68 HAL UART Generic Driver

68.1 UART Firmware driver registers structures

68.1.1 UART_InitTypeDef

Data Fields

- ***uint32_t BaudRate***
- ***uint32_t WordLength***
- ***uint32_t StopBits***
- ***uint32_t Parity***
- ***uint32_t Mode***
- ***uint32_t HwFlowCtl***
- ***uint32_t OverSampling***
- ***uint32_t OneBitSampling***
- ***uint32_t ClockPrescaler***

Field Documentation

- ***uint32_t UART_InitTypeDef::BaudRate***
  This member configures the UART communication baud rate. The baud rate register is computed using the following formula: UART: =====If oversampling is 16 or in LIN mode, Baud Rate Register = ((uart_ker_ckpres) / ((huart->Init.BaudRate)))If oversampling is 8, Baud Rate Register[15:4] = ((2 * uart_ker_ckpres) / ((huart->Init.BaudRate)))[15:4] Baud Rate Register[3] = 0 Baud Rate Register[2:0] = (((2 * uart_ker_ckpres) / ((huart->Init.BaudRate)))[3:0]) >> 1 LPUART: ======= Baud Rate Register = ((256 * lpuart_ker_ckpres) / ((huart->Init.BaudRate))) where (uart/lpuart)_ker_ck_pres is the UART input clock divided by a prescaler
- ***uint32_t UART_InitTypeDef::WordLength***
  Specifies the number of data bits transmitted or received in a frame. This parameter can be a value of ***UARTEx_Word_Length***.
- ***uint32_t UART_InitTypeDef::StopBits***
  Specifies the number of stop bits transmitted. This parameter can be a value of ***UART_Stop_Bits***.
- ***uint32_t UART_InitTypeDef::Parity***
  Specifies the parity mode. This parameter can be a value of ***UART_Parity***
  Note: When parity is enabled, the computed parity is inserted at the MSB position of the transmitted data (9th bit when the word length is set to 9 data bits; 8th bit when the word length is set to 8 data bits).
- ***uint32_t UART_InitTypeDef::Mode***
  Specifies whether the Receive or Transmit mode is enabled or disabled. This parameter can be a value of ***UART_Mode***.
- ***uint32_t UART_InitTypeDef::HwFlowCtl***
  Specifies whether the hardware flow control mode is enabled or disabled. This parameter can be a value of ***UART_Hardware_Flow_Control***.
- ***uint32_t UART_InitTypeDef::OverSampling***
  Specifies whether the Over sampling 8 is enabled or disabled, to achieve higher speed (up to f_PCLK/8). This parameter can be a value of ***UART_Over_Sampling***.
- ***uint32_t UART_InitTypeDef::OneBitSampling***
  Specifies whether a single sample or three samples' majority vote is selected. Selecting the single sample method increases the receiver tolerance to clock deviations. This parameter can be a value of ***UART_OneBit_Sampling***.

- ***uint32_t UART_InitTypeDef::ClockPrescaler***
  Specifies the prescaler value used to divide the UART clock source. This parameter can be a value of ***UART_ClockPrescaler***.

68.1.2 UART_AdvFeatureInitTypeDef

Data Fields

- ***uint32_t AdvFeatureInit***
- ***uint32_t TxPinLevelInvert***
- ***uint32_t RxPinLevelInvert***
- ***uint32_t DataInvert***
- ***uint32_t Swap***
- ***uint32_t OverrunDisable***
- ***uint32_t DMADisableonRxError***
- ***uint32_t AutoBaudRateEnable***
- ***uint32_t AutoBaudRateMode***
- ***uint32_t MSBFirst***

Field Documentation

- ***uint32_t UART_AdvFeatureInitTypeDef::AdvFeatureInit***
  Specifies which advanced UART features is initialized. Several Advanced Features may be initialized at the same time . This parameter can be a value of ***UART_Advanced_Features_Initialization_Type***.
- ***uint32_t UART_AdvFeatureInitTypeDef::TxPinLevelInvert***
  Specifies whether the TX pin active level is inverted. This parameter can be a value of ***UART_Tx_Inv***.
- ***uint32_t UART_AdvFeatureInitTypeDef::RxPinLevelInvert***
  Specifies whether the RX pin active level is inverted. This parameter can be a value of ***UART_Rx_Inv***.
- ***uint32_t UART_AdvFeatureInitTypeDef::DataInvert***
  Specifies whether data are inverted (positive/direct logic vs negative/inverted logic). This parameter can be a value of ***UART_Data_Inv***.
- ***uint32_t UART_AdvFeatureInitTypeDef::Swap***
  Specifies whether TX and RX pins are swapped. This parameter can be a value of ***UART_Rx_Tx_Swap***.
- ***uint32_t UART_AdvFeatureInitTypeDef::OverrunDisable***
  Specifies whether the reception overrun detection is disabled. This parameter can be a value of ***UART_Overrun_Disable***.
- ***uint32_t UART_AdvFeatureInitTypeDef::DMADisableonRxError***
  Specifies whether the DMA is disabled in case of reception error. This parameter can be a value of ***UART_DMA_Disable_on_Rx_Error***.
- ***uint32_t UART_AdvFeatureInitTypeDef::AutoBaudRateEnable***
  Specifies whether auto Baud rate detection is enabled. This parameter can be a value of ***UART_AutoBaudRate_Enable***
- ***uint32_t UART_AdvFeatureInitTypeDef::AutoBaudRateMode***
  If auto Baud rate detection is enabled, specifies how the rate detection is carried out. This parameter can be a value of ***UART_AutoBaud_Rate_Mode***.
- ***uint32_t UART_AdvFeatureInitTypeDef::MSBFirst***
  Specifies whether MSB is sent first on UART line. This parameter can be a value of ***UART_MSB_First***.

68.1.3 __UART_HandleTypeDef

Data Fields

- ***USART_TypeDef * Instance***
- ***UART_InitTypeDef Init***
- ***UART_AdvFeatureInitTypeDef AdvancedInit***
- ***uint8_t * pTxBuffPtr***
- ***uint16_t TxXferSize***
- ***__IO uint16_t TxXferCount***
- ***uint8_t * pRxBuffPtr***
- ***uint16_t RxXferSize***
- ***__IO uint16_t RxXferCount***
- ***uint16_t Mask***
- ***uint16_t NbRxDataToProcess***
- ***uint16_t NbTxDataToProcess***
- ***uint32_t FifoMode***
- ***uint32_t SlaveMode***
- ***void(* RxISR***
- ***void(* TxISR***
- ***DMA_HandleTypeDef * hdmatx***
- ***DMA_HandleTypeDef * hdmarx***
- ***HAL_LockTypeDef Lock***
- ***__IO HAL_UART_StateTypeDef gState***
- ***__IO HAL_UART_StateTypeDef RxState***
- ***__IO uint32_t ErrorCode***

Field Documentation

- ***USART_TypeDef* __UART_HandleTypeDef::Instance***
  UART registers base address
- ***UART_InitTypeDef __UART_HandleTypeDef::Init***
  UART communication parameters
- ***UART_AdvFeatureInitTypeDef __UART_HandleTypeDef::AdvancedInit***
  UART Advanced Features initialization parameters
- ***uint8_t* __UART_HandleTypeDef::pTxBuffPtr***
  Pointer to UART Tx transfer Buffer
- ***uint16_t __UART_HandleTypeDef::TxXferSize***
  UART Tx Transfer size
- ***__IO uint16_t __UART_HandleTypeDef::TxXferCount***
  UART Tx Transfer Counter
- ***uint8_t* __UART_HandleTypeDef::pRxBuffPtr***
  Pointer to UART Rx transfer Buffer
- ***uint16_t __UART_HandleTypeDef::RxXferSize***
  UART Rx Transfer size
- ***__IO uint16_t __UART_HandleTypeDef::RxXferCount***
  UART Rx Transfer Counter
- ***uint16_t __UART_HandleTypeDef::Mask***
  UART Rx RDR register mask
- ***uint16_t __UART_HandleTypeDef::NbRxDataToProcess***
  Number of data to process during RX ISR execution
- ***uint16_t __UART_HandleTypeDef::NbTxDataToProcess***
  Number of data to process during TX ISR execution
- ***uint32_t __UART_HandleTypeDef::FifoMode***
  Specifies if the FIFO mode is being used. This parameter can be a value of ***UARTEx_FIFO_mode***.

- ***uint32_t __UART_HandleTypeDef::SlaveMode***
  Specifies if the UART SPI Slave mode is being used. This parameter can be a value of ***UARTEx_Slave_Mode***.
- ***void(* __UART_HandleTypeDef::RxISR)(struct __UART_HandleTypeDef *huart)***
  Function pointer on Rx IRQ handler
- ***void(* __UART_HandleTypeDef::TxISR)(struct __UART_HandleTypeDef *huart)***
  Function pointer on Tx IRQ handler
- ***DMA_HandleTypeDef* __UART_HandleTypeDef::hdmatx***
  UART Tx DMA Handle parameters
- ***DMA_HandleTypeDef* __UART_HandleTypeDef::hdmarx***
  UART Rx DMA Handle parameters
- ***HAL_LockTypeDef __UART_HandleTypeDef::Lock***
  Locking object
- ***__IO HAL_UART_StateTypeDef __UART_HandleTypeDef::gState***
  UART state information related to global Handle management and also related to Tx operations. This parameter can be a value of **HAL_UART_StateTypeDef**
- ***__IO HAL_UART_StateTypeDef __UART_HandleTypeDef::RxState***
  UART state information related to Rx operations. This parameter can be a value of **HAL_UART_StateTypeDef**
- ***__IO uint32_t __UART_HandleTypeDef::ErrorCode***
  UART Error code

68.2 UART Firmware driver API description

68.2.1 How to use this driver

The UART HAL driver can be used as follows:

1. Declare a UART_HandleTypeDef handle structure (eg. UART_HandleTypeDef huart).
2. Initialize the UART low level resources by implementing the HAL_UART_MspInit() API:
   - Enable the USARTx interface clock.
   - UART pins configuration:
     - Enable the clock for the UART GPIOs.
     - Configure these UART pins as alternate function pull-up.
   - NVIC configuration if you need to use interrupt process (HAL_UART_Transmit_IT() and HAL_UART_Receive_IT() APIs):
     - Configure the USARTx interrupt priority.
     - Enable the NVIC USART IRQ handle.
   - UART interrupts handling: The specific UART interrupts (Transmission complete interrupt, RXNE interrupt, RX/TX FIFOs related interrupts and Error Interrupts) are managed using the macros __HAL_UART_ENABLE_IT() and __HAL_UART_DISABLE_IT() inside the transmit and receive processes.
   - DMA Configuration if you need to use DMA process (HAL_UART_Transmit_DMA() and HAL_UART_Receive_DMA() APIs):
     - Declare a DMA handle structure for the Tx/Rx channel.
     - Enable the DMAx interface clock.
     - Configure the declared DMA handle structure with the required Tx/Rx parameters.
     - Configure the DMA Tx/Rx channel.
     - Associate the initialized DMA handle to the UART DMA Tx/Rx handle.
     - Configure the priority and enable the NVIC for the transfer complete interrupt on the DMA Tx/Rx channel.

3. Program the Baud Rate, Word Length, Stop Bit, Parity, Prescaler value , Hardware flow control and Mode (Receiver/Transmitter) in the huart handle Init structure.
4. If required, program UART advanced features (TX/RX pins swap, auto Baud rate detection,...) in the huart handle AdvancedInit structure.
5. For the UART asynchronous mode, initialize the UART registers by calling the HAL_UART_Init() API.
6. For the UART Half duplex mode, initialize the UART registers by calling the HAL_HalfDuplex_Init() API.
7. For the UART LIN (Local Interconnection Network) mode, initialize the UART registers by calling the HAL_LIN_Init() API.
8. For the UART Multiprocessor mode, initialize the UART registers by calling the HAL_MultiProcessor_Init() API.
9. For the UART RS485 Driver Enabled mode, initialize the UART registers by calling the HAL_RS485Ex_Init() API.

These API's (HAL_UART_Init(), HAL_HalfDuplex_Init(), HAL_LIN_Init(), HAL_MultiProcessor_Init(), also configure the low level Hardware GPIO, CLOCK, CORTEX...etc) by calling the customized HAL_UART_MspInit() API.

68.2.2 Initialization and Configuration functions

This subsection provides a set of functions allowing to initialize the USARTx or the UARTy in asynchronous mode.

- For the asynchronous mode the parameters below can be configured:
  - Baud Rate
  - Word Length
  - Stop Bit
  - Parity: If the parity is enabled, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit.
  - Hardware flow control
  - Receiver/transmitter modes
  - Over Sampling Method
  - One-Bit Sampling Method
- For the asynchronous mode, the following advanced features can be configured as well:
  - TX and/or RX pin level inversion
  - data logical level inversion
  - RX and TX pins swap
  - RX overrun detection disabling
  - DMA disabling on RX error
  - MSB first on communication line
  - auto Baud rate detection

The HAL_UART_Init(), HAL_HalfDuplex_Init(), HAL_LIN_Init()and HAL_MultiProcessor_Init()API follow respectively the UART asynchronous, UART Half duplex, UART LIN mode and UART multiprocessor mode configuration procedures (details for the procedures are available in reference manual).

This section contains the following APIs:

- ***HAL_UART_Init()***
- ***HAL_HalfDuplex_Init()***
- ***HAL_LIN_Init()***
- ***HAL_MultiProcessor_Init()***

- ***HAL_UART_DeInit()***
- ***HAL_UART_MspInit()***
- ***HAL_UART_MspDeInit()***

68.2.3 IO operation functions

This section contains the following APIs:

- ***HAL_UART_Transmit()***
- ***HAL_UART_Receive()***
- ***HAL_UART_Transmit_IT()***
- ***HAL_UART_Receive_IT()***
- ***HAL_UART_Transmit_DMA()***
- ***HAL_UART_Receive_DMA()***
- ***HAL_UART_DMAPause()***
- ***HAL_UART_DMAResume()***
- ***HAL_UART_DMAStop()***
- ***HAL_UART_Abort()***
- ***HAL_UART_AbortTransmit()***
- ***HAL_UART_AbortReceive()***
- ***HAL_UART_Abort_IT()***
- ***HAL_UART_AbortTransmit_IT()***
- ***HAL_UART_AbortReceive_IT()***
- ***HAL_UART_IRQHandler()***
- ***HAL_UART_TxCpltCallback()***
- ***HAL_UART_TxHalfCpltCallback()***
- ***HAL_UART_RxCpltCallback()***
- ***HAL_UART_RxHalfCpltCallback()***
- ***HAL_UART_ErrorCallback()***
- ***HAL_UART_AbortCpltCallback()***
- ***HAL_UART_AbortTransmitCpltCallback()***
- ***HAL_UART_AbortReceiveCpltCallback()***

68.2.4 Peripheral Control functions

This subsection provides a set of functions allowing to control the UART.

- HAL_MultiProcessor_EnableMuteMode() API enables mute mode
- HAL_MultiProcessor_DisableMuteMode() API disables mute mode
- HAL_MultiProcessor_EnterMuteMode() API enters mute mode
- HAL_MultiProcessor_EnableMuteMode() API enables mute mode
- UART_SetConfig() API configures the UART peripheral
- UART_AdvFeatureConfig() API optionally configures the UART advanced features
- UART_CheckIdleState() API ensures that TEACK and/or REACK are set after initialization
- UART_Wakeup_AddressConfig() API configures the wake-up from stop mode parameters
- HAL_HalfDuplex_EnableTransmitter() API disables receiver and enables transmitter
- HAL_HalfDuplex_EnableReceiver() API disables transmitter and enables receiver
- HAL_LIN_SendBreak() API transmits the break characters

This section contains the following APIs:

- ***HAL_MultiProcessor_EnableMuteMode()***
- ***HAL_MultiProcessor_DisableMuteMode()***
- ***HAL_MultiProcessor_EnterMuteMode()***

- *HAL_HalfDuplex_EnableTransmitter()*
- *HAL_HalfDuplex_EnableReceiver()*
- *HAL_LIN_SendBreak()*

68.2.5 Peripheral State and Error functions

This subsection provides functions allowing to:

- Return the UART handle state.
- Return the UART handle error code

This section contains the following APIs:

- *HAL_UART_GetState()*
- *HAL_UART_GetError()*

68.2.6 Detailed description of functions

HAL_UART_Init

Function name	**HAL_StatusTypeDef HAL_UART_Init (UART_HandleTypeDef * huart)**
Function description	Initialize the UART mode according to the specified parameters in the UART_InitTypeDef and initialize the associated handle.
Parameters	• **huart:** UART handle.
Return values	• **HAL:** status

HAL_HalfDuplex_Init

Function name	**HAL_StatusTypeDef HAL_HalfDuplex_Init (UART_HandleTypeDef * huart)**
Function description	Initialize the half-duplex mode according to the specified parameters in the UART_InitTypeDef and creates the associated handle.
Parameters	• **huart:** UART handle.
Return values	• **HAL:** status

HAL_LIN_Init

Function name	**HAL_StatusTypeDef HAL_LIN_Init (UART_HandleTypeDef * huart, uint32_t BreakDetectLength)**
Function description	Initialize the LIN mode according to the specified parameters in the UART_InitTypeDef and creates the associated handle .
Parameters	• **huart:** UART handle. • **BreakDetectLength:** Specifies the LIN break detection length. This parameter can be one of the following values: – UART_LINBREAKDETECTLENGTH_10B 10-bit break detection – UART_LINBREAKDETECTLENGTH_11B 11-bit break detection
Return values	• **HAL:** status

HAL_MultiProcessor_Init

Function name **HAL_StatusTypeDef HAL_MultiProcessor_Init (UART_HandleTypeDef * huart, uint8_t Address, uint32_t WakeUpMethod)**

Function description Initialize the multiprocessor mode according to the specified parameters in the UART_InitTypeDef and initialize the associated handle.

Parameters
- **huart:** UART handle.
- **Address:** UART node address (4-, 6-, 7- or 8-bit long).
- **WakeUpMethod:** Specifies the UART wakeup method. This parameter can be one of the following values:
  - UART_WAKEUPMETHOD_IDLELINE WakeUp by an idle line detection
  - UART_WAKEUPMETHOD_ADDRESSMARK WakeUp by an address mark

Return values
- **HAL:** status

Notes
- If the user resorts to idle line detection wake up, the Address parameter is useless and ignored by the initialization function.
- If the user resorts to address mark wake up, the address length detection is configured by default to 4 bits only. For the UART to be able to manage 6-, 7- or 8-bit long addresses detection, the API HAL_MultiProcessorEx_AddressLength_Set() must be called after HAL_MultiProcessor_Init().

HAL_UART_DeInit

Function name **HAL_StatusTypeDef HAL_UART_DeInit (UART_HandleTypeDef * huart)**

Function description DeInitialize the UART peripheral.

Parameters
- **huart:** UART handle.

Return values
- **HAL:** status

HAL_UART_MspInit

Function name **void HAL_UART_MspInit (UART_HandleTypeDef * huart)**

Function description Initialize the UART MSP.

Parameters
- **huart:** UART handle.

Return values
- **None:**

HAL_UART_MspDeInit

Function name **void HAL_UART_MspDeInit (UART_HandleTypeDef * huart)**

Function description DeInitialize the UART MSP.

Parameters
- **huart:** UART handle.

Return values
- **None:**

HAL_UART_Transmit

Function name **HAL_StatusTypeDef HAL_UART_Transmit (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size, uint32_t Timeout)**

Function description Send an amount of data in blocking mode.

Parameters
- **huart:** UART handle.
- **pData:** Pointer to data buffer.
- **Size:** Amount of data to be sent.
- **Timeout:** Timeout duration.

Return values
- **HAL:** status

Notes
- When FIFO mode is enabled, writing a data in the TDR register adds one data to the TXFIFO. Write operations to the TDR register are performed when TXFNF flag is set. From hardware perspective, TXFNF flag and TXE are mapped on the same bit-field.

HAL_UART_Receive

Function name **HAL_StatusTypeDef HAL_UART_Receive (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size, uint32_t Timeout)**

Function description Receive an amount of data in blocking mode.

Parameters
- **huart:** UART handle.
- **pData:** Pointer to data buffer.
- **Size:** Amount of data to be received.
- **Timeout:** Timeout duration.

Return values
- **HAL:** status

Notes
- When FIFO mode is enabled, the RXFNE flag is set as long as the RXFIFO is not empty. Read operations from the RDR register are performed when RXFNE flag is set. From hardware perspective, RXFNE flag and RXNE are mapped on the same bit-field.

HAL_UART_Transmit_IT

Function name **HAL_StatusTypeDef HAL_UART_Transmit_IT (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size)**

Function description Send an amount of data in interrupt mode.

Parameters
- **huart:** UART handle.
- **pData:** Pointer to data buffer.
- **Size:** Amount of data to be sent.

Return values
- **HAL:** status

HAL_UART_Receive_IT

Function name **HAL_StatusTypeDef HAL_UART_Receive_IT (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size)**

Function description Receive an amount of data in interrupt mode.

Parameters
- **huart:** UART handle.
- **pData:** Pointer to data buffer.
- **Size:** Amount of data to be received.

Return values
- **HAL:** status

HAL_UART_Transmit_DMA

Function name **HAL_StatusTypeDef HAL_UART_Transmit_DMA (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size)**

Function description Send an amount of data in DMA mode.

Parameters
- **huart:** UART handle.
- **pData:** Pointer to data buffer.
- **Size:** Amount of data to be sent.

Return values
- **HAL:** status

HAL_UART_Receive_DMA

Function name **HAL_StatusTypeDef HAL_UART_Receive_DMA (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size)**

Function description Receive an amount of data in DMA mode.

Parameters
- **huart:** UART handle.
- **pData:** Pointer to data buffer.
- **Size:** Amount of data to be received.

Return values
- **HAL:** status

Notes
- When the UART parity is enabled (PCE = 1), the received data contain the parity bit (MSB position).

HAL_UART_DMAPause

Function name **HAL_StatusTypeDef HAL_UART_DMAPause (UART_HandleTypeDef * huart)**

Function description Pause the DMA Transfer.

Parameters
- **huart:** UART handle.

Return values
- **HAL:** status

HAL_UART_DMAResume

Function name **HAL_StatusTypeDef HAL_UART_DMAResume (UART_HandleTypeDef * huart)**

Function description Resume the DMA Transfer.

Parameters
- **huart:** UART handle.

Return values
- **HAL:** status

HAL_UART_DMAStop

Function name **HAL_StatusTypeDef HAL_UART_DMAStop (UART_HandleTypeDef * huart)**

Function description Stop the DMA Transfer.

Parameters
- **huart:** UART handle.

Return values
- **HAL:** status

HAL_UART_Abort

Function name **HAL_StatusTypeDef HAL_UART_Abort (UART_HandleTypeDef * huart)**

Function description Abort ongoing transfers (blocking mode).

Parameters
- **huart:** UART handle.

Return values
- **HAL:** status

Notes
- This procedure could be used for aborting any ongoing transfer started in Interrupt or DMA mode. This procedure performs following operations: Disable UART Interrupts (Tx and Rx)Disable the DMA transfer in the peripheral register (if enabled)Abort DMA transfer by calling HAL_DMA_Abort (in case of transfer in DMA mode)Set handle State to READY
- This procedure is executed in blocking mode: when exiting function, Abort is considered as completed.

HAL_UART_AbortTransmit

Function name **HAL_StatusTypeDef HAL_UART_AbortTransmit (UART_HandleTypeDef * huart)**

Function description Abort ongoing Transmit transfer (blocking mode).

Parameters
- **huart:** UART handle.

Return values
- **HAL:** status

Notes
- This procedure could be used for aborting any ongoing Tx transfer started in Interrupt or DMA mode. This procedure performs following operations: Disable UART Interrupts (Tx)Disable the DMA transfer in the peripheral register (if enabled)Abort DMA transfer by calling HAL_DMA_Abort (in case of transfer in DMA mode)Set handle State to READY
- This procedure is executed in blocking mode: when exiting function, Abort is considered as completed.

HAL_UART_AbortReceive

Function name **HAL_StatusTypeDef HAL_UART_AbortReceive (UART_HandleTypeDef * huart)**

Function description Abort ongoing Receive transfer (blocking mode).

Parameters
- **huart:** UART handle.

Return values
- **HAL:** status

Notes
- This procedure could be used for aborting any ongoing Rx transfer started in Interrupt or DMA mode. This procedure performs following operations: Disable UART Interrupts (Rx)Disable the DMA transfer in the peripheral register (if enabled)Abort DMA transfer by calling HAL_DMA_Abort (in case of transfer in DMA mode)Set handle State to READY
- This procedure is executed in blocking mode: when exiting function, Abort is considered as completed.

HAL_UART_Abort_IT

Function name **HAL_StatusTypeDef HAL_UART_Abort_IT (UART_HandleTypeDef * huart)**

Function description Abort ongoing transfers (Interrupt mode).

Parameters
- **huart:** UART handle.

Return values
- **HAL:** status

Notes
- This procedure could be used for aborting any ongoing transfer started in Interrupt or DMA mode. This procedure performs following operations: Disable UART Interrupts (Tx and Rx)Disable the DMA transfer in the peripheral register (if enabled)Abort DMA transfer by calling HAL_DMA_Abort_IT (in case of transfer in DMA mode)Set handle State to READYAt abort completion, call user abort complete callback
- This procedure is executed in Interrupt mode, meaning that abort procedure could be considered as completed only when user abort complete callback is executed (not when exiting function).

HAL_UART_AbortTransmit_IT

Function name **HAL_StatusTypeDef HAL_UART_AbortTransmit_IT (UART_HandleTypeDef * huart)**

Function description Abort ongoing Transmit transfer (Interrupt mode).

Parameters
- **huart:** UART handle.

Return values
- **HAL:** status

Notes
- This procedure could be used for aborting any ongoing Tx transfer started in Interrupt or DMA mode. This procedure performs following operations: Disable UART Interrupts (Tx)Disable the DMA transfer in the peripheral register (if enabled)Abort DMA transfer by calling HAL_DMA_Abort_IT (in case of transfer in DMA mode)Set handle State to READYAt abort completion, call user abort complete callback
- This procedure is executed in Interrupt mode, meaning that abort procedure could be considered as completed only when user abort complete callback is executed (not when exiting function).

HAL_UART_AbortReceive_IT

Function name **HAL_StatusTypeDef HAL_UART_AbortReceive_IT**

(UART_HandleTypeDef * huart)

Function description: Abort ongoing Receive transfer (Interrupt mode).

Parameters:
- **huart:** UART handle.

Return values:
- **HAL:** status

Notes:
- This procedure could be used for aborting any ongoing Rx transfer started in Interrupt or DMA mode. This procedure performs following operations: Disable UART Interrupts (Rx)Disable the DMA transfer in the peripheral register (if enabled)Abort DMA transfer by calling HAL_DMA_Abort_IT (in case of transfer in DMA mode)Set handle State to READYAt abort completion, call user abort complete callback
- This procedure is executed in Interrupt mode, meaning that abort procedure could be considered as completed only when user abort complete callback is executed (not when exiting function).

HAL_UART_IRQHandler

Function name: **void HAL_UART_IRQHandler (UART_HandleTypeDef * huart)**

Function description: Handle UART interrupt request.

Parameters:
- **huart:** UART handle.

Return values:
- **None:**

HAL_UART_TxHalfCpltCallback

Function name: **void HAL_UART_TxHalfCpltCallback (UART_HandleTypeDef * huart)**

Function description: Tx Half Transfer completed callback.

Parameters:
- **huart:** UART handle.

Return values:
- **None:**

HAL_UART_TxCpltCallback

Function name: **void HAL_UART_TxCpltCallback (UART_HandleTypeDef * huart)**

Function description: Tx Transfer completed callback.

Parameters:
- **huart:** UART handle.

Return values:
- **None:**

HAL_UART_RxHalfCpltCallback

Function name: **void HAL_UART_RxHalfCpltCallback (UART_HandleTypeDef * huart)**

Function description: Rx Half Transfer completed callback.

Parameters:
- **huart:** UART handle.

Return values
- **None:**

HAL_UART_RxCpltCallback

Function name **void HAL_UART_RxCpltCallback (UART_HandleTypeDef * huart)**

Function description Rx Transfer completed callback.

Parameters
- **huart:** UART handle.

Return values
- **None:**

HAL_UART_ErrorCallback

Function name **void HAL_UART_ErrorCallback (UART_HandleTypeDef * huart)**

Function description UART error callback.

Parameters
- **huart:** UART handle.

Return values
- **None:**

HAL_UART_AbortCpltCallback

Function name **void HAL_UART_AbortCpltCallback (UART_HandleTypeDef * huart)**

Function description UART Abort Complete callback.

Parameters
- **huart:** UART handle.

Return values
- **None:**

HAL_UART_AbortTransmitCpltCallback

Function name **void HAL_UART_AbortTransmitCpltCallback (UART_HandleTypeDef * huart)**

Function description UART Abort Complete callback.

Parameters
- **huart:** UART handle.

Return values
- **None:**

HAL_UART_AbortReceiveCpltCallback

Function name **void HAL_UART_AbortReceiveCpltCallback (UART_HandleTypeDef * huart)**

Function description UART Abort Receive Complete callback.

Parameters
- **huart:** UART handle.

Return values
- **None:**

HAL_LIN_SendBreak

Function name **HAL_StatusTypeDef HAL_LIN_SendBreak**

(UART_HandleTypeDef * huart)

Function description: Transmit break characters.

Parameters:
- **huart:** UART handle.

Return values:
- **HAL:** status

HAL_MultiProcessor_EnableMuteMode

Function name: **HAL_StatusTypeDef HAL_MultiProcessor_EnableMuteMode (UART_HandleTypeDef * huart)**

Function description: Enable UART in mute mode (does not mean UART enters mute mode; to enter mute mode, HAL_MultiProcessor_EnterMuteMode() API must be called).

Parameters:
- **huart:** UART handle.

Return values:
- **HAL:** status

HAL_MultiProcessor_DisableMuteMode

Function name: **HAL_StatusTypeDef HAL_MultiProcessor_DisableMuteMode (UART_HandleTypeDef * huart)**

Function description: Disable UART mute mode (does not mean the UART actually exits mute mode as it may not have been in mute mode at this very moment).

Parameters:
- **huart:** UART handle.

Return values:
- **HAL:** status

HAL_MultiProcessor_EnterMuteMode

Function name: **void HAL_MultiProcessor_EnterMuteMode (UART_HandleTypeDef * huart)**

Function description: Enter UART mute mode (means UART actually enters mute mode).

Parameters:
- **huart:** UART handle.

Return values:
- **None:**

Notes:
- To exit from mute mode, HAL_MultiProcessor_DisableMuteMode() API must be called.

HAL_HalfDuplex_EnableTransmitter

Function name: **HAL_StatusTypeDef HAL_HalfDuplex_EnableTransmitter (UART_HandleTypeDef * huart)**

Function description: Enable the UART transmitter and disable the UART receiver.

Parameters:
- **huart:** UART handle.

Return values:
- **HAL:** status

HAL_HalfDuplex_EnableReceiver

Function name	**HAL_StatusTypeDef HAL_HalfDuplex_EnableReceiver (UART_HandleTypeDef * huart)**
Function description	Enable the UART receiver and disable the UART transmitter.
Parameters	• **huart:** UART handle.
Return values	• **HAL:** status.

HAL_UART_GetState

Function name	**HAL_UART_StateTypeDef HAL_UART_GetState (UART_HandleTypeDef * huart)**
Function description	Return the UART handle state.
Parameters	• **huart:** Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified UART.
Return values	• **HAL:** state

HAL_UART_GetError

Function name	**uint32_t HAL_UART_GetError (UART_HandleTypeDef * huart)**
Function description	Return the UART handle error code.
Parameters	• **huart:** Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified UART.
Return values	• **UART:** Error Code

UART_SetConfig

Function name	**HAL_StatusTypeDef UART_SetConfig (UART_HandleTypeDef * huart)**
Function description	Configure the UART peripheral.
Parameters	• **huart:** UART handle.
Return values	• **HAL:** status

UART_CheckIdleState

Function name	**HAL_StatusTypeDef UART_CheckIdleState (UART_HandleTypeDef * huart)**
Function description	Check the UART Idle State.
Parameters	• **huart:** UART handle.
Return values	• **HAL:** status

UART_WaitOnFlagUntilTimeout

Function name	**HAL_StatusTypeDef UART_WaitOnFlagUntilTimeout (UART_HandleTypeDef * huart, uint32_t Flag, FlagStatus Status, uint32_t Tickstart, uint32_t Timeout)**

Function description	Handle UART Communication Timeout.
Parameters	• **huart:** UART handle. • **Flag:** Specifies the UART flag to check • **Status:** Flag status (SET or RESET) • **Tickstart:** Tick start value • **Timeout:** Timeout duration
Return values	• **HAL:** status

UART_AdvFeatureConfig

Function name	**void UART_AdvFeatureConfig (UART_HandleTypeDef * huart)**
Function description	Configure the UART peripheral advanced features.
Parameters	• **huart:** UART handle.
Return values	• **None:**

68.3 UART Firmware driver defines

68.3.1 UART

UART Advanced Feature Initialization Type

UART_ADVFEATURE_NO_INIT	No advanced feature initialization
UART_ADVFEATURE_TXINVERT_INIT	TX pin active level inversion
UART_ADVFEATURE_RXINVERT_INIT	RX pin active level inversion
UART_ADVFEATURE_DATAINVERT_INIT	Binary data inversion
UART_ADVFEATURE_SWAP_INIT	TX/RX pins swap
UART_ADVFEATURE_RXOVERRUNDISABLE_INIT	RX overrun disable
UART_ADVFEATURE_DMADISABLEONERROR_INIT	DMA disable on Reception Error
UART_ADVFEATURE_AUTOBAUDRATE_INIT	Auto Baud rate detection initialization
UART_ADVFEATURE_MSBFIRST_INIT	Most significant bit sent/received first

UART Advanced Feature Auto BaudRate Enable

UART_ADVFEATURE_AUTOBAUDRATE_DISABLE	RX Auto Baud rate detection enable
UART_ADVFEATURE_AUTOBAUDRATE_ENABLE	RX Auto Baud rate detection disable

UART Advanced Feature AutoBaud Rate Mode

UART_ADVFEATURE_AUTOBAUDRATE_ONSTARTBIT	Auto Baud rate detection on start bit
UART_ADVFEATURE_AUTOBAUDRATE_ONFALLINGEDGE	Auto Baud rate detection on falling edge
UART_ADVFEATURE_AUTOBAUDRATE_ON0X7FFRAME	Auto Baud rate detection on 0x7F frame detection

UART_ADVFEATURE_AUTOBAUDRATE_ON0X55FRAME	Auto Baud rate detection on 0x55 frame detection

UART Clock Prescaler

UART_PRESCALER_DIV1	fclk_pres = fclk
UART_PRESCALER_DIV2	fclk_pres = fclk/2
UART_PRESCALER_DIV4	fclk_pres = fclk/4
UART_PRESCALER_DIV6	fclk_pres = fclk/6
UART_PRESCALER_DIV8	fclk_pres = fclk/8
UART_PRESCALER_DIV10	fclk_pres = fclk/10
UART_PRESCALER_DIV12	fclk_pres = fclk/12
UART_PRESCALER_DIV16	fclk_pres = fclk/16
UART_PRESCALER_DIV32	fclk_pres = fclk/32
UART_PRESCALER_DIV64	fclk_pres = fclk/64
UART_PRESCALER_DIV128	fclk_pres = fclk/128
UART_PRESCALER_DIV256	fclk_pres = fclk/256

UART Driver Enable Assertion Time LSB Position In CR1 Register

UART_CR1_DEAT_ADDRESS_LSB_POS	UART Driver Enable assertion time LSB position in CR1 register

UART Driver Enable DeAssertion Time LSB Position In CR1 Register

UART_CR1_DEDT_ADDRESS_LSB_POS	UART Driver Enable de-assertion time LSB position in CR1 register

UART Address-matching LSB Position In CR2 Register

UART_CR2_ADDRESS_LSB_POS	UART address-matching LSB position in CR2 register

UART Advanced Feature Binary Data Inversion

UART_ADVFEATURE_DATAINV_DISABLE	Binary data inversion disable
UART_ADVFEATURE_DATAINV_ENABLE	Binary data inversion enable

UART Advanced Feature DMA Disable On Rx Error

UART_ADVFEATURE_DMA_ENABLEONRXERROR	DMA enable on Reception Error
UART_ADVFEATURE_DMA_DISABLEONRXERROR	DMA disable on Reception Error

UART DMA Rx

UART_DMA_RX_DISABLE	UART DMA RX disabled
UART_DMA_RX_ENABLE	UART DMA RX enabled

UART DMA Tx

UART_DMA_TX_DISABLE	UART DMA TX disabled
UART_DMA_TX_ENABLE	UART DMA TX enabled

UART DriverEnable Polarity

UART_DE_POLARITY_HIGH	Driver enable signal is active high

UART_DE_POLARITY_LOW Driver enable signal is active low

UART Exported Macros

__HAL_UART_RESET_HANDLE_STATE

Description:

- Reset UART handle states.

Parameters:

- __HANDLE__: UART handle.

Return value:

- None

__HAL_UART_FLUSH_DRREGISTER

Description:

- Flush the UART Data registers.

Parameters:

- __HANDLE__: specifies the UART Handle.

Return value:

- None

__HAL_UART_CLEAR_FLAG

Description:

- Clear the specified UART pending flag.

Parameters:

- __HANDLE__: specifies the UART Handle.
- __FLAG__: specifies the flag to check. This parameter can be any combination of the following values:
  - UART_CLEAR_PEF Parity Error Clear Flag
  - UART_CLEAR_FEF Framing Error Clear Flag
  - UART_CLEAR_NEF Noise detected Clear Flag
  - UART_CLEAR_OREF Overrun Error Clear Flag
  - UART_CLEAR_IDLEF IDLE line detected Clear Flag
  - UART_CLEAR_TXFECF TXFIFO empty clear Flag
  - UART_CLEAR_TCF Transmission Complete Clear Flag
  - UART_CLEAR_LBDF LIN Break Detection Clear Flag
  - UART_CLEAR_CTSF CTS Interrupt Clear Flag
  - UART_CLEAR_CMF Character Match Clear Flag
  - UART_CLEAR_WUF Wake Up from stop mode Clear Flag

Return value:

- None

__HAL_UART_CLEAR_PEFLAG

Description:

- Clear the UART PE pending flag.

Parameters:

- __HANDLE__: specifies the UART Handle.

Return value:

- None

__HAL_UART_CLEAR_FEFLAG

Description:

- Clear the UART FE pending flag.

Parameters:

- __HANDLE__: specifies the UART Handle.

Return value:

- None

__HAL_UART_CLEAR_NEFLAG

Description:

- Clear the UART NE pending flag.

Parameters:

- __HANDLE__: specifies the UART Handle.

Return value:

- None

__HAL_UART_CLEAR_OREFLAG

Description:

- Clear the UART ORE pending flag.

Parameters:

- __HANDLE__: specifies the UART Handle.

Return value:

- None

__HAL_UART_CLEAR_IDLEFLAG

Description:

- Clear the UART IDLE pending flag.

Parameters:

- __HANDLE__: specifies the UART Handle.

Return value:

- None

__HAL_UART_CLEAR_TXFECF

Description:

- Clear the UART TX FIFO empty clear flag.

Parameters:

- __HANDLE__: specifies the UART Handle.

Return value:

- None

__HAL_UART_GET_FLAG

Description:

- Check whether the specified UART flag is set or not.

Parameters:

- __HANDLE__: specifies the UART Handle.
- __FLAG__: specifies the flag to check. This parameter can be one of the following values:
  - UART_FLAG_TXFT TXFIFO threshold flag
  - UART_FLAG_RXFT RXFIFO threshold flag
  - UART_FLAG_RXFF RXFIFO Full flag
  - UART_FLAG_TXFE TXFIFO Empty flag
  - UART_FLAG_REACK Receive enable acknowledge flag
  - UART_FLAG_TEACK Transmit enable acknowledge flag
  - UART_FLAG_WUF Wake up from stop mode flag
  - UART_FLAG_RWU Receiver wake up flag (if the UART in mute mode)
  - UART_FLAG_SBKF Send Break flag
  - UART_FLAG_CMF Character match flag
  - UART_FLAG_BUSY Busy flag
  - UART_FLAG_ABRF Auto Baud rate detection flag
  - UART_FLAG_ABRE Auto Baud rate detection error flag
  - UART_FLAG_CTS CTS Change flag
  - UART_FLAG_LBDF LIN Break detection flag
  - UART_FLAG_TXE Transmit data register empty flag
  - UART_FLAG_TXFNF UART TXFIFO not full flag
  - UART_FLAG_TC Transmission Complete flag
  - UART_FLAG_RXNE Receive data register not empty flag
  - UART_FLAG_RXFNE UART RXFIFO not empty flag
  - UART_FLAG_IDLE Idle Line detection flag
  - UART_FLAG_ORE Overrun Error flag
  - UART_FLAG_NE Noise Error flag
  - UART_FLAG_FE Framing Error flag
  - UART_FLAG_PE Parity Error flag

Return value:

- The: new state of __FLAG__ (TRUE or FALSE).

__HAL_UART_ENABLE_IT

Description:

- Enable the specified UART interrupt.

Parameters:

- __HANDLE__: specifies the UART Handle.
- __INTERRUPT__: specifies the UART interrupt source to enable. This parameter can be one of the following values:
  - UART_IT_RXFF RXFIFO Full interrupt
  - UART_IT_TXFE TXFIFO Empty interrupt
  - UART_IT_RXFT RXFIFO threshold interrupt
  - UART_IT_TXFT TXFIFO threshold interrupt
  - UART_IT_WUF Wakeup from stop mode interrupt
  - UART_IT_CM Character match interrupt
  - UART_IT_CTS CTS change interrupt
  - UART_IT_LBD LIN Break detection interrupt
  - UART_IT_TXE Transmit Data Register empty interrupt
  - UART_IT_TXFNF TX FIFO not full interrupt
  - UART_IT_TC Transmission complete interrupt
  - UART_IT_RXNE Receive Data register not empty interrupt
  - UART_IT_RXFNE RXFIFO not empty interrupt
  - UART_IT_IDLE Idle line detection interrupt
  - UART_IT_PE Parity Error interrupt
  - UART_IT_ERR Error interrupt (Frame error, noise error, overrun error)

Return value:

- None

__HAL_UART_DISABLE_IT

Description:

- Disable the specified UART interrupt.

Parameters:

- __HANDLE__: specifies the UART Handle.
- __INTERRUPT__: specifies the UART interrupt source to disable. This parameter can be one of the following values:

- UART_IT_RXFF RXFIFO Full interrupt
- UART_IT_TXFE TXFIFO Empty interrupt
- UART_IT_RXFT RXFIFO threshold interrupt
- UART_IT_TXFT TXFIFO threshold interrupt
- UART_IT_WUF Wakeup from stop mode interrupt
- UART_IT_CM Character match interrupt
- UART_IT_CTS CTS change interrupt
- UART_IT_LBD LIN Break detection interrupt
- UART_IT_TXE Transmit Data Register empty interrupt
- UART_IT_TXFNF TX FIFO not full interrupt
- UART_IT_TC Transmission complete interrupt
- UART_IT_RXNE Receive Data register not empty interrupt
- UART_IT_RXFNE RXFIFO not empty interrupt
- UART_IT_IDLE Idle line detection interrupt
- UART_IT_PE Parity Error interrupt
- UART_IT_ERR Error interrupt (Frame error, noise error, overrun error)

Return value:

- None

__HAL_UART_GET_IT

Description:

- Check whether the specified UART interrupt has occurred or not.

Parameters:

- __HANDLE__: specifies the UART Handle.
- __INTERRUPT__: specifies the UART interrupt to check. This parameter can be one of the following values:
  - UART_IT_RXFF RXFIFO Full interrupt
  - UART_IT_TXFE TXFIFO Empty interrupt
  - UART_IT_RXFT RXFIFO threshold interrupt
  - UART_IT_TXFT TXFIFO threshold interrupt
  - UART_IT_WUF Wakeup from stop mode interrupt
  - UART_IT_CM Character match interrupt
  - UART_IT_CTS CTS change interrupt

- UART_IT_LBD LIN Break detection interrupt
- UART_IT_TXE Transmit Data Register empty interrupt
- UART_IT_TXFNF TX FIFO not full interrupt
- UART_IT_TC Transmission complete interrupt
- UART_IT_RXNE Receive Data register not empty interrupt
- UART_IT_RXFNE RXFIFO not empty interrupt
- UART_IT_IDLE Idle line detection interrupt
- UART_IT_PE Parity Error interrupt
- UART_IT_ERR Error interrupt (Frame error, noise error, overrun error)

Return value:

- The: new state of __INTERRUPT__ (SET or RESET).

__HAL_UART_GET_IT_SOURCE

Description:

- Check whether the specified UART interrupt source is enabled or not.

Parameters:

- __HANDLE__: specifies the UART Handle.
- __INTERRUPT__: specifies the UART interrupt source to check. This parameter can be one of the following values:
  - UART_IT_RXFF RXFIFO Full interrupt
  - UART_IT_TXFE TXFIFO Empty interrupt
  - UART_IT_RXFT RXFIFO threshold interrupt
  - UART_IT_TXFT TXFIFO threshold interrupt
  - UART_IT_WUF Wakeup from stop mode interrupt
  - UART_IT_CM Character match interrupt
  - UART_IT_CTS CTS change interrupt
  - UART_IT_LBD LIN Break detection interrupt
  - UART_IT_TXE Transmit Data Register empty interrupt
  - UART_IT_TXFNF TX FIFO not full interrupt
  - UART_IT_TC Transmission complete interrupt
  - UART_IT_RXNE Receive Data register not empty interrupt
  - UART_IT_RXFNE RXFIFO not empty

interrupt
- UART_IT_IDLE Idle line detection interrupt
- UART_IT_PE Parity Error interrupt
- UART_IT_ERR Error interrupt (Frame error, noise error, overrun error)

Return value:

- The: new state of __INTERRUPT__ (SET or RESET).

__HAL_UART_CLEAR_IT

Description:

- Clear the specified UART ISR flag, in setting the proper ICR register flag.

Parameters:

- __HANDLE__: specifies the UART Handle.
- __IT_CLEAR__: specifies the interrupt clear register flag that needs to be set to clear the corresponding interrupt This parameter can be one of the following values:
  - UART_CLEAR_PEF Parity Error Clear Flag
  - UART_CLEAR_FEF Framing Error Clear Flag
  - UART_CLEAR_NEF Noise detected Clear Flag
  - UART_CLEAR_OREF Overrun Error Clear Flag
  - UART_CLEAR_IDLEF IDLE line detected Clear Flag
  - UART_CLEAR_TXFECF TXFIFO empty Clear Flag
  - UART_CLEAR_TCF Transmission Complete Clear Flag
  - UART_CLEAR_LBDF LIN Break Detection Clear Flag
  - UART_CLEAR_CTSF CTS Interrupt Clear Flag
  - UART_CLEAR_CMF Character Match Clear Flag
  - UART_CLEAR_WUF Wake Up from stop mode Clear Flag

Return value:

- None

__HAL_UART_SEND_REQ

Description:

- Set a specific UART request flag.

Parameters:

- __HANDLE__: specifies the UART Handle.
- __REQ__: specifies the request flag to set This parameter can be one of the following

values:
- UART_AUTOBAUD_REQUEST Auto-Baud Rate Request
- UART_SENDBREAK_REQUEST Send Break Request
- UART_MUTE_MODE_REQUEST Mute Mode Request
- UART_RXDATA_FLUSH_REQUEST Receive Data flush Request
- UART_TXDATA_FLUSH_REQUEST Transmit data flush Request

Return value:

- None

__HAL_UART_ONE_BIT_SAMPLE_ENABLE

Description:

- Enable the UART one bit sample method.

Parameters:

- __HANDLE__: specifies the UART Handle.

Return value:

- None

__HAL_UART_ONE_BIT_SAMPLE_DISABLE

Description:

- Disable the UART one bit sample method.

Parameters:

- __HANDLE__: specifies the UART Handle.

Return value:

- None

__HAL_UART_ENABLE

Description:

- Enable UART.

Parameters:

- __HANDLE__: specifies the UART Handle.

Return value:

- None

__HAL_UART_DISABLE

Description:

- Disable UART.

Parameters:

- __HANDLE__: specifies the UART Handle.

Return value:

- None

__HAL_UART_HWCONTROL_CTS_ENABLE

Description:

- Enable CTS flow control.

Parameters:

- __HANDLE__: specifies the UART Handle.

Return value:

- None

Notes:

- This macro allows to enable CTS hardware flow control for a given UART instance, without need to call HAL_UART_Init() function. As involving direct access to UART registers, usage of this macro should be fully endorsed by user. As macro is expected to be used for modifying CTS Hw flow control feature activation, without need for USART instance Deinit/Init, following conditions for macro call should be fulfilled: UART instance should have already been initialised (through call of HAL_UART_Init())macro could only be called when corresponding UART instance is disabled (i.e. __HAL_UART_DISABLE(__HANDLE__)) and should be followed by an Enable macro (i.e. __HAL_UART_ENABLE(__HANDLE__)).

__HAL_UART_HWCONTROL_CTS_DISABLE

Description:

- Disable CTS flow control.

Parameters:

- __HANDLE__: specifies the UART Handle.

Return value:

- None

Notes:

- This macro allows to disable CTS hardware flow control for a given UART instance, without need to call HAL_UART_Init() function. As involving direct access to UART registers, usage of this macro should be fully endorsed by user. As macro is expected to be used for modifying CTS Hw flow control feature activation, without need for USART instance Deinit/Init, following conditions for macro call should be fulfilled: UART instance should have already been initialised (through call of HAL_UART_Init())macro could only be called when corresponding UART instance is disabled (i.e. __HAL_UART_DISABLE(__HANDLE__)) and should be followed by an Enable macro

(i.e. __HAL_UART_ENABLE(__HANDLE__)).

__HAL_UART_HWCONTROL_RTS_ENABLE

Description:

- Enable RTS flow control.

Parameters:

- __HANDLE__: specifies the UART Handle.

Return value:

- None

Notes:

- This macro allows to enable RTS hardware flow control for a given UART instance, without need to call HAL_UART_Init() function. As involving direct access to UART registers, usage of this macro should be fully endorsed by user. As macro is expected to be used for modifying RTS Hw flow control feature activation, without need for USART instance Deinit/Init, following conditions for macro call should be fulfilled: UART instance should have already been initialised (through call of HAL_UART_Init())macro could only be called when corresponding UART instance is disabled (i.e. __HAL_UART_DISABLE(__HANDLE__)) and should be followed by an Enable macro (i.e. __HAL_UART_ENABLE(__HANDLE__)).

__HAL_UART_HWCONTROL_RTS_DISABLE

Description:

- Disable RTS flow control.

Parameters:

- __HANDLE__: specifies the UART Handle.

Return value:

- None

Notes:

- This macro allows to disable RTS hardware flow control for a given UART instance, without need to call HAL_UART_Init() function. As involving direct access to UART registers, usage of this macro should be fully endorsed by user. As macro is expected to be used for modifying RTS Hw flow control feature activation, without need for USART instance Deinit/Init, following conditions for macro call should be fulfilled: UART instance should have already been initialised (through call of HAL_UART_Init()

)macro could only be called when corresponding UART instance is disabled (i.e. __HAL_UART_DISABLE(__HANDLE__)) and should be followed by an Enable macro (i.e. __HAL_UART_ENABLE(__HANDLE__)).

UART Status Flags

UART_FLAG_TXFT	UART TXFIFO threshold flag
UART_FLAG_RXFT	UART RXFIFO threshold flag
UART_FLAG_RXFF	UART RXFIFO Full flag
UART_FLAG_TXFE	UART TXFIFO Empty flag
UART_FLAG_REACK	UART receive enable acknowledge flag
UART_FLAG_TEACK	UART transmit enable acknowledge flag
UART_FLAG_WUF	UART wake-up from stop mode flag
UART_FLAG_RWU	UART receiver wake-up from mute mode flag
UART_FLAG_SBKF	UART send break flag
UART_FLAG_CMF	UART character match flag
UART_FLAG_BUSY	UART busy flag
UART_FLAG_ABRF	UART auto Baud rate flag
UART_FLAG_ABRE	UART auto Baud rate error
UART_FLAG_CTS	UART clear to send flag
UART_FLAG_CTSIF	UART clear to send interrupt flag
UART_FLAG_LBDF	UART LIN break detection flag
UART_FLAG_TXE	UART transmit data register empty
UART_FLAG_TXFNF	UART TXFIFO not full
UART_FLAG_TC	UART transmission complete
UART_FLAG_RXNE	UART read data register not empty
UART_FLAG_RXFNE	UART RXFIFO not empty
UART_FLAG_IDLE	UART idle flag
UART_FLAG_ORE	UART overrun error
UART_FLAG_NE	UART noise error
UART_FLAG_FE	UART frame error
UART_FLAG_PE	UART parity error

UART Half Duplex Selection

UART_HALF_DUPLEX_DISABLE	UART half-duplex disabled
UART_HALF_DUPLEX_ENABLE	UART half-duplex enabled

UART Hardware Flow Control

UART_HWCONTROL_NONE	No hardware control
UART_HWCONTROL_RTS	Request To Send
UART_HWCONTROL_CTS	Clear To Send
UART_HWCONTROL_RTS_CTS	Request and Clear To Send

UART Interruptions Flag Mask

UART_IT_MASK	UART interruptions flags mask

UART Interrupts Definition

UART_IT_PE	UART parity error interruption
UART_IT_TXE	UART transmit data register empty interruption
UART_IT_TXFNF	UART TX FIFO not full interruption
UART_IT_TC	UART transmission complete interruption
UART_IT_RXNE	UART read data register not empty interruption
UART_IT_RXFNE	UART RXFIFO not empty interruption
UART_IT_IDLE	UART idle interruption
UART_IT_LBD	UART LIN break detection interruption
UART_IT_CTS	UART CTS interruption
UART_IT_CM	UART character match interruption
UART_IT_WUF	UART wake-up from stop mode interruption
UART_IT_RXFF	UART RXFIFO full interruption
UART_IT_TXFE	UART TXFIFO empty interruption
UART_IT_RXFT	UART RXFIFO threshold reached interruption
UART_IT_TXFT	UART TXFIFO threshold reached interruption
UART_IT_ERR	UART error interruption
UART_IT_ORE	UART overrun error interruption
UART_IT_NE	UART noise error interruption
UART_IT_FE	UART frame error interruption

UART Interruption Clear Flags

UART_CLEAR_PEF	Parity Error Clear Flag
UART_CLEAR_FEF	Framing Error Clear Flag
UART_CLEAR_NEF	Noise detected Clear Flag
UART_CLEAR_OREF	Overrun Error Clear Flag
UART_CLEAR_IDLEF	IDLE line detected Clear Flag
UART_CLEAR_TXFECF	TXFIFO empty clear flag
UART_CLEAR_TCF	Transmission Complete Clear Flag
UART_CLEAR_LBDF	LIN Break Detection Clear Flag

UART_CLEAR_CTSF CTS Interrupt Clear Flag

UART_CLEAR_CMF Character Match Clear Flag

UART_CLEAR_WUF Wake Up from stop mode Clear Flag

UART Local Interconnection Network mode

UART_LIN_DISABLE Local Interconnect Network disable

UART_LIN_ENABLE Local Interconnect Network enable

UART LIN Break Detection

UART_LINBREAKDETECTLENGTH_10B LIN 10-bit break detection length

UART_LINBREAKDETECTLENGTH_11B LIN 11-bit break detection length

UART Transfer Mode

UART_MODE_RX RX mode

UART_MODE_TX TX mode

UART_MODE_TX_RX RX and TX mode

UART Advanced Feature MSB First

UART_ADVFEATURE_MSBFIRST_DISABLE Most significant bit sent/received first disable

UART_ADVFEATURE_MSBFIRST_ENABLE Most significant bit sent/received first enable

UART Advanced Feature Mute Mode Enable

UART_ADVFEATURE_MUTEMODE_DISABLE UART mute mode disable

UART_ADVFEATURE_MUTEMODE_ENABLE UART mute mode enable

UART One Bit Sampling Method

UART_ONE_BIT_SAMPLE_DISABLE One-bit sampling disable

UART_ONE_BIT_SAMPLE_ENABLE One-bit sampling enable

UART Advanced Feature Overrun Disable

UART_ADVFEATURE_OVERRUN_ENABLE RX overrun enable

UART_ADVFEATURE_OVERRUN_DISABLE RX overrun disable

UART Over Sampling

UART_OVERSAMPLING_16 Oversampling by 16

UART_OVERSAMPLING_8 Oversampling by 8

UART Parity

UART_PARITY_NONE No parity

UART_PARITY_EVEN Even parity

UART_PARITY_ODD Odd parity

UART Receiver TimeOut

UART_RECEIVER_TIMEOUT_DISABLE UART receiver timeout disable

UART_RECEIVER_TIMEOUT_ENABLE UART receiver timeout enable

UART Request Parameters

UART_AUTOBAUD_REQUEST	Auto-Baud Rate Request
UART_SENDBREAK_REQUEST	Send Break Request
UART_MUTE_MODE_REQUEST	Mute Mode Request
UART_RXDATA_FLUSH_REQUEST	Receive Data flush Request
UART_TXDATA_FLUSH_REQUEST	Transmit data flush Request

UART Advanced Feature RX Pin Active Level Inversion

UART_ADVFEATURE_RXINV_DISABLE	RX pin active level inversion disable
UART_ADVFEATURE_RXINV_ENABLE	RX pin active level inversion enable

UART Advanced Feature RX TX Pins Swap

UART_ADVFEATURE_SWAP_DISABLE	TX/RX pins swap disable
UART_ADVFEATURE_SWAP_ENABLE	TX/RX pins swap enable

UART State

UART_STATE_DISABLE	UART disabled
UART_STATE_ENABLE	UART enabled

UART Number of Stop Bits

UART_STOPBITS_0_5	UART frame with 0.5 stop bit
UART_STOPBITS_1	UART frame with 1 stop bit
UART_STOPBITS_1_5	UART frame with 1.5 stop bits
UART_STOPBITS_2	UART frame with 2 stop bits

UART Advanced Feature Stop Mode Enable

UART_ADVFEATURE_STOPMODE_DISABLE	UART stop mode disable
UART_ADVFEATURE_STOPMODE_ENABLE	UART stop mode enable

UART polling-based communications time-out value

HAL_UART_TIMEOUT_VALUE	UART polling-based communications time-out value

UART Advanced Feature TX Pin Active Level Inversion

UART_ADVFEATURE_TXINV_DISABLE	TX pin active level inversion disable
UART_ADVFEATURE_TXINV_ENABLE	TX pin active level inversion enable

UART WakeUp From Stop Selection

UART_WAKEUP_ON_ADDRESS	UART wake-up on address
UART_WAKEUP_ON_STARTBIT	UART wake-up on start bit
UART_WAKEUP_ON_READDATA_NONEMPTY	UART wake-up on receive data register not empty or RXFIFO is not empty

UART WakeUp Methods

UART_WAKEUPMETHOD_IDLELINE	UART wake-up on idle line
UART_WAKEUPMETHOD_ADDRESSMARK	UART wake-up on address mark

69 HAL UART Extension Driver

69.1 UARTEx Firmware driver registers structures

69.1.1 UART_WakeUpTypeDef

Data Fields

- ***uint32_t WakeUpEvent***
- ***uint16_t AddressLength***
- ***uint8_t Address***

Field Documentation

- ***uint32_t UART_WakeUpTypeDef::WakeUpEvent***
  Specifies which event will activat the Wakeup from Stop mode flag (WUF). This parameter can be a value of ***UART_WakeUp_from_Stop_Selection***. If set to UART_WAKEUP_ON_ADDRESS, the two other fields below must be filled up.
- ***uint16_t UART_WakeUpTypeDef::AddressLength***
  Specifies whether the address is 4 or 7-bit long. This parameter can be a value of ***UARTEx_WakeUp_Address_Length***.
- ***uint8_t UART_WakeUpTypeDef::Address***
  UART/USART node address (7-bit long max).

69.2 UARTEx Firmware driver API description

69.2.1 UART peripheral extended features

69.2.2 Initialization and Configuration functions

This subsection provides a set of functions allowing to initialize the USARTx or the UARTy in asynchronous mode.

- For the asynchronous mode the parameters below can be configured:
  - Baud Rate
  - Word Length
  - Stop Bit
  - Parity: If the parity is enabled, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit.
  - Hardware flow control
  - Receiver/transmitter modes
  - Over Sampling Method
  - One-Bit Sampling Method
- For the asynchronous mode, the following advanced features can be configured as well:
  - TX and/or RX pin level inversion
  - data logical level inversion
  - RX and TX pins swap
  - RX overrun detection disabling
  - DMA disabling on RX error
  - MSB first on communication line
  - auto Baud rate detection

The HAL_RS485Ex_Init() API follows the UART RS485 mode configuration procedures (details for the procedures are available in reference manual).

This section contains the following APIs:

- ***HAL_RS485Ex_Init()***

69.2.3 IO operation functions

This section contains the following APIs:

- ***HAL_UARTEx_WakeupCallback()***
- ***HAL_UARTEx_RxFifoFullCallback()***
- ***HAL_UARTEx_TxFifoEmptyCallback()***

69.2.4 Peripheral Control functions

This section provides the following functions:

- HAL_UARTEx_EnableClockStopMode() API enables the UART clock (HSI or LSE only) during stop mode
- HAL_UARTEx_DisableClockStopMode() API disables the above functionality
- HAL_MultiProcessorEx_AddressLength_Set() API optionally sets the UART node address detection length to more than 4 bits for multiprocessor address mark wake up.
- HAL_UARTEx_StopModeWakeUpSourceConfig() API defines the wake-up from stop mode trigger: address match, Start Bit detection or RXNE bit status.
- HAL_UARTEx_EnableStopMode() API enables the UART to wake up the MCU from stop mode
- HAL_UARTEx_DisableStopMode() API disables the above functionality
- HAL_UARTEx_WakeupCallback() called upon UART wakeup interrupt
- HAL_UARTEx_EnableSPISlaveMode() API enables the SPI slave mode
- HAL_UARTEx_DisableSPISlaveMode() API disables the SPI slave mode
- HAL_UARTEx_ConfigNSS API configures the Slave Select input pin (NSS)
- HAL_UARTEx_EnableFifoMode() API enables the FIFO mode
- HAL_UARTEx_DisableFifoMode() API disables the FIFO mode
- HAL_UARTEx_SetTxFifoThreshold() API sets the TX FIFO threshold
- HAL_UARTEx_SetRxFifoThreshold() API sets the RX FIFO threshold

This section contains the following APIs:

- ***HAL_MultiProcessorEx_AddressLength_Set()***
- ***HAL_UARTEx_StopModeWakeUpSourceConfig()***
- ***HAL_UARTEx_EnableStopMode()***
- ***HAL_UARTEx_DisableStopMode()***
- ***HAL_UARTEx_EnableSlaveMode()***
- ***HAL_UARTEx_DisableSlaveMode()***
- ***HAL_UARTEx_ConfigNSS()***
- ***HAL_UARTEx_EnableFifoMode()***
- ***HAL_UARTEx_DisableFifoMode()***
- ***HAL_UARTEx_SetTxFifoThreshold()***
- ***HAL_UARTEx_SetRxFifoThreshold()***

69.2.5 Detailed description of functions

HAL_RS485Ex_Init

Function name **HAL_StatusTypeDef HAL_RS485Ex_Init (UART_HandleTypeDef * huart, uint32_t Polarity, uint32_t AssertionTime, uint32_t DeassertionTime)**

Function description Initialize the RS485 Driver enable feature according to the specified parameters in the UART_InitTypeDef and creates the associated handle.

Parameters
- **huart:** UART handle.
- **Polarity:** Select the driver enable polarity. This parameter can be one of the following values:
  - UART_DE_POLARITY_HIGH DE signal is active high
  - UART_DE_POLARITY_LOW DE signal is active low
- **AssertionTime:** Driver Enable assertion time: 5-bit value defining the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate)
- **DeassertionTime:** Driver Enable deassertion time: 5-bit value defining the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

Return values
- **HAL:** status

HAL_UARTEx_WakeupCallback

Function name **void HAL_UARTEx_WakeupCallback (UART_HandleTypeDef * huart)**

Function description UART wakeup from Stop mode callback.

Parameters
- **huart:** UART handle.

Return values
- **None:**

HAL_UARTEx_RxFifoFullCallback

Function name **void HAL_UARTEx_RxFifoFullCallback (UART_HandleTypeDef * huart)**

Function description UART RX Fifo full callback.

Parameters
- **huart:** UART handle.

Return values
- **None:**

HAL_UARTEx_TxFifoEmptyCallback

Function name **void HAL_UARTEx_TxFifoEmptyCallback (UART_HandleTypeDef * huart)**

Function description UART TX Fifo empty callback.

Parameters
- **huart:** UART handle.

Return values
- **None:**

HAL_UARTEx_StopModeWakeUpSourceConfig

Function name
HAL_StatusTypeDef HAL_UARTEx_StopModeWakeUpSourceConfig (UART_HandleTypeDef * huart, UART_WakeUpTypeDef WakeUpSelection)

Function description
Set Wakeup from Stop mode interrupt flag selection.

Parameters
- **huart:** UART handle.
- **WakeUpSelection:** Address match, Start Bit detection or RXNE/RXFNE bit status. This parameter can be one of the following values:
  - UART_WAKEUP_ON_ADDRESS
  - UART_WAKEUP_ON_STARTBIT
  - UART_WAKEUP_ON_READDATA_NONEMPTY

Return values
- **HAL:** status

Notes
- It is the application responsibility to enable the interrupt used as usart_wkup interrupt source before entering low-power mode.

HAL_UARTEx_EnableStopMode

Function name
HAL_StatusTypeDef HAL_UARTEx_EnableStopMode (UART_HandleTypeDef * huart)

Function description
Enable UART Stop Mode.

Parameters
- **huart:** UART handle.

Return values
- **HAL:** status

Notes
- The UART is able to wake up the MCU from Stop 1 mode as long as UART clock is HSI or LSE.

HAL_UARTEx_DisableStopMode

Function name
HAL_StatusTypeDef HAL_UARTEx_DisableStopMode (UART_HandleTypeDef * huart)

Function description
Disable UART Stop Mode.

Parameters
- **huart:** UART handle.

Return values
- **HAL:** status

HAL_MultiProcessorEx_AddressLength_Set

Function name
HAL_StatusTypeDef HAL_MultiProcessorEx_AddressLength_Set (UART_HandleTypeDef * huart, uint32_t AddressLength)

Function description
By default in multiprocessor mode, when the wake up method is set to address mark, the UART handles only 4-bit long addresses

detection; this API allows to enable longer addresses detection (6-, 7- or 8-bit long).

Parameters
- **huart:** UART handle.
- **AddressLength:** This parameter can be one of the following values:
  - UART_ADDRESS_DETECT_4B 4-bit long address
  - UART_ADDRESS_DETECT_7B 6-, 7- or 8-bit long address

Return values
- **HAL:** status

Notes
- Addresses detection lengths are: 6-bit address detection in 7-bit data mode, 7-bit address detection in 8-bit data mode, 8-bit address detection in 9-bit data mode.

HAL_UARTEx_EnableSlaveMode

Function name **HAL_StatusTypeDef HAL_UARTEx_EnableSlaveMode (UART_HandleTypeDef * huart)**

Function description Enable the SPI slave mode.

Parameters
- **huart:** UART handle.

Return values
- **HAL:** status

Notes
- When the UART operates in SPI slave mode, it handles data flow using the serial interface clock derived from the external SCLK signal provided by the external master SPI device.
- In SPI slave mode, the UART must be enabled before starting the master communications (or between frames while the clock is stable). Otherwise, if the UART slave is enabled while the master is in the middle of a frame, it will become desynchronized with the master.
- The data register of the slave needs to be ready before the first edge of the communication clock or before the end of the ongoing communication, otherwise the SPI slave will transmit zeros.

HAL_UARTEx_DisableSlaveMode

Function name **HAL_StatusTypeDef HAL_UARTEx_DisableSlaveMode (UART_HandleTypeDef * huart)**

Function description Disable the SPI slave mode.

Parameters
- **huart:** UART handle.

Return values
- **HAL:** status

HAL_UARTEx_ConfigNSS

Function name **HAL_StatusTypeDef HAL_UARTEx_ConfigNSS (UART_HandleTypeDef * huart, uint32_t NSSConfig)**

Function description Configure the Slave Select input pin (NSS).

Parameters
- **huart:** UART handle.
- **NSSConfig:** NSS configuration. This parameter can be one

of the following values:
  - UART_NSS_HARD
  - UART_NSS_SOFT

Return values
- **HAL:** status

Notes
- Software NSS management: SPI slave will always be selected and NSS input pin will be ignored.
- Hardware NSS management: the SPI slave selection depends on NSS input pin. The slave is selected when NSS is low and deselected when NSS is high.

HAL_UARTEx_EnableFifoMode

Function name **HAL_StatusTypeDef HAL_UARTEx_EnableFifoMode (UART_HandleTypeDef * huart)**

Function description Enable the FIFO mode.

Parameters
- **huart:** UART handle.

Return values
- **HAL:** status

HAL_UARTEx_DisableFifoMode

Function name **HAL_StatusTypeDef HAL_UARTEx_DisableFifoMode (UART_HandleTypeDef * huart)**

Function description Disable the FIFO mode.

Parameters
- **huart:** UART handle.

Return values
- **HAL:** status

HAL_UARTEx_SetTxFifoThreshold

Function name **HAL_StatusTypeDef HAL_UARTEx_SetTxFifoThreshold (UART_HandleTypeDef * huart, uint32_t Threshold)**

Function description Set the TXFIFO threshold.

Parameters
- **huart:** UART handle.
- **Threshold:** TX FIFO threshold value This parameter can be one of the following values:
  - UART_TXFIFO_THRESHOLD_1_8
  - UART_TXFIFO_THRESHOLD_1_4
  - UART_TXFIFO_THRESHOLD_1_2
  - UART_TXFIFO_THRESHOLD_3_4
  - UART_TXFIFO_THRESHOLD_7_8
  - UART_TXFIFO_THRESHOLD_8_8

Return values
- **HAL:** status

HAL_UARTEx_SetRxFifoThreshold

Function name **HAL_StatusTypeDef HAL_UARTEx_SetRxFifoThreshold (UART_HandleTypeDef * huart, uint32_t Threshold)**

Function description Set the RXFIFO threshold.

Parameters
- **huart:** UART handle.
- **Threshold:** RX FIFO threshold value This parameter can be one of the following values:
  - UART_RXFIFO_THRESHOLD_1_8
  - UART_RXFIFO_THRESHOLD_1_4
  - UART_RXFIFO_THRESHOLD_1_2
  - UART_RXFIFO_THRESHOLD_3_4
  - UART_RXFIFO_THRESHOLD_7_8
  - UART_RXFIFO_THRESHOLD_8_8

Return values
- **HAL:** status

69.3 UARTEx Firmware driver defines

69.3.1 UARTEx

UARTEx FIFO mode

UART_FIFOMODE_DISABLE FIFO mode disable

UART_FIFOMODE_ENABLE FIFO mode enable

UARTEx RXFIFO threshold level

UART_RXFIFO_THRESHOLD_1_8 RXFIFO FIFO reaches 1/8 of its depth

UART_RXFIFO_THRESHOLD_1_4 RXFIFO FIFO reaches 1/4 of its depth

UART_RXFIFO_THRESHOLD_1_2 RXFIFO FIFO reaches 1/2 of its depth

UART_RXFIFO_THRESHOLD_3_4 RXFIFO FIFO reaches 3/4 of its depth

UART_RXFIFO_THRESHOLD_7_8 RXFIFO FIFO reaches 7/8 of its depth

UART_RXFIFO_THRESHOLD_8_8 RXFIFO FIFO becomes full

UARTEx Synchronous Slave mode

UART_SLAVEMODE_DISABLE USART SPI Slave Mode Enable

UART_SLAVEMODE_ENABLE USART SPI Slave Mode Disable

UARTEx Slave Select Management

UART_NSS_HARD SPI slave selection depends on NSS input pin

UART_NSS_SOFT SPI slave is always selected and NSS input pin is ignored

UARTEx TXFIFO threshold level

UART_TXFIFO_THRESHOLD_1_8 TXFIFO reaches 1/8 of its depth

UART_TXFIFO_THRESHOLD_1_4 TXFIFO reaches 1/4 of its depth

UART_TXFIFO_THRESHOLD_1_2 TXFIFO reaches 1/2 of its depth

UART_TXFIFO_THRESHOLD_3_4 TXFIFO reaches 3/4 of its depth

UART_TXFIFO_THRESHOLD_7_8 TXFIFO reaches 7/8 of its depth

UART_TXFIFO_THRESHOLD_8_8 TXFIFO becomes empty

UARTEx WakeUp Address Length

UART_ADDRESS_DETECT_4B 4-bit long wake-up address

UART_ADDRESS_DETECT_7B 7-bit long wake-up address

UARTEx Word Length

UART_WORDLENGTH_7B 7-bit long UART frame

UART_WORDLENGTH_8B 8-bit long UART frame

UART_WORDLENGTH_9B 9-bit long UART frame

70 HAL USART Generic Driver

70.1 USART Firmware driver registers structures

70.1.1 USART_InitTypeDef

Data Fields

- ***uint32_t BaudRate***
- ***uint32_t WordLength***
- ***uint32_t StopBits***
- ***uint32_t Parity***
- ***uint32_t Mode***
- ***uint32_t CLKPolarity***
- ***uint32_t CLKPhase***
- ***uint32_t CLKLastBit***
- ***uint32_t ClockPrescaler***

Field Documentation

- ***uint32_t USART_InitTypeDef::BaudRate***
  This member configures the Usart communication baud rate. The baud rate is computed using the following formula: Baud Rate Register[15:4] = ((2 * fclk_pres) / ((huart->Init.BaudRate)))[15:4] Baud Rate Register[3] = 0 Baud Rate Register[2:0] = (((2 * fclk_pres) / ((huart->Init.BaudRate)))[3:0]) >> 1 where fclk_pres is the USART input clock frequency (fclk) divided by a prescaler.
  Note: Oversampling by 8 is systematically applied to achieve high baud rates.
- ***uint32_t USART_InitTypeDef::WordLength***
  Specifies the number of data bits transmitted or received in a frame. This parameter can be a value of ***USARTEx_Word_Length***.
- ***uint32_t USART_InitTypeDef::StopBits***
  Specifies the number of stop bits transmitted. This parameter can be a value of ***USART_Stop_Bits***.
- ***uint32_t USART_InitTypeDef::Parity***
  Specifies the parity mode. This parameter can be a value of ***USART_Parity***
  Note: When parity is enabled, the computed parity is inserted at the MSB position of the transmitted data (9th bit when the word length is set to 9 data bits; 8th bit when the word length is set to 8 data bits).
- ***uint32_t USART_InitTypeDef::Mode***
  Specifies whether the Receive or Transmit mode is enabled or disabled. This parameter can be a value of ***USART_Mode***.
- ***uint32_t USART_InitTypeDef::CLKPolarity***
  Specifies the steady state of the serial clock. This parameter can be a value of ***USART_Clock_Polarity***.
- ***uint32_t USART_InitTypeDef::CLKPhase***
  Specifies the clock transition on which the bit capture is made. This parameter can be a value of ***USART_Clock_Phase***.
- ***uint32_t USART_InitTypeDef::CLKLastBit***
  Specifies whether the clock pulse corresponding to the last transmitted data bit (MSB) has to be output on the SCLK pin in synchronous mode. This parameter can be a value of ***USART_Last_Bit***.