8-Bit Shift Register Datasheet SHIFTREG8 V 1.0

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

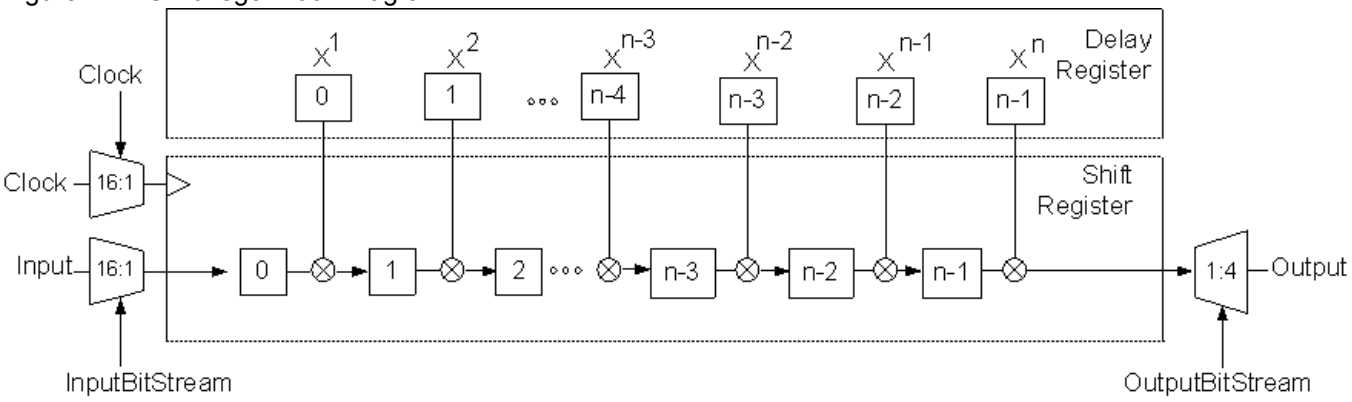
Resources	PSoC® Blocks			API Memory (Bytes)		Pins (per External I/O)
	Digital	Analog CT	Analog SC	Flash	RAM	
CY8C21x45, CY8C22x45, CY8C22x45H, CY8C28xxx						
	0	0	0	25	0	1 to 4

Features and Overview

- 8-bit Shift Register uses one PSoC block
- Source clock rates up to 48 MHz
- Data input for clocks up to 24 MHz
- Programmable Delay Cycle value
- Serial output bit stream

The ShiftReg8 User Module is a variation of a modular linear feedback shift register (LFSR) that delays an input bit stream. The Delay Cycle Number value can be specified to define its output delayed up to 8 PSoC block clocks.

Figure 1. ShiftReg8 Block Diagram



Functional Description

The ShiftReg8 User Module shifts digital signals in applications such as frequency-shift keying (FSK). The digital signal can be delayed by a specific digital block (DB) clock period when passing through the user module.

The Delay and Control registers of the ShiftReg8 PSoC block are used to define and control the generation of the output bit sequence delayed relative to the input. The Delay register is write-only. The highest bit position in the Delay register containing a ‘1’ determines the number of delay cycles.

The configuration of the underlying connective hardware of the digital PSoC blocks coordinates the operation of the PSoC blocks as a single ShiftReg8 User Module. The Delay register sets up the delay cycle of the ShiftReg from 1 to 8 PSoC block clocks.

The ShiftReg User Module requires that the Delay be initialized before the user module is started. The ShiftReg User Module is stopped while the register is updated.

DC and AC Electrical Characteristics

Table 1. ShiftReg AC Electrical Characteristics for the CY8C22x45 Device Family

Parameter	Typical	Limit	Units	Conditions and Notes
Maximum clock frequency	--	48[1]	MHz	4.75V < V_{DD} < 5.25V
	--	24[1]	MHz	3.0V < V_{DD} < 4.75V
Maximum input frequency	--	24[1]	MHz	4.75V < V_{DD} < 5.25V
Maximum output frequency	--	24[1]	MHz	V_{DD}=5.0V 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.

Placement

The ShiftReg consumes one digital PSoC block per 8 bits of resolution. Each block is given a symbolic name that is displayed by the device editor during and after placement. The Application Programming Interface (API) qualifies all register names with the user-assigned instance name and block name to provide direct access to the ShiftReg8 registers through the API include files. The block name is SHIFTREG8.

Parameters and Resources

Clock

The ShiftReg8 User Module is clocked by one of 16 possible sources. The Global I/O buses may be used to connect the clock input to an external pin or a clock function generated by a different PSoC block. 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.

Input

The input selects a data input source from one of the available data paths.

Output

The output may be routed to one of four Global Output buses or be disabled.

Length

Sets up the delay cycle of the ShiftReg. This parameter sets the delay cycle register value. The possible values are listed in the following table:

Parameter	Description
1	The input data is delayed to 1 PSoC block clock
2	The input data is delayed to 2 PSoC block clocks
3	The input data is delayed to 3 PSoC block clocks
4	The input data is delayed to 4 PSoC block clocks
5	The input data is delayed to 5 PSoC block clocks
6	The input data is delayed to 6 PSoC block clocks
7	The input data is delayed to 7 PSoC block clocks
8	The input data is delayed to 8 PSoC block clocks

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. These skews are more critical in the CY8C22x45 and CY8C21345 PSoC device families because of various data-path optimizations, particularly those applied to the system buses. 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 are determined from the following table.

ClockSync value	Description
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), 32 kHz, and digital PSoC blocks with SysClk-based sources. Externally generated clock sources 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 required. 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 required. In general, this use is advisable only when interrupt generation is the sole application of the Counter.

Application Programming Interface (API)

The Application Programming Interface (API) functions are provided as part of the user module to enable you 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 SHIFTREG8_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 SHIFTREG8 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.

SHIFTREG8_Start

Description:

Enables the ShiftReg8 User Module for operation. Before the module is started, the delay cycle values should be initialized.

Note An unexpected pulse may be seen on the output of the shift register when SHIFTREG8_Start is called. If this pulse is unwanted, call SHIFTREG8_WriteRegister(0x00) before calling Start.

C Prototype:

```
void  SHIFTREG8_Start(void)
```

Assembler:

```
lcall  SHIFTREG8_Start
```

Parameters:

None

Return Value:

None

Side Effects:

When the ShiftReg8 is started, a seed value written to the Seed register is not latched into the Shift register.

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

SHIFTREG8_Stop

Description:

Disables the ShiftReg8 User Module.

C Prototype:

```
void  SHIFTREG8_Stop(void)
```

Assembler:

```
lcall  SHIFTREG8_Stop
```

Parameters:

None

Return Value:

None

Side Effects:

Writing the seed value into the Seed register latches the seed value into the Shift register.

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

SHIFTREG8_SetLength

Description:

Loads the delay cycle register (DR1) with the delay cycle value. The ShiftReg User Module is stopped when the register is updated.

C Prototype:

```
void  SHIFTREG8_SetLength(BYTE bLength)
```

Assembler:

```
mov   A, [bLength]
lcall  SHIFTREG8_SetLength
```

Parameters:

bLength: 8-bit delay cycle value. The delay cycle can be set in a range of 1-8. For example, If bit0 is set, then the length is 1. If bit1 is set, then the length is 2, and so on. This length value is passed through symbolic names listed in the parameter table. These symbolic names have the form: ShiftReg8_LENGTH_x, where ‘x’ is the length (1, 2, 3, and so on.)

Symbolic name	Value	Description
SHIFTREG8_LENGTH_1	1	The input data is delayed to 1 PSoC block clock
SHIFTREG8_LENGTH_2	2	The input data is delayed to 2 PSoC block clocks
SHIFTREG8_LENGTH_3	3	The input data is delayed to 3 PSoC block clocks
SHIFTREG8_LENGTH_4	4	The input data is delayed to 4 PSoC block clocks
SHIFTREG8_LENGTH_5	5	The input data is delayed to 5 PSoC block clocks
SHIFTREG8_LENGTH_6	6	The input data is delayed to 6 PSoC block clocks
SHIFTREG8_LENGTH_7	7	The input data is delayed to 7 PSoC block clocks
SHIFTREG8_LENGTH_8	8	The input data is delayed to 8 PSoC block clocks

Return Value:

None

Side Effects:

The ShiftReg User Module is stopped while the register is updated.

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

SHIFTREG8_bReadRegister

Description:

This API reads the DR0 register, which transfers the LFSR value into DR2. This API then reads and returns the value in DR2. The shift register must be stopped before calling this API.

C Prototype:

```
BYTE  ShiftReg8_bReadRegister (void)
```

Assembler:

```
lcall  ShiftReg8_bReadRegister
```

Parameters:

None

Return Value:

This function returns a byte which is the contents of the shift register.

Side Effects:

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

SHIFTREG8_WriteRegister

Description:

This API writes the value passed into the DR2 register (the value is also simultaneously transferred into the LFSR). The shift register must be stopped before calling this API.

C Prototype:

```
void  ShiftReg8_WriteRegister(bValue)
```

Assembler:

```
mov A, [bValue]
lcall  ShiftReg8_WriteRegister
```

Parameters:

bValue: This is the value that is written into the shift register.

Return Value:

None

Side Effects:

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

Sample Firmware Source Code

In the following examples, the correspondence between the C and assembly code is simple and direct. The values shown for period and compare value are each “off-by-1” from the cardinal values because the registers are zero-based; that is, zero is the terminal count in their down-count cycle. Passing a simple one byte parameter in the A register rather than on the stack is a performance optimization used by both the assembler and C compiler for user module APIs. The C compiler employs this mechanism for “INT” types instead of pushing the argument on the stack when it sees the #pragma fastcall declarations in the SHIFTREG8.h file.

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

```
;
;  This sample shows how SHIFTREG8 User Module shifts the it's input signal.
;
; OVERVIEW:
;
;  This sample code has two User Modules. First User Module is PWM that sets the input
;  signal for SHIFTREG8 User Module. The input signal also can be routed to any pin.
;  In this example the PWM output (SHIFTREG8 input) is routed to P0[4] through
;  Row_0_Output_0.
;  The pin P0[4] has the 33% duty cycle output pulse with a frequency of 31,25 kHz.
;
;  The second User Module is SHIFTREG8 which shifts it's input signal (PWM output) to
;  eight PSoC block clocks. SHIFTREG8 input connects to PWM output through
; Row_0_Output_0.
;  The SHIFTREG8 output can be routed to any pin.
;  In this example the SHIFTREG8 output is routed to P1[3] through Row_0_Output_3.
;  The pin P0[4] has the 33% duty cycle output pulse with a frequency of 31,25 kHz
;  but this signal is shifted to eight PSoC block clocks (or 5,33 usec) relative
;  to SHIFTREG8 input.
;
;The following changes need to be made to the default settings in the Device Editor:
;
;   1. Select PWM8 user module.
;   2. The User Module will occupy the space in dedicated system resources.
;   3. Rename User Module's instance name to PWM8.
;   4. Set PWM8's Clock Parameter to VC1.
;   5. Set PWM8's Enable Parameter to High.
;   6. Set PWM8's CompareOut Parameter to Row_0_Output_0.
;   7. Set PWM8's CompareType Parameter to Less Than Or Equal.
;   8. Set PWM8's ClockSync Parameter to SyncSysClk.
;   9. Click on Row_0_Output_0 and connect Row_0_Output_0 to GlobalOutEven_4.
;   10.Select GlobalOutEven_4 for P0[4] in the Pinout.
;   11. Select SHIFTREG8 user module.
;   12. The User Module will occupy the space in dedicated system resources.
;   13. Rename User Module's instance name to SHIFTREG8.
;   14. Set SHIFTREG8's Clock Parameter to VC1.
;   15. Set SHIFTREG8's Input Parameter to Row_0_Output_0.
;   16. Set SHIFTREG8's Output Parameter to Row_0_Output_3.
;   17. Set SHIFTREG8's ClockSync Parameter to Sync to SysClk.
;   18. Click on Row_0_Output_3 and connect Row_0_Output_3 to GlobalOutOdd_3.
;   19. Select GlobalOutOdd_3 for P1[3] in the Pinout.
;   20. Generate the project (Ctrl + F6).
;
; CONFIGURATION DETAILS:
```

```
;
;  1. The clock selected should be 48 times the required period.
;  2. The UM's instance name must be shortened to PWM8.
;
; PROJECT SETTINGS:
;
;     IMO setting (SysClk)  = 24MHz     System clock is set to 24MHz
;     VC1=SysClk/1  = 16 (default)
;
; USER MODULE PARAMETER SETTINGS:
;
;  ------------------------------------------------------------------------------
;  UM         Parameter         Value                 Comments
;  ------------------------------------------------------------------------------
;  PWM8       Name              PWM8                  UM's instance name
;             Clock             VC1
;             Enable            High
;             CompareOut        Row_0_Output_0
;             TerminalCountOut  None
;             Period            47
;             PulseWidth        15
;             CompareType       Less Than Or Equal
;             InterruptType     Terminal Count
;             ClockSync         Sync to SysClk
;
;  SHIFTREG8  Name              SHIFTREG8             UM's instance name
;             Clock             VC1
;             Length            1                     The Code changes it.
;             Input             Row_0_Output_0
;             Output            Row_0_Output_3
;             ClockSync         Sync to SysClk
;             InvertInput       Normal
;
;  ------------------------------------------------------------------------------

;-----------------------------------------------------------------------------
; Assembly Code begins here
;-----------------------------------------------------------------------------

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           ; Uncomment this line to enable Global Interrupts
    lcall PWM8_Start           ; start the PWM8 - counter will start to

    mov   A, SHIFTREG8_LENGTH_8 ; load the ShiftReg8 Delay cycle value
    lcall SHIFTREG8_SetLength   ; delay 8 PSoC block clocks
    lcall SHIFTREG8_Start       ;start the ShiftReg8
    ; Insert your main assembly code here.
.terminate:
    jmp .terminate
```

The same code in C is:

```
//
//  This sample shows how SHIFTREG8 User Module shifts the it's input signal.
//
// OVERVIEW:
//
//  This sample code has two User Modules. First User Module is PWM that sets the input
//  signal for SHIFTREG8 User Module. The input signal also can be routed to any pin.
//  In this example the PWM output (SHIFTREG8 input) is routed to P0[4] through
// Row_0_Output_0.
//  The pin P0[4] has the 33% duty cycle output pulse with a frequency of 31,25 kHz.
//
//  The second User Module is SHIFTREG8 which shifts it's input signal (PWM output) to
//  eight PSoC block clocks. SHIFTREG8 input connects to PWM output through
// Row_0_Output_0.
//  The SHIFTREG8 output can be routed to any pin.
//  In this example the SHIFTREG8 output is routed to P1[3] through Row_0_Output_3.
//  The pin P0[4] has the 33% duty cycle output pulse with a frequency of 31,25 kHz
//  but this signal is shifted to eight PSoC block clocks (or 5,33 usec) relative
//  to SHIFTREG8 input.
//
//The following changes need to be made to the default settings in the Device Editor:
//
//   1. Select PWM8 user module.
//   2. The User Module will occupy the space in dedicated system resources.
//   3. Rename User Module's instance name to PWM8.
//   4. Set PWM8's Clock Parameter to VC1.
//   5. Set PWM8's Enable Parameter to High.
//   6. Set PWM8's CompareOut Parameter to Row_0_Output_0.
//   7. Set PWM8's CompareType Parameter to Less Than Or Equal.
//   8. Set PWM8's ClockSync Parameter to SyncSysClk.
//   9. Click on Row_0_Output_0 and connect Row_0_Output_0 to GlobalOutEven_4.
//   10.Select GlobalOutEven_4 for P0[4] in the Pinout.
//   11. Select SHIFTREG8 user module.
//   12. The User Module will occupy the space in dedicated system resources.
//   13. Rename User Module's instance name to SHIFTREG8.
//   14. Set SHIFTREG8's Clock Parameter to VC1.
//   15. Set SHIFTREG8's Input Parameter to Row_0_Output_0.
//   16. Set SHIFTREG8's Output Parameter to Row_0_Output_3.
//   17. Set SHIFTREG8's ClockSync Parameter to Sync to SysClk.
//   18. Click on Row_0_Output_3 and connect Row_0_Output_3 to GlobalOutOdd_3.
//   19. Select GlobalOutOdd_3 for P1[3] in the Pinout.
//   20. Generate the project (Ctrl + F6).
//
// CONFIGURATION DETAILS:
//
//  1. The clock selected should be 48 times the required period.
//  2. The UM's instance name must be shortened to PWM8.
//
// PROJECT SETTINGS:
//     IMO setting (SysClk)  = 24MHz     System clock is set to 24MHz
//     VC1=SysClk/1  = 16 (default)
//
// USER MODULE PARAMETER SETTINGS:
//
```

```
//  ------------------------------------------------------------------------------
//  UM          Parameter          Value                  Comments
//  ------------------------------------------------------------------------------
//  PWM8        Name               PWM8                   UM's instance name
//              Clock              VC1    //              Enable            High
//              CompareOut         Row_0_Output_0
//              TerminalCountOut   None
//              Period             47
//              PulseWidth         15
//              CompareType        Less Than Or Equal
//              InterruptType      Terminal Count
//              ClockSync          Sync to SysClk
//
//  SHIFTREG8   Name               SHIFTREG8              UM's instance name
//              Clock              VC1
//              Length             1                      The Code changes it.
//              Input              Row_0_Output_0
//              Output             Row_0_Output_3
//              ClockSync          Sync to SysClk
//              InvertInput        Normal
//
//  ------------------------------------------------------------------------------

/* Code begins here */

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

void main(void)
{
    // M8C_EnableGInt ;             // Uncomment this line to enable Global Interrupts
    PWM8_Start();                   // Start the PWM8!
      SHIFTREG8_SetLength(SHIFTREG8_LENGTH_8); // Load the ShiftReg8 Delay cycle value
    SHIFTREG8_Start();                        // Start the ShiftReg8
    // Insert your main routine code here.
}
```

Configuration Registers

The ShiftReg8 User Module is personalized and parameterized through the registers. The following tables show the register 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 2. SHIFTREG8_FUNC_REG

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

Table 3. SHIFTREG8_INPUT_REG

Bit	7	6	5	4	3	2	1	0
Value	Input[3:0]				Clock[3:0]			

Input selects the data input from one of 16 sources. Clock selects the clock input from one of 16 sources. Both parameters are set in the Device Editor.

Table 4. SHIFTREG8_OUTPUT_REG

Bit	7	6	5	4	3	2	1	0
Value	AuxClk		0	0	0	0	Output	

The user module "ClockSync" parameter in the Device Editor determines the value of the AuxClk bits. Output selects output from one of four global buses. This parameter is set in the Device Editor.

Table 5. SHIFTREG8_DELAY_REG

Bit	7	6	5	4	3	2	1	0
Value	Delay Cycle Number							

Delay Cycle Number is the ShiftReg8 Delay register. It is modified using the ShiftReg8 API. The Delay register is write-only. The highest bit with one in Delay register determines the real delay cycles.

Table 6. SHIFTREG8_CONTROL0_REG

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

Enable bit indicates that the ShiftReg8 is enabled when set. It is modified using the ShiftReg8 API.

Version History

Version	Originator	Description
1.0	DHA	Initial version
1.0.b	DHA	The following updates were done for this user module datasheet: 1. Renamed InputBitStream and OutputBitStream parameters. 2. Updated SHIFTREG8_Start API description.

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.

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.