

## Simulating a 555 Timer with PSoC™

**Author:** Dave Van Ess  
**Associated Project:** Yes  
**Associated Part Family:** Any  
**PSoC Designer Version:** 4.2

### Abstract

The PSoC Mixed-Signal Array is a versatile system-on-chip that facilitates complete system design. But can it do everything? It is a truth universally acknowledged, that any single circuit worthy of construction can be built with 555 timers. It follows that if the PSoC can simulate a 555 timer, then PSoC can be used to design any circuit of significance. Included in this Application Note is a brief history of the 555 timer. A PSoC device simulation is developed and examples of retro 555 timer applications are shown.

### Introduction

The 555 timer IC was first introduced in 1971 by the Signetics Corporation and called the “**The IC Time Machine.**” Designed by Hans Camenzind, it became the highest-volume timing IC for its time. Although not as widely used today, in the 70s it met with excitement and approval from both Engineers and Hobbyists alike. The 555 timer still has a loyal following that refuses to die off. It can be thought of as the leisure suit of integrated circuits.

In the 30s, several mathematicians began to think about what it meant to be able to compute a function. Alan Turing proposed a hypothetical computing model. It is respectfully called a “**Turing Machine.**” Simply stated:

*A function is computable if it can be computed with a Turing Machine.*

The 555 timer’s versatility had many believing that most anything could be built with one. Simply stated:

*A circuit is build-able if it can be built with a 555 Timer.*

This postulate has never really achieved popularity nor did Hans Camenzind get the respect he deserved. Perhaps if the 555 timer had been called the “**Camenzind Machine**” it would have drawn more academic interest and respect.

Still, in the long run, Camenzind has designed over 100 ICs and written well over 25 articles and books on IC design. Turing, on the other hand, suffering from depression, did not make it past his 42<sup>nd</sup> birthday. All in all, being a live, content engineer beats being a despondent, dead mathematician.

### Theory

The 555 timer consists of two voltage comparators, a bistable flip-flop, a discharge transistor, and a resistor divider network. A block diagram is shown is Figure 1.

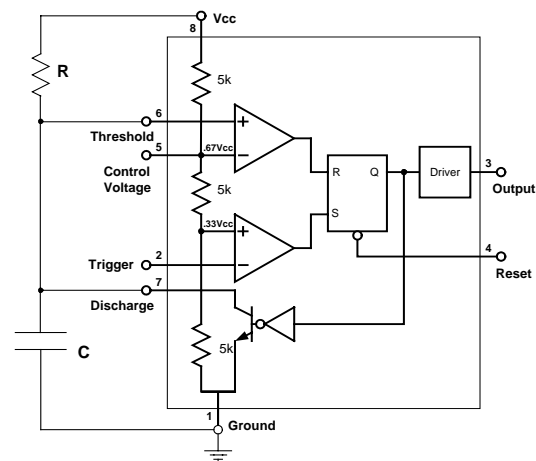


Figure 1. 555 Timer Block Diagram

The resistive divider network is used to set the comparator levels. Since all three resistors are of equal value, the threshold comparator is referenced internally to 2/3 of supply voltage and the trigger comparator is referenced to 1/3 of supply voltage. The outputs of the comparators are tied to the bistable flip-flop.

When the trigger voltage is moved below the 1/3  $V_{cc}$  point, the comparator changes state and sets the flip-flop driving the output high. When the output is high, the discharge transistor turns off. The threshold pin normally monitors the capacitor voltage of the external RC timing network. When the capacitor voltage exceeds  $2/3V_{cc}$ , the threshold comparator resets the flip-flop causing the output to go low. When the output is low, the discharge transistor turns on, thereby discharging the external timing capacitor. Once this capacitor is discharged, the timer awaits another trigger pulse, having completed the timing cycle. If a low impedance voltage level is placed on the Control Voltage pin, this voltage externally sets the comparator reference points. Equation (1) defines the comparator levels.

$$\begin{aligned} V_{Threshold} &= V_{ControlVoltage} \\ V_{Trigger} &= V_{ControlVoltage} / 2 \end{aligned} \quad (1)$$

The Reset pin is an internally pulled-up input. When pulled low, the timing cycle is immediately terminated.

Any part trying to simulate a 555 timer must meet all these requirements.

### Implementation

The 555 timer was originally packed as an 8-pin DIP. Literature from that period referred to it as an 8-pin **mini-DIP package**. Apparently its extremely small size was something to boast about back then. Figure 2 shows the original pinout.

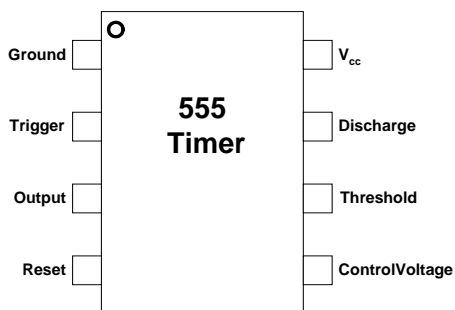


Figure 2. 555 Timer Pinout

To keep in the spirit of this simulation, an 8-pin DIP PSoC will be used. A **CY8C27143** is the PSoC 8-pin DIP package option.

The pin 1 location for ground on the original pinout makes a pin-for-pin match impossible. Figure 3 shows the PSoC pinout defined for easy construction.

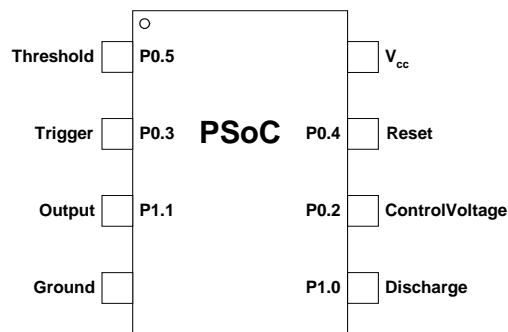


Figure 3. 555 Timer PSoC Simulation Pinout

Figure 4 shows the block diagram for implementing a simulation when the comparator reference values are internally generated.

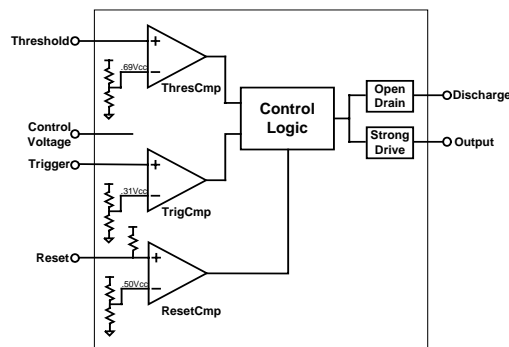
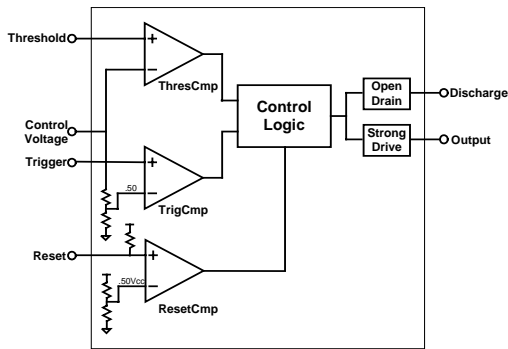


Figure 4. Internally Generated Comparator Reference Values

Figure 5 shows the block diagram for implementing a simulation when the comparator reference values are externally generated.



**Figure 5. Control Voltage Generated Comparator Reference Values**

Each particular 555 application determines which simulations will be used. Fortunately, all chips in the PSoC family have Flash memory to store their configuration. The chip can be easily reprogrammed with the correct configuration.

The Reset line must have an internal resistive pull up. The Output must have a strong drive output while the Discharge must be open drain. All the others are analog inputs. Figure 6 shows all the pins configured for type and drive strength.

Name	Port	Select	Drive
ControlVoltage	P0[2]	AnalogInput	High Z Analog
Trigger	P0[3]	AnalogInput	High Z Analog
Reset	P0[4]	StdCPU	Pull Up
Threshold	P0[5]	AnalogInput	High Z Analog
Discharge	P1[0]	GlobalOutOdd_0	Open Drain Low
Output	P1[1]	GlobalOutOdd_1	Strong

**Figure 6. Selection of Pin and Drive Types**

Reset is actually configured as a pull-up output. When logic high is written to the data register for this pin, it appears as a resistor pulling this pin to  $V_{cc}$ . The input to the analog blocks is still valid. ResetCmp is a comparator that converts this analog signal back to a digital level accessible to the digital blocks.

Given the description of the operation as described earlier, Equation (2) defines the logical equation to implement.

$$Output = ((Output * Threshold) + !Trigger) * Reset \quad (2)$$

$$Discharge = Output$$

The digital blocks, configured as buffers, along with global logic connections are used to implement Equation (2). Appendix A shows the routing while the routing path is described below.

- The output on **GOO[1]** routes back to **GIO[1]**.
- **GIO[1]** routes back to **RIO[0]**.
- **RIO[0]** passes through a **ThrashBuf** buffer and connects to **RO0[3]**.
- The threshold comparator connects to the other **ThrashBuf** buffer and outputs to **RO0[2]**.
- **RO0[2]** and **RO0[3]** logically combine and connect to **GOE[6]**.
- **GOE[6]** routes back to **GIE[6]**.
- **GIE[6]** connects to **RI1[2]**.
- **RIE[2]** passes through a **TrigBuf** buffer and connects to **RO1[3]**.
- The trigger comparator connects to the other **TrigBuf** buffer and outputs to **RO[0]**.
- **TRO1[0]** and **RO1[3]** logically combine and connect to **GOE[7]**.
- **GOE[7]** routes back to **GIE[7]**.
- **RIE[7]** is connected to **RI1[3]**.
- **RIE[3]** passes through a **ResetBuf** buffer and connects to **RO1[1]**.
- The reset comparator connects to the other **ResetBuf** buffer and outputs to **RO1[2]**.
- **RO1[1]** and **RO1[2]** logically combine and connect back to the original **GOO[1]**.
- The **Output** also routes to **Discharge** where it eventually connects to **GOO[0]**.

Appendix B shows the placement of the analog blocks required for the analog front-end shown in Figure 4. It requires only three continuous time block comparators. It outputs on Compactor0, Comparator2 and Comparator3.

Appendix C shows the placement of the analog blocks required for the analog front-end shown in Figure 5. It requires three analog buffers, one switched capacitor block for signal routing, two switched capacitor block comparators, and one continuous time reset comparator, for a total of seven analog blocks.

Code Example 1 shows the software required to control this application.

```

;-----
; 555 Timer Example
;-----
include "m8c.inc"
include "PSoCAPI.inc"
export _main
_main:
  mov reg[PRT0DR],10h ;activate pullup
  mov A,ThreshCmp_HIGHPOWER
  call ThreshCmp_Start
  mov A,TrigCmp_HIGHPOWER
  call TrigCmp_Start
  mov A,ResetCmp_HIGHPOWER
  call ResetCmp_Start
loop:
  jmp loop
ret

```

### Code Example 1

Note that the software is used only to start the comparator user modules and set P0[4] high. The CPU is not used during actual operation.

## Results

Figure 7 shows the schematic for a 555 oscillator.

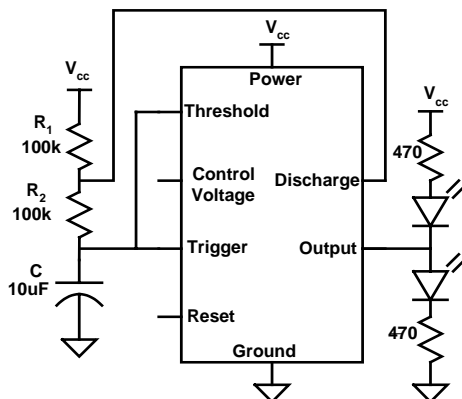


Figure 7. Oscillator Configuration

When the output is high, the discharge path is open and capacitor charges through  $R_1$  and  $R_2$ . This continues until the capacitor voltage reaches the threshold value causing the output to turn off. When the output is low, the discharge path closes and the capacitor discharges through  $R_2$  to ground. When the capacitor voltage drops to the trigger value, the output goes high and the cycle starts again. Equation (3) allows the on and off time to be calculated given the resistive and capacitance values.

$$\begin{aligned}
 t_{on} &= .785 \cdot (R_1 + R_2) \cdot C \\
 t_{off} &= .785 \cdot R_2 \cdot C
 \end{aligned}
 \quad (3)$$

For 100k resistors and a 10 uF capacitor, the on time should be about 1.8 seconds and the off time 0.8 seconds.

Figure 8 shows the oscilloscope traces of the circuit in Figure 7.

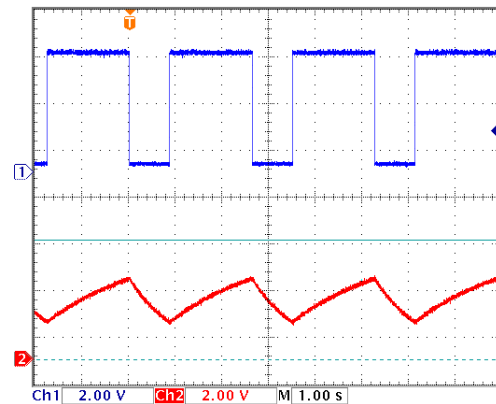


Figure 8. 555 Results of 555 Simulation

The top trace is the output. Note that it is high for 1.8 seconds and low for 0.9 seconds. With resistor tolerances of 10% and a capacitor tolerance of 10%, this agrees with values generated by Equation (3). The lower trace is the capacitor voltage. Please note that it stays between the threshold and trigger values.

Figure 9 shows the test circuit used to generate the scope traces above.

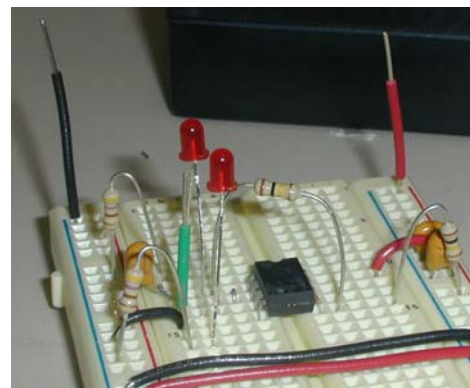


Figure 9. Photograph of Test Circuit

Figures 10 through 15 illustrate different circuits, and their associated waveforms implemented with 555 timers. All of these can be implemented with the PSoC 555 simulator.

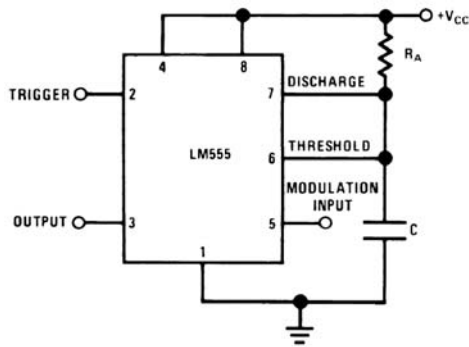
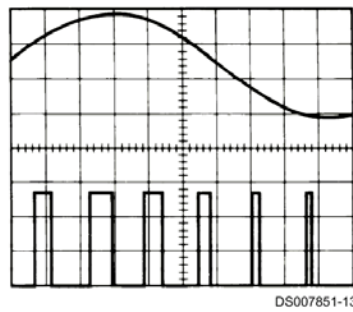


Figure 10. Pulse Width Modulator



DS007851-13  
 $V_{CC} = 5V$  Top Trace: Modulation 1V/Div.  
 TIME = 0.2 ms/DIV. Bottom Trace: Output Voltage 2V/Div.  
 $R_A = 9.1k\Omega$   
 $C = 0.01\mu F$

Figure 11. Pulse Width Modulator Waveforms

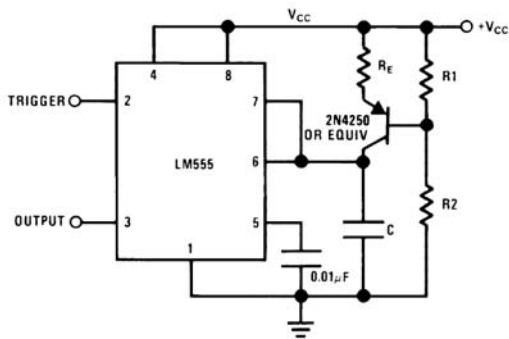
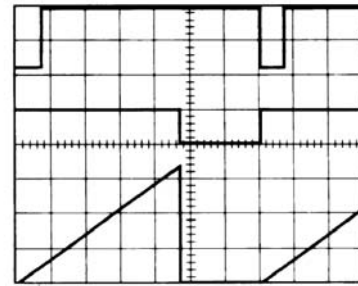


Figure 12. Linear Ramp



DS007851-17  
 $V_{CC} = 5V$  Top Trace: Input 3V/Div.  
 TIME = 20μs/DIV. Middle Trace: Output 5V/Div.  
 $R_1 = 47k\Omega$  Bottom Trace: Capacitor Voltage 1V/Div.  
 $R_2 = 100k\Omega$   
 $R_E = 2.7 k\Omega$   
 $C = 0.01 \mu F$

Figure 13. Linear Ramp Waveforms

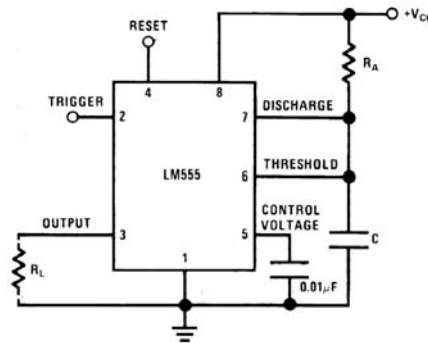
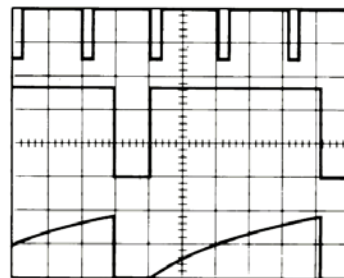


Figure 14. Frequency Divider



DS007851-11  
 $V_{CC} = 5V$  Top Trace: Input 4V/Div.  
 TIME = 20μs/DIV. Middle Trace: Output 2V/Div.  
 $R_A = 9.1k\Omega$  Bottom Trace: Capacitor 2V/Div.  
 $C = 0.01\mu F$

Figure 15. Frequency Divider Waveforms

## Conclusion

It has been shown that a PSoC Mixed-Signal Array can successfully simulate a 555 timer. Since PSoC can simulate a 555 timer, then PSoC can be used to design any circuit of significance.

## Epilogue

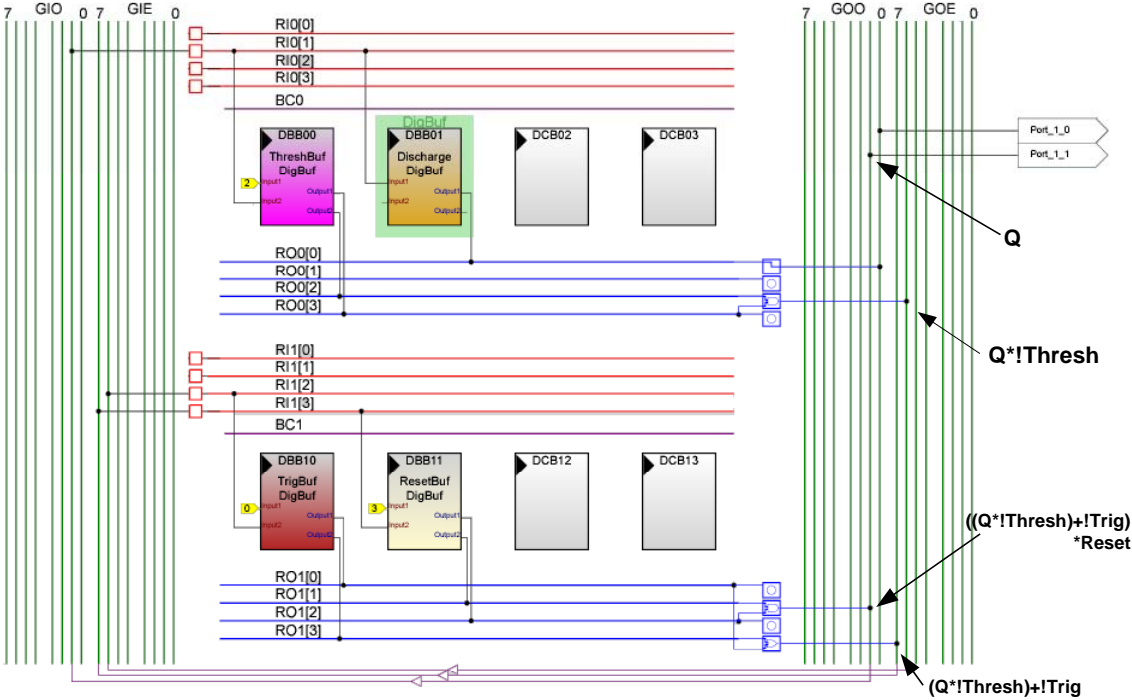
What appears to be a humorous Application Note about simulating old parts with new is actually a demonstration of PSoC versatility. The **CY8YC27xxx** part family has the ability to successfully combine analog and complex digital functions with no direct software assistance. This allows many functions to be easily implemented in hardware, thereby reducing the overall CPU requirement for an efficient system design.

## References

- [1] The 555 Timer Applications Sourcebook, Howard M. Berlin, Howard W. Sams & Co. Inc, 1976.
- [2] LM555 Data Sheet, National Semiconductor Corporation, 2000.

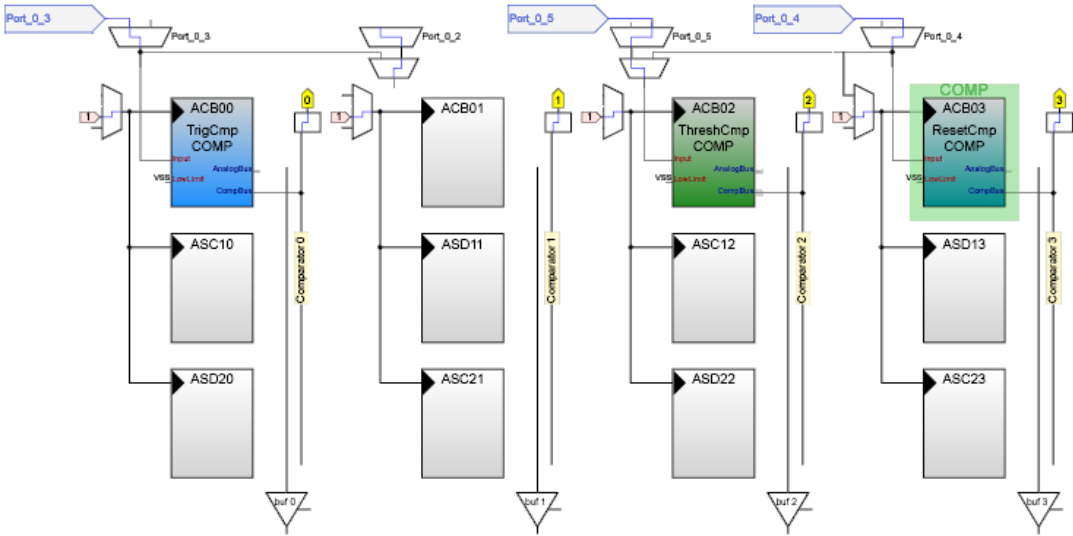
# Appendix A

## PSoC User Module Placement for 555 Control Logic



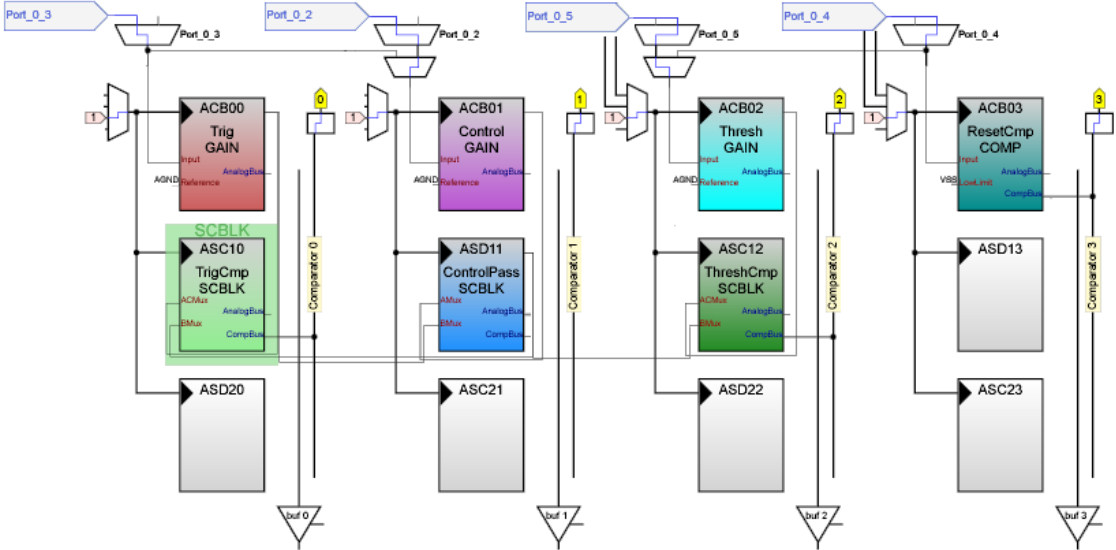
# Appendix B

## User Module Placement for Analog Section Internal Reference Selection



# Appendix C

## PSoC User Module Placement For Analog Section Control Voltage Reference Selection



---

## About the Author

**Name:** Dave Van Ess  
**Title:** Principal Application Engineer  
Cypress Semiconductor

**Background:** An Engineer by training, a poet by temperament, and an outlaw in Nebraska. Dave is capable of abstract thought, concrete analysis, and ruthless implementation. BSEE from University of California, Berkeley. More than 27 Years experience in circuit, signal processing, digital, software, analog, and system design. Holder of six U.S. Patents (plus three pending) for medical systems, signal processing, and digital block enhancements. Author of numerous Application Notes, web casts, and technical articles. Joined Cypress MicroSystems in 2000.

**Contact:** [dww@cypress.com](mailto:dww@cypress.com)

---

Cypress Semiconductor  
2700 162<sup>nd</sup> Street SW, Building D  
Lynnwood, WA 98037  
Phone: 800.669.0557  
Fax: 425.787.4641

<http://www.cypress.com/>

Copyright © 2005 Cypress Semiconductor Corporation. All rights reserved.  
PSoC™, Programmable System-on-Chip™, and PSoC Designer™ are PSoC-related trademarks of Cypress.  
All other trademarks or registered trademarks referenced herein are the property of their respective owners.  
The information contained herein is subject to change without notice. Made in the U.S.A.