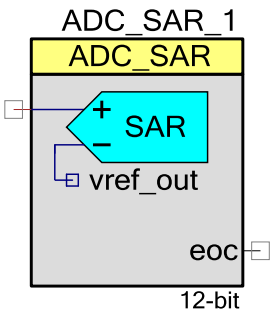
ADC Successive Approximation Register (ADC_SAR)

1.10

Features



- Resolution 12 bit at 1 Msps max.
- Four Power modes
- Selectable resolution and sample rate
- 2 conversion modes
- Differential input

General Description

The ADC Successive Approximation Register (ADC_SAR) component covers any medium-speed (max 1MHz sampling), medium-resolution case (max 12 bits) applications.

When to use a ADC_SAR

Example applications for the ADC_SAR component include:

- **VoIP headset:** sample voice signal from a microphone at a rate of 8Ksps, and digitize it for ADPCM coder to compress in firmware
- **Radio controlled hobbyist transmitter:** sample values from Muxbus, and digitize potentiometer values of a variety of digital buttons and switches
- **Radio controlled hobbyist receiver:** sample values from Muxbus, and digitize values of main and auxiliary voltages, air speed, temperature, and battery levels
- **LED lighting control:** sample values from Muxbus, and digitize LED colors and thermistor output for color and temperature compensation
- **Heart rate monitor:** sample and digitize the amplified biopotential; decimation in firmware to detect QRS peaks
- **Fitness shoes:** sample values from Muxbus, and digitize values from pace transducer and heel strike sensor

Input/Output Connections

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

+Input – Analog

This input is the positive analog signal input to the ADC_SAR. The conversion result is a function of the +Input minus the voltage reference, which is either the –Input or Vssa.

–Input – Analog *

This input is the negative analog signal input to the ADC_SAR. It can also be thought of as the reference input. The conversion result is a function of the +Input minus the –Input. This pin will be visible when the **Input Range** parameter is set to one of the differential modes.

vdac_ref – Input *

The VDAC reference (vdac_ref) is an optional pin. It is shown if you have selected to use "Vssa to VDAC*2" or "0.0 +/- VDAC" **Input Range**, otherwise this I/O will be hidden. This pin is usable for VDAC component output only. No other signal can be connected here.

soc – Input *

The start of conversion (soc) is an optional pin. It is shown if you have selected to use "Triggered" sample mode. A rising edge on this pin will start an ADC conversion if the option is selected. If the **Sample Mode** parameter is set to "Free Running" this I/O will be hidden.

aclk – Input *

This optional pin is present if the **Clock Source** parameter is set to "External", otherwise the pin will not be shown. This clock determines the conversion rate as a function of conversion method and resolution. If the Clock Source is set to "Internal", this I/O will be hidden.

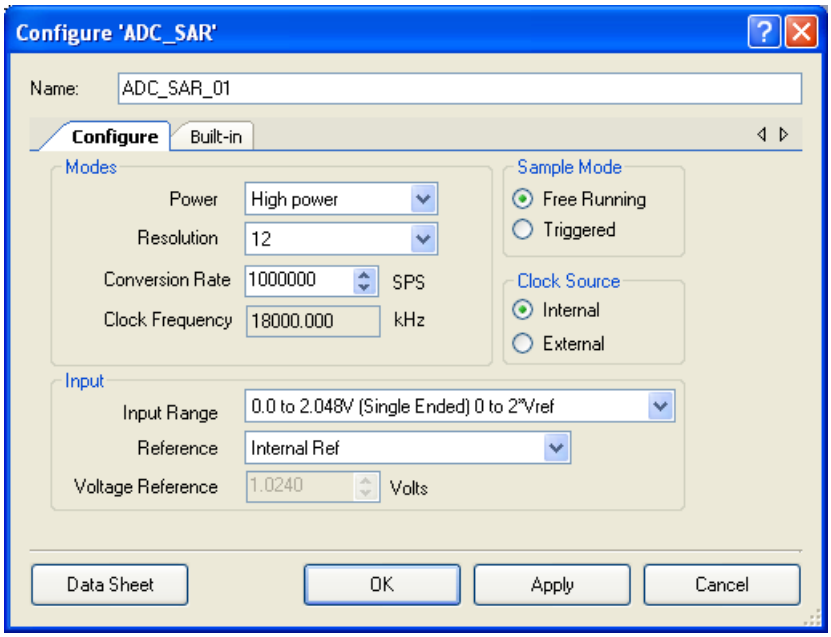
eoc – Output

A rising edge on the End Of Conversion (eoc) signals that a conversion is complete. A DMA request may be connected to this pin to transfer the conversion output to system RAM, DFB, or other component. The internal interrupt is also connected to this signal.

Parameters and Setup

The ADC_SAR component is a highly configurable analog to digital converter. Drag an ADC_SAR component onto your design and double-click it to open the Configure dialog.

Figure 1: Configure ADC_SAR Dialog



The ADC_SAR has the following parameters. The option shown in bold is the default.

Modes

Power

This parameter sets the power level of the ADC. The higher power settings allow the ADC to operate at higher clock rates for faster conversion times. The lower power setting provides a low power alternative when fast conversion speed is not critical.

Parameters Name	Value	Description
High Power	0	Normal power
Medium Power	1	1/2 power
Low Power	2	1/3 power
Minimum Power	3	1/4 power

Resolution

Sets the resolution of the ADC.

ADC_Resolution	Value	Description
12	12	Sets resolution to 12 bits.
10	10	Sets resolution to 10 bits.
8	8	Sets resolution to 8 bits.

Conversion Rate

The ADC conversion rate is selected with this parameter. The conversion time will simply be the reciprocal of the conversion rate. The conversion rate is entered in samples per second. The maximum conversion rate depends on the resolution. To convert one sample, it takes 18 cycles at 12 bit resolution, 16 cycles at 10 bits, and 14 cycles at 8 bits. So, the conversion rate of SAR will be clock frequency / (resolution + 6 cycles).

Clock Frequency

The ADC clock frequency is selected with this parameter in kHz. The maximum frequency of the clock source is 18MHz. Clock frequency can be anywhere between 1MHz to 18MHz.

Sample Mode

This parameter determines how the ADC operates in either Free Running or triggered mode.

Start_of_Conversion	Description
Free Running	Start a conversion which runs continuously.
Triggered	A rising edge pulse on the SOC pin will cause a conversion to start. All conversions need to be triggered individually.

Clock Source

This parameter allows you to select either a clock that is internal to the ADC_SAR module or an external clock.

ADC_Clock	Description
Internal	Use an internal clock that is part of the ADC_SAR component.

ADC_Clock	Description
External	Use an external clock. The clock source may be analog, digital, or generated by another component.

Input

Input Range

This parameter configures the ADC for a given input range. This configures the input to the ADC and is independent of the input buffer gain setting. The analog signals connected to the IC must be between Vssa and Vdda no mater what input range settings are used.

Input Range	Description
0.0 to 2.048V (Single Ended) **0 to Vref*2**	When using the internal reference (1.024V), the usable input voltage to the ADC is 0 .0 to 2.048 Volts. The ADC will be configured to operate in a single ended input mode with the –Input connected internally to Vrefhi_out. If an external reference voltage is used, the usable input range to the ADC will be 0.0 to Vref*2.
Vssa to Vdda (Single Ended)	This mode uses the Vdda/2 reference and the usable input range will cover the full analog supply voltage. The ADC is put in a single ended input mode with the –Input connected internally to Vrefhi_out.
Vssa to VDAC*2 (Single Ended)	This mode uses the VDAC reference, which should be connected to the vdac_ref pin. The usable input voltage to the ADC is 0.0 to VDAC*2 Volts. The ADC will be configured to operate in a single ended input mode with the –Input connected internally to Vrefhi_out.
0.0 +/- 1.024V (Differential) -Input +/- Vref	This mode is configured for differential inputs. When using the internal reference (1.024V), the input range will be –Input +/- 1.024V. If –Input is connected to 2.048 volts the usable input range is 2.048 +/- 1.024V or 1.024 to 3.072V. For systems where both single ended and differential signals are scanned, connect the –Input to Vssa when scanning a single ended input. An external reference may be used to provide a wider operating range. The usable input range can be calculated with the same equation, -Input +/- Vref.
0.0 +/- Vdda (Differential) -Input +/- Vdda	This mode is configured for differential inputs and uses the Vdda reference. The input range will be –Input +/- Vdda. For systems where both single ended and differential signals are scanned, connect the –Input to Vssa when scanning a single ended input.
0.0 +/- Vdda/2 (Differential) -Input +/- Vdda/2	This mode is configured for differential inputs and uses the Vdda/2 reference. The input range will be –Input +/- Vdda/2. For systems where both single

Input Range	Description
	ended and differential signals are scanned, connect the –Input to Vssa when scanning a single ended input.
0.0 +/- VDAC (Differential) -Input +/- VDAC	This mode is configured for differential inputs and uses the VDAC reference, which should be connected to the vdac_ref pin. The input range will be –Input +/- VDAC. For systems where both single ended and differential signals are scanned, connect the –Input to Vssa when scanning a single ended input.

Reference

This parameter selects the ADC_SAR reference voltage and configuration. The reference voltage sets the range of the ADC.

ADC_Reference	Description
Internal Vref	Use the internal 1.024V reference
Internal Vref, bypassed	Use internal 1.024V reference, and place a bypass capacitor on pin P02* for SAR1 or on pin P04* for SAR0.
External Vref	Use external reference on pin P02 for SAR1 or on pin P04 for SAR0.

Note The use of an external bypass capacitor is recommended if the internal noise caused by internal digital switching exceeds what is required for the applications analog performance. To use this option, configure either port pin P02 or P04 as an analog "Hi-Z" pin and connect an external capacitor between 0.01uF and 10uF.

Voltage Reference

The voltage reference is used for the ADC count to voltage conversion functions discussed in the API section. This parameter is non-editable when using the internal 1.024 volt reference. When using an external reference, you may edit this value to match the external reference voltage.

- When selecting input range "Vssa to Vdda", "-Input +/- Vdda" or "-Input +/- Vdda/2" , the Vdda supply voltage value should be entered.
- When selecting the input range "Vssa to VDAC*2" or "–Input +/- VDAC", the VDAC supply voltage value should be entered.

Placement

The ADC_SAR component is placed in one of two available SAR blocks and placement information is provided to the API through the *cyfitter.h* file. If necessary to change default

placement, use the Design-Wide Resources – Directives Editor (in the project's .cydwr file) to edit the parameters '*Component Name', 'Device type' and 'Directive value'* for the required instance of the ADC_SAR component.

The following shows example placement information for an instance of ADC_SAR named ADC_SAR_1 :

- Component Name: \ADC_SAR_1:ADC_SAR\
- Device type: ForceComponentFixed
- Directive value: F(SAR,1)

Resources

The ADC_SAR uses an ADC_SAR primitive and a clock source.

Resolution	Digital Blocks					API Memory (Bytes)		Pins (per External I/O)
	Datapaths	Macro cells	Status Registers	Control Registers	Counter 7	Flash	RAM	
8-12 Bits	0	0	0	0	0	TBD	TBD	-

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 "ADC_SAR_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 "ADC".

Function	Description
void ADC_Start(void)	Power up ADC and reset all states
void ADC_Stop(void)	Stop ADC conversions and power down

Function	Description
void ADC_SetPower(uint8 power)	Set the power mode
void ADC_SetResolution(uint8 resolution)	Set the resolution of the ADC
void ADC_StartConvert(void)	Start conversions
void ADC_StopConvert(void)	Stop Conversions
void ADC_IRQ_Enable(void)	Enables interrupts at the end of conversion
void ADC_IRQ_Disable(void)	Disables interrupts
uint8 ADC_IsEndConversion(uint8 retMode)	Returns a non-zero value if conversion is complete
int8 ADC_GetResult8(void)	Returns an 8-bit conversion result, right justified
int16 ADC_GetResult16(void)	Returns a 16-bit conversion result, right justified
void ADC_SetOffset(int32 offset)	Set offset of ADC
void ADC_SetGain(int32 offset)	Sets ADC gain in counts per volt
int16 ADC_CountsTo_mVolts(int32 adcCounts)	Convert ADC counts to mVolts

void ADC_Start(void)

Description: Configures and powers up the ADC, but does not start conversions.

Parameters: None

Return Value: None

Side Effects: None

void ADC_Stop(void)

Description: Disables and powers down the ADC.

Parameters: None

Return Value: None

Side Effects: None

void ADC_SetPower(uint8 power)

Description: Sets the operational power of the ADC. The higher power settings should be used with faster clock speeds.

Parameters: (uint8) power: Power setting.

Parameters Name	Value	Description
ADC__HIGHPOWER	0	Normal power
ADC__MEDPOWER	1	1/2 power
ADC__LOWPOWER	2	1/3 power
ADC__MINPOWER	3	1/4 power

Return Value: None

Side Effects: Power setting may affect conversion accuracy.

void ADC_SetResolution(uint8 resolution)

Description: Sets the resolution of the ADC.

Parameters: (uint8) resolution: Resolution setting.

Parameters Name	Value	Description
ADC__BITS_12	12	Sets resolution to 12 bits.
ADC__BITS_10	10	Sets resolution to 10 bits.
ADC__BITS_8	8	Sets resolution to 8 bits.

Return Value: None

Side Effects: Affects the resolution. If this function is called when ADC is in conversion state, the new resolution will be valid after end of current conversion.

Affects the ADC_CountsTo_mVolts function by calculating the correct conversion between ADC counts and the applied input voltage. Calculation depends on resolution, input range and voltage reference.

void ADC_StartConvert(void)

Description: Forces the ADC to initiate a conversion. In Free Running mode, the ADC will run continuously. In a triggered mode the function also acts as a software version of the SOC. Here every conversion has to be triggered by the routine. This writes into the SOC bit in SAR_CTRL reg.

Parameters: None

Return Value: None

Side Effects: None

void ADC_StopConvert(void)

Description: Forces the ADC to stop all conversions.

Parameters: None

Return Value: None

Side Effects: None

void ADC_IRQ_Enable(void)

Description: Enables interrupts to occur at the end of a conversion. Global interrupts must also be enabled for the ADC interrupts to occur. To enable global interrupts, use the enable global interrupt macro "`CYGlobalIntEnable;`" in your *main.c*, prior to when interrupts should occur.

Parameters: None

Return Value: None

Side Effects: Enables interrupts to occur. Reading the result will clear the interrupt.

void ADC_IRQ_Disable(void)

Description: Disables interrupts at the end of a conversion.

Parameters: None

Return Value: None

Side Effects: None

uint8 ADC_IsEndConversion(uint8 retMode)

Description: Check for ADC end of conversion. This function provides the programmer with two options. In one mode this function immediately returns with the conversion status. In the other mode, the function does not return (blocking) until the conversion has completed.

Parameters: (uint8) retMode: Check conversion return mode. See table below for options.

Options	Description
ADC_RETURN_STATUS	Immediately returns conversion result status.
ADC_WAIT_FOR_RESULT	Does not return until ADC conversion is complete.

Return Value: (uint8) If a non-zero value is returned, the last conversion has completed. If the returned value is zero, the ADC is still calculating the last result.

Side Effects: None

int8 ADC_GetResult8(void)

Description: This function will return the result of an 8-bit conversion. If the resolution is set greater than 8-bits, the LSB of the result will be returned.

Parameters: None

Return Value: (int8) The LSB of the last ADC conversion.

Side Effects: None

int16 ADC_GetResult16(void)

Description: Returns a 16-bit result for a conversion with a result that has a resolution of 8 to 12 bits.

Parameters: None

Return Value: (int16) The 16-bit result of the last ADC conversion.

Side Effects: None

void ADC_SetOffset(int16 offset)

Description: Sets the ADC offset which is used by the function ADC_CountsTo_mVolts to subtract the offset from the given reading before calculating the voltage conversion.

Parameters: (int16) offset: This value is measured when the inputs are shorted or connected to the same input voltage.

Return Value: None.

Side Effects: Affects ADC_CountsTo_mVolts function by subtracting the given offset.

void ADC_SetGain(int16 adcGain)

Description: Sets the ADC gain in counts per volt for the voltage conversion functions below. This value is set by default by the reference and input range settings. It should only be used to further calibrate the ADC with a known input or if an external reference is used.

Parameters: (int16) adcGain: ADC gain in counts per volt.

Return Value: None.

Side Effects: Affects the ADC_CountsTo_mVolts function by supplying the correct conversion between ADC counts and the applied input voltage.

int16 ADC_CountsTo_mVolts(int16 adcCounts)

Description: Converts the ADC output to mVolts as a 16-bit integer. For example, if the ADC measured 0.534 volts, the return value would be 534 mVolts.

Parameters: (int16) adcCounts: Result from the ADC conversion.

Return Value: (int16) Result in mVolts.

Side Effects: None

Sample Firmware Source Code

The following is a C language example demonstrating the basic functionality of the ADC_SAR component. This example assumes the component has been placed in a design with the default name "ADC_SAR_1."

Note If you rename your component you must also edit the example code as appropriate to match the component name you specify.

```
#include <device.h>

void main()
{
    int16  result;
    ADC_SAR_1_Start();
    ADC_SAR_1_StartConvert();
    ADC_SAR_1_IsEndConversion(ADC_SAR_1_WAIT_FOR_RESULT);
    result = ADC_SAR_1_GetResult16();
    while(1);
}
```

Interrupt Service Routine

The ADC_SAR contains a blank interrupt service routine in the file ADC_SAR_1_INT.c file, where "ADC_SAR_1" is the instance name. You may place custom code in the designated areas to perform whatever function is required at the end of a conversion. A copy of the blank interrupt service routine is shown below. Place custom code between the " /* `#START MAIN_ADC_ISR` */ " and "/* `#END` */ " comments. This ensures that the code will be preserved when a project is regenerated.

```
CY_ISR( ADC_SAR_1_ISR )
{
    /* Place user ADC ISR code here. This can be a good place */
    /* to place code that is used to switch the input to the  */
    /* ADC. It may be good practice to first stop the ADC     */
    /* before switching the input then restart the ADC.       */

    /* `#START MAIN_ADC_ISR` */
    /* Place user code here. */
    /* `#END`  */
}
```

A second designated area is made available to place variable definitions and constant definitions.

```
/* System variables */

/* `#START ADC_SYS_VAR`  */
    /* Place user code here. */
/* `#END`  */
```

Below is example code using an interrupt to capture data. The main is similar to the previous example except that the interrupt must be enabled.

```
#include <device.h>

int16  result = 0;
uint8 dataReady = 0;
void main()
```

```
{
    int16 newReading = 0;
    CYGlobalIntEnable;           /* Enable Global interrupts */
    ADC_SAR_1_Start();           /* Initialize ADC */
    ADC_SAR_1_IRQ_Enable();      /* Enable ADC interrupts */
    ADC_SAR_1_StartConvert();    /* Start ADC conversions */
    for(;;)
    {
        if (dataReady != 0)
        {
           dataReady = 0;
           newReading = result;
           /* More user code */
        }
    }
}
```

Interrupt code segments in the file ADC_SAR_1_INT.c.

```
    /*********************************
    *      System variables
    *********************************/
    /* `#START ADC_SYS_VAR`  */
     extern int16 result;
     extern uint8 dataReady;
     /* `#END`  */

CY_ISR(ADC_SAR_1_ISR )
{
    /*********************************************/
    /* Place user ADC ISR code here.             */
    /* This can be a good place to place code    */
    /* that is used to switch the input to the   */
    /* ADC. It may be good practice to first     */
    /* Stop the ADC before switching the input   */
    /* then restart the ADC.                     */
    /*********************************************/
    /* `#START MAIN_ADC_ISR`  */
        result = ADC_SAR_1_GetResult16();
        dataReady = 1;
    /* `#END`  */
}
```

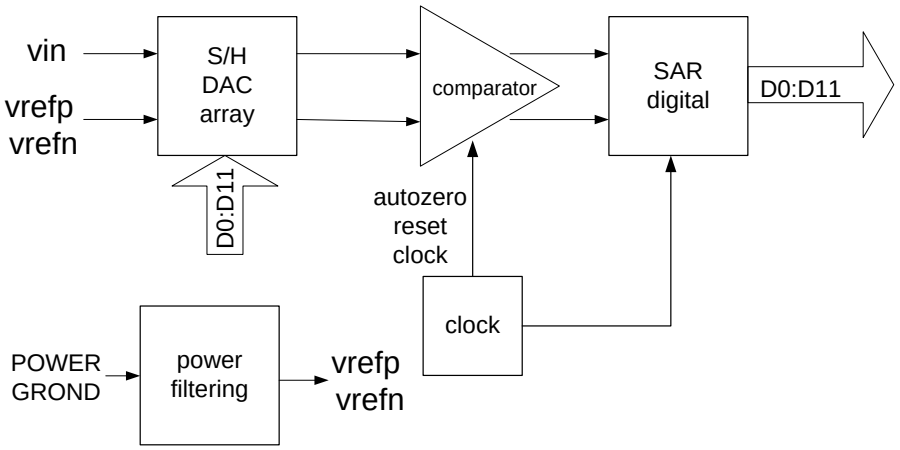
It is important to set up correct parameters: Conversion Rate and Master Clock for safety of ISR handler. For example for the maximum conversion rate (1MSPS) the Master Clock should be selected to 72Mhz in DRW, and ISR routine should be optimized, otherwise the processor will not be able to handle the ISR. If lower Master Clock is selected, the run time of ISR will be longer than SAR ADC conversion time. It is possible to optimize ISR by reading sample registers directly:

```
CY_ISR(ADC_SAR_1_ISR )
{
    /*********************************************/
    /* Place user ADC ISR code here.             */
```

```
        /* This can be a good place to place code     */
        /* that is used to switch the input to the    */
        /* ADC. It may be good practice to first      */
        /* Stop the ADC before switching the input    */
        /* then restart the ADC.                      */
        /********************************************/
        /* `#START MAIN_ADC_ISR`  */
            result = (ADC_SAR_1_SAR_WRK1 << 8) | ADC_SAR_1_SAR_WRK0;
            dataReady = 1;
        /* `#END`  */
    }
```

Functional Description

The block diagram is shown in the following figure. An input analog signal is sampled and compared with the output of a DAC using a binary search algorithm to determine the conversion bits in succession from MSB to LSB.



Registers

Sample Registers

The ADC results may be between 8 and 12 bits of resolution. The output is divided into two 8 bit registers. The CPU or DMA may access these register to read the ADC result.

ADC_SAR_WRK0 (SAR working register 0)

Bits	7	6	5	4	3	2	1	0
Value	Data[7:0]							

ADC_SAR_WRK1 (SAR working register 1)

Bits	7	6	5	4	3	2	1	0
Value	overrun_det NA	NA			Data[11:8]			

- Data[11:0] - the ADC results
- overrun_det - data overrun detection flag, this function is disabled by default.

DC and AC Electrical Characteristics

The following values are indicative of expected performance and based on initial characterization data.

DC Specifications

Parameter	Description	Conditions	Min	Typ	Max	Units
	Resolution		-	-	12	bits
	Number of channels - single ended		-	-	No of GPIO	
	Number of channels - differential	Differential pair is formed using a pair of neighboring GPIO.	-	-	No of GPIO/2	
	Monotonicity		Yes	-	-	

Parameter	Description	Conditions	Min	Typ	Max	Units
	Gain error		-	-	±0.1	%
	Input offset voltage		-	-	±0.2	mV
	Current consumption		-	-	500	µA
	Input voltage range - single ended		Vssa	-	Vdda	V
	Input voltage range - differential		Vssa	-	Vdda	V
	Input resistance		-	-	2	kOhms
Cin	Input capacitance		7	8	9	pF

AC Specifications

Parameter	Description	Conditions	Min	Typ	Max	Units
	Sample & hold droop		-	-	1	µV/µs
PSRR	Power supply rejection ratio		80	-	-	dB
CMRR	Common mode rejection ratio		80	-	-	dB
	Sample rate		-	-	1	Msps
SNR	Signal-to-noise ratio (SNR)		70	-	-	dB
	Input bandwidth		-	500	-	KHz
INL	Integral non linearity	Internal reference	-	-	±1	LSB
DNL	Differential non linearity	Internal reference	-	-	±1	LSB
THD	Total harmonic distortion		-	-	0.005	%

© Cypress Semiconductor Corporation, 2010-2016. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC (“Cypress”). This document, including any software or firmware included or referenced in this document (“Software”), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress’s patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage (“Unintended Uses”). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.

