Enhanced Comparator Datasheet CMPEn V 1.0

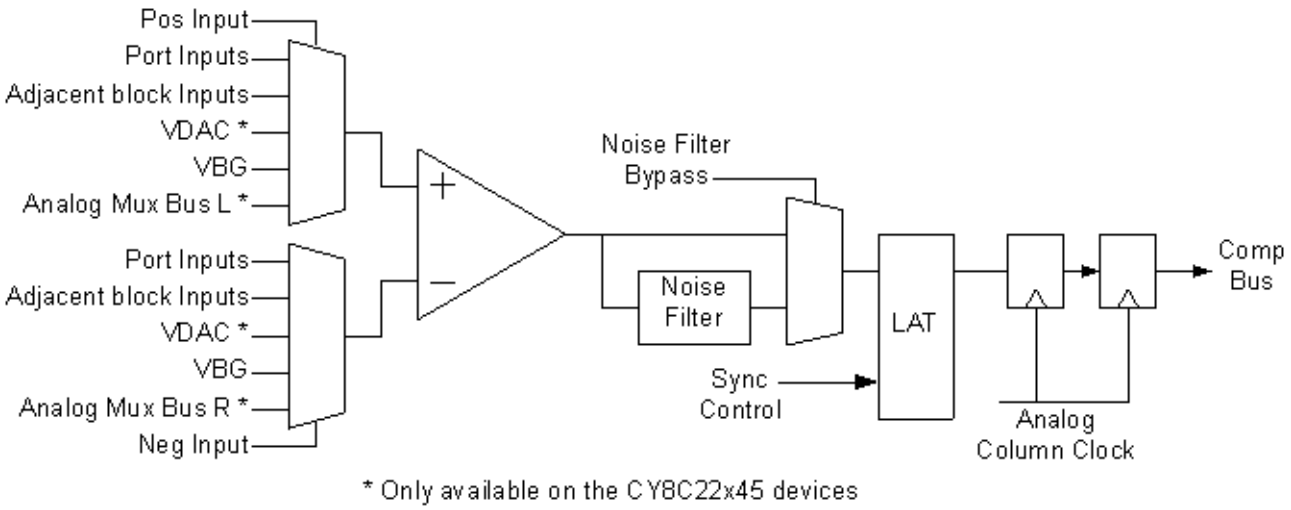
Resources	PSoC® Blocks			API Memory (Bytes)		Pins (per External I/O)
	Digital	Analog CT	Analog SC	Flash	RAM	
CY8C21x45, CY8C22x45	0	1	0	24	0	0...2

Features and Overview

- Flexible input sources
- Direct connection to digital PSoC block and interrupt
- Output signal filtering
- Output signal latching

The Enhanced Comparator (CMPEn) User Module provides a comparison of two input options. Both inputs have the same set of possible connections to choose from, which enables you to select the polarity of the output.

Figure 1. CMPEn Block Diagram



Functional Description

The ACE block provides 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 enables you to create any combination of inputs with either negative or positive polarity. You can compare the values of two pins by using a pin connected through the Analog Column Input Mux, and another pin connected through the Analog Mux Bus. You can also compare an input signal against a reference. A fixed 1.3V reference is provided internally by using the VBG analog reference.

The analog compare output signal can pass through a noise filter. That is, a comparator output pulse with width less than a selected number of clock cycles is filtered out. A “Noise Filter Bypass” property is also available to disable this function.

An RS-Latch function is present in the analog comparator output data path. The latch block is shared between two compare columns. The compare column 0 is the master input, and the compare column 1 is the slave input. The master output (Q) and the slave output (QB) go back to compare column 0 and compare column 1, respectively. If the master output is a primary output, then the 'high' signal from the master channel clears the primary output (that is, the target is cut off for protection). The 'high' signal from the slave channel sets the primary output (that is, the target is turned on for normal operation). The RS-Latch function is bypassed if the RS-Latch is not selected. An RS_EN bit is used to enable the RS function, and the RS_SEL bit in each column is used to select the RS function output.

DC and AC Electrical Characteristics

See the device datasheet for the electrical characteristics of this device.

Placement

The CMPEn block maps to any ACE02 or ACE03 block of the CY8C22x45 and CY8C21x45 device families.

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

Parameters and Resources

There are seven parameters:

- Pos Input
- Neg Input
- Noise Filter Bypass
- Filter Clk Selection
- Noise Filter Selection
- Sync Latch Enable
- Select Auto Trigger Source

Pos Input

The Positive Input can be selected from one of seven sources. These include the pin input multiplexer, VBG (1.3V), Analog Mux Bus*, VDAC*, and the adjacent Analog CT block.

Note * Only available on CY8C22x45 devices.

Neg Input

The Negative Input can be selected from one of six sources. These include the pin input multiplexer, VBG (1.3V), Analog Mux Bus*, VDAC*, and the adjacent Analog CT block.

Note * Only available on CY8C22x45 devices.

Noise Filter Bypass

This parameter can enable or disable the comparator output Noise Filter.

Filter Clk Selection

This parameter gives you two options to clock the Noise Filter: System Clock or Column Clock.

Noise Filter Selection

This parameter determines the Noise Filter Width.

Sync Latch Enable

The Sync Latch Enable parameter enables latching the output signal. This parameter contains the following selections:

Auto Trigger Global Enable	Description
Disable	Latching of output signal is disabled.
Enable	Latching of output signal is enabled. The Auto-Trig Selection System must be activated. See Note ** at the end of the Parameters and Resources section.

Select Auto Trigger Source

The Select Auto-Trigger Source sets the current auto-trigger source for comparator output signal latching. This parameter contains the following selections:

Select Auto Trigger Source	Description
TGL	The low 8-bit digital path is used as an auto-trigger source.
TGH	The high 8-bit digital path is used as an auto-trigger source.
TG16Bit	The combined high/low 8-bit digital path is used as an auto-trigger source.
TGINCMP	The GIE or internal comparators is used as an auto-trigger source.

PSoC digital blocks can be used as a trigger source for comparator output signal latching. The output signal latches when a one-shot pulse occurs on the selected trigger source.

Note ** For example, the control registers of the Auto-Trig Selection System can be configured as:

```
TSCMPH = 1; // set CMP_H compare value
TSCR1 = 0x10; // select CMP_H source DBC01
TSCR0 = 0x02; // enable high channel trigger source
```

This means that the trig signal passes from the occupied DBC01 digital block. So that, for example, the Counter8 User Module should be placed on DBC01 to generate an auto-trig signal.

Application Programming Interfaces (APIs)

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.

Each time a user module is placed, it is assigned an instance name. By default, PSoC Designer assigns CMPEn_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 CMPEn for simplicity

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 functions 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.

CMPEn_Start

Description:

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

C Prototype:

```
void  CMPEn_Start(void)
```

Assembler:

```
lcall  CMPEn_Start
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section.

CMPEn_Stop

Description:

Powers the user module off. The outputs are not driven.

C Prototype:

```
void  CMPEn_Stop(void)
```

Assembler:

```
lcall  CMPEn_Stop
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section.

CMPEn_EnableInt

Description:

Enables the interrupt associated with the Analog Comparator Bus.

C Prototype:

```
void  CMPEn_EnableInt(void)
```

Assembler:

```
lcall  CMPEn_EnableInt
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section.

CMPEn_DisableInt

Description:

Disables the interrupt associated with the Analog Comparator Bus.

C Prototype:

```
void  CMPEn_DisableInt(void)
```

Assembler:

```
lcall  CMPEn_DisableInt
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section.

Sample Firmware Source Code

The sample code given here creates a simple comparator that generates an interrupt whenever the Positive Input rises above the negative input:

```
;;-------------------------------------------------------------------
;; Sample Code for the CMPEn in Column 2
;;
;; The parameters are configured as such:
;; Interrupt: enabled
;; Power: switched on
;;-------------------------------------------------------------------

export _main

include "m8c.inc"
include "CMPEn.inc"

_main:
M8C_EnableGInt
call CMPEn_EnableInt             ; Enable the interrupt
call CMPEn_Start                 ; and turn it on
```

The same code in C is:

```
//-------------------------------------------------------------------
// Sample Code for the CMPEn in Column 2
//
// The parameters are configured:
// Interrupt: enabled
// Power: switched on
//-------------------------------------------------------------------

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

void main(void)
{
    M8C_EnableGInt;
    CMPEn_EnableInt();
    CMPEn_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 1. Block CMPEn, 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, and the Analog SC block if an internal programmable reference is used.

Table 2. Block CMPEn, Register: ACE_CR2

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

The Enable bit turns on the comparator. This is controlled by the Start and Stop API functions.

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.