Stochastic Signal Density Modulation Datasheet SSDM V 1.0

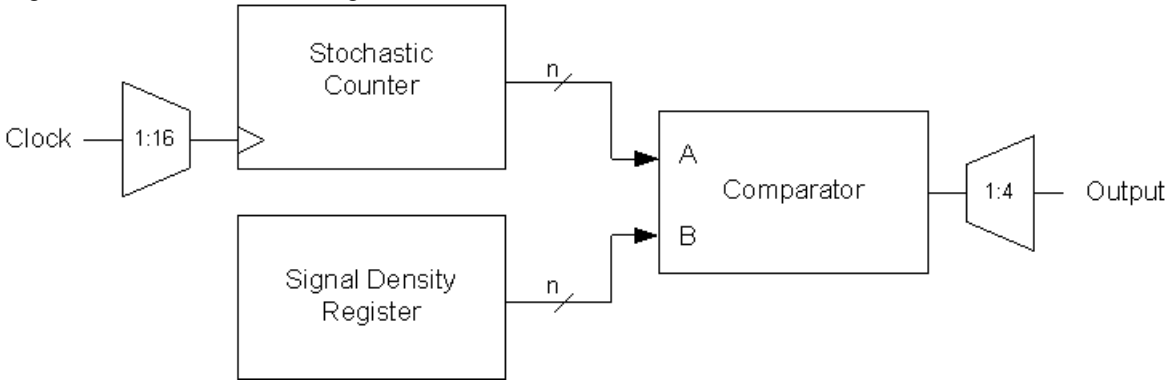
Resources	PSoC® Blocks			API Memory (Bytes)		Pins (per External I/O)
	Digital	Analog CT	Analog SC	Flash	RAM	
CY8CLED02/04/08/16, CY8CLED16P01, CY8CLED0xD, CY8CLED0xG, CY8C29x66, CY8C27x43, CY8C24x94, CY8C21x23, CY8CPLC20						
8-bit	1	0	0	40	0	1
16-bit	2	0	0	76	0	1
24-bit	3	0	0	145	0	1
32-bit	4	0	0	205	0	1

Features and Overview

- LED brightness control
- 2- to 8-, 16-, 24- or 32-bit resolution
- Input clocking up to 48 MHz
- Selectable output signal density
- Interrupt on Compare true

The Stochastic Signal Density Modulation (SSDM) User Module provides the compare output of a SSDM hardware block to a row interconnect. The stochastic counter produces all codes in range 1..2^n-1, but the codes are randomly ordered. The resolution is selected in the device editor, and the resolution setting automatically chooses the correct irreducible simple polynomial. The signal density may be chosen in the device editor or using an API call, the setting corresponds to the Signal Density register in the SSDM hardware block. The compare type can be selected in the device editor.

Figure 1. SSDM Block Diagram



Functional Description

The SSDM User Module employs one to four digital PSoC blocks, each contributing 8 bits to the total resolution. It implements a modular 2- to 8-, 16-, 24-, or 32-bit stochastic counter that generates a pseudo-random bit stream. The Shift, Polynomial, Signal Density, and Control registers of the SSDM PSoC blocks are used to define and control the generation of the output signal.

The configuration of the underlying connective hardware, of the digital PSoC blocks, coordinate the operation of the PSoC blocks as a single SSDM User Module. The Polynomial, Shift, and Signal Density registers refer to the combined registers. The SSDM_MSB registers form the most significant byte of the register set and the SSDM_LSB registers form the least significant byte. For example, the 32-bit SSDM Polynomial register is composed of the SSDM_MSB, SSDM_ISB2, SSDM_ISB1, and SSDM_LSB Polynomial registers.

The SSDM User Module requires that the Signal Density and Polynomial registers be initialized prior to setting the start bit in the SSDM's Control register. Writing the value into the Signal Density register, while the SSDM start bit is not set, causes the signal density value to be latched into the Shift register. Writing the Signal Density value, after the SSDM has been started, only changes the compare value.

The SSDM stochastic counter is an LFSR that is implemented using the modular form where there is one XOR gate between each bit of the shift register.

The 32-bit polynomial [32,31,30,10] is specified as 1110 0000 0000 0000 0000 0010 0000 0000b or E0000200h.

The following table lists modular irreducible polynomials that define 2- to 32-bit pseudo-random number sequences (including 8, 16, and 24 bits). Note that the table is not complete, in that only one of the many polynomials is specified. This table is for reference purposes. The polynomial is set automatically by the SSDM User Module.

Table 1. Modular Irreducible Polynomials

Number of Bits	Code Length	Modular Form Polynomial	Polynomial value
2	3	[2,1]	0x03
3	7	[3,2]	0x06
4	15	[4,3]	0x0C
5	31	[5,4,3,2]	0x1E
6	63	[6,5,4,1]	0x39
7	127	[7,6,5,2]	0x72
8	255	[8,6,5,4]	0xB8
9	511	[9,6,5,3]	0x134
10	1,023	[10,8,7,2]	0x2C2
11	2,047	[11,9,6,3]	0x524
12	4,095	[12,11,8,6]	0xCA0
13	8,191	[13,12,10,9]	0x1B00
14	16,383	[14,13,12,2]	0x3802
15	32,767	[15,13,10,8]	0x5280

Number of Bits	Code Length	Modular Form Polynomial	Polynomial value
16	65,535	[16,15,13,4]	0xD008
17	131,071	[17,16,15,14]	0x1E000
18	262,143	[18,13,11,8]	0x21480
19	524,287	[19,18,17,14]	0x72000
20	1,048,575	[20,17,15,14]	0x96000
21	2,097,151	[21,19,14,7]	0x142040
22	4,194,303	[22,21,17,13]	0x311000
23	8,388,607	[23,18,13,5]	0x421010
24	16,777,215	[24,23,22,17]	0xE10000
25	33,554,431	[25,24,23,22]	0x1E00000
26	67,108,863	[26,25,24,20]	0x3880000
27	134,217,727	[27,26,25,22]	0x7200000
28	268,435,455	[28,25,23,19,17,15]	0x9454000
29	536,870,911	[29,27,18,9]	0x14020100
30	1,073,741,823	[30,29,28,7]	0x60200008
31	2,147,483,647	[31,30,22,4]	0x60200008
32	4,294,967,295	[32,31,23,4]	0xE0000200

The maximal code length, for an N-bit stochastic counter, is 2^n-1. Zero is the missing value.

The polynomial length tap bit, N, is configured by the SSDM PSoC block, as both the feedback and output bit.

When the signal density value and polynomial are initialized, the SSDM User Module is started and the rising edge of the input clock generates the stochastic counter next state in the specified pseudo-random sequence. The output value of the comparator is sent to the specified output bit stream synchronous with the clock.

DC and AC Electrical Characteristics

Table 2. SSDM AC Electrical Characteristics

Parameter	Typical	Limit	Units	Conditions and Notes
Maximum input frequency	--	48[1]	MHz	Vdd=5.0V
	--	24[2]	MHz	Vdd=3.3V
Maximum output frequency	--	24[1]	MHz	Vdd=5.0V and 48 MHz input clock
	--	12[2]	MHz	Vdd=3.3V and 24 MHz input clock

Electrical Characteristics Notes

1. If the input or output is routed through the global buses, then the frequency is limited to a maximum of 12 MHz.
2. Fastest clock available to PSoC blocks is 24 MHz at 3.3V operation.

Placement

The SSDM consumes one digital PSoC block per 8 bits of resolution. When more than one block is allocated, all blocks will be placed consecutively by the Device Editor in order of increasing block number from the least-significant byte (LSB) to the most significant (MSB). Each block is given a symbolic name displayed by the device editor during and after placement. The API qualifies all register names with user assigned instance name and block name to provide direct access to the SSDM registers through the API include files. The block names used by the various widths are given in the following table.

PSoC Blocks	2- to 8-Bit Dimming Resolution	9- to 16-Bit Dimming Resolution	17- to 24-Bit Dimming Resolution	25- to 32-Bit Dimming Resolution
1	SSDM8	SSDM16_LSB	SSDM24_LSB	SSDM32_LSB
2	--	SSDM16_MSB	SSDM24_ISB	SSDM32_ISB1
3	--	--	SSDM24_MSB	SSDM32_ISB2
4	--	--	--	SSDM32_MSB

Parameters and Resources

Clock

The SSDM User Module is clocked by one of 16 possible sources. The Global I/O busses may be used to connect the clock input to an external pin or a clock function generated by a different PSoC block. When using an external digital clock for the block, the row input synchronization should be turned off for best accuracy, and sleep operation. The 48 MHz clock, the CPU_32 kHz clock, one of the divided clocks (24V1 or 24V2), or another PSoC block output can be specified as the clock input. Note only one 8-bit SSDM User module can use the previous PSoC block as a clock source.

SSDMOut

SSDMOut may be routed to one of four Global Output buses or disabled. The output is the auxiliary output of the MSB block. This output cannot be connected to the broadcast bus.

CompareType

Each cycle the SSDM compares the value of the Shift register to the value of the Signal Density register. This parameter sets the type of compare function to be performed. The possible settings are given in the following table.

Parameter	Description
Less Than or Equal	SSDM output goes high when Shift register is less than or equal to Signal Density value. (Recommended.)
Less Than	SSDM output goes high when Shift register is less than Signal Density value.

DimmingResolution

The DimmingResolution parameter contains a value from 2-8 (for 8-bit SSDM), 2-16 (for 16-bit SSDM), 2-24 (for 24-bit SSDM) and 2-32 (for 16-bit SSDM). This parameter defines the stochastic counter period as 2^n-1 where n is DimmingResolution.

SignalDensity

The SignalDensity parameter can contain values from 0 to 2^n-1 where n is value selected in the Dimming Resolution parameter. The interface does not enforce the 2^n-1 limit on the SignalDensity parameter. If you set the DimmingResolution to 7 and set the SignalDensity to 255, the extra bit is discarded and the SignalDensity is 127. The output average duty ratio will be SignalDensity/DimmingResolution. For a CompareType of Less Than or Equal, use the following equation:

Equation 1

$$\frac{SignalDensity + 1}{DimmingResolution + 1}$$

For a CompareType of Less Than, us the following equation:

Equation 2

$$\frac{SignalDensity}{DimmingResolution + 1}$$

ClockSync

In the PSoC devices, digital blocks may provide clock sources in addition to the system clocks. Digital clock sources may even be chained in ripple fashion. This introduces skew with respect to the system clocks. This parameter may be used to control clock skew and ensure proper operation when reading and writing PSoC block register values. Appropriate values for this parameter should be determined from the following table.

ClockSync Value	Use
Sync to SysClk	Use this setting for any 24 MHz (SysClk) derived clock source that is divided by two or more. Examples include VC1, VC2, VC3 (when VC3 is driven by SysClk), 32KHz, and digital PSoC blocks with SysClk-based sources. Externally generated clock sources should also use this value to ensure that proper synchronization occurs.
Sync to SysClk*2	Use this setting for any 48 MHz (SysClk*2) based clock unless the resulting frequency is 48 MHz (in other words, when the product of all divisors is 1).
Use SysClk Direct	Use when a 24 MHz (SysClk/1) clock is desired. This does not actually perform synchronization but provides low-skew access to the system clock itself. If selected, this option overrides the setting of the Clock parameter, above. It should always be used instead of VC1, VC2, VC3 or Digital Blocks where the net result of all dividers in combination produces a 24 Mhz output.
Unsynchronized	Use when the 48 MHz (SysClk*2) input is selected. Use when unsynchronized inputs are desired. In general this use is advisable only when interrupt generation is the sole application of the Counter.

Application Programming Interface

The Application Programming Interface (API) routines are provided as part of the user module to allow the designer to deal with the module at a higher level. This section specifies the interface to each function together with related constants provided by the “include” files.

Note

In this, as in all user module APIs, the values of the A and X register may be altered by calling an API function. It is the responsibility of the calling function to preserve the values of A and X prior to the call if those values are required after the call. This “registers are volatile” policy was selected for efficiency reasons and has been in force since version 1.0 of PSoC Designer. The C compiler automatically takes care of this requirement. Assembly language programmers must ensure their code observes the policy, too. Though some user module API function may leave A and X unchanged, there is no guarantee they will do so in the future.

For Large Memory Model devices, it is also the caller's responsibility to preserve any value in the CUR_PP, IDX_PP, MVR_PP, and MVW_PP registers. Even though some of these registers may not be modified now, there is no guarantee that will remain the case in future releases.

There are five functions in this API, and some functions will vary depending on the number of bits (8, 16, 24, or 32):

- SSDM_Start
- SSDM_Stop
- SSDM_WriteSignalDensity
- SSDM_WriteResolution

- SSDM_EnableInt
- SSDM_DisableInt

SSDM_Start

Description:

Enables the SSDM User Module for operation. This function is the same no matter how many bits are used.

C Prototype:

```
void  SSDM_Start(void)
```

Assembler:

```
lcall  SSDM_Start
```

Parameters:

None.

Return Value:

None.

Side Effects:

While the SSDM is started, a signal density value written to the Signal Density register will not be latched into the Shift register. The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8CLED16). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

SSDM_Stop

Description:

Disables the SSDM User Module. This function is the same no matter how many bits are used.

C Prototype:

```
void  SSDM_Stop(void)
```

Assembler:

```
lcall  SSDM_Stop
```

Parameters:

None

Return Value:

None

Side Effects:

Writing the value into the Signal Density register will latch the signal density value into the Shift register. The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8CLED16). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions. SSDM_WriteSignalDensity

SSDM_WriteSignalDensity

Description:

Initializes the SSDM Signal Density register with an initial value. This function varies depending on how many bits are used.

C Prototype:

- 8-Bit Prototype: **void SSDM_WriteSignalDensity(BYTE bSignalDensity)**
- 16-Bit Prototype: **void SSDM_WriteSignalDensity(WORD wSignalDensity)**
- 24-Bit Prototype: **void SSDM_WriteSignalDensity(DWORD dwSignalDensity)**
- 32-Bit Prototype: **void SSDM_WriteSignalDensity(DWORD dwSignalDensity)**

Assembler:

- 8-Bit Assembler:

```
mov   A, [bSignalDensity]
lcall  SSDM_WriteSignalDensity
```

- 16-Bit Assembler:

```
mov   X, >wSignalDensity
mov   A, <wSignalDensity
lcall  SSDM_WriteSignalDensity
```

- 24-Bit Assembler:

```
mov   X, SP
add   SP, 4
mov   [X], 0
mov   [X+1], (dwSignalDensity >> 16) & ffh
mov   [X+2], (dwSignalDensity >> 8 ) & ffh
mov   [X+3], (dwSignalDensity & ffh)
lcall  SSDM_WriteSignalDensity
add   SP, -4
```

- 32-Bit Assembler:

```
mov   X, SP
add   SP, 4
mov   [X],   (dwSignalDensity >> 24) & ffh
mov   [X+1], (dwSignalDensity >> 16) & ffh
mov   [X+2], (dwSignalDensity >> 8 ) & ffh
mov   [X+3], (dwSignalDensity & ffh)
lcall  SSDM_WriteSignalDensity
add   SP, -4
```

Parameters:

- 8-Bit Prototype: bSignalDensity: 8-bit Signal Density value and is passed in the A register.
- 16-Bit Prototype: wSignalDensity: 16-bit signal density value. MSB is passed in the X register and LSB is passed in the Accumulator.
- 24-Bit Prototype: dwSignalDensity: 24-bit value where the MSB is always zero. The parameter is passed through stack.
- 32-Bit Prototype: dwSignalDensity: 32-bit signal density value. The parameter is passed through stack.

Return Value:

None

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8CLED16). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

SSDM_WriteResolution

Description:

Initializes the stochastic counter Polynomial Register with a polynomial that matches the desired resolution.

C Prototype:

```
void  SSDM_WriteResolution(BYTE bResolution)
```

Assembler:

```
mov   A, [bResolution]
lcall  SSDM_WriteResolution
```

Parameters:

bSignalResolution: value from 2 to 8/16/24/32 depending on which version of the user module you use. The resolution value is passed in the A register.

Return Value:

None

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8CLED16). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

SSDM_EnableInt

Description:

Enables the block interrupt for the SSDM block, the only valid interrupt is the compare out interrupt. This function is the same no matter how many bits are used.

C Prototype:

```
void  SSDM_EnableInt(void)
```

Assembler:

```
lcall  SSDM_EnabeInt
```

Parameters:

None.

Return Value:

None

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8CLED16). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

SSDM_DisableInt

Description:

Enables the block interrupt for the SSDM block, the only valid interrupt is compare out interrupt.

C Prototype:

```
void  SSDM_DisableInt(void)
```

Assembler:

```
lcall  SSDM_DisableInt
```

Parameters:

None.

Return Value:

None

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8CLED16). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

Sample Firmware Source Code

Sample code is provided for all four possible bit widths of the SSDM User Module.

8-Bit SSDM Sample Firmware Source Code

In the following examples, the correspondence between the C and assembly code is simple and direct.

The following is assembly language source that illustrates the use of the APIs.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Description:
;   This demonstrate how to use SSDM and change Signal Density.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
include "m8c.inc"       ; part specific constants and macros
include "memory.inc"    ; Constants & macros for SMM/LMM and Compiler
include "PSoCAPI.inc"   ; PSoC API definitions for all user modules

export _main

_main:

   M8C_EnableGInt               ; enable Global interrupts
   lcall SSDM_EnableInt         ; enable SSDM interrupts
   lcall SSDM_Start             ; start SSDM user module
   mov   A, 50h                 ; copy Signal Density value to A
   lcall SSDM_WriteSignalDensity ; set Signal Density value
```

```
   ; Insert your main assembly code here.

.terminate:
   jmp   .terminate
```

The same code in C is:

```
#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules


void main(void)
{
    M8C_EnableGInt ;                            // enable Global interrupts
    SSDM_EnableInt();                           // enable SSDM interrupts
    SSDM_Start();                               // start SSDM User Module
    SSDM_WriteSignalDensity(0x50);  // change the Signal Density value

    // Insert your main routine code here.
}
```

16-Bit SSDM Sample Firmware Source Code

In the following examples, the correspondence between the C and assembly code is simple and direct.

The following is assembly language source that illustrates the use of the APIs.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Description:
;   This demonstrate how to use SSDM and change Signal Density.
;*****************************************************************
include "m8c.inc"       ; part specific constants and macros
include "memory.inc"    ; Constants & macros for SMM/LMM and Compiler
include "PSoCAPI.inc"   ; PSoC API definitions for all user modules

export _main

_main:

   M8C_EnableGInt               ; enable Global interrupts
   lcall SSDM_EnableInt         ; enable SSDM interrupts
   lcall SSDM_Start             ; start SSDM user module
   mov   A, 10h      ; copy LSB value of Signal Density to A
   mov   X, 20h      ; copy MSB value of Signal Density to X
   call  SSDM_WriteSignalDensity ; set Signal Density value

   ; Insert your main assembly code here.

.terminate:
   jmp   .terminate
```

The same code in C is:

```
#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all user modules


void main(void)
{
    M8C_EnableGInt ;                            // enable Global interrupts
    SSDM_EnableInt();                           // enable SSDM interrupts
    SSDM_Start();                               // start SSDM User Module
    SSDM_WriteSignalDensity(0x1020);    // change the Signal Density value

    // Insert your main routine code here.
}
```

24-Bit SSDM Sample Firmware Source Code

In the following examples, the correspondence between the C and assembly code is simple and direct.

The following is assembly language source that illustrates the use of the APIs.

In the following examples, the correspondence between the C and assembly code is simple and direct.

The following is assembly language source that illustrates the use of the APIs.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Description:
;   This demonstrate how to use SSDM and change Signal Density.
;*****************************************************************
include "m8c.inc"       ; part specific constants and macros
include "memory.inc"    ; Constants & macros for SMM/LMM and Compiler
include "PSoCAPI.inc"   ; PSoC API definitions for all user modules

export _main

_main:

   M8C_EnableGInt               ; enable Global interrupts
   lcall SSDM_EnableInt         ; enable SSDM interrupts
   lcall SSDM_Start             ; start SSDM User Module

   mov   A, 10h    ; load MSB Signal Density value to A
   push  A
   mov   A, 20h    ; load ISB Signal Density value to A
   push  A
   mov   A, 30h    ; load LSB Signal Density value to A
   push  A
   call  SSDM_WriteSignalDensity ; set Signal Density value
   add   SP, -3h                 ; restore SP

   ; Insert your main assembly code here.

.terminate:
   jmp   .terminate
```

The same code in C is:

```
#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all user modules


void main(void)
{
    M8C_EnableGInt ;                            // enable Global interrupts
    SSDM_EnableInt();                           // enable SSDM interrupts
    SSDM_Start();                               // start SSDM User Module
    SSDM_WriteSignalDensity(0x102030);  // change the Signal Density value

    // Insert your main routine code here.
}
```

32-Bit SSDM Sample Firmware Source Code

In the following examples, the correspondence between the C and assembly code is simple and direct.

The following is assembly language source that illustrates the use of the APIs.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Description:
;   This demonstrate how to use SSDM and change Signal Density.
;*****************************************************************************
include "m8c.inc"       ; part specific constants and macros
include "memory.inc"    ; Constants & macros for SMM/LMM and Compiler
include "PSoCAPI.inc"   ; PSoC API definitions for all user modules

export _main

_main:
   dwSignalDensity:    blk 4     ; Signal Density variable

   M8C_EnableGInt                ; enable Global interrupts
   lcall SSDM_EnableInt          ; enable SSDM interrupts
   lcall SSDM_Start              ; start SSDM User Module

   mov   A, 10h  ; load MSB Signal Density value to A
   push  A
   mov   A, 20h  ; load ISB Signal Density value to A
   push  A
   mov   A, 30h  ; load ISB Signal Density value to A
   push  A
   mov   A, 40h  ; load LSB Signal Density value to A
   push  A
   call  SSDM_WriteSignalDensity ; set Signal Density value
   add   SP, -4h                 ; restore SP

   ; Insert your main assembly code here.

.terminate:
   jmp   .terminate
```

The same code in C is:

```
#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all user modules


void main(void)
{
    M8C_EnableGInt ;                            // enable Global interrupts
    SSDM_EnableInt();                           // enable SSDM interrupts
    SSDM_Start();                               // start SSDM User Module
    SSDM_WriteSignalDensity(0x10203040);    // change the Signal Density value

    // Insert your main routine code here.
}
```

Configuration Registers

8-Bit SSDM Configuration Registers

The 8-bit SSDM uses a single digital PSoC block named SSDM. Each block is personalized and parameterized through 7 registers. The following tables give the “personality” values as constants and the parameters as named bit-fields with brief descriptions. Symbolic names for these registers are defined in the user module instance’s C and assembly language interface files (the *.h* and *.inc* files).

Table 3. Function Register, Bank 1

Bit	7	6	5	4	3	2	1	0
Value	0	BCEN	1	Compare Type		0	1	0

BCEN gates the compare output onto the row broadcast bus line. This bit field is set in the Device Editor by directly configuring the broadcast line. Compare Type indicates whether the compare function is set to “Equal”, “Less Than or Equal”, or “Less Than.” The Compare Type parameter is set in the Device Editor.

Table 4. Input Register, Bank 1

Bit	7	6	5	4	3	2	1	0
Value	0	0	0	0	Clock			

Clock selects the input clock from one of 16 sources. This parameter is set in the Device Editor.

Table 5. Output Register, Bank 1

Bit	7	6	5	4	3	2	1	0
Value	ClockSync		SSDMOut			0	0	0

SSDMOut selects output from one of four global busses. This parameter is set in the Device Editor.

Table 6. Shift Register (DR0), Bank 0

Bit	7	6	5	4	3	2	1	0
Value	Shift Register							

Shift Register is the SSDM stochastic counter Shift register.

Table 7. Resolution Register (DR1), Bank 0

Bit	7	6	5	4	3	2	1	0
Value	Polynomial Register							

Polynomial Register is the SSDM stochastic counter Polynomial register.

Table 8. Signal Density (DR2), Bank 0

Bit	7	6	5	4	3	2	1	0
Value	Signal Density							

Signal Density Register is set in the Device Editor and also can be modified using the SSDM API.

Table 9. Control Register (CR0), Bank 0

Bit	7	6	5	4	3	2	1	0
Value	0	0	0	0	0	0	0	Start/Stop

Start/Stop indicates that the SSDM is enabled when set. It is modified using the SSDM API.

16-Bit SSDM Configuration Registers

The 16-bit SSDM uses two digital PSoC blocks. In placement order from left to right they are named SSDM_LSB and SSDM_MSB. Each block is personalized and parameterized through 7 registers. The following tables give the “personality” values as constants and the parameters as named bit-fields with brief descriptions. Symbolic names for these registers are defined in the user module instance’s C and assembly language interface files (the *.h* and *.inc* files).

Table 10. Function Register, Bank 1

Block/Bit	7	6	5	4	3	2	1	0
MSB	0	BCEN	1	Compare Type		0	1	0
LSB	0	0	0	Compare Type		0	1	0

BCEN gates the compare output onto the row broadcast bus line. This bit field is set in the Device Editor by directly configuring the broadcast line. Compare Type indicates whether the compare function is set to “Equal”, "Less Than or Equal", or "Less Than." The Compare Type parameter is set in the Device Editor.

Table 11. Input Register, Bank 1

Block/Bit	7	6	5	4	3	2	1	0
MSB	0	0	1	1	Clock			
LSB	0	0	0	0	Clock			

Clock selects the input clock from one of 16 sources. This parameter is set in the Device Editor.

Table 12. Output Register, Bank 1

Block/Bit	7	6	5	4	3	2	1	0
MSB	ClockSync		SSDMOut			0	0	0
LSB	ClockSync		0	0	0	0	0	0

SSDMOut selects output from one of four global busses. This parameter is set in the Device Editor.

Table 13. Shift Register (DR0), Bank 0

Block/Bit	7	6	5	4	3	2	1	0
MSB	Shift Register(MSB)							
LSB	Shift Register(LSB)							

Shift Register is the SSDM stochastic counter Shift register MSB and LSB.

Table 14. Resolution Register (DR1), Bank 0

Block/Bit	7	6	5	4	3	2	1	0
MSB	Polynomial Register(MSB)							
LSB	Polynomial Register(LSB)							

Polynomial Register is the SSDM stochastic counter Polynomial register MSB and LSB.

Table 15. Signal Density Register (DR2), Bank 0

Block/Bit	7	6	5	4	3	2	1	0
MSB	Signal Density Register(MSB)							
LSB	Signal Density Register(LSB)							

Signal Density Register is set in the Device Editor and also can be modified using the SSDM API.

Table 16. Control Register (CR0), Bank 0

Block/Bit	7	6	5	4	3	2	1	0
MSB	0	0	0	0	0	0	0	0
LSB	0	0	0	0	0	0	0	Start/Stop

Start/Stop indicates that the SSDM is enabled when set. It is modified using the SSDM API.

24-Bit SSDM Configuration Registers

The 24-bit SSDM uses three digital PSoC blocks. In placement order from left to right they are named SSDM_LSB, SSDM_ISB, and SSDM_MSB. Each block is personalized and parameterized through 7 registers. The following tables give the “personality” values as constants and the parameters as named bit-fields with brief descriptions. Symbolic names for these registers are defined in the user module instance’s C and assembly language interface files (the *.h* and *.inc* files).

Table 17. Function Register, Bank 1

Block/Bit	7	6	5	4	3	2	1	0
MSB	0	BCEN	1	Compare Type		0	1	0
ISB	0	0	0	Compare Type		0	1	0
LSB	0	0	0	Compare Type		0	1	0

BCEN gates the compare output onto the row broadcast bus line. This bit field is set in the Device Editor by directly configuring the broadcast line. Compare Type indicates whether the compare function is set to “Equal”, "Less Than or Equal", or "Less Than." The Compare Type parameter is set in the Device Editor.

Table 18. Input Register, Bank 1

Block/Bit	7	6	5	4	3	2	1	0
MSB	0	0	1	1	Clock			
ISB	0	0	1	1	Clock			
LSB	0	0	0	0	Clock			

Clock selects the input clock from one of 16 sources. This parameter is set in the Device Editor.

Table 19. Output Register, Bank 1

Block/Bit	7	6	5	4	3	2	1	0
MSB	ClockSync		SSDMOut			0	0	0
ISB	0	0	0	0	0	0	0	0
LSB	0	0	0	0	0	0	0	0

SSDMOut selects output from one of four global busses. This parameter is set in the Device Editor.

Table 20. Shift Register (DR0), Bank 0

Block/Bit	7	6	5	4	3	2	1	0
MSB	Shift Register(MSB)							
ISB	Shift Register(ISB)							
LSB	Shift Register(LSB)							

Shift Register is the SSDM stochastic counter Shift register MSB, ISB and LSB. It is read and configured using the SSDM API.

Table 21. Resolution Register (DR1), Bank 0

Block/Bit	7	6	5	4	3	2	1	0
MSB	Polynomial Register(MSB)							
ISB	Polynomial Register(ISB)							
LSB	Polynomial Register(LSB)							

Polynomial Register is the SSDM stochastic counter Polynomial register MSB, ISB and LSB. It is modified using the SSDM API.

Table 22. Signal Density Register (DR2), Bank 0

Block/Bit	7	6	5	4	3	2	1	0
MSB	Signal Density Register(MSB)							
ISB	Signal Density Register(ISB)							
LSB	Signal Density Register(LSB)							

Signal Density Register is set in the Device Editor and also can be modified using the SSDM API.

Table 23. Control Register (CR0), Bank 0

Block/Bit	7	6	5	4	3	2	1	0
MSB	0	0	0	0	0	0	0	0
ISB	0	0	0	0	0	0	0	0
LSB	0	0	0	0	0	0	0	Start/Stop

Start/Stop indicates that the SSDM is enabled when set. It is modified using the SSDM API.

32-Bit SSDM Configuration Registers

The 32-bit SSDM uses four digital PSoC blocks. In placement order from left to right they are named SSDM_LSB, SSDM_ISB1, SSDM_ISB2 and SSDM_MSB. Each block is personalized and parameterized through 7 registers. The following tables give the “personality” values as constants and the parameters as named bit-fields with brief descriptions. Symbolic names for these registers are defined in the user module instance’s C and assembly language interface files (the *.h* and *.inc* files).

Table 24. Function Register, Bank 1

Block/Bit	7	6	5	4	3	2	1	0
MSB	0	BCEN	1	Compare Type		0	1	0
ISB2	0	0	0	Compare Type		0	1	0
ISB1	0	0	0	Compare Type		0	1	0
LSB	0	0	0	Compare Type		0	1	0

BCEN gates the compare output onto the row broadcast bus line. This bit field is set in the Device Editor by directly configuring the broadcast line. Compare Type indicates whether the compare function is set to “Equal”, "Less Than or Equal", or "Less Than." The Compare Type parameter is set in the Device Editor.

Table 25. Input Register, Bank 1

Block/Bit	7	6	5	4	3	2	1	0
MSB	0	0	1	1	Clock			
ISB2	0	0	1	1	Clock			
ISB1	0	0	1	1	Clock			
LSB	0	0	0	0	Clock			

Clock selects the input clock from one of 16 sources. This parameter is set in the Device Editor.

Table 26. Output Register, Bank 1

Block/Bit	7	6	5	4	3	2	1	0
MSB	ClockSync		SSDMOut			0	0	0
ISB2	ClockSync		0	0	0	0	0	0
ISB1	ClockSync		0	0	0	0	0	0
LSB	ClockSync		0	0	0	0	0	0

SSDMOut selects output from one of four global busses. This parameter is set in the Device Editor.

Table 27. Shift Register (DR0), Bank 0

Block/Bit	7	6	5	4	3	2	1	0
MSB	Shift Register(MSB)							
ISB2	Shift Register(ISB2)							
ISB1	Shift Register(ISB2)							
LSB	Shift Register(LSB)							

Shift Register is the SSDM stochastic counter Shift register MSB, ISB2, ISB1, and LSB.

Table 28. Resolution Register (DR1), Bank 0

Block/Bit	7	6	5	4	3	2	1	0
MSB	Polynomial Register(MSB)							
ISB2	Polynomial Register(ISB2)							
ISB1	Polynomial Register(ISB1)							
LSB	Polynomial Register(LSB)							

Polynomial Register is the SSDM stochastic counter Polynomial register MSB, ISB2, ISB1, and LSB.

Table 29. Signal Density Register (DR2), Bank 0

Block/Bit	7	6	5	4	3	2	1	0
MSB	Signal Density Register(MSB)							
ISB2	Signal Density Register(ISB2)							
ISB1	Signal Density Register(ISB1)							
LSB	Signal Density Register(LSB)							

Signal Density Register is set in the Device Editor and also can be modified using the SSDM API.

Table 30. Control Register (CR0), Bank 0

Block/Bit	7	6	5	4	3	2	1	0
MSB	0	0	0	0	0	0	0	0
ISB2	0	0	0	0	0	0	0	0
ISB1	0	0	0	0	0	0	0	0
LSB	0	0	0	0	0	0	0	Start/Stop

Start/Stop indicates that the SSDM is enabled when set. It is modified using the SSDM API.

Version History

Version	Originator	Description
1.0	DHA	Initial version

Note PSoC Designer 5.1 introduces a Version History in all user module datasheets. This section documents high level descriptions of the differences between the current and previous user module versions.