Hardware Current Sense Amplifier Datasheet CurSenseHWV 1.0

Copyright © 2009-2013 Cypress Semiconductor Corporation. All Rights Reserved.

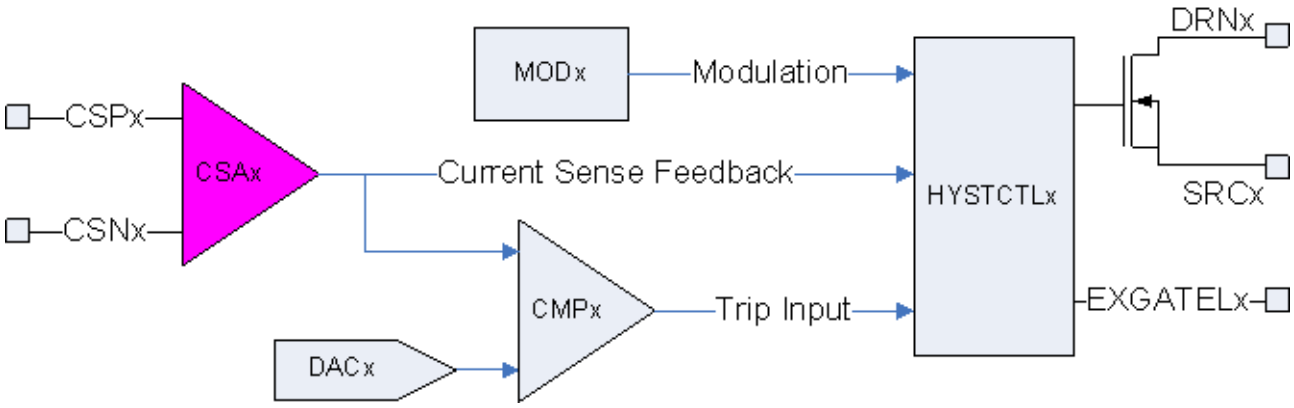
Resources	PSoC® Blocks	API Memory (Bytes)		Pins (per External I/O)
	CSA	Flash	RAM	
CY8CLED0xD, CY8CLED0xG	1	60	0	2

Features and Overview

- Operates with a high common mode voltage of 36V
- Possesses high common mode rejection ratio
- Gives bandwidth adjustment capability with high bandwidth
- Gives low input offset currents and low offset voltages
- Highly accurate output

The high side current sense amplifiers (CSA) give a differential sense capability to measure the voltage across current sense resistors in power systems.

Figure 1. CurSenseHW Block Diagram



Functional Description

The CurSenseHW User Module (UM) block consists of two amplifier stages, Stage 1 and Stage 2. Stage 1 is used to level shift and amplify a high side input. Stage 2 is used to amplify Stage 1 output. Stage 2 gain is fixed and is equal to 5. As a result, the total gain is fixed and is equal to 20.

The CurSenseHW performs continuous time differential amplification of the input voltage, VSENSE (V(INP)-V(INN)), to produce a single ended analog signal, VOUT, given by Equation 1.

Equation 1

$$Vout = 20 . Vsense$$

DC and AC Electrical Characteristics

See the device characterization data in the DC and AC Electrical Characteristics section of the device datasheet.

Placement

CurSenseHW occupies one of the four possible blocks: CSA0, CSA1, CSA2, or CSA3.

Parameters and Resources

Bandwidth

The Bandwidth parameter controls the capacitance load at the output of the Stage 1 amplifier and gives bandwidth adjustment capability, allowing tradeoffs in bandwidth, time delay, and power supply rejection ratio. The bandwidth can be selected from one of the following values:

Bandwidth	Description
Highest (Default)	Highest bandwidth, no capacitance added to Stage 1 output.
MediumHigh	Medium high bandwidth.
MediumLow	Medium low bandwidth.
Lowest	Lowest bandwidth, most capacitance added.

Application Programming Interface

The Application Programming Interface (API) functions are given as part of the user module to allow you to 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.

Each time a user module is placed, it is assigned an instance name. By default, PSoC Designer assigns CURSENSEHW_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 CURSENSEHW 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 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 also ensure their code observes the policy. Though some user module API functions 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.

CURSENSEHW_Start

Description:

Performs all required initialization for this user module and enables the block. The user module output is driven.

C Prototype:

```
void  CURSENSEHW_Start(void);
```

Assembly:

```
lcall  CURSENSEHW_Start
```

Parameters:

None.

Return Value:

None.

Side Effects:

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

CURSENSEHW_Stop

Description:

Stops CurSenseHW operation.

C Prototype:

```
void  CURSENSEHW_Stop(void);
```

Assembly:

```
lcall  CURSENSEHW_Stop
```

Parameters:

None.

Return Value:

None.

Side Effects:

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

CURSENSEHW_SetBandwidth

Description:

Controls the capacitance load at the output of the Stage 1 amplifier and gives bandwidth adjustment capability, allowing trade-offs in bandwidth, time delay, and power supply rejection ratio.

C Prototype:

```
void  CURSENSEHW_SetBandwidth(BYTE bBandwidth);
```

Assembly:

```
mov A, bBandwidth
lcall  CURSENSEHW_SetBandwidth
```

Parameters:

bBandwidth: Is the bandwidth value. Symbolic names are given in C and assembly. Their associated values are listed in the following table:

Symbolic Name	Value	Description
CURSENSEHW_BANDWIDTH_HIGHEST	0x00	Highest bandwidth, no capacitance added.
CURSENSEHW_BANDWIDTH_MEDIUM_HIGH	0x10	Medium high bandwidth.
CURSENSEHW_BANDWIDTH_MEDIUM_LOW	0x20	Medium low bandwidth.
CURSENSEHW_BANDWIDTH_LOWEST	0x30	Lowest bandwidth, most capacitance added.

Return Value:

None.

Side Effects:

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

Sample Firmware Source Code

The C code illustrated here shows you how to use the CurSenseHW User Module.

```
CURSENSEHW_SetBandwidth(CURSENSEHW_BANDWIDTH_HIGHEST); // set the higest bandwidth
CURSENSEHW_Start();                    // start user module
```

The same code in assembly is:

```
mov  A, CURSENSEHW_BANDWIDTH_HIGHEST
call CURSENSEHW_SetBandwidth     ; set the higest bandwidth
call CURSENSEHW_Start       ; start user module
```

Configuration Registers

Table 1. CURSENSEHW_CONTROL_REG

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

The Enable bit is modified by calling the Start or Stop API routine.

The Bandwidth bits determine the CurSenseHW bandwidth. The value of these bits is determined by the choice made for the parameter of the same name under user module parameters in the Device Editor. The value can also be changed by calling the SetBandwidth API.

Version History

Version	Originator	Description
1.0	DHA	Removed the Gain property and CURSENSEHW_SetGain API. This is because a CSA gain value of 20 is the only value that is guaranteed by the device datasheet.

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

Copyright © 2009-2013 Cypress Semiconductor Corporation. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC Designer™ and Programmable System-on-Chip™ are trademarks and PSoC® is a registered trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.