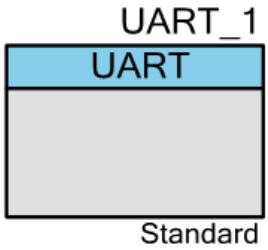
Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the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

UART (SCB_UART_PDL)
2.0



Features

- Baud Rate of up to 1 Mbps
- Detection of Framing, Parity, and Overrun Errors
- Full Duplex, TX only, and RX only modes
- 9-bit address mode with hardware address detection
- Break signal detection
- Hardware Flow control
- Smartcard and IrDA support
- DMA support

General Description

The SCB_UART_PDL Component provides asynchronous communications commonly referred to as RS232 or RS485. The Component can be configured for Full Duplex, RX only, or TX only versions. It can also be configured as a SmartCard interface, or an IrDA interface.

For most use cases, you can easily configure the UART by choosing the baud rate, parity, number of data bits, and number of start bits. The most common configuration for RS232 is often listed as “8N1,” which is shorthand for eight data bits, no parity, and one stop bit. This is the default configuration for the SCB_UART_PDL Component. Therefore, in most applications you only need to set the baud rate. A second common use for UARTs is in multi-drop RS485 networks. The SCB_UART_PDL Component supports 9-bit addressing mode with hardware address detect.

UARTs have been around a long time, so there have been many physical-layer and protocol-layer variations over time. These include, but are not limited to, RS423, DMX512, MIDI, LIN bus, legacy terminal protocols. To support the commonly used UART variations, the Component provides configuration support for the number of data bits, stop bits, parity, hardware flow control, and parity generation and detection.

The SCB_UART_PDL Component is a graphical configuration entity built on top of the cy_scb driver available in the Peripheral Driver Library (PDL). It allows schematic-based connections and hardware configuration as defined by the Component Configure dialog.

When to Use a SCB_UART_PDL Component

Use the SCB_UART_PDL Component any time a compatible asynchronous communications interface is required, especially RS232 and RS485 and other variations. You can also use the Component to create more advanced asynchronous based protocols such as Smartcard, and IrDA, or customer or industry proprietary.

Do not use a SCB_UART_PDL Component in those cases where a specific Component has already been created to address the protocol. For example, if a LIN or MIDI Component is provided, it has a specific implementation providing both hardware and protocol layer functionality. The SCB_UART_PDL is not needed in this case (subject to Component availability).

Definitions

- UART – Universal Asynchronous Receiver Transmitter commonly referred to as RS232. The UART supports following modes:
  - UART – This is the standard mode.
  - SmartCard – Transfer is similar to a UART transfer, with the addition of a negative acknowledgement (NACK) that may be sent from the receiver to the transmitter. A NACK is always ‘0’. Both transmitter and receiver may drive the same IO line, although never at the same time.
  - IrDA – the Infrared Data Association protocol adds a modulation scheme to the UART signaling. At the transmitter, bits are modulated. At the receiver, bits are demodulated. The modulation scheme uses a Return-to-Zero-Inverted (RZI) format. A bit value of ‘0’ is signaled by a short ‘1’ pulse on the line and a bit value of ‘1’ is signaled by holding the line to ‘0’.

Quick Start

1. Drag a UART (SCB) Component from the Component Catalog Cypress/Communications/UART folder onto your schematic (the placed instance takes the name UART_1).
2. Double-click to open the Configure dialog.
3. On the **Basic** tab, select the **Baud rate** at which the UART interface is expected to communicate.

   Note For successful communication, the calculated **Actual baud rate** should be within the acceptable accuracy. If this is not the case, iterate through the **Oversample** parameter values to achieve the desired baud rate.
4. On the Pin Editor, select pins for the UART interface. The choice of pins that can be used for the UART interface is limited and PSoC Creator restricts the selection.

5. Build the project in order to verify the correctness of your design. This will add the required PDL modules to the Workspace Explorer and generate the configuration data for the UART_1 instance.
6. In the *main.c* file, initialize the peripheral and start the application.

   Interrupt parameter is “Internal” (uses UART High-Level communication cy_scb driver functions; refer to Interrupt Service Routine section):

```
/* Implement ISR for UART_1 */
void UART_1_Isr(void)
{
    Cy_SCB_UART_Interrupt(UART_1_HW, &UART_1_context);
}

/* Allocate buffer */
#define BUFFER_SIZE (64UL)
uint8_t buffer[BUFFER_SIZE];

/* Initialize SCB for UART operation with GUI selected settings */
(void) Cy_SCB_UART_Init(UART_1_HW, &UART_1_config, &UART_1_context);

/* Hook interrupt service routine and enable interrupt */
Cy_SysInt_Init(&UART_1_SCB_IRQ_cfg, &UART_1_Isr);
NVIC_EnableIRQ(UART_1_SCB_IRQ_cfg.intrSrc);

/* Enable UART */
Cy_SCB_UART_Enable(UART_1_HW);

/* Transmit data */
(void) Cy_SCB_UART_Transmit(UART_1_HW, buffer, BUFFER_SIZE, &UART_1_context);
```

   Interrupt parameter is “External” (uses UART Low-Level communication cy_scb driver functions; refer to Interrupt Service Routine section):

```
char_t str[] = "Hello World/r/n";

/* Initialize SCB for UART operation with GUI selected settings */
(void) Cy_SCB_UART_Init(UART_1_HW, &UART_1_config, NULL);

/* Enable UART */
Cy_SCB_UART_Enable(UART_1_HW);

/* Put data into TX FIFO to be transferred */
Cy_SCB_UART_PutString(UART_1_HW, str);
```

7. Build and program the device.

Input/Output Connections

This section describes the various input and output connections for the 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
clock*	Digital Input	Clock that operates this block. The presence of this terminal varies depending on the **Enable Clock from terminal** parameter.
interrupt*	Digital Output	This signal can only be connected to an interrupt Component or left unconnected. The presence of this terminal varies depending on the **Interrupt** parameter.
rx_dma*	Digital Output	This signal is used to trigger a DMA transfer to get data from the RX FIFO and can only be connected to DMA channel trigger input. The output of this terminal is controlled by the RX FIFO Level parameter. The presence of this terminal varies depending **RX Output** parameter.
tx_dma*	Digital Output	This signal is used to trigger a DMA transfer to put data into the TX FIFO and can only be connected to DMA channel trigger input. The output of this terminal is controlled by the TX FIFO Level parameter. The presence of this terminal varies depending **TX Output** parameter.
rx_in*	Digital Input	The receive input receives the serial data from another device on the serial bus. The presence of this terminal varies depending on the **Show UART terminals** and **TX/RX Mode** parameters.
tx_out*	Digital Output	The transmitter output drives the output serial data to another device on the serial bus. The presence of this terminal varies depending on the **Show UART terminals** and **TX/RX Mode** parameters.
tx_en_out *	Digital Output	The transmitter enable keeps the output level high during serial data transfer. The presence of this terminal varies depending on the **Show UART terminals** and **TX/RX Mode** parameters.
cts_in*	Digital Input	The clear to send input accepts notification that another device is ready to receive data. The presence of this terminal varies depending on the **Show UART terminals** and **CTS** parameters.
rts_out*	Digital Output	The request to send output notifies another device that this device is ready to receive data. The presence of this terminal varies depending on the **Show UART terminals** and **RTS** parameters.
rx_tx_out*	Digital Output	The receive and transmit output receives and transmits data from/to another device on the bus. This pin presents if the Com Mode parameter is SmartCard. Note that Drive Mode of the connected pin must be Open Drain Drives Low. The presence of this terminal varies depending on the **Show UART terminals** parameter.

Internal Pins Configuration

By default, the UART pins are buried inside Component: UART_1_rx, UART_1_tx, UART_1_tx_en, UART_1_cts, UART_1_rts and UART_1_tx_rx. These pins are buried because they use dedicated connections and are not routable as general purpose signals. For more information, refer to the I/O System section in the device Technical Reference Manual (TRM).

The preferred method to change pins configuration is to enable **Show Terminals** option on the **Pins** tab and configure pins connected to the Component. Alternatively, the cy_gpio driver API can be used.

Note The instance name is not included in the Pin Names provided in the following tables:

Pin Name	Direction	Drive Mode	Initial Drive State	Threshold	Slew Rate	Descriptiion
rx	Input	High Impedance Digital	High	CMOS	–	The rx or input pin receives the serial data from another device on the serial bus.
tx	Output	Strong Drive	High	–	Fast	The tx output pin drives the output serial data to another device on the serial bus.
tx_en	Output	Strong Drive	High	–	Fast	The tx_en output pin keeps the output level high during serial data transfer.
cts	Input	High Impedance Digital	High	CMOS	–	The cts input pin accepts notification that another device is ready to receive data.
rts	Output	Strong Drive	High	–	Fast	The rts output pin notifies another device that this device is ready to receive data.
rx_tx	Bidirectional	Open Drain Drvies Low	High	CMOS	Fast	The rx_tx bi-directional pin receives and transmits data from/to another device on the bus. This pins configuration requires connection of external pulls on the line. The other option is applying internal pull-up setting pin Drive Mode to Resistive Pull-up.

The Input threshold level for **input pins** is CMOS, which should be used for the vast majority of application connections.

The Input Buffer for **output pins** is disabled so as not to cause current linkage in low power mode. Reading the status of these pins always returns zero. To get the current status, the input buffer must be enabled before a status read.

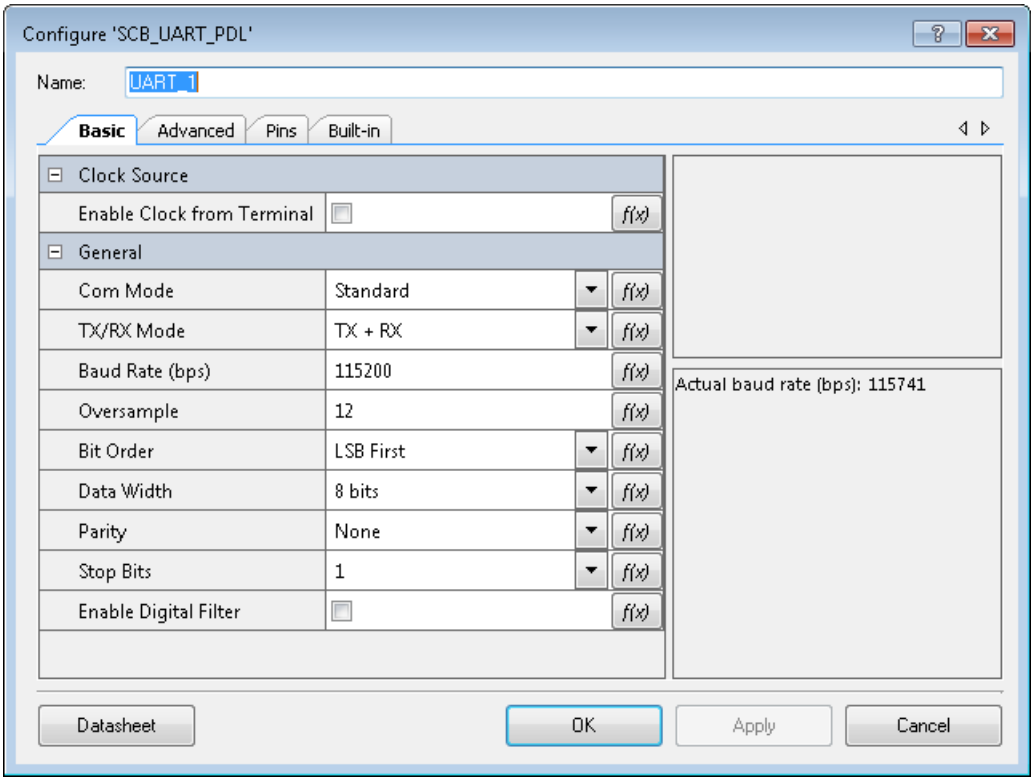
The other **input pins** and **output pins** parameters are set to default. Refer to pin component datasheet for more information about default parameters values.

Component Parameters

The SCB_UART_PDL Component Configure dialog allows you to edit the configuration parameters for the Component instance. Parameters are located on different tabs.

Basic Tab

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

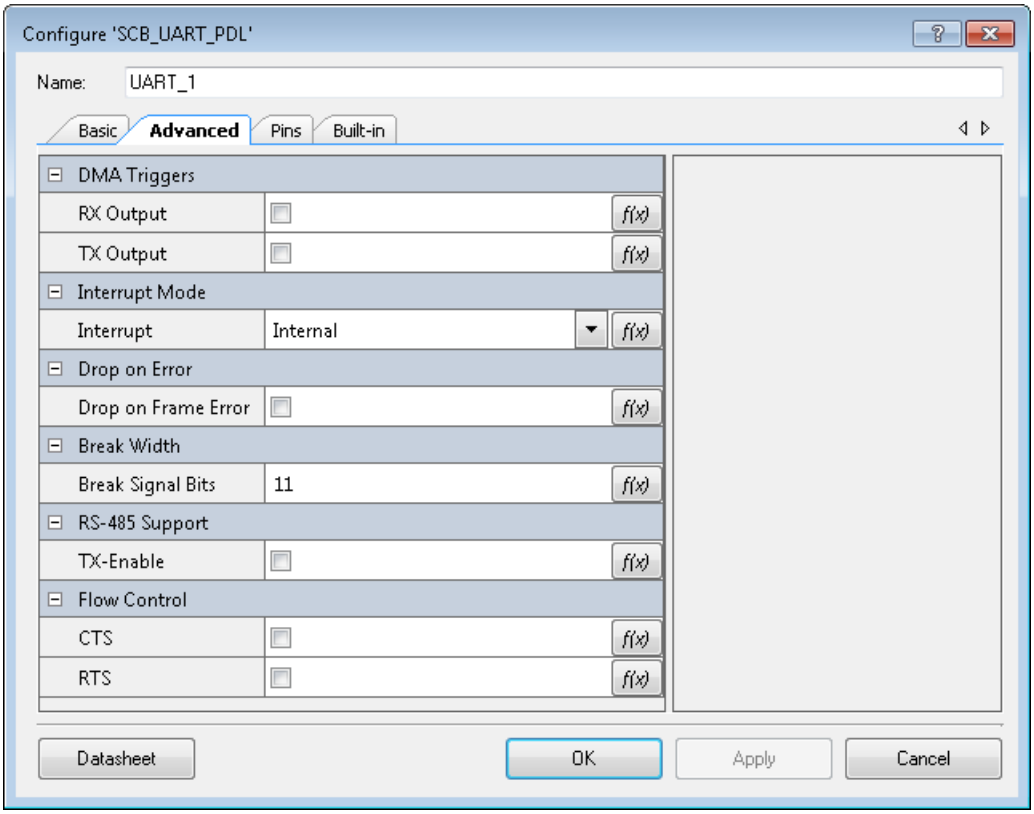


Parameter Name	Description
Enable Clock from Terminal	This parameter allows choosing between an internally configured clock (by the Component) or an externally configured clock (by the user) for the Component operation.
Com Mode	This parameter defines the sub-mode of UART as: Standard, SmartCard or IrDA.
TX/RX Mode	This parameter enables the receiver or transmitter functionality or both simultaneously.
Baud Rate (bps)	This parameter specifies the baud rate in bps. The actual baud rate may differ based on the available clock frequency and Component settings. Range: 1 - 1000000 bps.
Oversample	This parameter defines how many Component clocks oversample the selected baud rate. The range: 8 - 16 (except IrDA mode). The oversample values are predefined for IrDA mode.
Actual baud rate	The actual baud rate displays the data rate at which the Component will operate with current settings. The factors that affect the actual data rate calculation are: the nominal frequency of the Component clock (internal or external) and oversample parameter.

Parameter Name	Description
Bit Order	This parameter defines the direction in which the serial data is transmitted. When set to the MSB first, the most-significant bit is transmitted first. When set to the LSB first, the least-significant bit is transmitted first.
Data Width	This option defines the width of a single data element in bits. The range: 5-9.
Parity	This parameter defines the functionality of the parity bit location in the transfer as None, Odd or Even.
Stop Bits	This parameter defines the number of stop bits.
Enable Digital Filter	This parameter applies a digital 3-tap median filter to the UART input lines.
Retry on Nack	This parameter defines whether to send a message again when a NACK response is received. Only applicable when Com Mode is SmartCard.
Invert RX	This parameter enables the inversion of the incoming RX line signal. Only applicable when Com Mode is IrDA.
Low Power Receiving	This parameter enables the low-power receiver option. Only applicable when Com Mode is IrDA.

Advanced Tab

This tab contains the Interrupt configuration settings.



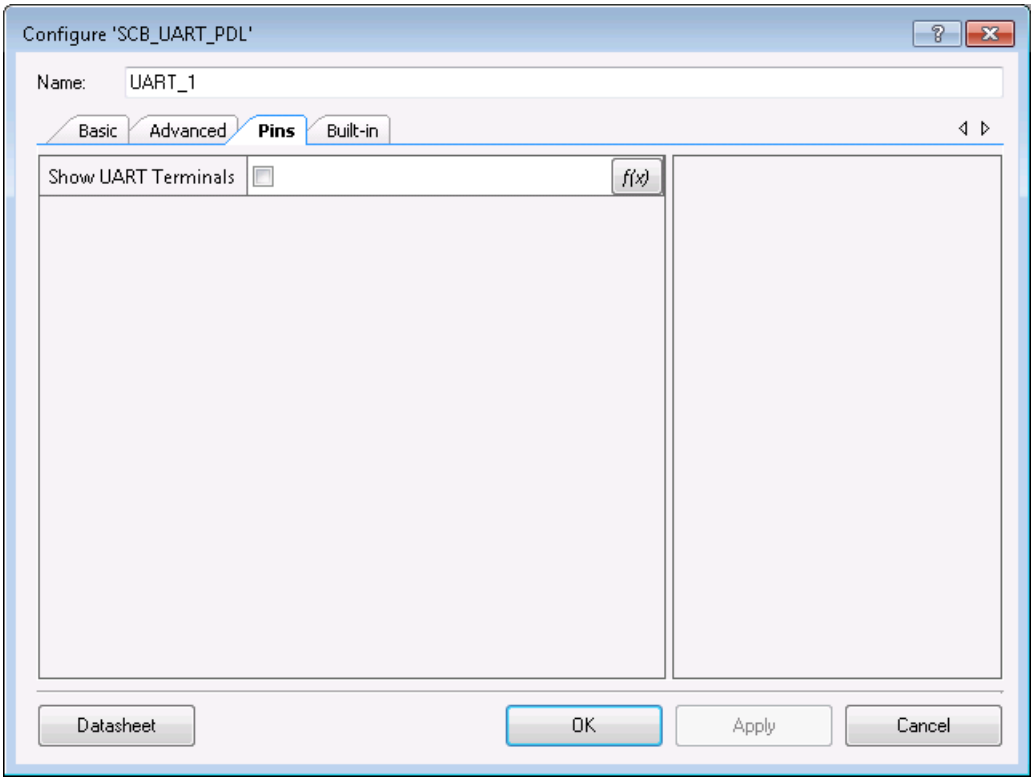
Parameter Name	Description
RX Output	This parameter enables the RX trigger output terminal (rx_dma) of the Component. This terminal must be connected to the DMA trigger input or left unconnected.
TX Output	This parameter enables the TX trigger output terminal (tx_dma) of the Component. This terminal must be connected to the DMA trigger input or left unconnected.
RX FIFO Level	This parameter determines the behavior of the signal that drives the RX FIFO Above Level interrupt source and the RX trigger output as follows: the signal is active while the number of data elements in the RX FIFO is greater than the value of the RX FIFO Level. For example, the RX FIFO has 8 data elements and the RX FIFO level is 0. The signal remains active until all data elements are read from the RX FIFO. The range: 0 – 127 (when Data Width <= 8), and 0 - 63 otherwise.
TX FIFO Level	This parameter determines the behavior of the signal that drives the TX FIFO Below Level interrupt source and the TX trigger output as follows: the signal is active while the number of data elements in the TX FIFO is less than the value of TX FIFO level. For example, the TX FIFO has 0 data elements (empty) and the TX FIFO level is 7. The signal remains active until TX FIFO is loaded to contain 7 data elements. The range: 0 – 127 (when Data Width <= 8), and 0 - 63 otherwise.
Interrupt	Internal or External. See Interrupt Service Routine section.

Parameter Name	Description
RX FIFO not Empty	This parameter enables the RX FIFO not-empty interrupt source.
RX FIFO Above Level	This parameter enables the RX FIFO above-level interrupt source to trigger the interrupt output.
RX FIFO Full	This parameter enables the RX FIFO full interrupt source to trigger the interrupt output.
RX FIFO Overflow	This parameter enables the RX FIFO overflow interrupt source to trigger the interrupt output.
RX FIFO Underflow	This parameter enables the RX FIFO underflow interrupt source to trigger the interrupt output.
RX Frame Error	This parameter enables the RX frame error interrupt source to trigger the interrupt output.
RX Parity Error	This parameter enables the RX parity error interrupt source to trigger the interrupt output.
Break Detected	This parameter enables the RX break detection interrupt source to trigger the interrupt output.
UART Done	This parameter enables the UART done interrupt source to trigger the interrupt output.
TX FIFO Empty	This parameter enables the TX FIFO empty interrupt source to trigger the interrupt output.
TX FIFO Below Level	This parameter enables the TX FIFO below-level interrupt source to trigger the interrupt output.
TX FIFO not Full	This parameter enables the TX FIFO not-full interrupt source to trigger the interrupt output.
TX FIFO Overflow	This parameter enables the TX FIFO overflow interrupt source to trigger the interrupt output.
TX FIFO Underflow	This parameter enables the TX FIFO underflow interrupt source to trigger the interrupt output.
TX Lost Arbitration	This parameter enables the TX lost arbitration interrupt source to trigger the interrupt output.
TX NACK	This parameter enables the TX NACK interrupt source to trigger the interrupt output.
Drop On Frame Error	This parameter determines if the data is dropped from the RX FIFO on a frame error event.
Drop On Parity Error	This parameter determines if the data is dropped from the RX FIFO on a parity error event.
Break Signal Bits	This parameter specifies the break width in bits. The range: 7-16.
TX-Enable	This parameter enables tx_en output, which is commonly used for RS-485 support. Note that TX-Enable and CTS parameters are mutually exclusive.
CTS	This parameter enables the cts input.
CTS Polarity	This parameter defines the active polarity of the CTS output signal as Active Low or Active High.
RTS	This parameter enables the rts output.
RTS Polarity	This parameter defines the active polarity of the RTS output signal as Active Low or Active High.
RTS Active When	This parameter determines the behavior of the RTS signal as follows: the signal is active while the number of data elements in the RX FIFO is less than the value of the RTS Active When. The range: 0 – 127 (when Data Width <= 8), and 0 - 63 otherwise.
Enable Multi Processor Mode	This parameter enables the UART multi-processor mode. Only applicable when Com Mode is Standard.
Address	This parameter specifies the UART address. Only applicable when the Enable Multi Processor Mode parameter is true.
Mask	This parameter specifies the UART address mask. Only applicable when the Enable Multi Processor Mode parameter is true. • Bit value 0 – excludes a bit from the address comparison. • Bit value 1 – the bit needs to match with the corresponding bit of the UART address.

Parameter Name	Description
Accept Matching Address in RX FIFO	This parameter determines whether to put the matched UART address into the RX FIFO. Only applicable when the Enable Multi Processor Mode parameter is true.

Pins Tab

This tab contains the Interrupt configuration settings.



Parameter Name	Description
Show UART Terminals	This parameter removes internal pins and expose signals to terminals. The exposed terminals must be connected to the pins or SmartIO Component.

Application Programming Interface

The Application Programming Interface (API) is provided by the cy_scb driver module from the PDL. The driver is copied into the “pdl\drivers\peripheral\scb\” directory of the application project after a successful build.

Refer to the PDL documentation for a detailed description of the complete API. To access this document, right-click on the Component symbol on the schematic and choose the “**Open PDL Documentation…**” option in the drop-down menu.

The Component generates the configuration structures and base address described in the Global Variables and Preprocessor Macros sections. Pass the generated data structure and the base address to the associated cy_scb driver function in the application initialization code to configure the peripheral. Once the peripheral is initialized, the application code can perform run-time changes by referencing the provided base address in the driver API functions.

By default, PSoC Creator assigns the instance name UART_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.

Global Variables

The SCB_UART_PDL Component populates the following peripheral initialization data structure(s). The generated code is placed in C source and header files that are named after the instance of the Component (e.g. UART_1.c). Each variable is also prefixed with the instance name of the Component.

cy_stc_scb_uart_config_t const UART_1_config

The instance-specific configuration structure. The pointer to this structure should be passed to Cy_SCB_UART_Init function to initialize Component with GUI selected settings.

cy_stc_scb_uart_context_t UART_1_context

The instance-specific context structure. It is used for internal configuration and data keeping for the UART. The user should not modify anything in this structure. If only Low Level functions are used this is not needed.

Preprocessor Macros

The SCB_UART_PDL Component generates the following preprocessor macro(s). Note that each macro is prefixed with the instance name of the Component (e.g. “UART_1”).

#define UART_1_HW ((CySCB_Type *) UART_1_SCB__HW)

The pointer to the base address of the hardware

Data in RAM

The generated data may be placed in flash memory (const) or RAM. The former is the more memory-efficient choice if you do not wish to modify the configuration data at run-time. Under the **Built-In** tab of the Configure dialog set the parameter CONST_CONFIG to make your selection. The default option is to place the data in flash.

Interrupt Service Routine

Interrupt processing is optional for the SCB_UART_PDL Component; therefore, the Component provides a parameter to choose between Internal and External placement of the Interrupt Component:

- Internal: This means that the Interrupt Component is placed inside the SCB_UART_PDL Component. This allows use of the UART High-Level communication functions in the cy_scb driver. However, you must configure the interrupt controller to call the cy_scb driver UART ISR, and you must enable the corresponding interrupt. The ISR must call Cy_SCB_UART_Interrupt function that implements cy_scb driver UART ISR functionality. Refer to the code example in the Quick Start section.
- External: This means that you are responsible for interrupt processing. There is no internal Interrupt Component. Instead, an output terminal is provided to connect to an external Interrupt Component. Use the **Interrupt Source** parameters to configure one or more sources to trigger the interrupt output. This external option allows you to implement your own interrupt handler. When this option is chosen only UART Low-Level communication functions from the cy_scb driver, can be used. You can also leave the output terminal not connected and the Component will not process an interrupt at all.

Refer to cy_scb driver documentation for the list of UART High-Level and Low-Level communication API.

Code Examples and Application Notes

Code Examples

PSoC Creator provides access to code examples in the 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 "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.

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.

Functional Description

Clock Selection

The SCB_UART_PDL Component provides an **Enable Clock from Terminal** parameter, which allows choosing between an internally configured clock (by the Component) or an externally configured clock for the Component operation:

- Internally configured means that the Component is responsible for clock configuration. It requests the system to provide the clock frequency required to operate with the selected data rate. The **Oversample** and **Baud Rate** parameters define clock frequency in Hz as (Baud Rate * Oversample).
- Externally configured means that the Component provides a clock terminal for Clock Component connection. You must then configure the Clock Component appropriately.

For more information about UART clock configuration, refer to the UART Clocking and Oversampling sub-section in the device TRM.

Baud Rate Configuration

The baud rate is determined by the connected clock source and the **Oversample** parameter. These two factors are used the set number of Component clocks within one UART bit time.

The Component provides the following methods to configure **Baud Rate**:

- Set the desired **Baud Rate** and **Oversample**. This option uses a clock that is internal to the Component (this clock still uses clock divider resources). The Component asks PSoC Creator to create a clock with a frequency equal to (Baud Rate * Oversample) in kHz. Iterate over the **Oversample** parameter to get the actual baud rate that meets your system UART accuracy requirements.
- Connect a user-configurable clock to the Component. This option is controlled by the **Enable Clock from Terminal** parameter, which must be enabled. You must also configure the **Oversample** parameter. This method provides full control of the data rate configuration.

Regardless of the chosen method, the Component will display the **Actual baud rate**. The difference between the actual baud rate and the desired baud rate must meet your system UART accuracy requirements.

DMA Support

The SCB_UART_PDL Component supports Direct Memory Access (DMA) transfers. The Component may transfer to/from the following sources.

Name of DMA Source / Destination	Length	Direction	DMA Req Signal	DMA Req Type	Description
RX_FIFO_RD	Word	Source	rx_dma	Level sensitive [1]	The data available in the RX FIFO is copied into place defined by destination. The DMA request signal is active while the number of data elements in the RX FIFO is greater than the value of RX FIFO Level.
TX_FIFO_WR	Word	Destination	tx_dma	Level sensitive [1]	The available data from source is copied into the TX FIFO. The DMA request signal is active while the number of data elements in the TX FIFO is less than the value of the TX FIFO Level.

Low-Power Modes

The SCB_UART_PDL Component requires specific processing to support Deep Sleep and Hibernate modes. The cy_scb driver module provides callback functions to handle power mode transition. These functions must be registered using the cy_syspm driver before entering Deep Sleep or Hibernate mode appropriately. Refer to the Low Power Support section of the cy_scb driver, or the System Power Management section of the PDL Documentation for more details about callback registration.

The SCB_UART_PDL Component is **not** capable of waking up the device from Deep Sleep mode. However, GPIO functionality can be used for this purpose. Refer to the Start Skipping section within the UART section of the device TRM for more information about waking up from Deep Sleep mode on UART RX activity.

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

[1] To properly handle DMA level request signal activation and de-activation from the SCB peripheral block, the DMA Descriptor typically must be configured to re-trigger after 16 Clk_Slow cycles.

This section provides information on Component-specific deviations. Refer to PSoC Creator Help > Building a PSoC Creator Project > Generated Files (PSoC 6) for information on MISRA compliance and deviations for files generated by PSoC Creator.

The SCB_UART_PDL Component does not have any specific deviations.

This Component uses firmware drivers from the cy_scb, cy_sysint, and cy_gpio PDL modules. For information on their MISRA compliance and specific deviations, refer to the PDL documentation.

This Component has the following embedded Components: clock, interrupt and pin. Refer to the corresponding Component datasheets for information on their MISRA compliance and specific deviations.

Registers

Refer to the Serial Communication Block Registers section in the device TRM.

Resources

The SCB_UART_PDL Component uses single SCB peripheral block configured for UART operation.

DC and AC Electrical Characteristics

Note Final characterization data for PSoC 6 devices is not available at this time. Once the data is available, the Component datasheet will be updated on the Cypress web site.

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
2.0.b	Updated the datasheet.	Added Low-Power Modes section.
2.0.a	Minor datasheet edits.	
2.0	Updated the underlying version of PDL driver.	
	Added support of tx_en signal commonly used for RS-485.	
	Datasheet updates.	
1.0.a	Datasheet edits.	Updated to clarify descriptions and update the Quick Start section.
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.

