8-Bit Software Serial Transmitter Datasheet TX8SWV 1.2

Copyright © 2008-2013 Cypress Semiconductor Corporation. All Rights Reserved.

Resources	PSoC Blocks			API Memory (Bytes)		Pins (per External I/O and Clock)
	Digital	Analog CT	Analog SC	Flash	RAM	
CY8C29/27/24/21xxx, CY8C20x34, CY8CLED02/04/08/16, CY8CLED0xD, CY8CLED0xG, CY8C20x66, CY8C20x36, CY8C20336AN, CY8C20436AN, CY8C20636AN, CY8C20XX6L,.CY8C20xx6AS, CY8C20x46, CY8C20x96, CY7C604xx, CY7C643xx, CYONS2xxx, CYONSTB2010, CYONSTB2011, CYONSFN2010-BFXC, CYONSCN2024-BFXC, CYONSCN2028-BFXC, CYONSCN2020-BFXC, CYONSKN2033-BFXC, CYONSKN2035-BFXC, CYONSKN2030-BFXC, CY8CTMG2xx, CY8CTMA30xx, CY8C28x45, CY8C21x12, CYONSTN2040,CY8CTMA140, CY8C20xx7/7S, CYRF89x35, CY8C20065, CY8C24x93, CY7C69xxx						
	0	0	0	383	2	1

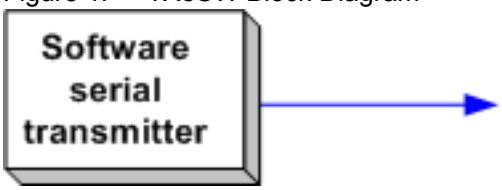
For one or more fully configured, functional example projects that use this user module go to www.cypress.com/psocexampleprojects

Features and Overview

- 7/8-bit software serial transmitter
- Data framing consists of start, optional parity, and one or two stop bits
- RS-232 serial-data compatible format with optional parity

The TX8SW User Module is an 7- or 8-bit RS-232 data-format compliant serial transmitter. The data transmitted is framed with a leading start bit and a final one or two stop bits. Transmitter firmware is used to start and stop device and control transmission of complex structures like strings, HEX value representations, and so on.

Figure 1. TX8SW Block Diagram



Functional Description

The TX8SW User Module is an RS-232 data-format compliant software serial transmitter. The transmitter supports 115200, 57600, 38400, 19200, 9600, 4800, 2400, and 1200 bps transfer speed with or without parity. It does not use any PSoC blocks and can be used when all digital resources are occupied. Transmitter firmware is used to start and stop the user module and to write data to the dedicated pin. During transmission, the user module masks interrupt and change the CPU clock frequency to assure the proper timing of signals. Interrupts and CPU frequency are restored after the byte is sent. The SysClk does not change and change of the CPU clock does not affect the hardware resource clock during transmission.

For all baud rates used, the CPU clock is less than 3 MHz. All supported baud rates are available regardless of the current CPU clock. IMO settings are detected using conditional compiler directives. The TX8SW User Module works with all IMO settings (24/12/6MHz). All CPU clock changes and previous value restores are completed by TX8SW routines and do not require manual frequency changes. The following tables show the CPU clock and CPU cycles used for different combinations of baud rate and SysClock:

Table 1. CPU Clock Used for Byte Transmit Time

SysClock	Baud Rate							
	115200	57600	38400	19200	9600	4800	2400	1200
24 MHz (Note 1)	3 MHz	1.5 MHz	1.5 MHz	0.75 MHz	0.75 MHz	187 kHz	93 kHz	93 kHz
12 MHz (Note 2)	3 MHz	1.5 MHz	1.5 MHz	0.75 MHz	375 kHz	375 kHz	93 kHz	46 kHz
6 MHz (Note 3)	3 MHz	1.5 MHz	1.5 MHz	0.75 MHz	375 kHz	187 kHz	187 kHz	46 kHz

Table 2. CPU Cycles per One Bit Transfer

SysClock	Baud Rate							
	115200	57600	38400	19200	9600	4800	2400	1200
24 MHz (Note 1)	26	26	39	39	78	39	39	78
12 MHz (Note 2)	26	26	39	39	39	78	39	39
6 MHz (Note 3)	26	26	39	39	39	39	78	39

Note

1. Not available in CY8C20x34 parts.
2. Not available in CY7C64215, CY8C24x94, CY8C27xxx, CY8C29xxx, or CY8CLED04/08/16 parts.
3. Not available in CY7C64215, CY8C24x94, CY8C27xxx or CY8CLED04/08 parts.

During transmit time interrupts are masked. The following table shows the delay introduced before ISR for each baud rate selection:

Table 3. Interrupt Latency Table

Baud Rate	Min (Note 1)	Typ2 (Note 2)	Max3(Note 3)	Units
115200	69.4	78.1	95.5	µs
57600	139	156	191	µs
38400	208	234	286	µs
19200	417	469	573	µs
9600	833	938	1145	µs
4800	1.66	1.88	2.29	ms
2400	3.33	3.75	4.58	ms
1200	6.67	7.50	9.17	ms

Note

1. For 7-bit data format, one stop bit and no parity.
2. For 8-bit data format, one stop bit and no parity.
3. For 8-bit data format, two stop bits and parity bit.

DC and AC Electrical Characteristics

Table 4. DC and AC Electrical Characteristics

Parameter	Value	Units	Conditions and Notes (See Note)
F115200	115385	baud	When SysCPU is 24/12/6 MHz exactly
F57600	57692	baud	When SysCPU is 24/12/6 MHz exactly
F38400	38462	baud	When SysCPU is 24/12/6 MHz exactly
F19200	19231	baud	When SysCPU is 24/12/6 MHz exactly
F9600	9615	baud	When SysCPU is 24/12/6 MHz exactly
F4800	4808	baud	When SysCPU is 24/12/6 MHz exactly
F2400	2404	baud	When SysCPU is 24/12/6 MHz exactly
F1200	1202	baud	When SysCPU is 24/12/6 MHz exactly

Note Information about IMO tolerances can be found in the datasheet for your chosen device.

Parameters and Resources

Port

Select the port to send data out.

Pin

Select which pin of the selected port to send data out.

BaudRate

Select baud rate from the list. Possible choices are 115200, 57600, 38400, 19200, 9600, 4800, 2400 and 1200 bits per second.

Parity

Select parity type from the list. The choices are Odd, Even, and None.

StopBits

Select the number of stop bits from the list. One or two stop bits are supported.

DataBits

Select the number of data bits from the list. Seven or eight data bits are supported.

Placement

The TX8SW User Module does not require any digital or analog PSoC blocks. There are no placement restrictions.

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.

Each time a user module is placed, it is assigned an instance name. By default, PSoC Designer assigns TX8SW_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 TX8SW for simplicity.

** 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 prior to 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 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.

The API routines allow programmatic control of the TX8SW User Module. The following tables list the low level and high level TX8SW supplied API functions.

Table 5. Low Level TX8SW API

Function	Description
void TX8SW_Start(void)	Enables user module.
void TX8SW_Stop(void)	Stops user module.
void TX8SW_SendData(BYTE bTxData)	Sends one byte bTxData via TX pin.

Table 6. High Level TX8SW API

Function	Description
void TX8SW_PutString(BYTE * szStr)	Sends RAM located string szStr out through TX pin.
void TX8SW_CPutString(const BYTE * azStr)	Sends ROM located string azStr out through TX pin.
void TX8SW_PutChar(BYTE bData)	Sends one byte via TX pin.
void TX8SW_Write(BYTE * aStr, BYTE bCnt)	Sends RAM located data pointed by aStr with bCnt length.
void TX8SW_CWrite(const BYTE * aStr, INT iCnt)	Sends ROM located data pointed by aStr with iCnt length.
void TX8SW_PutSHexByte(BYTE bValue)	Sends a bValue BYTE in Hex representation (two characters) via TX pin.
void TX8SW_PutSHexInt(INT iValue)	Sends a iValue INT in Hex representation (four characters) via TX pin.
void TX8SW_PutCRLF(void)	Sends a CR and LF characters.

TX8SW_Start

Description:

Enables the TX8SW module, sets the output pin to strong mode and high (serial inactive) state.

C Prototype:

```
void  TX8SW_Start(void)
```

Assembly:

```
lcall  TX8SW_Start
```

Parameters:

None

Return Value:

None

Side Effects:

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

TX8SW_Stop

Description:

Stops the TX8SW module, sets the output pin to high impedance (hi-z analog) mode.

C Prototype:

```
void  TX8SW_Stop(void)
```

Assembly:

```
lcall  TX8SW_Stop
```

Parameters:

None

Return Value:

None

Side Effects:

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

TX8SW_SendData

Description:

Sends one byte bTxData via the TX pin.

C Prototype:

```
void  TX8SW_SendData(BYTE bTxData)
```

Assembly:

```
mov   A, bTxData
lcall  TX8SW_SendData
```

Parameters:

bTxData: data to be sent, passed in the Accumulator.

Return Value:

None

Side Effects:

This function does not return until the byte is transmitted.

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

TX8SW_PutString

Description:

Sends a RAM located null terminated string, szRamString, out through the TX pin.

C Prototype:

```
void TX8SW_PutString(BYTE * szRamString)
```

Assembler:

```
mov  A, >szRamString    ; Load MSB part of pointer to RAM based null
                        ; terminated string.
mov  X, <szRamString    ; Load LSB part of pointer to RAM based null
                        ; terminated string.
lcall TX8SW_PutString    ; Call function to send string out TX8SW port
```

Parameters:

BYTE * szRamString: A pointer to the string to be sent to the TX8SW pin. The MSB is passed in the Accumulator and the LSB is passed in the X register.

Return Value:

None

Side Effects:

This function does not return until the last character is sent out.

See Note ** at the beginning of the API section. Currently, only the IDX_PP page pointer register is modified.

TX8SW_CPutString

Description:

Sends a constant (ROM), null terminated string out the TX port.

C Prototype:

```
void TX8SW_CPutString(const BYTE * szRomString)
```

Assembler:

```
mov  A, >szRomString  ; Load MSB part of pointer to ROM based null
                      ; terminated string.
mov  X, <szRomString  ; Load LSB part of pointer to ROM based null
                      ; terminated string.
lcall TX8SW_CPutString ; Call function to send constant string out
                      ; TX8SW port
```

Parameters:

const BYTE * szRomString: A pointer to a string to be sent to the TX8SW pin. The MSB of the string pointer is passed in the Accumulator and the LSB of the pointer is passed in the X register.

Return Value:

None

Side Effects:

This function does not return until the last character is sent out.

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

TX8SW_PutChar

Description:

Sends one byte via the TX pin.

C Prototype:

```
void TX8SW_PutChar(BYTE cData)
```

Assembler:

```
mov  A, cData            ; Load ASCII character in A
lcall TX8SW_PutChar       ; Call function to send single character to
                         ; TX8SW port.
```

Parameters:

BYTE cData: The character to be sent to the TX8SW pin. The data is passed in the Accumulator.

Return Value:

None

Side Effects:

This function does not return until the data is sent out.

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

TX8SW_Write

Description:

Sends RAM located data pointed to by sRamData with bCount length.

C Prototype:

```
void TX8SW_Write(BYTE * sRamData, BYTE bCount)
```

Assembler:

```
mov   A, bCount           ; Load string/array count
push  A
mov   A, >sRamData        ; Load MSB part of pointer to RAM string
push  A
mov   A, <sRamData        ; Load LSB part of pointer to RAM string
push  A
lcall  TX8SW_Write         ; Make call to function
add   SP, 253             ; Reset stack pointer to original position
```

Parameters:

BYTE * szRomString: Pointer to the buffer to be sent to the TX8SW pin. BYTE bCount: Number of bytes to be sent to the TX8SW pin.

Return Value:

None

Side Effects:

This function does not return until the last byte is sent out.

See Note ** at the beginning of the API section. Currently, only the IDX_PP page pointer register is modified.

TX8SW_CWrite

Description:

Sends ROM located data pointed to by sRomData with iCount length.

C Prototype:

```
void TX8SW_CWrite(const BYTE * sRomData, INT iCount)
```

Assembler:

```
mov   A, >iCount          ; Load MSB of count
push  A
mov   A, <iCount          ; Load LSB of count
push  A
mov   A, >sRomData        ; Load MSB part of pointer to ROM string
push  A
mov   A, <sRomData        ; Load LSB part of pointer to ROM string
push  A
lcall  TX8SW_CWrite        ; Make call to function
add   SP, 252             ; Reset stack pointer to original position
```

Parameters:

BYTE * szRomData: A pointer to the buffer to be sent to the TX8SW pin.

int iCount: Number of bytes to be sent to the TX8SW pin.

Return Value:

None

Side Effects:

This function does not return until the last character is sent out.

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

TX8SW_PutSHexByte

Description:

Sends a byte in Hex representation (two characters) via TX8SW pin.

C Prototype:

```
void TX8SW_PutSHexByte(BYTE bData)
```

Assembler:

```
mov  A, bData             ; Load data to be sent to TX8SW
lcall TX8SW_PutSHexByte    ; Call function
```

Parameters:

BYTE bData: The byte to be converted to an ASCII string.

Return Value:

None

Side Effects:

This function does not return until both characters are sent out.

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

TX8SW_PutSHexInt

Description:

Sends an integer value in Hex representation (four characters) via the TX8SW pin.

C Prototype:

```
void TX8SW_PutSHexInt(INT iData)
```

Assembler:

```
mov  A, <iData          ; Load LSB in A
mov  X, >iData          ; Load MSB in X
lcall TX8SW_PutSHexInt   ; Function call
```

Parameters:

int iData: Integer to be converted to ASCII string and sent via TX8SW pin.

Return Value:

None

Side Effects:

This function does not return until all four characters are sent out.

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

TX8SW_PutCRLF

Description:

This function sends a carriage return (0x0D) and a line feed (0x0A) character out the TX8SW port.

C Prototype:

```
void TX8SW_PutCRLF(void)
```

Assembler:

```
lcall TX8SW_PutCRLF      ; Send a carriage return and line feed out TX8SW pin
```

Parameters:

None

Return Value:

None

Side Effects:

This function does not return until the two characters are sent out.

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

Sample Firmware Source Code

The following is an example of a function that transmits a constant zero-terminated string using the TX8SW User Module. The code sample in assembly language is:

```
include "m8c.inc"       ; part specific constants and macros
include "memory.inc"    ; Constants & macros for1 SMM/LMM and Compiler
include "PSoCAPI.inc"   ; PSoC API definitions for all User Modules

export _main

_main:

.LITERAL
    MyString: ASCIZ "Sending_Message"
.ENDLITERAL

;TX8SW module enabling
    call TX8SW_Start

;Put null-termined string through TX pin
    mov A, >MyString
    mov X, <MyString
    call TX8SW_CPutString
```

Here is the same code written in C:

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


void main(void)
{
    TX8SW_Start();
TX8SW_CPutString("Sending_Message\0");
}
```

Version History

Version	Originator	Description
1.2	DHA	1. Modified Wizard to prevent used pin selection 2. Modified Start API to improve memory management.
1.2.b	DHA	Added CYRF89x35 device support.

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 © 2008-2013 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.