Comparator Datasheet CMP V 1.2

Resources	PSoC® Blocks			API Memory (Bytes)		Pins (per External I/O)
	Digital	Analog CT	Analog SC	Flash	RAM	
CY7C603xx	0	1	0	17	0	1
CY8C21xxx, CY8CLED02, CY8CTST110, CY8CTMG110, CY8C21x45, CY8C22x45, CY8C28x13, CY8C28x45, CY8C28x52	0	1	0	17	0	1
CYWUSB6953	0	1	0	17	0	1

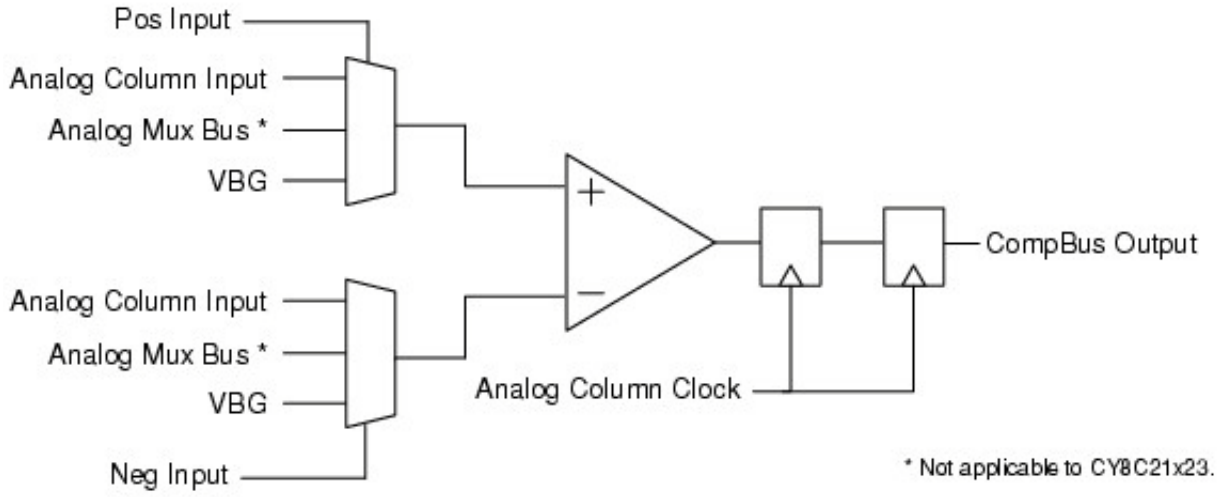
For one or more fully configured, functional code examples that use this user module go to www.cypress.com/psocexampleprojects.

Features and Overview

- Flexible input sources
- Direct connection to digital PSoC block and interrupt

The CMP User Module compares two selectable inputs. Both inputs have the same set of possible connections to choose from. This enables you to select the polarity of the output.

Figure 1. CMP Block Diagram



Functional Description

The CT Block gives a simple comparator which is at the heart of this user module. The comparator has two inputs: a positive and a negative. Both the positive and negative inputs have identical connection options at their input muxes. This allows creating any combination of inputs with either negative or positive polarity. The value of two pins can be compared by using a pin connected through the Analog Column Input Mux, and one connected through the Analog Mux Bus. An input signal can also be compared against a reference. A fixed 1.3-V reference is given internally by using the VBG analog reference. For this user module, the comparator output is always directly connected to the corresponding comparator bus in the same column

DC and AC Electrical Characteristics

The following values are indicative of expected performance and based on initial characterization data. Unless otherwise specified in the following tables, T_A = 25 °C, VDD = 5.0 V, LowLimit = VSS.

Table 1. 5.0 V CMP DC Electrical Characteristics

Parameter	Typical	Limit	Units	Conditions and Notes
Input				
Input voltage range	--	VSS to VDD	V	
Leakage[1]	1	--	nA	
Input capacitance[1]	3	--	pF	
Output swing	0.05 to VDD-0.05	--	V	

The following values are indicative of expected performance and based on initial characterization data. Unless otherwise specified in the following tables, T_A = 25 °C, VDD = 3.3 V, LowLimit = VSS.

Table 2. 3.3 V CMP DC Electrical Characteristics

Parameter	Typical	Limit	Units	Conditions and Notes
Input				
Input voltage range	--	VSS to VDD	V	
Leakage[1]	1	--	nA	
Input capacitance[1]	3	--	pF	
Output swing	0.05 to VDD-0.05	--	V	

The following values are indicative of expected performance and based on initial characterization data. Unless otherwise specified in the following tables, T_A = 25 °C, VDD = 2.7 V, LowLimit = VSS.

Table 3. 2.7 V CMP DC Electrical Characteristics

Parameter	Typical	Limit	Units	Conditions and Notes
Input				
Input voltage range	--	VSS to VDD	V	
Leakage[1]	1	--	nA	
Input capacitance[1]	3	--	pF	
Output swing	0.05 to Vdd-0.05	--	V	

Placement

The CMP block maps on any of the Analog CT blocks of the CY8C21xxx.

Parameters and Resources

Pos Input

You can select the positive input from one of these four sources: the pin input multiplexer, the analog mux bus, VBG(1.3 V), and either of the two analog SC blocks.

Neg Input

You can select the negative input from one of these four sources: the pin input multiplexer, the analog mux bus, VBG(1.3 V), and either of the two analog SC blocks.

Analog Column Clock

The analog column clock resource should be set to a source that outputs a maximum frequency of SysClk/2 for the user module to function properly. It is recommended to clock the analog column with slow frequencies to limit the oscillation produced when the two comparator inputs are near each other.

Interrupt Generation Control

There are two additional parameters that become available when the **Enable interrupt generation control** check box in PSoC Designer is checked. This is available under **Project > Settings > Chip Editor**. Interrupt Generation Control is important when multiple overlays are used with interrupts shared by multiple user modules across overlays:

InterruptAPI

The InterruptAPI parameter allows conditional generation of a user module's interrupt service routine (ISR) and interrupt vector table entry. Select "Enable" to generate the ISR and interrupt vector table entry. Select "Disable" to bypass the generation of the ISR and interrupt vector table entry. Properly selecting this parameter is particularly important for projects with multiple overlays where a single block resource is shared by the different overlays. You can eliminate the need to generate the interrupt dispatch code by enabling Interrupt API generation only when it is necessary. This reduces code overhead.

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 small amount of RAM.

Application Programming Interface

The Application Programming Interface (API) routines are given as part of the user module to help you deal with the module at a higher level. This section specifies the interface to each function together with related constants given 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. 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.

CMP_Start

Description:

Performs all required initialization for this user module and turns on the comparator.

C Prototype:

```
void  CMP_Start(void)
```

Assembler:

```
lcall  CMP_Start
```

Parameters:

None

Return Value:

None

Side Effects:

The A and X registers may be altered by this function.

CMP_Stop

Description:

Turns off the power in the user module. The outputs are not driven.

C Prototype:

```
void  CMP_Stop(void)
```

Assembler:

```
lcall  CMP_Stop
```

Parameters:

None

Return Value:

None

Side Effects:

The A and X registers may be altered by this function.

CMP_EnableInt

Description:

Enables the interrupt associated with the Analog Comparator Bus.

C Prototype:

```
void  CMP_EnableInt(void)
```

Assembler:

```
lcall  CMP_EnableInt
```

Parameters:

None

Return Value:

None

Side Effects:

The A and X registers may be altered by this function.

CMP_DisableInt

Description:

Disables the interrupt associated with the Analog Comparator Bus.

C Prototype:

```
void  CMP_DisableInt(void)
```

Assembler:

```
lcall  CMP_DisableInt
```

Parameters:

None

Return Value:

None

Side Effects:

The A and X registers may be altered by this function.

Sample Firmware Source Code

The sample code creates a simple comparator that generates an interrupt whenever the positive input rises above the negative input.

```
;;-------------------------------------------------------------------
;; Sample Code for the CMP in Column 0
;;
;; The parameters are configured as such:
;; Pos Input: AnalogColumn_InputMux
;; Neg Input: AnalogMuxBus
;;
;; Resources are configured as such:
;; Pin 0.1: connected to AnalogColumn_InputMux
;; Pin 1.1: connected to AnalogMuxBus
;;-------------------------------------------------------------------

export _main

include "m8c.inc"
include "CMP.inc"

_main:

M8C_EnableGInt
call CMP_EnableInt             ; Enable the interrupt
call CMP_Start                 ; and turn it on

ret
```

The same sample code in C:

```
//-------------------------------------------------------------------
// Sample Code for the CMP in Column 0
//
// The parameters are configured:
// Pos Input: AnalogColumn_InputMux
// Neg Input: AnalogMuxBus
//
// Resources are configured as such:
// Pin 0.1: connected to AnalogColumn_InputMux
// Pin 1.1: connected to AnalogMuxBus
//-------------------------------------------------------------------

#include <m8c.h>
#include "PSoCAPI.h"

void main(void)
{
    M8C_EnableGInt;
    CMP_EnableInt();
    CMP_Start();                    //Start the comparator
}
```

Configuration Registers

The basic topology of the comparator sets most of the bits in the register configuration for the analog CT block that is used.

Table 4. Block CMP, Register: ACE_CR1

Bit	7	6	5	4	3	2	1	0
Value	0	1	Negative Input			Positive Input		

The positive and the negative inputs offer the same set of possible connections. The Analog Column Input, the Analog Mux Bus, AGND, and the Analog SC block right below if an internal programmable reference is to be used.

Table 5. Block CMP, Register: ACE_CR2

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

The Enable bit turns on the comparator, it is controlled through the Start and Stop API functions.

Version History

Version	Originator	Description
1.2	DHA	Added Version History
1.2.b	DHA	Explained comparator output connection in the user module datasheet.

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.