Incremental ADC Datasheet ADCINC V 1.20

Resources	PSoC® Blocks				API Memory (Bytes)			Pins (per External I/O)
	Digital	Analog CT	Analog SC		Flash		RAM	
Modulators (1st or 2nd Order)		1st and 2nd	1st	2nd	1st	2nd		
CY8C29xxx, CY8C24x94, CY8C23x33, CY7C64215, CY8CLED04/08/16, CY8CLED0xD, CY8CLED0xG, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C28x45, CY8C28x43, CY8C28x52, CY8CPLC20, CY8CLED16P01								
	1	0	1	2	273	322	8	1
CY8C27/24/22xxx, CY8CLED08	1	0	1	2	226	275	8	1

See AN2239, ADC Selection Guide for other converters.

For one or more fully configured, functional example projects that use this user module go to www.cypress.com/psocexampleprojects.

Features and Overview

- 6- to 14-bit resolution
- Optional synchronous 8-bit PWM output
- Optional differential Input
- Signed or unsigned data format
- Sample rate up to 15.6 ksps (6-bit resolution)
- Input range defined by internal and external reference options
- Internal or external clock

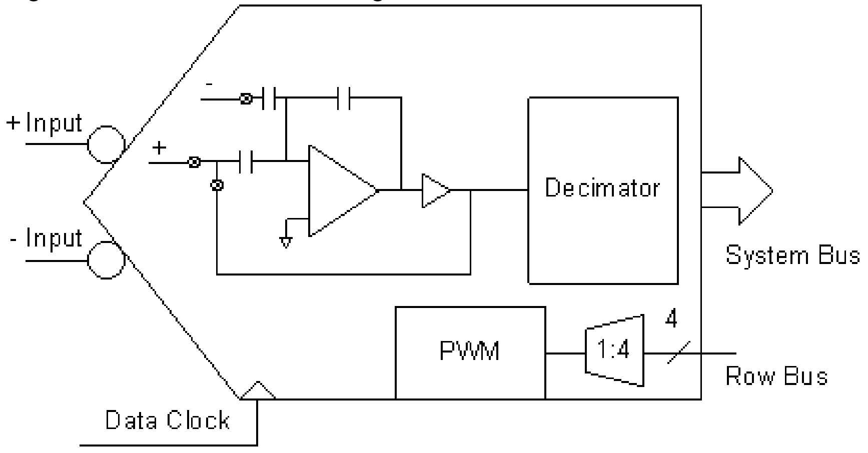
Note If this user module is used with the 29K family, it consumes an additional 6 mA. As an alternative, use the ADCINCVR User Module.

The ADCINC is a differential or single input ADC that returns a 6- to 14-bit result. The maximum DataClock frequency is 8 MHz, but 2 MHz is the maximum frequency recommended for improved linearity. This ADC may only be placed once due to its implementation, which uses the hardware decimator rather than a digital block. This is the most resource-efficient ADC. A second order modulator may be implemented with an additional switch-capacitor block, allowing better linearity with an 8-MHz DataClock.

Timing is implemented with an 8-bit PWM that gives you a modulated pulse width that is synchronous to the input sample.

The ADCINC requires 2^n–1 integration cycles to generate an output with n bits of resolution.

Figure 1. ADCINC Block Diagram



Quick Start

You can download a preconfigured example project from www.cypress.com/psocexampleprojects. To create an ADCINC project:

1. In PSoC Designer, select **New Project.**
2. In the New Project dialog, choose **Designer Only Project**. Choose a name and location for the project and click **OK**.
3. In the **Select New Base Part** dialog, select one of the devices that supports this user module. See the Resources table at the beginning of this datasheet.
4. In the **User Module Catalog**, click to expand the selection of **ADCs**, then expand the **ADCINC** folder. If the User Module Catalog is not visible, select **View > User Module Catalog**.
5. The ADCINC is available with a single stage modulator or a dual stage modulator. If you do not know which version to use, you can review the datasheet by right clicking on one of the modulators and selecting **Datasheet** to review the datasheet before deciding.
6. Right click on the user module and select **Place**.

Functional Description

The ADCINC gives a first order modulator formed from a single analog switched capacitor PSoC block, one digital PSoC block, and the decimator, as shown in the following figures:

Figure 2. Schematic of the ADCINC with First Order Modulator

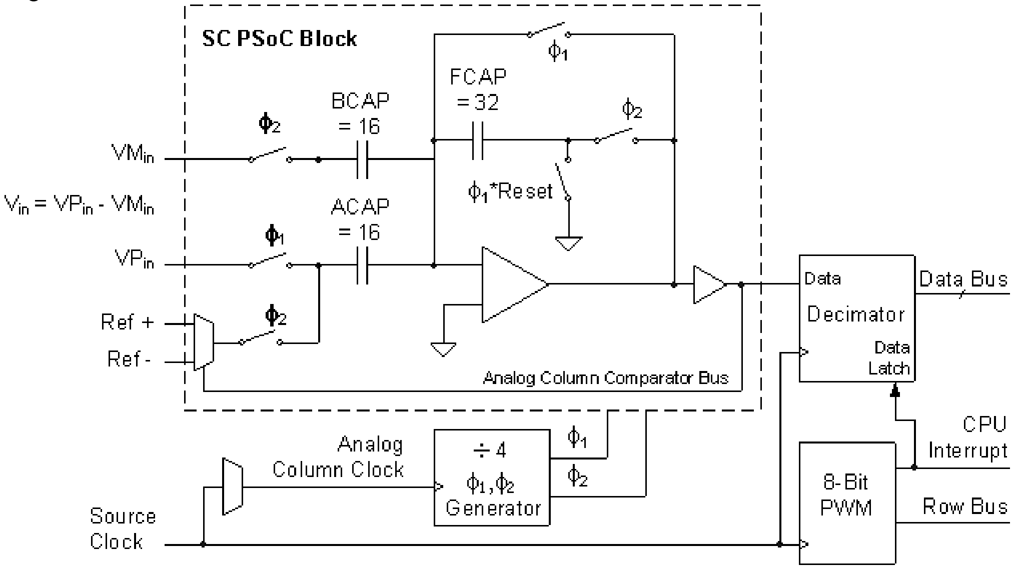
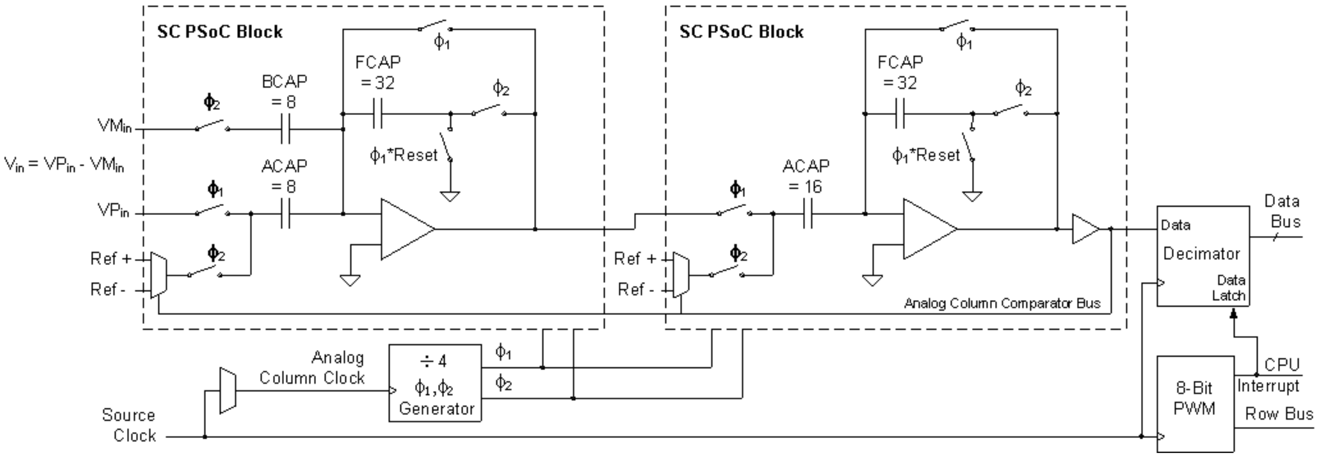


Figure 3. Schematic of the ADCINC with Second Order Modulator



The range of the ADCINC is set at $\pm V_{Ref}$. You can set V_{Ref} in the Global Resources window of PSoC Designer. For fixed scale, V_{Ref} is set to $V_{Bandgap}$, or 1.6 $V_{Bandgap}$. For adjustable scale, V_{Ref} is set to Port 2[6]. For supply ratio metric scale, V_{Ref} is set to $V_{DD}/2$.

The analog block is configured as an integrator that can be reset. Depending on the output polarity, the reference control is configured so that the reference voltage is either added or subtracted from the input and placed in the integrator. This reference control attempts to pull the integrator output back towards AGND. If the integrator is operated 2^{Bits} times and the output voltage comparator is positive "n" of those times, the residual voltage (V_{resid}) at the output is:

Equation 1

$$V_{resid} = 2^{Bits} \cdot V_{in} - n \cdot V_{ref} + (2^{Bits} - n) \cdot V_{ref}$$

Equation 2

$$V_{in} = \frac{n - 2^{Bits - 1}}{2^{Bits - 1}} V_{ref} + \frac{V_{resid}}{2^{Bits}}$$

This equation states that the range of this ADC is $\pm V_{Ref}$, the resolution (LSB) is $V_{Ref}/2^{Bits-1}$, and the voltage on the output at the end of the computation is defined as the residue. Since V_{resid} is always less than V_{Ref}, $V_{resid}/2^{Bits}$ less than half a LSB and can be ignored.

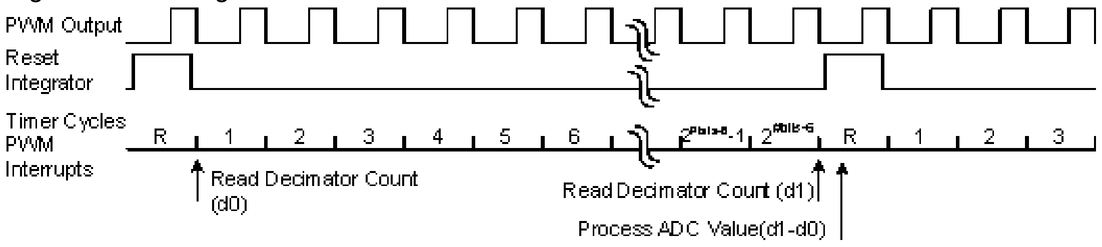
To enable the integrator to function as an incremental ADC, the digital resources used are:

- A PWM to count the proper number of integration cycles.
- A decimator, configured in the incremental mode, to accumulate the number of cycles that the output comparator is positive.

Note **CAUTION:** When placing this module, it must be configured with the same source clock for both the analog and digital blocks. Failure to do so causes it to operate incorrectly.

The PWM is set up to generate an interrupt every 256 counts. This causes the input to be sampled 64 times, which is equivalent to one integrate cycle. The decimator counter is set up to accumulate $2^{Bits}/64$ of these integrate cycles. The accumulated value is sampled at the start and finish of the integrate time. A single cycle is added to reset the integrator and process the answer.

Figure 4. Timing for ADCINC

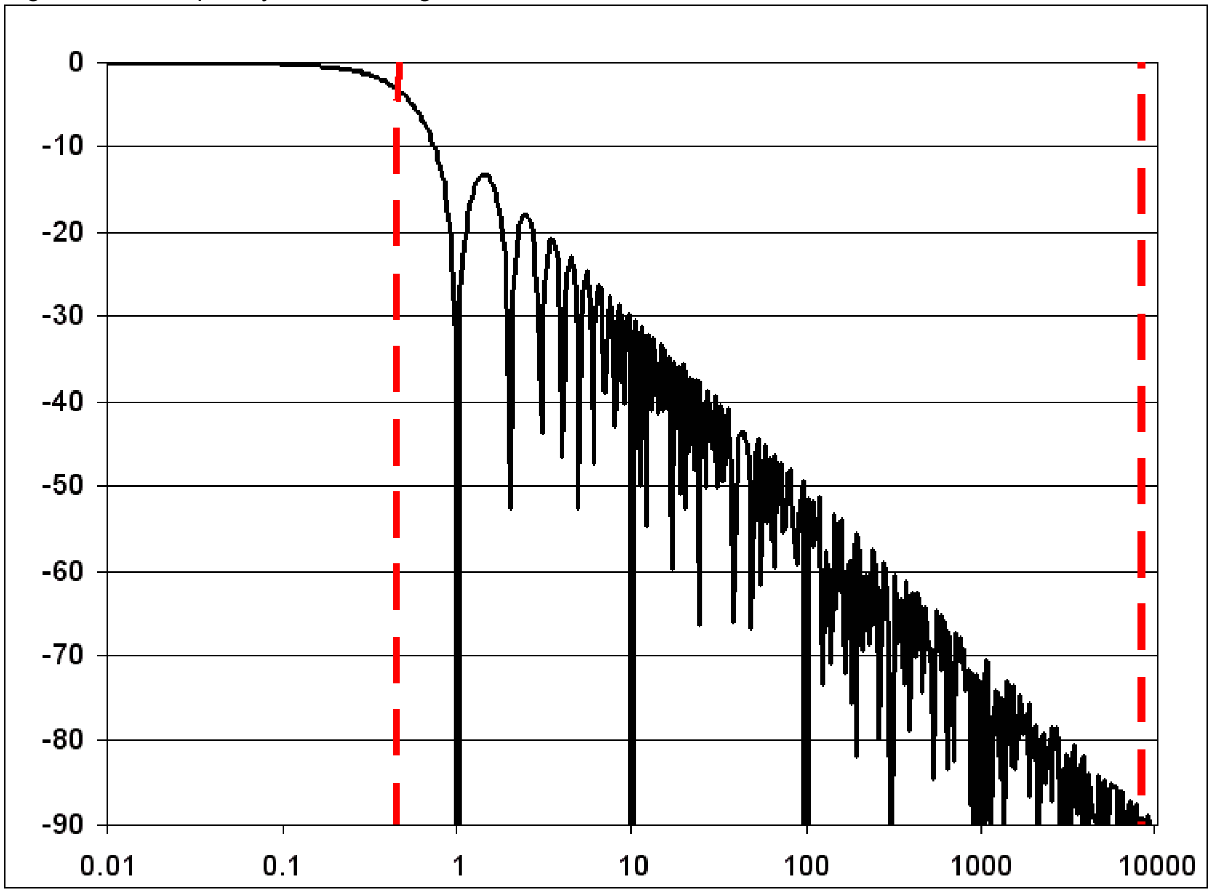


Because the ACDINC control is interrupt based and the sample time is relatively long, it is unreasonable to expect the processor to wait while a sample is being processed. The primary communication between the ADC routine and the main program is a data-available flag that may be polled. APIs are available to check the data flag and retrieve data.

The data handler is designed to be poll-based. If you need an interrupt-based data handler, application-specific data handler code can be added to the interrupt routine, _ADCINC_ADConversion_ISR, located in the assembly file ANDINCINT.asm. The point where you must insert the code is marked clearly.

The following frequency domain magnitude plot normalizes the frequency so the 14-bit sample rate, F_{nom} = 1.0. The -3 dB point occurs at .443 ∞F_{nom} and zeros of the function occur at each integer multiple of F_S. Since the ADCINCPWM is set for a resolution of 14 bits, actually samples 16385 times faster than the nominal output rate, the Nyquist limit is 8,192 higher, 13 octaves above F_{nom}, which significantly reduces the requirements for an anti-alias filter. The Nyquist limit is 12 octaves for 13 bits of resolution, 11 octaves for 12 bits of resolution, and so on.

Figure 5. Frequency Domain Magnitude Plot



DC and AC Electrical Characteristics

The following values are indicative of expected performance and based on initial characterization data. Unless otherwise specified, TA = 25 °C, Vdd = 5.0 V, Power HIGH, Opamp bias LOW, output referenced to 2.5 V external Analog Ground on P2[4] with 1.25 external Vref on P2[6].

Table 1. 5.0 V Second Order Modulator DC and AC Electrical Characteristics

Parameter	Typical	Limit	Units	Conditions and Notes
Input				
Input voltage range	---	Vss to Vdd	V	Ref Mux = Vdd/2 ± Vdd/2
Input capacitance[1]	3	---	pF	
Input impedance	1/(C*clk)	---	Ω	
Resolution	---	8	Bits	
Sample rate	---	.125 to 31.25	ksps	
SNR	46	---	dB	
DC accuracy				
DNL	0.1	---	LSB	Column Clock 2 MHz
INL	0.5	---	LSB	
Offset error	10	---	mV	
Gain error				
Including reference gain error	3.0	--	% FSR	
Excluding reference gain error[2]	0.1	--	% FSR	
Operating current				
Low power	180	---	uA	
Med power	840	---	uA	
High power	3450	---	uA	
Data clock	---	0.032 to 8.0	MHz	Input to digital blocks and analog column clock

Table 2. 5.0 V First Order Modulator DC and AC 5.0 V Electrical Characteristics (For 8-bit and 10-bit Resolutions)

Parameter	Conditions and Notes	Typical	Limit	Units
Input				
Input voltage range	Ref Mux = Vdd/2 ± Vdd/2	---	Vss to Vdd	
Input capacitance[1]		3	---	pF
Input impedance		1/(C*clk)	---	Ω
Resolution		---	8, 10	Bits
Sample rate		---	.125 to 31.25	ksps
SNR	8-bit resolution	44	---	dB
	10-bit resolution	56		
DC accuracy				
DNL	Column Clock 2 MHz	0.6	---	LSB
INL	Column Clock 2 MHz (8-bit resolution)	0.7	---	LSB
	Column Clock 2 MHz (10-bit resolution)	0.8		
Offset error	Column Clock 2 MHz	5.0	---	mV
Gain error				
Including reference gain error		3.0	--	% FSR
Excluding reference gain error[2]		0.1	--	% FSR
Operating current				
Low power	8-bit resolution	50	---	uA
	10-bit resolution	60		
Med power	8-bit resolution	500	---	uA
	10-bit resolution	520		
High power	8-bit resolution	1900	---	uA
	10-bit resolution	2000		
Data clock	Input to digital blocks and analog column clock	---	0.032 to 8.0	MHz

The following values are indicative of expected performance and based on initial characterization data. Unless otherwise specified, TA = 25 °C, Vdd = 3.3 V, Power HIGH, OpAmp bias LOW, output referenced to 1.64 V external Analog Ground on P2[4] with 1.25 external Vref on P2[6].

Table 3. 3.3 V Second Order Modulator DC and AC Electrical Characteristics

Parameter	Typical	Limit	Units	Conditions and Notes
Input				
Input voltage range	---	Vss to Vdd	V	Ref Mux = Vdd/2 ± Vdd/2
Input capacitance[1]	3	---	pF	
Input impedance	1/(C*clk)	---	Ω	
Resolution	---	8	Bits	
Sample rate	---	.125 to 31.25	ksps	
SNR	46	---	dB	
DC accuracy				
DNL	0.1	---	LSB	Column Clock 2 MHz
INL	0.5	---	LSB	
Offset error	10	---	mV	
Gain error				
Including reference gain error	3.0	--	% FSR	
Excluding reference gain error[2]	0.3	--	% FSR	
Operating current				
Low power	130	---	uA	
Med power	840	---	uA	
High power	3370	---	uA	
Data clock	---	0.032 to 8.0	MHz	Input to digital blocks and analog column clock

Table 4. 3.3 V First Order Modulator DC and AC Electrical Characteristics

Parameter	Typical	Limit	Units	Conditions and Notes
Input				
Input voltage range	---	Vss to Vdd	V	Ref Mux = Vdd/2 ± Vdd/2
Input capacitance[1]	3	---	pF	
Input impedance	1/(C*clk)	---	Ω	
Resolution	---	8	Bits	
Sample rate	---	.125 to 31.25	ksps	
SNR	44	---	dB	
DC accuracy				
DNL	0.6	---	LSB	Column Clock 2 MHz
INL	0.8	---	LSB	
Offset error	6	---	mV	
Gain error				
Including reference gain error	3.0	--	% FSR	
Excluding reference gain error[2]	0.3	--	% FSR	
Operating current				
Low power	50	---	uA	
Med power	500	---	uA	
High power	1900	---	uA	
Data clock	---	0.032 to 8.0	MHz	Input to digital blocks and analog column clock

Electrical Characteristics Notes

1. Includes I/O pin.
2. Reference Gain Error measured by comparing the external reference to VRefHigh and VRefLow routed through the test mux and back out to a pin.

Placement

The first order modulator design requires two PSoC blocks, one digital, and one analog. The analog block may be placed in any switched capacitor PSoC Block. The only considerations are input and clock availability. The digital block, however, must be able to feed the hardware decimator. In the CY8C27xxx family the qualified digital blocks are DBB01, DBB02, DBB11, and DCB12. In other device families any of the digital blocks can be used. The same clock must be selected for both the digital block and the analog block or this user module does not function correctly.

The second order modulator design requires a second switched capacitor PSoC block. Both analog blocks must lie in the same column so they can share the column comparator bus. The digital block is subject to the same restrictions for both first and second order modulators.

Although there are a number of placements possible for the analog and digital blocks, the ADCINC also uses the PSoC device's only hardware decimator. Only one ADCINC instance may be placed for a given configuration. With dynamic reconfiguration, it is possible to load more than one configuration if the blocks do not overlap. Though both instances appear to work, only the output of the one most recently loaded is functional.

Parameters and Resources

After an ADCINC PWM instance is placed, these parameters must be configured for proper operation: the Resolution, DataFormat, DataClock, PosInput Signal Multiplexer selection, NegInput Multiplexer selection, NegInput gain, Clock Phase, Pulse Width, and PWM Output.

DataFormat

This selection determines the data format of the return result. Signed results are two's complement values with the selected resolution.

Resolution

This selection determines the data format of the return result. Valid resolution options are from 6 to 14 bits.

Data Clock

The Data Clock determines the sample rate. This clock goes to both PSoC blocks of the first order modulator design and to all three PSoC block of the second order design.

Note **IMPORTANT:** It is imperative that the same clock is selected for both the digital block and the analog column clock or this user module does not function correctly.

The Data Clock must not be set to less than 250 kHz when the CPU is running at 24 MHz. Otherwise, it may be set as low as 125 kHz. The Data Clock may not exceed the CPU clock, it must always be equal to or less than the CPU Clock. The PWM is set to provide an interrupt every 256 counts of the Data Clock.The counter integrates the signal for 2^{Bits-6} of these cycles. An additional cycle is required to reset the integrator and process the data. The sample rate is defined in Equation 3:

Equation 3

$$SampleRate = \frac{DataClock}{256 \cdot (2^{Bits-6} + 1)}$$

The maximum DataClock that can be used is 8 MHz. This is because of limitations in the Switched Cap blocks. The maximum sample rate for each of the various bit rates can be calculated using an 8 MHz clock rate. The sample rates are listed in the following table:

Resolution	Maximum Sample Rate
6-bit	15.6 ksps
7-bit	10.4 ksps
8-bit	6.25 ksps
9-bit	3.4 ksps
10-bit	1.8 ksps
11-bit	976 sps
12-bit	480 sps
13-bit	242 sps
14-bit	121 sps

The sample window determines the normal mode frequencies the ADC rejects. It is defined as:

Equation 4

$$SampleWindow = \frac{2^{Bits}}{DataClock}$$

To reject a higher frequency and its harmonics, select the sample window such that it is an even multiple of the frequency-to-reject.

Clock Phase

The selection of the Clock Phase is used to synchronize the output of one analog PSoC block to the input of another. The switched capacitor analog PSoC blocks use a two-phase clock (phi1, phi2) to acquire and transfer signals. Normally, the input to the ADCINC is sampled on phi1. A problem arises in that many of the user modules autozero their output during phi1 and only provide a valid output during phi2. If such a module's output is fed to the ADCINC’s input, it acquires an autozeroed output instead of a valid signal. The Clock Phase selection allows the phases to be swapped, so that the input signal is acquired during phi2.

PosInput

The main input to the ADC. PSoC Designer allows you to select any legal input.

NegInput

Allows for the creation of a differential input for the ADC. This input can be weighted through the use of the NegInput Gain parameter. If a single input as opposed to a differential input is desired then set the NegInput Gain parameter value to “Disconnected". For the NegInput parameter, PSoC Designer allows you to select any legal input.

NegInput Gain

Selects the Gain for the negative input. If single-ended input is desired, set this value to “Disconnected”.

PulseWidth

Allows PWM pulsewidth to set from a value 1 to 255 counts. If no value is set then the user module automatically sets the PWM pulsewidth to 1 when the GetSamples function is called.

PWM Output

The Output parameter may be disabled or routed to one of four global output signals.

Interrupt Generation Control

The following parameter is only accessible when the Enable Interrupt Generation Control check box in PSoC Designer is checked. This is available under **Project > Settings... > Device Editor.**

IntDispatchMode

The IntDispatchMode parameter is used to specify how an interrupt request is handled for interrupts shared by multiple user modules existing in the same block but in different overlays. Selecting “ActiveStatus" causes firmware to test which overlay is active before servicing the shared interrupt request. This test occurs every time the shared interrupt is requested. This adds latency and also produces a nondeterministic procedure of servicing shared interrupt requests, but does not require any RAM. Selecting “OffsetPreCalc" causes firmware to calculate the source of a shared interrupt request only when an overlay is initially loaded. This calculation decreases interrupt latency and produces a deterministic procedure for servicing shared interrupt requests, but at the expense of a byte of RAM.

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 before 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 may 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.

Each time a user module is placed, it is assigned an instance name. By default, PSoC Designer assigns the ADCINC_1 to the first instance of this user module in a given project. It can be changed to any unique value that follows the syntactic rules for identifiers. The assigned instance name becomes the prefix of every global function name, variable and constant symbol. In the following descriptions the instance name has been shortened to ADCINC for simplicity.

ADCINC_Start

Description:

Performs all required initialization for this user module and sets the power level for the switched capacitor PSoC block. The PWM is started.

C Prototype:

```
void  ADCINC_Start (BYTE bPowerSetting)
```

Assembly:

```
mov   A, bPowerSetting
lcall  ADCINC_Start
```

Parameters:

bPowerSetting: One byte that specifies the power level. Following reset and configuration, the analog PSoC block assigned to ADCINC is powered down. The symbolic names provided in C and assembly, and their associated values, are listed in the following table:

Symbolic Name	Value
ADCINC_OFF	0
ADCINC_LOWPOWER	1
ADCINC_MEDPOWER	2
ADCINC_HIGHPOWER	3

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 (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

ADCINC_SetPower

Description:

Sets the power level for the switched capacitor PSoC block.

C Prototype:

```
void  ADCINC_SetPower (BYTE bPowerSetting)
```

Assembly:

```
mov   A, bPowerSetting
lcall  ADCINC_SetPower
```

Parameters:

bPowerSetting: Same as the bPowerSetting parameter used for the Start entry point.

Return Value:

None

Side Effects:

The A and X registers may be modified by this or future implementations of this function. This is true for all RAM page pointer registers in the Large Memory Model (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

ADCINC_Stop

Description:

Sets the power level on the switched capacitor PSoC block to OFF.

C Prototype:

```
void  ADCINC_Stop (void)
```

Assembly:

```
lcall  ADCINC_Stop
```

Parameters:

None

Return Value:

None

Side Effects:

The A and X registers may be modified by this or future implementations of this function. This is true for all RAM page pointer registers in the Large Memory Model (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

ADCINC_GetSamples

Description:

Runs the ADC for the specified number of samples.

C Prototype:

```
void  ADCINC_GetSamples (BYTE bNumSamples)
```

Assembly:

```
mov   A, bNumSamples
lcall  ADCINC_GetSamples
```

Parameters:

bNumSamples: 8-bit value that sets the number of samples to be converted. As a value of ‘0’ causes the ADC to run continuously.

Return Value:

None

Side Effects:

The A and X registers may be modified by this or future implementations of this function. This is true for all RAM page pointer registers in the Large Memory Model (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions. Currently, only the CUR_PP page pointer register is modified.

ADCINC_StopADC

Description:

Immediately stops the ADC. PWM continues to run.

C Prototype:

```
void  ADCINC_StopADC (void)
```

Assembly:

```
lcall  ADCINC_StopADC
```

Parameters:

None

Return Value:

None

Side Effects:

The A and X registers may be modified by this or future implementations of this function. This is true for all RAM page pointer registers in the Large Memory Model (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

ADCINC_fIsDataAvailable

Description:

Checks the availability of sampled data.

C Prototype:

```
BYTE  ADCINC_fIsDataAvailable(void)
```

Assembly:

```
lcall  ADCINC_fIsDataAvailable
```

Parameters:

None

Return Value:

Returns a nonzero value if data has been converted and is ready to read.

Side Effects:

The A and X registers may be modified by this or future implementations of this function. This is true for all RAM page pointer registers in the Large Memory Model (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions. Currently, only the CUR_PP page pointer register is modified.

ADCINC_iGetData

Description:

Returns converted data as a signed integer. ADCINC_fIsDataAvailable() must be called to verify that the data sample is ready.

C Prototype:

```
INT   ADCINC_iGetData(void)
```

Assembly:

```
lcall  ADCINC_iGetData            ; Data will be in A and X upon return
```

Parameters:

None

Return Value:

Returns the converted data sample in 16-bit 2’s complement format.

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 (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions. Currently, only the CUR_PP page pointer register is modified.

ADCINC_wGetData

Description:

Returns converted data as an unsigned integer. ADCINC_fIsDataAvailable() must be called to verify that the data sample is ready.

C Prototype:

```
WORD  ADCINC_wGetData(void)
```

Assembly:

```
lcall  ADCINC_wGetData            ; Data will be in A and X upon return
```

Parameters:

None

Return Value:

Returns the converted 16-bit unsigned data sample.

Side Effects:

The A and X registers may be modified by this or future implementations of this function. This is true for all RAM page pointer registers in the Large Memory Model (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions. Currently, only the CUR_PP page pointer register is modified.

ADCINC_cGetData

Description:

Returns converted data as a signed char. ADCINC_fIsDataAvailable() must be called to verify that the data sample is ready.

C Prototype:

```
CHAR  ADCINC_cGetData(void)
```

Assembly:

```
lcall  ADCINC_cGetData            ; Data will be in A upon return
```

Parameters:

None

Return Value:

Returns the converted data sample in 8-bit 2’s complement format.

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 (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions. Currently, only the CUR_PP page pointer register is modified.

ADCINC_bGetData

Description:

Returns converted data as an unsigned char. ADCINC_fIsDataAvailable() must be called to verify that the data sample is ready.

C Prototype:

```
BYTE  ADCINC_bGetData(void)
```

Assembly:

```
lcall  ADCINC_bGetData            ; Data will be in A upon return
```

Parameters:

None

Return Value:

Returns the converted 8-bit unsigned data sample.

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 (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions. Currently, only the CUR_PP page pointer register is modified.

ADCINC_iClearFlagGetData

Description:

Clears the data ready flag and gets converted data as signed integer. Checks to see that data flag is still reset. If not the data is retrieved again. This ensures that the ADC interrupt routine did not update the answer while it was being collected.

C Prototype:

```
INT   ADCINC_iClearFlagGetData(void)
```

Assembly:

```
lcall  ADCINC_iClearFlagGetData   ; Data will be in A and X upon return
```

Parameters:

None

Return Value:

Returns the converted data sample in 16-bit 2’s complement format.

Side Effects:

The global variable ADCINC_bfStatus is set to zero. 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 (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions. Currently, only the CUR_PP page pointer register is modified.

ADCINC_wClearFlagGetData

Description:

Clears the data ready flag and gets converted data as unsigned integer. Checks if the data-flag is still reset. If the data-flag is not reset, the data is retrieved again. This ensures that the ADC interrupt routine does not update the answer while it is being collected.

C Prototype:

```
WORD  ADCINC_wClearFlagGetData(void)
```

Assembly:

```
lcall  ADCINC_wClearFlagGetData   ; Data will be in A and X upon return
```

Parameters:

None

Return Value:

Returns the converted 16-bit unsigned data sample.

Side Effects:

The global variable ADCINC_bfStatus is set to zero. 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 (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions. Currently, only the CUR_PP page pointer register is modified.

ADCINC_cClearFlagGetData

Description:

Clears the data ready flag and gets converted data as signed char. Checks to see that data-flag is still reset. If not the data is retrieved again. This makes sure that the ADC interrupt routine did not update the answer while it was being collected.

C Prototype:

```
CHAR  ADCINC_cClearFlagGetData(void)
```

Assembly:

```
lcall  ADCINC_cClearFlagGetData   ; Data will be in A upon return
```

Parameters:

None

Return Value:

Returns the converted data sample in 8-bit 2's complement format.

Side Effects:

The global variable ADCINC_bfStatus is set to zero. 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 (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions. Currently, only the CUR_PP page pointer register is modified.

ADCINC_bClearFlagGetData

Description:

Clears the data ready flag and gets converted data as an unsigned char. Checks to see that data-flag is still reset. If not the data is retrieved again. This makes sure that the ADC interrupt routine did not update the answer while it was being collected.

C Prototype:

```
BYTE  ADCINC_bClearFlagGetData(void)
```

Assembly:

```
lcall  ADCINC_bClearFlagGetData   ; Data will be in A upon return
```

Parameters:

None

Return Value:

Returns the converted 8-bit unsigned data sample.

Side Effects:

The global variable ADCINC_bfStatus is set to zero. 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 (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions. Currently, only the CUR_PP page pointer register is modified.

ADCINC_fClearFlag

Description:

Returns the contents of the data available variable and resets the flag.

C Prototype:

```
BYTE  ADCINC_fClearFlag(void)
```

Assembly:

```
lcall  ADCINC_fClearFlag
```

Parameters:

None

Return Value:

The returns the value of the status register.

Side Effect:

The global variable ADCINC_bfStatus is set to zero.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 (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions. Currently, only the CUR_PP page pointer register is modified.

ADCINC_WritePulseWidth

Description:

Changes the pulse width of the PWM.

C Prototype:

```
void  ADCINC_WritePulseWidth(BYTE bPulseWidth)
```

Assembly:

```
mov   A, bPulseWidth
lcall  ADCINC_WritePulseWidth
```

Parameters:

bPulseWidth: This sets the width of the PWM. This value must not be zero or the ADC stops functioning.

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 (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

Sample Firmware Source Code

The following sample code polls the Flag register and sends the data to a routine that shifts the data out one of the I/O pins.

```
;;; Sample Code for the ADCINC
;;; Continuously Sample and Output Data to a pin.
;;;
;;; The user must provide the function to shift the data out.
;;;
include "m8c.inc"                ; part specific constants and macros
include "PSoCAPI.inc"            ; PSoC API definitions for all User Modules
export _main
_main:
M8C_EnableGInt                  ; enable global interrupts
mov  a,ADCINC_HIGHPOWER         ; set Power
call ADCINC_Start
mov  a,00h                      ; set ADC to continuous sampling
call ADCINC_GetSamples
loop1:
wait:
call ADCINC_fIsDataAvailable    ; poll flag
jz wait
call ADCINC_iClearFlagGetData   ; reset flag and retrieve data
;; call shift_it_out            ; (user provided) send data to output pin
jmp loop1
```

The same project written in C is:

```
//-------------------------------------------------------------------------
//  Sample C Code for the ADCINC
//  Continuously Sample input voltage
//
//-------------------------------------------------------------------------
#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules
INT  iData;
void main(void)
{
   M8C_EnableGInt;                         // Enable Global Interrupts
ADCINC_Start(ADCINC_HIGHPOWER);       // Apply power to the SC Block
ADCINC_GetSamples(0);                 // Have ADC run continuously
for(;;){
while(ADCINC_fIsDataAvailable() == 0);   // Loop until value ready
ADCINC_iClearFlagGetData();              // Clear ADC flag and get data
// Add user code here to use or display result
}
 }
```

Configuration Registers

Table 5. Registers used by the “ADC" Analog Switched Capacitor PSoC Block

Register	7	6	5	4	3	2	1	0
CR0	1	ClockPhase	0	ACap				
CR1	PosInputSource			NegInputGain				
CR2	0	1	AZ	0	0	0	0	0
CR3	1	1	1	FSW0	NegInputSource		0	0

The ADC uses one or two switched capacitor PSoC blocks. The blocks are configured to make an analog modulator. To build the modulator, the blocks are configured to be an integrator with reference feedback that converts the input value into a digital pulse stream. The input multiplexer determines what signal is digitized.

ClockPhase controls the clock phase of the comparator within the switched cap blocks, as well as the clock phase of the switches.

ACap contains binary encoding for 32 possible capacitor sizes for capacitor ACap.

InputSource field selects the input signal digitized by the converter. This parameter is set in the Device Editor.

The AZ and FSW0 are used by the PWM interrupt handler and various APIs to reset the integrator.

Table 6. Registers used by the PWM Digital PSoC Block

Register	7	6	5	4	3	2	1	0
Function	0	0	1	1	0	0	0	1
Input	0	0	0	1	Clock			
Output	0	0	0	0	0	0	0	0
DR0	Timer Down Count Value (Never Accessed by the API)							
DR1	1	1	1	1	1	1	1	1
DR2	PulseWidth							
CR0	0	0	1	0	0	0	0	Enable

The PWM is a digital PSoC block configured to have a timer with a period of 256 counts. At the interrupt, the decimator is read and the ADC value is calculated.

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

Note The source chosen must also be used to control the analog clock for the column where the ADC block resides.

Enable empowers the PWM when set. It is modified and controlled by the ADCINC API.

Table 7. Decimation Control Registers

Bit	7	6	5	4	3	2	1	0
DEC_CR0	0	0	0	0	ICLKS0	0	DCol	DCLKS0
DEC_CR1	1	1	0	0	ICLKS1			DCLKS1
DEC_DH	High Byte Output of Decimator							
DEC_DL	Low Byte Output of Decimator							

Version History

Version	Originator	Description
1.1	DHA	Added DRC to check if: 1. The source clock is different between digital and analog resources. 2. The ADC Clock is higher than CPU Clock.
1.20	DHA	Restored VC3 as the source for the data clock.
1.20.b	DHA	Added 10-bit resolution data in DC-AC Characteristics table.
1.20.c	HPHA	Added design rules check for the situation when the ADC clock is faster than 8 MHz.

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.