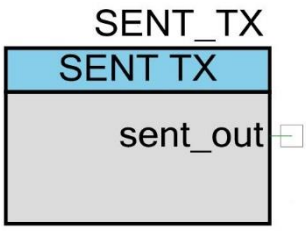
Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

Single Edge Nibble Transmission (SENT_TX)
1.0



Features

- Compliant with SAE J2716 APR2016 (Issued 2007-04, Revised 2016-04) without any serial message formats
- Selectable clock tick period (3 to 90 µs)
- Optional Pause Pulse for constant frame length of 282 clock ticks
- Supports both Legacy CRC & New CRC implementations
- Selectable Transmit Mode

General Description

The Single Edge Nibble Transmission (SENT) encoding scheme is intended for automotive applications as a simple low cost alternative to CAN or LIN. It transmits sensor data from a sensor module to an Engine Control Unit (ECU).

SENT is a one wire (apart from Supply & Ground), unidirectional communications scheme from a sensor/transmitting device to a controller/receiving device, which does not include a coordination signal from the controller/receiving device. Transmission occurs independently of any action of the receiver module; that is, the transmission shall not require a synchronization signal from the receiver module. The sensor signal is transmitted as a series of pulses with data encoded as falling to falling edge periods.

Assumptions used to design the encoding scheme:

- Actual Transmission time may be dependent on the data values being sent and the actual clock variation of the Component.
- Message pulse order is fixed & always containing 6 data nibbles.
- Maximum allowed clock tick time variation is <= ±20%.
- Transmission time for the longest data message and max transmitter clock variation is less than 1.0 millisecond at 3 microsecond clock tick time and 6 data nibbles.

Note that this SENT_TX Component defines only the physical and data link layers of SAE J2716 APR2016 (Issued 2007-04, Revised 2016-04) protocol implementation. You can add your stack on top of this based on your application. At the end, top level stack you will need to provide

necessary values for Status/Communication, Data, and pulses, which will be converted into pulse widths and sent out by this SENT_TX Component.

When to Use a SENT_TX Component

Use SENT_TX Component as a simple one-wire low cost alternative to CAN or LIN Components in Automotive applications.

Definitions

- **SENT**: Single Edge Nibble Transmission
- **CRC***:* Cyclic Redundancy Check
- **ECU**: Engine Control Unit
- **CAN**: Controller Area Network
- **LIN**: Local Interconnect Network
- **SAE**: Society of Automotive Engineers

Quick Start

1. Drag a SENT_TX Component schematic macro from the Component Catalog *Cypress/Communications* folder onto your schematic. The placed instance takes the name SENT_TX_1). The schematic macro has default configuration settings for the SENT_TX and cy_pin Components.
2. Double-click to open the SENT_TX Configure dialog
3. Set the required Clock Period, CRC method, Transmit mode, Enable, and if required, the Pause Pulse.
4. Select the pin in the Pin Editor.
5. Set the highest priority level in the Interrupt Editor for the SENT_TX interrupt.
6. In *main.c*, define the 7-byte prepared data array, where the first byte is the status nibble, and the next six bytes are data nibbles.
7. Build the project in order to verify the correctness of your design and generate the configuration data for the SENT_TX_1 instance.
8. In *main.c*, initialize the peripheral and start the application.

```
uint8 data[7] = {0, 4, 10, 12, 4, 10, 12};

CyGlobalIntEnable; /* Enable global interrupts. */
SENT_TX_1_Start(data);
```

9. Build and program the device.

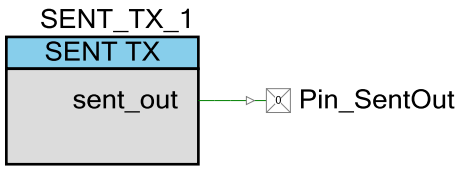
Input/Output Connections

This section describes the various input and output connections for the SENT_TX Component. An asterisk (*) in the following list indicates that it may not be shown on the Component symbol for the conditions listed in the description of that I/O.

Terminal Name	I/O Type	Description
sent_out	Digital Output	SENT_TX output

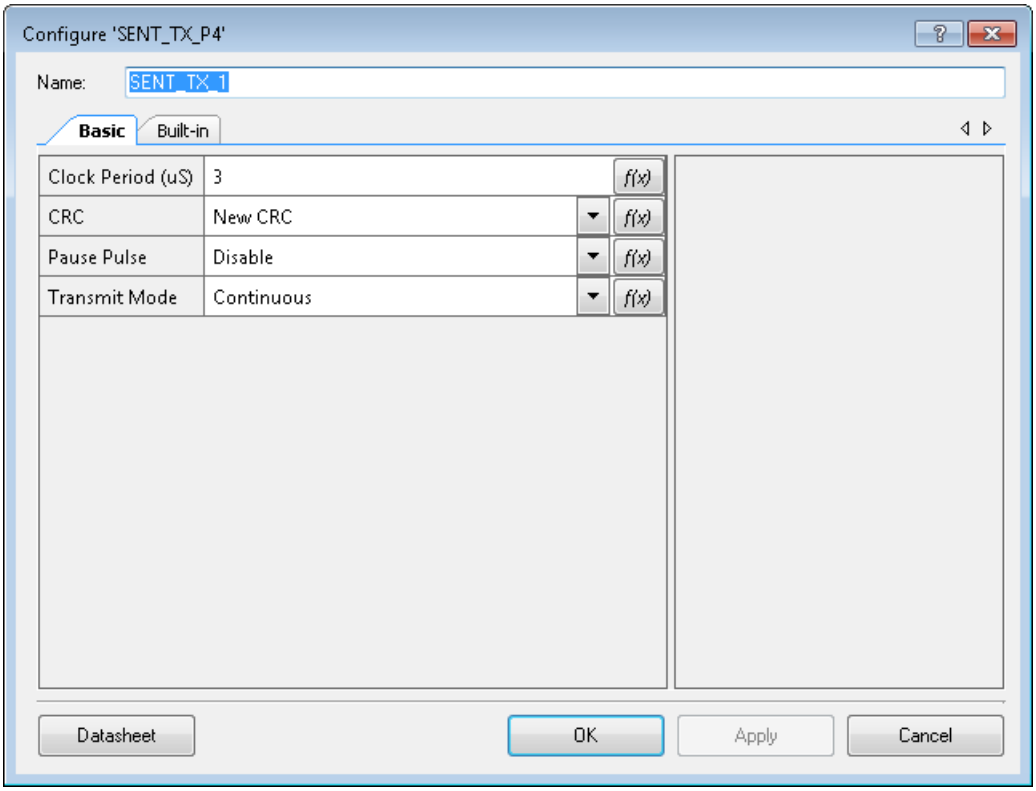
Schematic Macro Information

The Component Catalog provides a schematic macro for the SENT_TX Component with sent_out terminal connected to an Output Pin Component. The schematic macro has the default configuration settings for the SENT_TX and cy_pin Components.



Component Parameters

The SENT_TX Component Configure dialog allows you to edit the configuration parameters for the Component instance.



General Tab

This tab contains the Component parameters used in the general peripheral initialization settings.

Parameter Name	Description
Clock Period (uS)	Selects the time period of one Clock Tick in microseconds. Valid range: 3 - 90 uS with 0.1 uS resolution Note the input is a float value. This value will be rounded to one character after the decimal point during build and compile time
CRC	The two options of the drop-down box: • New CRC • Legacy CRC The selected CRC method will be used in the Component. Refer to J2716-APR2016 spec for more details about CRC methods implementation.
Pause Pulse	The drop-down box selects whether to add the optional Pause Pulse. With enabled Pause Pulse option all messages are aligned to constant 282 ticks length

Parameter Name	Description
Transmit Mode	The drop-down box selects the SENT_TX mode: • Continuous – If no new data is ready at the end of transmitting one full frame, the Component will be sending the same data. • Single – If no new data is ready, the output line will be in High state, until new data is loaded into the transmit buffer. The new data can be loaded into transmit buffer using the SENT_TX_UpdateData() function.

Application Programming Interface

By default, PSoC Creator assigns the instance name **SENT_TX_1** to the first instance of a component in a given design. You can rename it to any unique value that follows the syntactic rules for identifiers. The instance name becomes the prefix of every global function name, variable, and constant symbol. For readability, the instance name used in the following table is **SENT_TX**.

General API

General API functions are used for run-time configuration of the component during active power mode.

These include, initializing, starting, stopping, reading from registers and writing to registers.

- void SENT_TX_Init (void)
- void SENT_TX_Enable (void)
- void SENT_TX_Start (uint8 const *data)
- void SENT_TX_Stop (void)
- void SENT_TX_UpdateData (uint8 const *data)
- uint8 SENT_TX_GetStatus (void)

void SENT_TX_Init (void)

Initialize/Restore default SENT_TX configuration.

void SENT_TX_Enable (void)[inline]

Enables the PWM that is used in SENT_TX to generate pulses.

void SENT_TX_Start (uint8 const *data)

Initializes SENT_TX with default customizer values when called the first time and enables SENT_TX. For subsequent calls the configuration is left unchanged and only the component is enabled.

Parameters:

data	The pointer to the uint8 array of length SENT_TX_USER_NIBBLES, where the 1st byte is the status nibble and the next 6 bytes are data nibbles.

Global Variables

SENT_TX_initVar

void SENT_TX_Stop (void)[inline]

Disables SENT_TX.

void SENT_TX_UpdateData (uint8 const *data)

Writes the frame with new data into the transfer buffer.

Parameters:

data	The pointer to the uint8 array of length SENT_TX_USER_NIBBLES, where the first byte is the status nibble and the next six bytes are data nibbles.

uint8 SENT_TX_GetStatus (void)[inline]

Returns the state of SENT readiness to send new data message.

Returns:

Status of readiness to send new data, see SENT TX status definitions.

Low Power Functions

Low power functions perform the necessary configurations to the components to prepare it for entering low power modes.

These API functions must be used if the intent is to put the chip to sleep, then to continue the component operation when it comes back to active power mode.

- uint32 SENT_TX_Sleep (void)
- void SENT_TX_Wakeup (void)

uint32 SENT_TX_Sleep (void)[inline]

Stops the component operation and saves the user configuration.

Before call this function, ensure that SENT is ready to sleep by checking the SENT_TX_status flag. For this purpose, use the SENT_TX_GetStatus() function.

Returns:

CYRET_SUCCESS the SENT went into sleep.
CYRET_CANCELED the SENT did not go into sleep, because SENT still sending previous data, SENT_TX_status = SENT_TX_BUSY.

void SENT_TX_Wakeup (void)[inline]

Restores the user configuration and restores the enable state.

Global Variables

Global variables used in the component.

The following global variables are used in the component.

- uint8 SENT_TX_initVar
- uint8 SENT_TX_status

uint8 SENT_TX_initVar

The global variable that indicates the initialization status of the SENT_TX component

uint8 SENT_TX_status

The variable that indicates a component status

API Constants

Component API functions are designed to work with pre-defined enumeration values.

These values should be used with the API functions that reference them.

- SENT TX status definitions

SENT TX status definitions

Definitions of SENT TX status

- #define SENT_TX_READY (0U)
- #define SENT_TX_BUSY (1U)

#define SENT_TX_READY (0U)

The define to show the data is sent and the component is ready to send a new data

#define SENT_TX_BUSY (1U)

The define to show the component is sending previous data

Interrupt Service Routine

The SENT_TX Component uses interrupts generated from TCPWM IP block. The interrupt is generated on the Terminal Count (TC) event of TCPWM instance. In the interrupt handler function is prepared the next period of the TCPWM interrupt. Next period of the interrupt corresponds to next nibble pulse period. The interrupt behavior is different based upon the Transition Mode.

Note, it is highly important to set the highest interrupt priority for SENT interrupt in your application to do not have timing issue related to other interrupts. The SENT pulse period can be corrupted when the other interrupt, with higher priority, is handled and at this moment the next period of SENT_TX message should be prepared in the SENT interrupt.

Code Examples and Application Notes

This section lists the projects that demonstrate the use of the Component.

Code Examples

PSoC Creator provides access to code examples in the Find Code Example dialog. For Component-specific examples, open the dialog from the Component Catalog or an instance of the Component in a schematic. For general examples, open the dialog from the Start Page or **File** menu. As needed, use the **Filter Options** in the dialog to narrow the list of projects available to select.

Refer to the "Find Code Example" topic in the PSoC Creator Help for more information.

There are also numerous code examples that include schematics and example code available online at the Cypress Code Examples web page. Examples that use this Component include:

- CE219033 - SENT TX Basic Code Example

Application Notes

Cypress provides a number of application notes describing how PSoC can be integrated into your design. You can access the Cypress Application Notes search web page at www.cypress.com/appnotes. Component

API Memory Usage

The Component memory usage varies significantly depending on the compiler, device, number of APIs used and Component configuration. The following table provides the memory usage for all APIs available in the given Component configuration.

The measurements have been done with an associated compiler configured in Release mode with optimization set for Size. For a specific design, the map file generated by the compiler can be analyzed to determine the memory usage.

PSoC 4 (GCC)

Configuration	PSoC 4000		PSoC 4000S	
	Flash Bytes	SRAM Bytes	Flash Bytes	SRAM Bytes
Continuous Mode, Pause Pulse Disabled, Legacy CRC	964	30	1323	50
Single Mode, Pause Pulse Disabled, New CRC	1044	30	1403	50
Single Mode, Pause Pulse Enabled, New CRC	1176	32	1435	52

Configuration	PSoC 4000DS / PSoC 4200DS		PSoC 4100 / PSoC 4200	
	Flash Bytes	Flash Bytes	Flash Bytes	SRAM Bytes
Continuous Mode, Pause Pulse Disabled, Legacy CRC	924	30	1458	86
Single Mode, Pause Pulse Disabled, New CRC	1004	30	1538	86
Single Mode, Pause Pulse Enabled, New CRC	1036	32	1570	88

Configuration	PSoC 4100 M / PSoC 4200 M / SHM35x2M		PSoC 4100 S / PSoC Analog Coprocessor	
	Flash Bytes	Flash Bytes	Flash Bytes	SRAM Bytes
Continuous Mode, Pause Pulse Disabled, Legacy CRC	1677	98	1683	78
Single Mode, Pause Pulse Disabled, New CRC	1757	98	1763	78
Single Mode, Pause Pulse Enabled, New CRC	1789	100	1795	82

Configuration	PSoC 4100 BLE / PSoC 4200 BLE		PSoC 4200 L / SHM35x2L	
	Flash Bytes	Flash Bytes	Flash Bytes	SRAM Bytes
Continuous Mode, Pause Pulse Disabled, Legacy CRC	1518	98	2089	126
Single Mode, Pause Pulse Disabled, New CRC	1598	98	2169	126
Single Mode, Pause Pulse Enabled, New CRC	1630	100	2201	128

Functional Description

The SENT_TX Component functionality is based on TCPWM interrupts. The Component takes user's input data, as 7-elements array, where the first element is a status nibble and six others are data nibbles. Based on configuration selected, the Component calculates the CRC and Pause Pulse values. After all preparations, all input data and calculated values are stored into Prepare buffer. On the first SENT_TX interrupt, the data from Prepare buffer is stored into Transmit buffer. In the next SENT_TX interrupts the data from Transmit buffer is sent. After the last nibble is sent the further behavior depends on Transition mode selected.

For the Single transition mode, the Component generates one more pulse to indicate the end of message (provide falling edge for the latest nibble of message) and after the TCPWM counter is stopped. If TCPWM counter was already stopped the counter is restarted in the SENT_TX_UpdateData() function. After TCPWM counter restarts the interrupts generation and SENT_TX Component starts sending new data.

For the Continuous transition mode, the TCPWM counter is never stopped and the SENT_TX Component generates interrupts and send the data continuously.

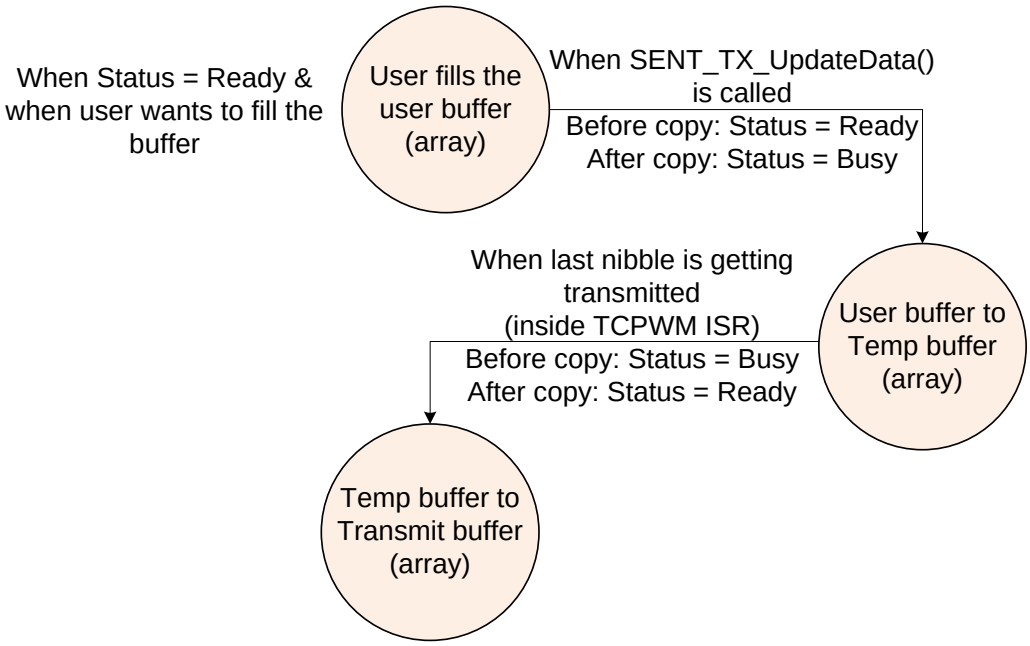
If you want to update data in run-time, it is required to check status flag. You can call SENT_TX_UpdateData() only when status is in READY state. See section below for more details.

Handling status flags

It is very important to not to corrupt the buffer when it is being used for transmission. So, Component uses a status bit (Ready & Busy) & two buffers – one temp buffer & another transmit buffer. User is expected to have another buffer called user buffer.

Ready state (0) of status signal indicates that user can fill the user data & call SENT_TX_UpdateData() function to copy this user buffer to temp buffer. Busy state (1) of status signal indicates that data from temp buffer is yet to be copied into transmit buffer which will happen when the last nibble is getting transmitted.

Below flow chart depicts the usage of status signals & when exactly data is getting from which buffer to which other buffer:



Industry Standards

The SENT_TX Component is compliant with SAE J2716 SENT (rev. 2016-04) standard.

MISRA Compliance

This section describes the MISRA-C:2004 compliance and deviations for the Component. There are two types of deviations defined:

- project deviations – deviations that are applicable for all PSoC Creator Components
- specific deviations – deviations that are applicable only for this Component

This section provides information on Component-specific deviations. Project deviations are described in the MISRA Compliance section of the *System Reference Guide* along with information on the MISRA compliance verification environment.

MISRA-C:2004 Rule	Rule Class (Required/ Advisory)	Rule Description	Description of Deviation(s)
17.4	R	Array indexing shall be the only allowed form of pointer arithmetic.	The component use array indexing that is applied to an object of pointer type to access elements of array, allocated by firmware.
19.16	R	Unrecognized preprocessing directive has been ignored because of conditional inclusion directives.	Violated to set correct supply voltages in the project.

Registers

See the Timer, Counter, PWM Counter (CNT) Registers section in the chip Technical Reference Manual (TRM) for more information about the registers.

Component Debug Window

SENT_TX Component uses TCPWM debug window.

Resources

The SENT_TX Component uses the TCPWM HW peripheral block.

DC and AC Electrical Characteristics

Refer to TCPWM datasheet for the DC and AC Characteristics

Component Changes

This section lists the major changes in the Component from the previous version.

Version	Description of Changes	Reason for Changes / Impact
1.0.a	Updated API Memory Usage section. Updated MISRA Compliance section.	Updated data for various devices. Added additional deviation description.
1.0	Initial Version	

© Cypress Semiconductor Corporation, 2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC (“Cypress”). This document, including any software or firmware included or referenced in this document (“Software”), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress’s patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical Components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage (“Unintended Uses”). A critical Component is any Component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.