**SPECIFICATION OF UART DRIVER**

1. Introduction and Functional Overview

This specification specifies the functionality, API and the configuration of the module UART Driver (called “UART module” in this document). The UART module is part of the lowest layer, performs the hardware access and offers a hardware independent API to the upper layer.

The UART module provides Names for initiating transmissions, config and calls the callback functions of the App layer.

2. Software Requirements

|  |  |  |
| --- | --- | --- |
| **Requirement ID** | **Reuirement description** | **Test case id to cover requirements of modules** |
| SRS\_UART\_01 | The program shall provide an initialization function for UART. The initialization function shall include initial configuration of parameters such as the UART baud rate, and other settings  The program shall configure parameters for the UART module as follows:  + Baud Rate: 115200  + Data bit: 8 bits  + Parity bit: disable  + Channel: LPUART channel 1 | UART\_TC\_01,  UART\_TC\_02, UART\_TC\_03,  UART\_TC\_06,  UART\_TC\_14 |
| SRS\_UART\_02 | The program shall provide a function to transmit/receiver data via UART, that accepts the UART instance, data buffer, and length as parameters. The function allows sending/receiving required data into the UART transmission/receive buffer and then posting it onto the transmission/receive bus | UART\_TC\_04,  UART\_TC\_05,  UART\_TC\_08,  UART\_TC\_09,  UART\_TC\_10,  UART\_TC\_11, UART\_TC\_12 |
| SRS\_UART\_03 | The program shall include functionality to de-initialize or disable a UART instance. De-initialization involves returning the UART module to a default or disabled state | UART\_TC\_07, UART\_TC\_13 |

3. UART Driver Layer

3.1. Macro

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Description** | **Type** | **Value** | **Defined in** |
| DRV\_UART\_CTRL\_ERROR\_REC\_INTERRUPT\_MASK | This macro defines bit fields that will be set in CTRL register | #define | 0xF000000u | DRV\_LPUART.h |
| DRV\_UART\_STAT\_ERROR\_REC\_FLAG\_MASK | This macro defines bit fields that will be set in STAT register | #define | 0xF0000u | DRV\_LPUART.h |

3.2. Type Definition

3.1.1. DRV\_UART\_DataBitCountType

|  |  |  |  |
| --- | --- | --- | --- |
| Name | DRV\_UART\_DataBitCountType | | |
| Kind | Enumeration | | |
| Range | DRV\_UART\_dataBitCount\_7 | 0xFFU | 7-bit data characters. |
| DRV\_UART\_dataBitCount\_8 | 0x00U | 8-bit data characters. |
| DRV\_UART\_dataBitCount\_9 | 0x01U | 9-bit data characters. |
| DRV\_UART\_dataBitCount\_10 | 0x02U | 10-bit data characters. |
| Description | This enum is responsible for configuring the number of bits per character in UART. | | |
| Available via | DRV\_LPUART.h | | |

3.1.2. DRV\_UART\_ParityModeType

|  |  |  |  |
| --- | --- | --- | --- |
| Name | DRV\_UART\_ParityModeType | | |
| Kind | Enumeration | | |
| Range | DRV\_UART\_parityModeDisabled | 0x00U | Parity disabled. |
| DRV\_UART\_parityModeEven | 0x02U | Parity enabled, type even, bit setting: PE|PT = 10. |
| DRV\_UART\_parityModeOdd | 0x03U | Parity enabled, type odd, bit setting: PE|PT = 11. |
| Description | This enum is responsible for configuring the parity mode. | | |
| Available via | DRV\_LPUART.h | | |

3.1.3. DRV\_UART\_StopBitCountType

|  |  |  |  |
| --- | --- | --- | --- |
| Name | DRV\_UART\_StopBitCountType | | |
| Kind | Enumeration | | |
| Range | DRV\_UART\_stopBitCountOne | 0x00U | One stop bit. |
| DRV\_UART\_stopBitCountTwo | 0x01U | Two stop bits. |
| Description | This enum is responsible for configuring the number of stop bits in UART communication. | | |
| Available via | DRV\_LPUART.h | | |

3.1.4. DRV\_UART\_TransferType

|  |  |  |  |
| --- | --- | --- | --- |
| Name | DRV\_UART\_TransferType | | |
| Kind | Enumeration | | |
| Range | DRV\_UART\_notUsingInterrupts | 0x00U | The driver will not use interrupts to perform UART transfer. |
| DRV\_UART\_usingInterrupts | 0x01U | The driver will use interrupts to perform UART transfer. |
| Description | This enum is responsible for deciding to use the type of transmission | | |
| Available via | DRV\_LPUART.h | | |

3.1.5. DRV\_UART\_CallBackFunctionType

|  |  |  |  |
| --- | --- | --- | --- |
| Name | DRV\_UART\_CallBackFunctionType | | |
| Kind | Enumeration | | |
| Range | DRV\_UART\_callBackError | 0x00U | Callback function type for handling error. |
| DRV\_UART\_callBackTransmitter | 0x01U | Callback function type for handling transmitter. |
| DRV\_UART\_callBackReceiver | 0x02U | Callback function type for handling receiver. |
| Description | This enum is responsible for configuring type for the callback function. | | |
| Available via | DRV\_LPUART.h | | |

3.1.6. DRV\_UART\_BaudRateValueType

|  |  |  |
| --- | --- | --- |
| Name | DRV\_UART\_BaudRateValueType | |
| Kind | Enumeration | |
| Range | DRV\_UART\_baudRateValue\_600 | 600U |
| DRV\_UART\_ baudRateValue \_9600 | 9600U |
| DRV\_UART\_ baudRateValue \_12800 | 12800U |
| DRV\_UART\_ baudRateValue \_38400 | 38400U |
| DRV\_UART\_ baudRateValue \_128000 | 128000U |
| DRV\_UART\_ baudRateValue \_230400 | 230400U |
| DRV\_UART\_ baudRateValue \_256000 | 256000U |
| DRV\_UART\_ baudRateValue \_115200 | 115200U |
| Description | This enum is responsible for configuring UART baudrate values. | |
| Available via | DRV\_LPUART.h | |

3.1.7. DRV\_UART\_InstanceType

|  |  |  |  |
| --- | --- | --- | --- |
| Name: | DRV\_UART\_InstanceType | | |
| Kind | Enumeration | | |
| Range | DRV\_UART\_instance\_0 | 0x00U | UART instance 0. |
| DRV\_UART\_ instance \_1 | 0x01U | UART instance 1. |
| DRV\_UART\_ instance \_2 | 0x02U | UART instance 2. |
| DRV\_UART\_instanceCount | 0x03U | Instance count |
| Description | This enum is responsible for configuring UART instance. | | |
| Available via | DRV\_LPUART.h | | |

3.1.8. DRV\_UART\_StatusType

|  |  |  |  |
| --- | --- | --- | --- |
| Name | DRV\_UART\_StatusType | | |
| Kind | Enumeration | | |
| Range | DRV\_UART\_txBusy | 0xFFU | Operation was busy due to data transfer in progress. |
| DRV\_UART\_rxBusy | 0xFEU | Operation was busy due to data receive in progress. |
| DRV\_UART\_stateReady | 0x22U | UART free to use. |
| DRV\_UART\_stateDefault | 0x55U | UART in default state. |
| DRV\_UART\_stateInitialized | 0xEEU | UART module initialized. |
| DRV\_UART\_stateRxOverrunError | 0x01U | RX overrun error state. |
| DRV\_UART\_stateFramingError | 0x02U | Framing error state. |
| DRV\_UART\_stateParityError | 0x03U | Parity error state. |
| DRV\_UART\_stateNoiseError | 0x04U | Noise error state. |
| DRV\_UART\_ok | 0x00U | Operation OK. |
| DRV\_UART\_error | 0x05U | Operation ERROR. |
| DRV\_UART\_busy | 0x06U | Operation was busy. |
| Description | This enum is responsible for configuring UART parameters. | | |
| Available via | DRV\_LPUART.h | | |

3.1.9. DRV\_UART\_ClkSourceType

|  |  |  |  |
| --- | --- | --- | --- |
| Name | DRV\_UART\_ClkSourceType | | |
| Kind | Enumeration | | |
| Range | DRV\_UART\_soscClkSouce | 0x00U | SOSCCLK source = 8000000U Hz |
|  | DRV\_UART\_fircClkSouce | 0x01U | FIRCCLK source = 48000000U Hz |
| Description | This enum is responsible for configuring UART clock source. | | |
| Available via | DRV\_LPUART.h | | |

3.1.10. DRV\_UART\_ConfigType

|  |  |  |
| --- | --- | --- |
| Name | DRV\_UART\_ConfigType | |
| Kind | Structure | |
| Range | bitCountPerChar | |
| ***Type*** | [DRV\_UART\_DataBitCountType](#3.1.1. DRV_UART_DataBitCountType) |
| ***Comment*** | Number of bits in a character. |
| parityMode | |
| ***Type*** | [DRV\_UART\_ParityModeType](#3.1.2.DRV_UART_ParityModeType) |
| ***Comment*** | Parity mode, disabled (default), even, odd. |
| stopBit | |
| ***Type*** | [DRV\_UART\_StopBitCountType](#3.1.3.DRV_UART_StopBitCountType) |
| ***Comment*** | Number of stop bits, 1 stop bit (default) or 2 stop bits. |
| baudRate | |
| ***Type*** | [DRV\_UART\_BaudrateValueType](#3.1.6. DRV_UART_BaudRateValueType) |
| ***Comment*** | Baud rate. |
| transferType | |
| ***Type*** | [DRV\_UART\_TransferType](#3.1.4. DRV_UART_TransferType) |
| ***Comment*** | Type of the transfer (polling/interrupt). |
| clockSource |  |
| ***Type*** | [DRV\_UART\_ClkSourceType](#3.1.9. DRV_UART_ClkSourceType) |
| ***Comment*** | UART module clock source |
| Description | This structure is responsible for defining the UART configuration structure. | |
| Available via | DRV\_LPUART.h | |

3.1.11. DRV\_UART\_RxBuffType

|  |  |  |
| --- | --- | --- |
| Name | DRV\_UART\_RxBuffType | |
| Kind | Structure | |
| Range | prxBuff | |
|  | ***Type*** | uint8\_t\* |
|  | ***Comment*** | Pointer to RX buffer. |
|  | rxBuffSize | |
|  | ***Type*** | uint16\_t |
|  | ***Comment*** | Size of Rx buffer. |
|  | rxCount | |
|  | ***Type*** | uint16\_t |
|  | ***Comment*** | LPUart RX Transfer Counter. |
|  | rxStatus |  |
|  | ***Type*** | [DRV\_UART\_StatusType](#3.1.8. DRV_UART_StatusType) |
|  | ***Comment*** | the status of receive |
|  | isRxBusy |  |
|  | ***Type*** | bool |
|  | ***Comment*** | Check the status of receiver |
| Description | This structure is responsible for defining the RX buffer. | |
| Available via | DRV\_LPUART.h | |

3.1.12. DRV\_UART\_TxBuffType

|  |  |  |
| --- | --- | --- |
| Name | DRV\_UART\_TxBuffType | |
| Kind | Structure | |
| Range | ptxBuff | |
| ***Type*** | uint8\_t\* |
| ***Comment*** | Pointer to TX buffer. |
| txBuffSize | |
| ***Type*** | uint16\_t |
| ***Comment*** | Size of Tx buffer. |
| txCount | |
| ***Type*** | uint16\_t |
| ***Comment*** | LPUart TX Transfer Counter. |
| txStatus |  |
| ***Type*** | [DRV\_UART\_StatusType](#3.1.8. DRV_UART_StatusType) |
| ***Comment*** | the status of transmitter |
| isTxBusy |  |
| ***Type*** | bool |
| ***Comment*** | Check the status of transmitter |
| Description | This structure is responsible for defining the TX buffer. | |
| Available via | DRV\_LPUART.h | |

3.1.13. DRV\_CallBackErrorLPUART

|  |  |
| --- | --- |
| Name | DRV\_CallBackErrorLPUART |
| Kind | Function pointer |
| Range | None |
| Description | typedef void (\*DRV\_CallBack\_LPUART)(void);  This function pointer is responsible for registering the callback function for transmitting, receiving, and handling errors if needed |
| Available via | DRV\_LPUART.h |

3.1.14. DRV\_UART\_InstallCallBackE

|  |  |
| --- | --- |
| Name | DRV\_UART\_InstallCallBackE |
| Kind | Function pointer |
| Range | None |
| Description | typedef void DRV\_UART\_InstallCallBackE(DRV\_CallBackErrorLPUART cbFunctionE);  This function pointer is responsible for registering the callback function for handling errors and providing UART module status |
| Available via | DRV\_LPUART.h |

3.3. Function Definition

3.3.1. DRV\_UART\_Init

**[SWS\_UART\_001]**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | DRV\_UART\_Init | | |
| Syntax | DRV\_UART\_StatusType DRV\_UART\_Init(const UART InstanceType instance ,const UART\_ConfigType\* uartConfig) | | |
| Parameters | [uartConfig](#_2.1.2.13._UART_ConfigType) [in] | The pointer points to a parameter for UART. | |
| [instance](#_2.1.2.15._UART_InstanceType) [in] | UART instance for initializing. | |
| Return value | [DRV\_UART\_StatusType](#3.1.8. DRV_UART_StatusType) | DRV\_UART\_stateReady | UART initialize success full and ready to use |
| DRV\_UART\_error | UART module encountered an error while initializing |
| Description | This function is responsible for initializing the UART module based on the provided configuration: data bit, parity mode, stop bit, baud rate and transfer type | | |
| Available via | DRV\_LPUART.h | | |

**[SWS\_UART\_002]: [**The [DRV\_UART\_Init](#3.3.1. DRV_UART_Init) function shall initialize the UART module for a specific instance using the provided configuration.]

**[SWS\_UART\_003]**: [The [DRV\_UART\_Init](#3.3.1. DRV_UART_Init) function shall validate the input parameters and return DRV\_UART\_error if:

* The instance is out of range
* The uartConfig parameter is a null pointer.]

**[SWS\_UART\_004]:** [The function shall configure the UART bit count per character, parity mode, stop bit, and baud rate.]

**[SWS\_UART\_005]:**[If the transferType in uartConfig is DRV\_UART\_usingInterrupts, the function shall enable the UART interrupt in the NVIC for the corresponding instance.]

**[SWS\_UART\_006]:** [The function shall initialize the transmit and receive buffers' status to DRV\_UART\_stateReady and mark them as not busy.]

**[SWS\_UART\_007]:** [If initialization is successful, the function shall return DRV\_UART\_stateReady. Otherwise, it shall return DRV\_UART\_error]

3.3.2. DRV\_UART\_Deinit

**[SWS\_UART\_008]**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | DRV\_UART\_Deinit | | |
| Syntax | DRV\_UART\_StatusType DRV\_UART\_Deinit(const UART\_InstanceType instance) | | |
| Parameters | [instance](#_2.1.2.15._UART_InstanceType) [in] | The UART instance to deinitialize the UART module | |
| Return value | [DRV\_UART\_StatusType](#3.1.8. DRV_UART_StatusType) | DRV\_UART\_ok | Deinitialize UART module success |
| DRV\_UART\_error | UART module encountered an error while deinitializing |
| Description | This function is responsible for deinitializing the UART module:  +, Set default STAT, CTRL, MATCH, BAUD registers  +, Disable NVIC module  +, Disable transmitter and receiver  +, Set default status for transmitter and receiver | | |
| Available via | DRV\_LPUART.h | | |

**[SWS\_UART\_009]:** [The DRV\_UART\_Deinit function shall deinitialize the UART module for a specific instance, resetting all related settings to their default states.]

**[SWS\_UART\_010]: [**The function shall validate the instance parameter and return DRV\_UART\_error if the instance is out of range.]

**[SWS\_UART\_011]:** [The function shall reset the UART to their default values.]

**[SWS\_UART\_012]:** [The function shall set all UART callback function pointers to NULL.]

**[SWS\_UART\_013]:** [The function shall reset the transmit and receive buffers' status to DRV\_UART\_stateDefault and mark them as not busy.

**[SWS\_UART\_014]**: [The function shall disable the UART interrupt in the NVIC for the specified instance.]

**[SWS\_UART\_015]:** [If the deinitialization is successful, the function shall return DRV\_UART\_ok. Otherwise, it shall return DRV\_UART\_error.]

3.3.3. DRV\_UART\_SetBaudRate

**[SWS\_UART\_016]**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | DRV\_UART\_SetBaudRate | | |
| Syntax | DRV\_UART\_StatusType DRV\_UART\_SetBaudRate(const UART\_InstanceType instance, const UART\_BaudrateValueType baudRate) | | |
| Parameters | [instance](#_2.1.2.15._UART_InstanceType) [in] | UART instance for setting baudrate. | |
| [baudRate](#_2.1.2.10._UART_BaudrateValueType) [in] | Baudrate parameter to be set. | |
| Return value | [DRV\_UART\_StatusType](#3.1.8. DRV_UART_StatusType) | DRV\_UART\_ok | Set baudrate for the UART module success |
| DRV\_UART\_error | The UART module encountered an error |
| Description | This function is responsible for setting the baud rate for UART communication. | | |
| Available via | DRV\_LPUART.h | | |

**[SWS\_UART\_017]:** [The function shall validate the instance parameter to ensure it is within the range of available UART instances. If the instance is invalid, the function shall return an error code.]

**[SWS\_UART\_018]:** [The function shall return DRV\_UART\_ok if the baud rate is successfully set.]

3.3.4. DRV\_UART\_SendDataPolling

**[SWS\_UART\_019]**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | DRV\_UART\_SendDataPolling | | |
| Syntax | DRV\_UART\_StatusType DRV\_UART\_SendDataPolling(const DRV\_UART\_InstanceType instance, const uint8\_t \*txBuff, uint16\_t txSize); | | |
| Parameters | [instance](#_2.1.2.15._UART_InstanceType) [in] | Given UART instance to send data polling. | |
| txBuff [in-out] | The pointer points to the transmit buffer. | |
| txSize [in] | The size of the transmit buffer. | |
| Return value | [DRV\_UART\_StatusType](#3.1.8. DRV_UART_StatusType) | DRV\_UART\_ok | Send data via polling method success |
| DRV\_UART\_error | The UART module encountered an error |
| DRV\_UART\_busy | The UART module is busy |
| Description | This function is responsible for sending data using polling method in UART communication. | | |
| Available via | DRV\_LPUART.h | | |

**[SWS\_UART\_020]:** [The function shall validate the instance parameter to ensure it is within the range of available UART instances. If the instance is invalid, the function shall return an error code DRV\_UART\_error.]

**[SWS\_UART\_021]:** [The function shall validate the txBuff pointer to ensure it is not NULL and the txSize parameter to ensure it is greater than 0. If either is invalid, the function shall return the error code DRV\_UART\_error.]

**[SWS\_UART\_022]:** [If the UART instance is currently busy transmitting data, the function shall return the status DRV\_UART\_txBusy.]

**[SWS\_UART\_023]:** [The function shall configure the UART transmitter by enabling the transmit (TE) control bit before sending data.]

**[SWS\_UART\_024]:** [The function shall transmit the data byte-by-byte using polling, waiting for the transmit data register (TDRE) and transmission complete (TC) status flags.]

**[SWS\_UART\_025]:** [The function shall support multiple data formats including 7-bit, 8-bit, 9-bit, and 10-bit configurations, as specified in the UART instance configuration.]

**[SWS\_UART\_026]:** [The function shall disable the UART transmitter after all data has been successfully transmitted.]

**[SWS\_UART\_027]:** [The function shall update the transmission status of the UART instance to DRV\_UART\_stateReady after the transmission completes.]

3.3.5. DRV\_UART\_SendDataInterrupt

**[SWS\_UART\_028]**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Name*** | DRV\_UART\_SendDataInterrupt | | |
| ***Syntax*** | DRV\_UART\_StatusType DRV\_UART\_SendDataInterrupt(const DRV\_UART\_InstanceType instance, uint8\_t\* data, uint16\_t length); | | |
| ***Parameters*** | [instance](#_2.1.2.12._DRV_UART_InstanceType) [in] | Given UART instance to transmit data | |
| txBuffer [in] | The pointer points to the transmit buffer. | |
| txSize [in] | The size of the transmit buffer. | |
| ***Return value*** | [DRV\_UART\_StatusType](#3.1.8. DRV_UART_StatusType) | DRV\_UART\_ok | Send data via interrupt method success |
| DRV\_UART\_error | The UART module encountered an error |
| DRV\_UART\_busy | The UART module is busy |
| ***Description*** | This function is responsible for transmitting data using the interrupt method in UART communication. | | |
| ***Available via*** | DRV\_LPUART.h | | |

**[SWS\_UART\_029]:** [The function shall validate the instance parameter to ensure it is within the range of available UART instances. If the instance is invalid, the function shall return an error code DRV\_UART\_error.]

**[SWS\_UART\_030]:** [The function shall validate the data pointer to ensure it is not NULL and the length parameter to ensure it is greater than 0. If either is invalid, the function shall return the error code DRV\_UART\_error.]

**[SWS\_UART\_031]:** [If the UART instance is already busy transmitting data, the function shall return the status DRV\_UART\_txBusy without altering the current transmission state.]

**[SWS\_UART\_032]:** [The function shall return DRV\_UART\_txBusy to indicate that the interrupt-driven transmission has been successfully initiated.]

3.3.6. DRV\_UART\_ReceiveDataPolling

**[SWS\_UART\_033]:**

|  |  |  |  |
| --- | --- | --- | --- |
| Name name: | DRV\_UART\_ReceiveDataPolling | | |
| Syntax | DRV\_UART\_StatusType DRV\_UART\_ ReceiveDataPolling(const DRV\_UART\_InstanceType instance, const uint8\_t\* rxBuff, uint16\_t rxSize); | | |
| Parameters | [instance](#_2.1.2.15._UART_InstanceType) [in] | Given UART instance to receive data | |
| rxBuff [in-out] | The pointer points to the receive buffer. | |
| rxSize [in] | The size of the receive buffer. | |
| Return value | [DRV\_UART\_StatusType](#3.1.8. DRV_UART_StatusType) | DRV\_UART\_ok | Receive data via polling method success |
| DRV\_UART\_error | The UART module encountered an error |
| DRV\_UART\_busy | The UART module is busy |
| Description | This function is responsible for receiving data using polling method in UART communication. | | |
| Available via | DRV\_LPUART.h | | |

**[SWS\_UART\_034]:** [The function shall validate the instance parameter to ensure it is within the range of available UART instances. If the instance is invalid, the function shall return an error code DRV\_UART\_error.]

**[SWS\_UART\_035]:** [The function shall validate the rxBuff pointer to ensure it is not NULL and the rxSize parameter to ensure it is greater than 0. If either is invalid, the function shall return the error code DRV\_UART\_error.]

**[SWS\_UART\_036]:** [If the UART instance is currently busy receiving data, the function shall return the status DRV\_UART\_txBusy.]

**[SWS\_UART\_037]:** [The function shall support receiving data in multiple formats including 7-bit, 8-bit, 9-bit, and 10-bit configurations, as specified in the UART instance configuration.]

**[SWS\_UART\_038]:** [The function shall update the status of the UART instance to DRV\_UART\_stateReady after the data reception completes.]

**[SWS\_UART\_039]:** [If the data reception completes successfully, the function shall return DRV\_UART\_stateReady. Otherwise, it shall return an appropriate error code.]

3.3.7. DRV\_UART\_ReceiveDataInterrupt

**[SWS\_UART\_040]**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Name*** | DRV\_UART\_ReceiveDataInterrupt | | |
| ***Syntax*** | DRV\_UART\_StatusType DRV\_UART\_ReceiveDataInterrupt(const DRV\_UART\_InstanceType instance, const uint8\_t \*rxBuff, const uint16\_t rxSize) | | |
| ***Parameters*** | [instance](#_2.1.2.15._UART_InstanceType) [in] | Given UART instance to receive data | |
| rxBuffer [in] | The pointer points to the receive buffer. | |
| rxSize [in] | The size of the receive buffer. | |
| ***Return value*** | [DRV\_UART\_StatusType](#3.1.8. DRV_UART_StatusType) | DRV\_UART\_ok | Receive data via interrupt method success |
| DRV\_UART\_error | The UART module encountered an error |
| DRV\_UART\_busy | The UART module is busy |
| ***Description*** | This function is responsible for receving data using the interrupt method in UART communication. | | |
| ***Available via*** | DRV\_LPUART.h | | |

**[SWS\_UART\_041]:** [The function shall validate the instance parameter to ensure it is within the range of available UART instances. If the instance is invalid, the function shall return an error code DRV\_UART\_error.]

**[SWS\_UART\_042]:** [The function shall validate the rxBuff pointer to ensure it is not NULL and the rxSize parameter to ensure it is greater than 0. If either is invalid, the function shall return the error code DRV\_UART\_error.]

**[SWS\_UART\_043]:** [If the UART instance is already busy receiving data, the function shall return the status DRV\_UART\_rxBusy.]

**[SWS\_UART\_044**]: [The function shall return DRV\_UART\_rxBusy to indicate that the interrupt-driven reception has been successfully initiated.]

3.3.8. DRV\_UART\_InstallCallBack

**[SWS\_UART\_045]**

|  |  |  |
| --- | --- | --- |
| ***Name*** | DRV\_UART\_InstallCallBack | |
| ***Syntax*** | void DRV\_UART\_InstallCallBack(DRV\_UART\_CallBackFunctionType callBackType, DRV\_CallBack\_LPUART cbFunction) | |
| ***Parameters*** | [callBackType](#_2.1.2.6._DRV_UART_CallBackFunctionT) [in] | Choose type of the callback function |
| [cbFunction](#_2.1.2.12._DRV_CallBack_LPUART) [in] | The pointer points to a function that be called when an interrupt happens. |
| ***Return value*** | void |  |
| ***Description*** | This function is responsible for registering function callback depending on the callback function type. | |
| ***Available via*** | DRV\_LPUART.h | |

**[SWS\_UART\_046]:** [The function shall validate the callBackType parameter to ensure it matches one of the defined callback types (DRV\_UART\_callBackError, DRV\_UART\_callBackTransmitter, or DRV\_UART\_callBackReceiver).]

**[SWS\_UART\_047**]: [The function shall assign the provided callback function pointer (cbFunction) to the corresponding entry in the s\_UARTfunctionPointer array based on the callBackType parameter.]

**[SWS\_UART\_048]:** [If the callBackType is DRV\_UART\_callBackError, the function shall assign the callback to the error callback entry

(s\_UARTfunctionPointer[DRV\_UART\_callBackError]).]

**[SWS\_UART\_049**]: [If the callBackType is DRV\_UART\_callBackTransmitter, the function shall assign the callback to the transmitter callback entry

(s\_UARTfunctionPointer[DRV\_UART\_callBackTransmitter]).]

**[SWS\_UART\_050**]: [If the callBackType is DRV\_UART\_callBackReceiver, the function shall assign the callback to the receiver callback entry

(s\_UARTfunctionPointer[DRV\_UART\_callBackReceiver]).]

3.3.9. DRV\_UART\_InstallCallBackE

**[SWS\_UART\_051]**

|  |  |  |
| --- | --- | --- |
| ***Name*** | DRV\_UART\_InstallCallBackE | |
| ***Syntax*** | void DRV\_UART\_InstallCallBackE(DRV\_CallBackErrorLPUART cbFunctionE) | |
| ***Parameters*** | [cbFunctionE](#_2.1.2.13._DRV_CallBackErrorLPUART) [in] | The pointer points to a function that be called when an errors via UART interrupt happens |
| ***Return value*** | void |  |
| ***Description*** | This function is responsible for detecting an error from the interrupt receive. | |
| ***Available via*** | DRV\_LPUART.h | |

**[SWS\_UART\_052]:** [The function shall assign the provided error callback function pointer (cbFunctionE) to the global variable s\_UARTx\_ErrorCallBack.]

3.3.10. DRV\_UART\_EnableTx

**[SWS\_UART\_053]**

|  |  |  |
| --- | --- | --- |
| ***Name*** | DRV\_UART\_EnableTx | |
| ***Syntax*** | void DRV\_UART\_EnableTx(const DRV\_UART\_InstanceType instance); | |
| ***Parameters*** | [instance](#_2.1.2.8._DRV_UART_InstanceType) [in] | Given UART instance to enable transmitter |
| ***Return value*** | void |  |
| ***Description*** | This function is responsible for enabling the transmitter. | |
| ***Available via*** | DRV\_LPUART.h | |

**[SWS\_UART\_054]:** [The function shall enable the transmitter for the specified UART instance by setting the TE (Transmitter Enable) bit in the CTRL register.]

3.3.11. DRV\_UART\_DisableTx

**[SWS\_UART\_055]**

|  |  |  |
| --- | --- | --- |
| ***Name*** | DRV\_UART\_DisableTx | |
| ***Syntax*** | void DRV\_UART\_DisableTx(const DRV\_UART\_InstanceType instance); | |
| ***Parameters*** | [instance](#_2.1.2.8._DRV_UART_InstanceType) [in] | Given UART instance to disable transmitter |
| ***Return value*** | void |  |
| ***Description*** | This function is responsible for disabling the transmitter. | |
| ***Available via*** | DRV\_LPUART.h | |

**[SWS\_UART\_056]:** [The function shall disable the transmitter for the specified UART instance by clearing the TE (Transmitter Enable) bit in the CTRL register.]

3.3.12. DRV\_UART\_EnableRx

**[SWS\_UART\_057]**

|  |  |  |
| --- | --- | --- |
| ***Name*** | DRV\_UART\_EnableRx | |
| ***Syntax*** | void DRV\_UART\_EnableRx(const DRV\_UART\_InstanceType instance); | |
| ***Parameters*** | [instance](#_2.1.2.8._DRV_UART_InstanceType) [in] | Given UART instance to enable the receiver |
| ***Return value*** | void |  |
| ***Description*** | This function is responsible for enabling the receiver. | |
| ***Available via*** | DRV\_LPUART.h | |

**[SWS\_UART\_058]:** [The function shall enable the receiver for the specified UART instance by setting the RE (Receiver Enable) bit in the CTRL register.]

3.3.13. DRV\_UART\_DisableRx

**[SWS\_UART\_059]**

|  |  |  |
| --- | --- | --- |
| ***Name*** | DRV\_UART\_DisableRx | |
| ***Syntax*** | void DRV\_UART\_DisableRx(const DRV\_UART\_InstanceType instance); | |
| ***Parameters*** | [instance](#_2.1.2.8._DRV_UART_InstanceType) [in] | Given UART instance to disable receiver |
| ***Return value*** | void |  |
| ***Description*** | This function is responsible for disabling the receiver. | |
| ***Available via*** | DRV\_LPUART.h | |

**[SWS\_UART\_059]:** [The function shall disable the receiver for the specified UART instance by clearing the RE (Receiver Enable) bit in the CTRL register.]

3.3.14. DRV\_UART\_AbortReceiving

**[SWS\_UART\_060]**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Name*** | DRV\_UART\_ AbortReceiving | | |
| ***Syntax*** | DRV\_UART\_StatusType DRV\_UART\_AbortReceiving(const DRV\_UART\_InstanceType instance); | | |
| ***Parameters*** | [instance](#_2.1.2.8._DRV_UART_InstanceType) [in] | Given UART instance to abort receiver | |
| ***Return value*** | [DRV\_UART\_StatusType](#3.1.8. DRV_UART_StatusType) | DRV\_UART\_stateReady | Return ready status after aborting |
| ***Description*** | This function is responsible for aborting the receiving. | | |
| ***Available via*** | DRV\_LPUART.h | | |

**[SWS\_UART\_061]:** [The function shall reset the receive buffer by setting the buffer pointer to NULL, buffer size to 0, and receive count to 0 for the specified UART instance.]

**[SWS\_UART\_062]:** [The function shall update the receive status (rxStatus) of the specified UART instance to DRV\_UART\_stateReady.]

**[SWS\_UART\_063]:** [The function shall disable the receiver by clearing the RE (Receiver Enable) bit in the CTRL register of the specified UART instance.]

**[SWS\_UART\_064]:** [The function shall disable the receive interrupt by clearing the RIE (Receive Interrupt Enable) bit in the CTRL register of the specified UART instance.]

**[SWS\_UART\_065]:** [The function shall return the status DRV\_UART\_stateReady after successfully aborting the receive operation.]

3.3.15. DRV\_UART\_AbortTransmitting

**[SWS\_UART\_066]**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Name*** | DRV\_UART\_ AbortTransmitting | | |
| ***Syntax*** | DRV\_UART\_StatusType DRV\_UART\_ AbortTransmitting(const DRV\_UART\_InstanceType instance); | | |
| ***Parameters*** | [instance](#_2.1.2.8._DRV_UART_InstanceType) [in] | Given UART instance to abort receiver | |
| ***Return value*** | [DRV\_UART\_StatusType](#3.1.8. DRV_UART_StatusType) | DRV\_UART\_stateReady | Return ready status after aborting |
| ***Description*** | This function is responsible for aborting the transmitting. | | |
| ***Available via*** | DRV\_LPUART.h | | |

**[SWS\_UART\_067]:** [The function shall update the transmit status (txStatus) of the specified UART instance to DRV\_UART\_stateReady.]

**[SWS\_UART\_068]:** [The function shall disable the transmitter by clearing the TE (Transmitter Enable) bit in the CTRL register of the specified UART instance.]

**[SWS\_UART\_069]:** [The function shall disable the transmit interrupt by clearing the TIE (Transmit Interrupt Enable) bit in the CTRL register of the specified UART instance.]

**[SWS\_UART\_070]:** [The function shall return the status DRV\_UART\_stateReady after successfully aborting the transmit operation.]