene	0.5%	10 μF

Example Projects

There are two example projects that accompany this application note. Both of them implement a form of the solution presented in this application note. Both projects have the user modules required for this application in a second configuration of the project. To see the user modules used for this application, click on the *Sleep* tab near the top of the Device Editor view. The example projects are discussed in the following sections.

[4] Source of data is www.wikipedia.org and www.avx.com. Always consult a capacitor manufacturer's documentation to obtain the most reliable and accurate specifications for capacitors.

Fixed Reference Project

The first example project is named “FixedRef”. It implements the solution discussed in the Fixed Reference Implementation section of this application note. Figure 4 is the circuit used with the example project. The CY8C21534-24PVXI device is used for the project. This project can be implemented with the CY3210-PSoCEval1 kit along with a 28-pin SSOP to DIP adapter socket.

The input to the comparator is on the P0[0] pin. This pin is chosen because it is an analog input pin. Therefore, there is a signal path from P0[0] to the input of the CMP User Module. Any of the Port 0 pins can also be used for this purpose.

The output pin that drives the RC oscillator is on the P1[0] pin. This pin was chosen because it is also the programming data signal pin for the device. In most applications, it is best to avoid sharing this GPIO with other functions of the device. However, this port pin can easily be shared between the programming and the RC oscillator functionality. To allow successful programming, P1[0] must not be loaded with less impedance than 120 pF in parallel with 1 kΩ[5]. In this application, the RC circuit loads this pin with a large resistance in series with a capacitance. If the RC circuit has proper R and C values as discussed earlier, then the impedance they introduce will not be too small to allow programming. Therefore, by sharing P1[0] as the RC circuit driver and a programming pin, a second GPIO need not be used as the RC circuit driver.

For this project, R_{OSC} is a 1 MΩ, ±1% resistor and C_{OSC} is a 0.1 μF, ±10% X7R ceramic capacitor. V_{Ref} is the fixed V_{BG} voltage which is nominally 1.3V and V_{DD} is 5 V. This creates a nominal T_{SLEEP} of 30 ms with an accuracy of approximately ±11%.

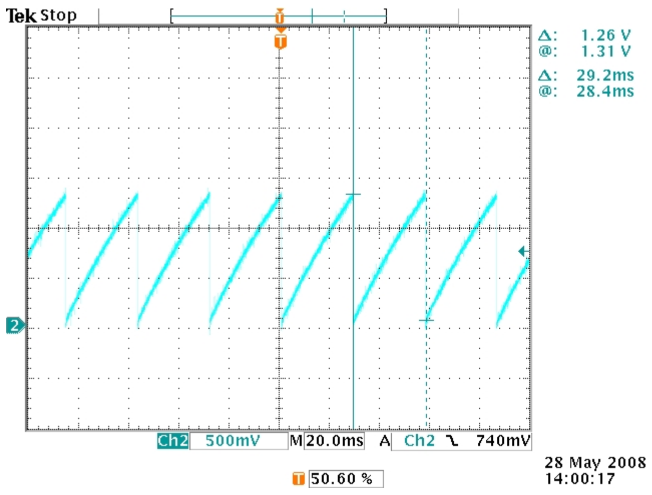
There are two primary functions in *main.c*. The first is named `Sleep` and is used to execute a number of sleep cycles determined by a parameter passed to the function. There is a defined label near the top of *main.c* called `TOTAL_SLEEP_TIME`. The number of desired milliseconds is entered for this label to determine the total length of sleep time. The `TSLEEP` definition is defined to be 30 to match the 30.1 ms solution for T_{SLEEP}. The software automatically calculates the number of sleep cycles to execute to achieve the total sleep time desired.

The second primary function is named `DoSomething`. All it does is toggle a GPIO port pin. This function represents the user’s application code. The HIGH time of this toggled GPIO pin should be approximately one second. This signal output is on P1[2].

This project uses dynamic reconfiguration to load all hardware used to time the sleep cycles. Any user modules and hardware used during normal program execution need not be used during sleep in most applications. Therefore, the *Sleep* configuration of the project is loaded just before the long sleep cycle. Although this project uses an analog block for the comparator, it is only used during the sleep cycle. Therefore, the same analog block can be used for something else during normal operation[6]. The base configuration of the project has nothing in it. It can be filled up with user modules to accomplish normal device operation.

Figure 7 shows an oscilloscope screenshot of the $v_{OSC}(t)$ signal for this project. The cursors in the figure show the T_{SLEEP} time to be 29.2 ms, which is close to the nominal of 30 ms.

Figure 7. $v_{OSC}(t)$ Waveform for the FixedRef Project



Dynamic Reference Project

The second example project is named “DynRef”. It implements the solution discussed in the Dynamic Reference Implementation section of this application note. Figure 4 shows the circuit that is also used with this example project. The CY8C27443-24PXI device is used for the project. This 28-pin DIP package device can be used with the CY3210-PSoCEval1 kit to easily implement this project.

This “DynRef” example project is similar to the “FixedRef” example project. The differences between the two are discussed in this section.

[5] See application note AN2014 for more information on ISSP programming.

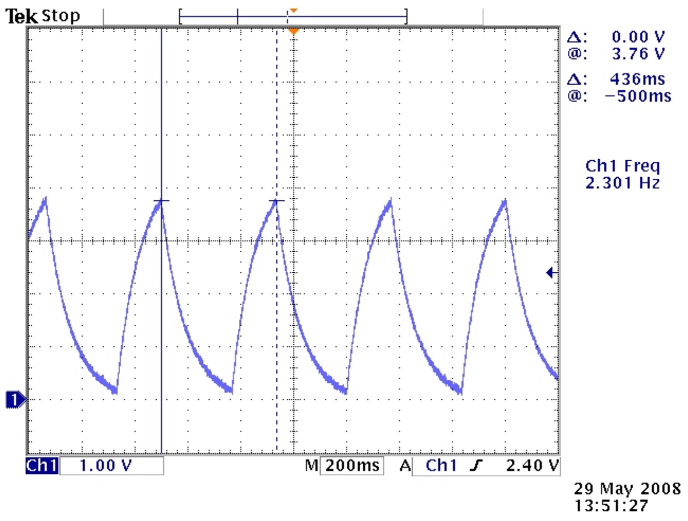
[6] See application note AN2014 for more information on Dynamic Reconfiguration.

For this project, R_{OSC} is a 1 MΩ, ±1% resistor and C_{OSC} is a 0.1 μF, ±10% X7R ceramic capacitor. V_{Ref1} is 3.75 V ($0.75V_{DD}$) and V_{Ref2} is 210 mV ($0.042V_{DD}$). V_{DD} is also 5 V for this project. This creates a nominal T_{SLEEP} of 411 ms (solved with Equations 6 and 7) with an accuracy of about ±11%.

This project also has a `Sleep` function and a `DoSomething` function. Both of these functions do the same thing they do in the “FixedRef” project. However, the `Sleep` function in this project must take care of executing one sleep cycle on the rising edge of $v_{OSC}(t)$ and a second sleep cycle on the falling edge of $v_{OSC}(t)$. Also, much of the code that manages the $v_{OSC}(t)$ signal is moved to the CMP_LP user module’s interrupt service routine (ISR). This ISR is written in C code and is located in *main.c*. Because of the longer sleep cycles in this project, only two of them are executed in *main.c*. This means the toggled pin output signal on P1[2] should have a HIGH time of approximately 822 ms because each sleep cycle is 411 ms long.

Figure 8 shows an oscilloscope screenshot of the $v_{OSC}(t)$ signal for this project. The cursors in the figure show the total T_{SLEEP} time to be 436 ms, which is close to the nominal of 411 ms and is within the expected accuracy of 11%.

Figure 8. $v_{OSC}(t)$ Waveform for the DynRef Project



Summary

This application note discusses methods to achieve sleep timing that is more accurate than when using the ILO. The methods are also less expensive than using an external crystal. This application note is also applicable for any other application where a reference frequency that is more accurate than the PSoC's internal sleep timer is needed, without the extra cost of a crystal. Two example projects that implement the methods discussed accompany this application note.

About the Author

Name:	Ben Kropf.
Title:	Applications Engineer Staff
Background:	Graduated with a B.S. degree in Electrical Engineering from Seattle Pacific University. Professional experience includes mixed-signal embedded system design, firmware and software design, high brightness LED applications, and switching power supplies.

Document History

Document Title: PSoC® RC Oscillator to Accurately Time Sleep Cycles - AN47215

Document Number: 001-47215

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	2525145	BTK	07/02/08	New application note.
*A	3320371	ANUP	07/19/11	Updated Software Version. Updated Example Projects.
*B	4461278	PMAD	07/30/2014	Updated Software Version as “PSoC® Designer 5.4”. Updated Example Projects. Updated attached Example Projects. Updated in new template. Completing Sunset Review.

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

Automotive	cypress.com/go/automotive
Clocks & Buffers	cypress.com/go/clocks
Interface	cypress.com/go/interface
Lighting & Power Control	cypress.com/go/powerpsoc cypress.com/go/plc
Memory	cypress.com/go/memory
PSoC	cypress.com/go/psoc
Touch Sensing	cypress.com/go/touch
USB Controllers	cypress.com/go/usb
Wireless/RF	cypress.com/go/wireless

PSoC® Solutions

psoc.cypress.com/solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Community | Forums | Blogs | Video | Training

Technical Support

cypress.com/go/support

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

Phone	: 408-943-2600
Fax	: 408-943-4730
Website	: www.cypress.com