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***.

- ***uint32_t USART_InitTypeDef::ClockPrescaler***
  Specifies the prescaler value used to divide the USART clock source. This parameter can be a value of ***USART_ClockPrescaler***.

70.1.2 __USART_HandleTypeDef

Data Fields

- ***USART_TypeDef * Instance***
- ***USART_InitTypeDef Init***
- ***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_USART_StateTypeDef State***
- ***__IO uint32_t ErrorCode***

Field Documentation

- ***USART_TypeDef* __USART_HandleTypeDef::Instance***
  USART registers base address
- ***USART_InitTypeDef __USART_HandleTypeDef::Init***
  USART communication parameters
- ***uint8_t* __USART_HandleTypeDef::pTxBuffPtr***
  Pointer to USART Tx transfer Buffer
- ***uint16_t __USART_HandleTypeDef::TxXferSize***
  USART Tx Transfer size
- ***__IO uint16_t __USART_HandleTypeDef::TxXferCount***
  USART Tx Transfer Counter
- ***uint8_t* __USART_HandleTypeDef::pRxBuffPtr***
  Pointer to USART Rx transfer Buffer
- ***uint16_t __USART_HandleTypeDef::RxXferSize***
  USART Rx Transfer size
- ***__IO uint16_t __USART_HandleTypeDef::RxXferCount***
  USART Rx Transfer Counter
- ***uint16_t __USART_HandleTypeDef::Mask***
  USART Rx RDR register mask
- ***uint16_t __USART_HandleTypeDef::NbRxDataToProcess***
  Number of data to process during RX ISR execution
- ***uint16_t __USART_HandleTypeDef::NbTxDataToProcess***
  Number of data to process during TX ISR execution

- ***uint32_t __USART_HandleTypeDef::FifoMode***
  Specifies if the FIFO mode is being used. This parameter can be a value of ***USARTEx_FIFO_mode***.
- ***uint32_t __USART_HandleTypeDef::SlaveMode***
  Specifies if the UART SPI Slave mode is being used. This parameter can be a value of ***USARTEx_Slave_Mode***.
- ***void(* __USART_HandleTypeDef::RxISR)(struct __USART_HandleTypeDef *husart)***
  Function pointer on Rx IRQ handler
- ***void(* __USART_HandleTypeDef::TxISR)(struct __USART_HandleTypeDef *husart)***
  Function pointer on Tx IRQ handler
- ***DMA_HandleTypeDef* __USART_HandleTypeDef::hdmatx***
  USART Tx DMA Handle parameters
- ***DMA_HandleTypeDef* __USART_HandleTypeDef::hdmarx***
  USART Rx DMA Handle parameters
- ***HAL_LockTypeDef __USART_HandleTypeDef::Lock***
  Locking object
- ***__IO HAL_USART_StateTypeDef __USART_HandleTypeDef::State***
  USART communication state
- ***__IO uint32_t __USART_HandleTypeDef::ErrorCode***
  USART Error code

70.2 USART Firmware driver API description

70.2.1 How to use this driver

The USART HAL driver can be used as follows:

1. Declare a USART_HandleTypeDef handle structure (eg. USART_HandleTypeDef husart).
2. Initialize the USART low level resources by implementing the HAL_USART_MspInit() API:
   - Enable the USARTx interface clock.
   - USART pins configuration:
     - Enable the clock for the USART GPIOs.
     - Configure these USART pins as alternate function pull-up.
   - NVIC configuration if you need to use interrupt process (HAL_USART_Transmit_IT(), HAL_USART_Receive_IT() and HAL_USART_TransmitReceive_IT() APIs):
     - Configure the USARTx interrupt priority.
     - Enable the NVIC USART IRQ handle.
   - USART interrupts handling: The specific USART interrupts (Transmission complete interrupt, RXNE interrupt and Error Interrupts) will be managed using the macros __HAL_USART_ENABLE_IT() and __HAL_USART_DISABLE_IT() inside the transmit and receive process.
   - DMA Configuration if you need to use DMA process (HAL_USART_Transmit_DMA() HAL_USART_Receive_DMA() and HAL_USART_TransmitReceive_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 USART 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, and Mode (Receiver/Transmitter) in the husart handle Init structure.
4. Initialize the USART registers by calling the HAL_USART_Init() API:
   - This API configures also the low level Hardware GPIO, CLOCK, CORTEX...etc) by calling the customized HAL_USART_MspInit(&husart) API.

To configure and enable/disable the USART to wake up the MCU from stop mode, resort to UART API's HAL_UARTEx_StopModeWakeUpSourceConfig(), HAL_UARTEx_EnableStopMode() and HAL_UARTEx_DisableStopMode() in casting the USART handle to UART type UART_HandleTypeDef.

70.2.2 Initialization and Configuration functions

This subsection provides a set of functions allowing to initialize the USART in asynchronous and in synchronous modes.

- For the asynchronous mode only these parameters 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.
  - USART polarity
  - USART phase
  - USART LastBit
  - Receiver/transmitter modes

The HAL_USART_Init() function follows the USART synchronous configuration procedure (details for the procedure are available in reference manual).

This section contains the following APIs:

- ***HAL_USART_Init()***
- ***HAL_USART_DeInit()***
- ***HAL_USART_MspInit()***
- ***HAL_USART_MspDeInit()***

70.2.3 IO operation functions

This subsection provides a set of functions allowing to manage the USART synchronous data transfers.

The USART supports master mode only: it cannot receive or send data related to an input clock (SCLK is always an output).

1. There are two modes of transfer:
   - Blocking mode: The communication is performed in polling mode. The HAL status of all data processing is returned by the same function after finishing transfer.
   - No-Blocking mode: The communication is performed using Interrupts or DMA, These APIs return the HAL status. The end of the data processing will be indicated through the dedicated USART IRQ when using Interrupt mode or the DMA IRQ when using DMA mode. The HAL_USART_TxCpltCallback(), HAL_USART_RxCpltCallback() and HAL_USART_TxRxCpltCallback() user callbacks will be executed respectively at the end of the transmit or Receive

process The HAL_USART_ErrorCallback()user callback will be executed when a communication error is detected

2. Blocking mode APIs are:
   - HAL_USART_Transmit()in simplex mode
   - HAL_USART_Receive() in full duplex receive only
   - HAL_USART_TransmitReceive() in full duplex mode
3. Non-Blocking mode APIs with Interrupt are:
   - HAL_USART_Transmit_IT()in simplex mode
   - HAL_USART_Receive_IT() in full duplex receive only
   - HAL_USART_TransmitReceive_IT()in full duplex mode
   - HAL_USART_IRQHandler()
4. No-Blocking mode APIs with DMA are:
   - HAL_USART_Transmit_DMA()in simplex mode
   - HAL_USART_Receive_DMA() in full duplex receive only
   - HAL_USART_TransmitReceive_DMA() in full duplex mode
   - HAL_USART_DMAPause()
   - HAL_USART_DMAResume()
   - HAL_USART_DMAStop()
5. A set of Transfer Complete Callbacks are provided in Non_Blocking mode:
   - HAL_USART_TxCpltCallback()
   - HAL_USART_RxCpltCallback()
   - HAL_USART_TxHalfCpltCallback()
   - HAL_USART_RxHalfCpltCallback()
   - HAL_USART_ErrorCallback()
   - HAL_USART_TxRxCpltCallback()
6. Non-Blocking mode transfers could be aborted using Abort APIs:
   - HAL_USART_Abort()
   - HAL_USART_Abort_IT()
7. For Abort services based on interrupts (HAL_USART_Abort_IT), a Abort Complete Callbacks is provided: HAL_USART_AbortCpltCallback()
8. In Non-Blocking mode transfers, possible errors are split into 2 categories. Errors are handled as follows:
   - Error is considered as Recoverable and non blocking: Transfer could go till end, but error severity is to be evaluated by user: this concerns Frame Error, Parity Error or Noise Error in Interrupt mode reception . Received character is then retrieved and stored in Rx buffer, Error code is set to allow user to identify error type, and HAL_USART_ErrorCallback() user callback is executed. Transfer is kept ongoing on USART side. If user wants to abort it, Abort services should be called by user.
   - Error is considered as Blocking: Transfer could not be completed properly and is aborted. This concerns Overrun Error In Interrupt mode reception and all errors in DMA mode. Error code is set to allow user to identify error type, and HAL_USART_ErrorCallback() user callback is executed.

This section contains the following APIs:

- ***HAL_USART_Transmit()***
- ***HAL_USART_Receive()***
- ***HAL_USART_TransmitReceive()***
- ***HAL_USART_Transmit_IT()***
- ***HAL_USART_Receive_IT()***
- ***HAL_USART_TransmitReceive_IT()***
- ***HAL_USART_Transmit_DMA()***
- ***HAL_USART_Receive_DMA()***
- ***HAL_USART_TransmitReceive_DMA()***

- ***HAL_USART_DMAPause()***
- ***HAL_USART_DMAResume()***
- ***HAL_USART_DMAStop()***
- ***HAL_USART_Abort()***
- ***HAL_USART_Abort_IT()***
- ***HAL_USART_IRQHandler()***
- ***HAL_USART_TxCpltCallback()***
- ***HAL_USART_TxHalfCpltCallback()***
- ***HAL_USART_RxCpltCallback()***
- ***HAL_USART_RxHalfCpltCallback()***
- ***HAL_USART_TxRxCpltCallback()***
- ***HAL_USART_ErrorCallback()***
- ***HAL_USART_AbortCpltCallback()***

70.2.4 Peripheral State and Error functions

This subsection provides functions allowing to:

- Return the USART handle state
- Return the USART handle error code

This section contains the following APIs:

- ***HAL_USART_GetState()***
- ***HAL_USART_GetError()***

70.2.5 Detailed description of functions

HAL_USART_Init

Function name	**HAL_StatusTypeDef HAL_USART_Init (USART_HandleTypeDef * husart)**
Function description	Initialize the USART mode according to the specified parameters in the USART_InitTypeDef and initialize the associated handle.
Parameters	• **husart:** USART handle.
Return values	• **HAL:** status

HAL_USART_DeInit

Function name	**HAL_StatusTypeDef HAL_USART_DeInit (USART_HandleTypeDef * husart)**
Function description	DeInitialize the USART peripheral.
Parameters	• **husart:** USART handle.
Return values	• **HAL:** status

HAL_USART_MspInit

Function name	**void HAL_USART_MspInit (USART_HandleTypeDef * husart)**
Function description	Initialize the USART MSP.
Parameters	• **husart:** USART handle.
Return values	• **None:**

HAL_USART_MspDeInit

Function name **void HAL_USART_MspDeInit (USART_HandleTypeDef * husart)**

Function description DeInitialize the USART MSP.

Parameters
- **husart:** USART handle.

Return values
- **None:**

HAL_USART_Transmit

Function name **HAL_StatusTypeDef HAL_USART_Transmit (USART_HandleTypeDef * husart, uint8_t * pTxData, uint16_t Size, uint32_t Timeout)**

Function description Simplex send an amount of data in blocking mode.

Parameters
- **husart:** USART handle.
- **pTxData:** Pointer to data buffer.
- **Size:** Amount of data to be sent.
- **Timeout:** Timeout duration.

Return values
- **HAL:** status

HAL_USART_Receive

Function name **HAL_StatusTypeDef HAL_USART_Receive (USART_HandleTypeDef * husart, uint8_t * pRxData, uint16_t Size, uint32_t Timeout)**

Function description Receive an amount of data in blocking mode.

Parameters
- **husart:** USART handle.
- **pRxData:** Pointer to data buffer.
- **Size:** Amount of data to be received.
- **Timeout:** Timeout duration.

Return values
- **HAL:** status

Notes
- To receive synchronous data, dummy data are simultaneously transmitted.

HAL_USART_TransmitReceive

Function name **HAL_StatusTypeDef HAL_USART_TransmitReceive (USART_HandleTypeDef * husart, uint8_t * pTxData, uint8_t * pRxData, uint16_t Size, uint32_t Timeout)**

Function description Full-Duplex Send and Receive an amount of data in blocking mode.

Parameters
- **husart:** USART handle.
- **pTxData:** pointer to TX data buffer.
- **pRxData:** pointer to RX data buffer.
- **Size:** amount of data to be sent (same amount to be received).
- **Timeout:** Timeout duration.

Return values
- **HAL:** status

HAL_USART_Transmit_IT

Function name **HAL_StatusTypeDef HAL_USART_Transmit_IT (USART_HandleTypeDef * husart, uint8_t * pTxData, uint16_t Size)**

Function description Send an amount of data in interrupt mode.

Parameters
- **husart:** USART handle.
- **pTxData:** pointer to data buffer.
- **Size:** amount of data to be sent.

Return values
- **HAL:** status

HAL_USART_Receive_IT

Function name **HAL_StatusTypeDef HAL_USART_Receive_IT (USART_HandleTypeDef * husart, uint8_t * pRxData, uint16_t Size)**

Function description Receive an amount of data in blocking mode.

Parameters
- **husart:** USART handle.
- **pRxData:** pointer to data buffer.
- **Size:** amount of data to be received.

Return values
- **HAL:** status

Notes
- To receive synchronous data, dummy data are simultaneously transmitted.

HAL_USART_TransmitReceive_IT

Function name **HAL_StatusTypeDef HAL_USART_TransmitReceive_IT (USART_HandleTypeDef * husart, uint8_t * pTxData, uint8_t * pRxData, uint16_t Size)**

Function description Full-Duplex Send and Receive an amount of data in interrupt mode.

Parameters
- **husart:** USART handle.
- **pTxData:** pointer to TX data buffer.
- **pRxData:** pointer to RX data buffer.
- **Size:** amount of data to be sent (same amount to be received).

Return values
- **HAL:** status

HAL_USART_Transmit_DMA

Function name **HAL_StatusTypeDef HAL_USART_Transmit_DMA (USART_HandleTypeDef * husart, uint8_t * pTxData, uint16_t Size)**

Function description Send an amount of data in DMA mode.

Parameters
- **husart:** USART handle.

- **pTxData:** pointer to data buffer.
- **Size:** amount of data to be sent.

Return values
- **HAL:** status

HAL_USART_Receive_DMA

Function name **HAL_StatusTypeDef HAL_USART_Receive_DMA (USART_HandleTypeDef * husart, uint8_t * pRxData, uint16_t Size)**

Function description Receive an amount of data in DMA mode.

Parameters
- **husart:** USART handle.
- **pRxData:** pointer to data buffer.
- **Size:** amount of data to be received.

Return values
- **HAL:** status

Notes
- When the USART parity is enabled (PCE = 1), the received data contain the parity bit (MSB position).
- The USART DMA transmit channel must be configured in order to generate the clock for the slave.

HAL_USART_TransmitReceive_DMA

Function name **HAL_StatusTypeDef HAL_USART_TransmitReceive_DMA (USART_HandleTypeDef * husart, uint8_t * pTxData, uint8_t * pRxData, uint16_t Size)**

Function description Full-Duplex Transmit Receive an amount of data in non-blocking mode.

Parameters
- **husart:** USART handle.
- **pTxData:** pointer to TX data buffer.
- **pRxData:** pointer to RX data buffer.
- **Size:** amount of data to be received/sent.

Return values
- **HAL:** status

Notes
- When the USART parity is enabled (PCE = 1) the data received contain the parity bit.

HAL_USART_DMAPause

Function name **HAL_StatusTypeDef HAL_USART_DMAPause (USART_HandleTypeDef * husart)**

Function description Pause the DMA Transfer.

Parameters
- **husart:** USART handle.

Return values
- **HAL:** status

HAL_USART_DMAResume

Function name **HAL_StatusTypeDef HAL_USART_DMAResume (USART_HandleTypeDef * husart)**

Function description	Resume the DMA Transfer.
Parameters	• **husart:** USART handle.
Return values	• **HAL:** status

HAL_USART_DMAStop

Function name	**HAL_StatusTypeDef HAL_USART_DMAStop (USART_HandleTypeDef * husart)**
Function description	Stop the DMA Transfer.
Parameters	• **husart:** USART handle.
Return values	• **HAL:** status

HAL_USART_Abort

Function name	**HAL_StatusTypeDef HAL_USART_Abort (USART_HandleTypeDef * husart)**
Function description	Abort ongoing transfers (blocking mode).
Parameters	• **husart:** USART 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 USART 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_USART_Abort_IT

Function name	**HAL_StatusTypeDef HAL_USART_Abort_IT (USART_HandleTypeDef * husart)**
Function description	Abort ongoing transfers (Interrupt mode).
Parameters	• **husart:** USART 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 USART 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_USART_IRQHandler

Function name **void HAL_USART_IRQHandler (USART_HandleTypeDef * husart)**

Function description Handle USART interrupt request.

Parameters
- **husart:** USART handle.

Return values
- **None:**

HAL_USART_TxHalfCpltCallback

Function name **void HAL_USART_TxHalfCpltCallback (USART_HandleTypeDef * husart)**

Function description Tx Half Transfer completed callback.

Parameters
- **husart:** USART handle.

Return values
- **None:**

HAL_USART_TxCpltCallback

Function name **void HAL_USART_TxCpltCallback (USART_HandleTypeDef * husart)**

Function description Tx Transfer completed callback.

Parameters
- **husart:** USART handle.

Return values
- **None:**

HAL_USART_RxCpltCallback

Function name **void HAL_USART_RxCpltCallback (USART_HandleTypeDef * husart)**

Function description Rx Transfer completed callback.

Parameters
- **husart:** USART handle.

Return values
- **None:**

HAL_USART_RxHalfCpltCallback

Function name **void HAL_USART_RxHalfCpltCallback (USART_HandleTypeDef * husart)**

Function description Rx Half Transfer completed callback.

Parameters
- **husart:** USART handle.

Return values
- **None:**

HAL_USART_TxRxCpltCallback

Function name **void HAL_USART_TxRxCpltCallback (USART_HandleTypeDef * husart)**

Function description Tx/Rx Transfers completed callback for the non-blocking process.

Parameters
- **husart:** USART handle.

Return values
- **None:**

HAL_USART_ErrorCallback

Function name **void HAL_USART_ErrorCallback (USART_HandleTypeDef * husart)**

Function description USART error callback.

Parameters
- **husart:** USART handle.

Return values
- **None:**

HAL_USART_AbortCpltCallback

Function name **void HAL_USART_AbortCpltCallback (USART_HandleTypeDef * husart)**

Function description USART Abort Complete callback.

Parameters
- **husart:** USART handle.

Return values
- **None:**

HAL_USART_GetState

Function name **HAL_USART_StateTypeDef HAL_USART_GetState (USART_HandleTypeDef * husart)**

Function description Return the USART handle state.

Parameters
- **husart:** pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART.

Return values
- **USART:** handle state

HAL_USART_GetError

Function name **uint32_t HAL_USART_GetError (USART_HandleTypeDef * husart)**

Function description Return the USART error code.

Parameters
- **husart:** pointer to a USART_HandleTypeDef structure that contains the configuration information for the specified USART.

Return values
- **USART:** handle Error Code

70.3 USART Firmware driver defines

70.3.1 USART

USART Clock

USART_CLOCK_DISABLE USART clock disable

USART_CLOCK_ENABLE USART clock enable

USART Clock Prescaler

USART_PRESCALER_DIV1 fclk_pres = fclk

USART_PRESCALER_DIV2 fclk_pres = fclk/2

USART_PRESCALER_DIV4 fclk_pres = fclk/4

USART_PRESCALER_DIV6 fclk_pres = fclk/6

USART_PRESCALER_DIV8 fclk_pres = fclk/8

USART_PRESCALER_DIV10 fclk_pres = fclk/10

USART_PRESCALER_DIV12 fclk_pres = fclk/12

USART_PRESCALER_DIV16 fclk_pres = fclk/16

USART_PRESCALER_DIV32 fclk_pres = fclk/32

USART_PRESCALER_DIV64 fclk_pres = fclk/64

USART_PRESCALER_DIV128 fclk_pres = fclk/128

USART_PRESCALER_DIV256 fclk_pres = fclk/256

USART Clock Phase

USART_PHASE_1EDGE USART frame phase on first clock transition

USART_PHASE_2EDGE USART frame phase on second clock transition

USART Clock Polarity

USART_POLARITY_LOW Driver enable signal is active high

USART_POLARITY_HIGH Driver enable signal is active low

USART Exported Macros

__HAL_USART_RESET_HANDLE_STATE

Description:

- Reset USART handle state.

Parameters:

- __HANDLE__: USART handle.

Return value:

- None

__HAL_USART_GET_FLAG

Description:

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

Parameters:

- __HANDLE__: specifies the USART Handle
- __FLAG__: specifies the flag to check. This parameter can be one of the following values:
  - USART_FLAG_TXFT TXFIFO threshold flag
  - USART_FLAG_RXFT RXFIFO threshold flag
  - USART_FLAG_RXFF RXFIFO Full flag

- USART_FLAG_TXFE TXFIFO Empty flag
- USART_FLAG_REACK Receive enable acknowledge flag
- USART_FLAG_TEACK Transmit enable acknowledge flag
- USART_FLAG_BUSY Busy flag
- USART_FLAG_UDR SPI slave underrun error flag
- USART_FLAG_TXE Transmit data register empty flag
- USART_FLAG_TXFNF TXFIFO not full flag
- USART_FLAG_TC Transmission Complete flag
- USART_FLAG_RXNE Receive data register not empty flag
- USART_FLAG_RXFNE RXFIFO not empty flag
- USART_FLAG_IDLE Idle Line detection flag
- USART_FLAG_ORE OverRun Error flag
- USART_FLAG_NE Noise Error flag
- USART_FLAG_FE Framing Error flag
- USART_FLAG_PE Parity Error flag

Return value:

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

__HAL_USART_CLEAR_FLAG

Description:

- Clear the specified USART pending flag.

Parameters:

- __HANDLE__: specifies the USART Handle.
- __FLAG__: specifies the flag to check. This parameter can be any combination of the following values:
  - USART_CLEAR_PEF Parity Error Clear Flag
  - USART_CLEAR_FEF Framing Error Clear Flag
  - USART_CLEAR_NEF Noise detected Clear Flag
  - USART_CLEAR_OREF Overrun Error Clear Flag
  - USART_CLEAR_IDLEF IDLE line detected Clear Flag
  - USART_CLEAR_TXFECF TXFIFO empty clear Flag
  - USART_CLEAR_TCF Transmission Complete Clear Flag
  - USART_CLEAR_UDRF SPI slave underrun error Clear Flag

Return value:

- None

__HAL_USART_CLEAR_PEFLAG

Description:

- Clear the USART PE pending flag.

Parameters:

- __HANDLE__: specifies the USART Handle.

Return value:

- None

__HAL_USART_CLEAR_FEFLAG

Description:

- Clear the USART FE pending flag.

Parameters:

- __HANDLE__: specifies the USART Handle.

Return value:

- None

__HAL_USART_CLEAR_NEFLAG

Description:

- Clear the USART NE pending flag.

Parameters:

- __HANDLE__: specifies the USART Handle.

Return value:

- None

__HAL_USART_CLEAR_OREFLAG

Description:

- Clear the USART ORE pending flag.

Parameters:

- __HANDLE__: specifies the USART Handle.

Return value:

- None

__HAL_USART_CLEAR_IDLEFLAG

Description:

- Clear the USART IDLE pending flag.

Parameters:

- __HANDLE__: specifies the USART Handle.

Return value:

- None

__HAL_USART_CLEAR_TXFECF

Description:

- Clear the USART TX FIFO empty clear flag.

Parameters:

- __HANDLE__: specifies the USART Handle.

Return value:

- None

__HAL_USART_CLEAR_UDRFLAG

Description:

- Clear SPI slave underrun error flag.

Parameters:

- __HANDLE__: specifies the USART Handle.

Return value:

- None

__HAL_USART_ENABLE_IT

Description:

- Enable the specified USART interrupt.

Parameters:

- __HANDLE__: specifies the USART Handle.
- __INTERRUPT__: specifies the USART interrupt source to enable. This parameter can be one of the following values:
  - USART_IT_RXFF RXFIFO Full interrupt
  - USART_IT_TXFE TXFIFO Empty interrupt
  - USART_IT_RXFT RXFIFO threshold interrupt
  - USART_IT_TXFT TXFIFO threshold interrupt
  - USART_IT_TXE Transmit Data Register empty interrupt
  - USART_IT_TXFNF TX FIFO not full interrupt
  - USART_IT_TC Transmission complete interrupt
  - USART_IT_RXNE Receive Data register not empty interrupt
  - USART_IT_RXFNE RXFIFO not empty interrupt
  - USART_IT_IDLE Idle line detection interrupt
  - USART_IT_PE Parity Error interrupt
  - USART_IT_ERR Error interrupt(Frame error, noise error, overrun error)

Return value:

- None

__HAL_USART_DISABLE_IT

Description:

- Disable the specified USART interrupt.

Parameters:

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

- USART_IT_RXFF RXFIFO Full interrupt
- USART_IT_TXFE TXFIFO Empty interrupt
- USART_IT_RXFT RXFIFO threshold interrupt
- USART_IT_TXFT TXFIFO threshold interrupt
- USART_IT_TXE Transmit Data Register empty interrupt
- USART_IT_TXFNF TX FIFO not full interrupt
- USART_IT_TC Transmission complete interrupt
- USART_IT_RXNE Receive Data register not empty interrupt
- USART_IT_RXFNE RXFIFO not empty interrupt
- USART_IT_IDLE Idle line detection interrupt
- USART_IT_PE Parity Error interrupt
- USART_IT_ERR Error interrupt(Frame error, noise error, overrun error)

Return value:

- None

__HAL_USART_GET_IT

Description:

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

Parameters:

- __HANDLE__: specifies the USART Handle.
- __INTERRUPT__: specifies the USART interrupt source to check. This parameter can be one of the following values:
  - USART_IT_RXFF RXFIFO Full interrupt
  - USART_IT_TXFE TXFIFO Empty interrupt
  - USART_IT_RXFT RXFIFO threshold interrupt
  - USART_IT_TXFT TXFIFO threshold interrupt
  - USART_IT_TXE Transmit Data Register empty interrupt
  - USART_IT_TXFNF TX FIFO not full interrupt
  - USART_IT_TC Transmission complete interrupt
  - USART_IT_RXNE Receive Data register not empty interrupt
  - USART_IT_RXFNE RXFIFO not empty interrupt
  - USART_IT_IDLE Idle line detection interrupt
  - USART_IT_ORE OverRun Error interrupt
  - USART_IT_NE Noise Error interrupt

– USART_IT_FE Framing Error interrupt
– USART_IT_PE Parity Error interrupt

Return value:

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

__HAL_USART_GET_IT_SOURCE

Description:

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

Parameters:

- __HANDLE__: specifies the USART Handle.
- __INTERRUPT__: specifies the USART interrupt source to check. This parameter can be one of the following values:
  – USART_IT_RXFF RXFIFO Full interrupt
  – USART_IT_TXFE TXFIFO Empty interrupt
  – USART_IT_RXFT RXFIFO threshold interrupt
  – USART_IT_TXFT TXFIFO threshold interrupt
  – USART_IT_TXE Transmit Data Register empty interrupt
  – USART_IT_TXFNF TX FIFO not full interrupt
  – USART_IT_TC Transmission complete interrupt
  – USART_IT_RXNE Receive Data register not empty interrupt
  – USART_IT_RXFNE RXFIFO not empty interrupt
  – USART_IT_IDLE Idle line detection interrupt
  – USART_IT_ORE OverRun Error interrupt
  – USART_IT_NE Noise Error interrupt
  – USART_IT_FE Framing Error interrupt
  – USART_IT_PE Parity Error interrupt

Return value:

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

__HAL_USART_CLEAR_IT

Description:

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

Parameters:

- __HANDLE__: specifies the USART 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:
  – USART_CLEAR_PEF Parity Error Clear

Flag
- USART_CLEAR_FEF Framing Error Clear Flag
- USART_CLEAR_NEF Noise detected Clear Flag
- USART_CLEAR_OREF Overrun Error Clear Flag
- USART_CLEAR_IDLEF IDLE line detected Clear Flag
- USART_CLEAR_TXFECF TXFIFO empty clear Flag
- USART_CLEAR_TCF Transmission Complete Clear Flag

Return value:

- None

__HAL_USART_SEND_REQ

Description:

- Set a specific USART request flag.

Parameters:

- __HANDLE__: specifies the USART Handle.
- __REQ__: specifies the request flag to set. This parameter can be one of the following values:
  - USART_RXDATA_FLUSH_REQUEST Receive Data flush Request
  - USART_TXDATA_FLUSH_REQUEST Transmit data flush Request

Return value:

- None

__HAL_USART_ONE_BIT_SAMPLE_ ENABLE

Description:

- Enable the USART one bit sample method.

Parameters:

- __HANDLE__: specifies the USART Handle.

Return value:

- None

__HAL_USART_ONE_BIT_SAMPLE_ DISABLE

Description:

- Disable the USART one bit sample method.

Parameters:

- __HANDLE__: specifies the USART Handle.

Return value:

- None

__HAL_USART_ENABLE

Description:

- Enable USART.

Parameters:

- __HANDLE__: specifies the USART Handle.

Return value:

- None

__HAL_USART_DISABLE

Description:

- Disable USART.

Parameters:

- __HANDLE__: specifies the USART Handle.

Return value:

- None

USART Flags

USART_FLAG_TXFT	USART TXFIFO threshold flag
USART_FLAG_RXFT	USART RXFIFO threshold flag
USART_FLAG_RXFF	USART RXFIFO Full flag
USART_FLAG_TXFE	USART TXFIFO Empty flag
USART_FLAG_TXE	USART transmit data register empty
USART_FLAG_TXFNF	USART TXFIFO not full
USART_FLAG_RXNE	USART read data register not empty
USART_FLAG_RXFNE	USART RXFIFO not empty
USART_FLAG_REACK	USART receive enable acknowledge flag
USART_FLAG_TEACK	USART transmit enable acknowledge flag
USART_FLAG_BUSY	USART busy flag
USART_FLAG_TC	USART transmission complete
USART_FLAG_IDLE	USART idle flag
USART_FLAG_ORE	USART overrun error
USART_FLAG_NE	USART noise error
USART_FLAG_FE	USART frame error
USART_FLAG_PE	USART parity error
USART_FLAG_UDR	SPI slave underrun error flag

USART Interruption Flags Mask

USART_IT_MASK	USART interruptions flags mask

USART Interrupts Definition

USART_IT_PE	USART parity error interruption
USART_IT_TXFNF	USART TX FIFO not full interruption
USART_IT_RXFNE	USART RXFIFO not empty interruption
USART_IT_RXFF	USART RXFIFO full interruption
USART_IT_TXFE	USART TXFIFO empty interruption

USART_IT_RXFT	USART RXFIFO threshold reached interruption
USART_IT_TXFT	USART TXFIFO threshold reached interruption
USART_IT_TXE	USART transmit data register empty interruption
USART_IT_TC	USART transmission complete interruption
USART_IT_RXNE	USART read data register not empty interruption
USART_IT_IDLE	USART idle interruption
USART_IT_ERR	USART error interruption
USART_IT_ORE	USART overrun error interruption
USART_IT_NE	USART noise error interruption
USART_IT_FE	USART frame error interruption

USART Interruption Clear Flags

USART_CLEAR_PEF	Parity Error Clear Flag
USART_CLEAR_FEF	Framing Error Clear Flag
USART_CLEAR_NEF	Noise detected Clear Flag
USART_CLEAR_OREF	OverRun Error Clear Flag
USART_CLEAR_IDLEF	IDLE line detected Clear Flag
USART_CLEAR_TCF	Transmission Complete Clear Flag
USART_CLEAR_TXFECF	TXFIFO Empty Clear Flag
USART_CLEAR_UDRF	SPI slave underrun error Clear Flag

USART Last Bit

USART_LASTBIT_DISABLE	USART frame last data bit clock pulse not output to SCLK pin
USART_LASTBIT_ENABLE	USART frame last data bit clock pulse output to SCLK pin

USART Mode

USART_MODE_RX	RX mode
USART_MODE_TX	TX mode
USART_MODE_TX_RX	RX and TX mode

USART Over Sampling

USART_OVERSAMPLING_16	Oversampling by 16
USART_OVERSAMPLING_8	Oversampling by 8

USART Parity

USART_PARITY_NONE	No parity
USART_PARITY_EVEN	Even parity
USART_PARITY_ODD	Odd parity

USART Request Parameters

USART_RXDATA_FLUSH_REQUEST	Receive Data flush Request
USART_TXDATA_FLUSH_REQUEST	Transmit data flush Request

USART Number of Stop Bits

USART_STOPBITS_0_5	USART frame with 0.5 stop bit
USART_STOPBITS_1	USART frame with 1 stop bit
USART_STOPBITS_1_5	USART frame with 1.5 stop bits
USART_STOPBITS_2	USART frame with 2 stop bits

71 HAL USART Extension Driver

71.1 USARTEx Firmware driver API description

71.1.1 USART peripheral extended features

71.1.2 IO operation functions

This section contains the following APIs:

- ***HAL_USARTEx_RxFifoFullCallback()***
- ***HAL_USARTEx_TxFifoEmptyCallback()***

71.1.3 Peripheral Control functions

This section provides the following functions:

- HAL_USARTEx_EnableSPISlaveMode() API enables the SPI slave mode
- HAL_USARTEx_DisableSPISlaveMode() API disables the SPI slave mode
- HAL_USARTEx_ConfigNSS API configures the Slave Select input pin (NSS)
- HAL_USARTEx_EnableFifoMode() API enables the FIFO mode
- HAL_USARTEx_DisableFifoMode() API disables the FIFO mode
- HAL_USARTEx_SetTxFifoThreshold() API sets the TX FIFO threshold
- HAL_USARTEx_SetRxFifoThreshold() API sets the RX FIFO threshold

This section contains the following APIs:

- ***HAL_USARTEx_EnableSlaveMode()***
- ***HAL_USARTEx_DisableSlaveMode()***
- ***HAL_USARTEx_ConfigNSS()***
- ***HAL_USARTEx_EnableFifoMode()***
- ***HAL_USARTEx_DisableFifoMode()***
- ***HAL_USARTEx_SetTxFifoThreshold()***
- ***HAL_USARTEx_SetRxFifoThreshold()***

71.1.4 Detailed description of functions

HAL_USARTEx_RxFifoFullCallback

Function name	**void HAL_USARTEx_RxFifoFullCallback (USART_HandleTypeDef * husart)**
Function description	USART RX Fifo full callback.
Parameters	• **husart:** USART handle.
Return values	• **None:**

HAL_USARTEx_TxFifoEmptyCallback

Function name	**void HAL_USARTEx_TxFifoEmptyCallback (USART_HandleTypeDef * husart)**
Function description	USART TX Fifo empty callback.
Parameters	• **husart:** USART handle.

Return values
- **None:**

HAL_USARTEx_EnableSlaveMode

Function name **HAL_StatusTypeDef HAL_USARTEx_EnableSlaveMode (USART_HandleTypeDef * husart)**

Function description Enable the SPI slave mode.

Parameters
- **husart:** USART handle.

Return values
- **HAL:** status

Notes
- When the USART 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 USART must be enabled before starting the master communications (or between frames while the clock is stable). Otherwise, if the USART 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_USARTEx_DisableSlaveMode

Function name **HAL_StatusTypeDef HAL_USARTEx_DisableSlaveMode (USART_HandleTypeDef * husart)**

Function description Disable the SPI slave mode.

Parameters
- **husart:** USART handle.

Return values
- **HAL:** status

HAL_USARTEx_ConfigNSS

Function name **HAL_StatusTypeDef HAL_USARTEx_ConfigNSS (USART_HandleTypeDef * husart, uint32_t NSSConfig)**

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

Parameters
- **husart:** USART handle.
- **NSSConfig:** NSS configuration. This parameter can be one of the following values:
  - USART_NSS_HARD
  - USART_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_USARTEx_EnableFifoMode

Function name **HAL_StatusTypeDef HAL_USARTEx_EnableFifoMode (USART_HandleTypeDef * husart)**

Function description Enable the FIFO mode.

Parameters
- **husart:** USART handle.

Return values
- **HAL:** status

HAL_USARTEx_DisableFifoMode

Function name **HAL_StatusTypeDef HAL_USARTEx_DisableFifoMode (USART_HandleTypeDef * husart)**

Function description Disable the FIFO mode.

Parameters
- **husart:** USART handle.

Return values
- **HAL:** status

HAL_USARTEx_SetTxFifoThreshold

Function name **HAL_StatusTypeDef HAL_USARTEx_SetTxFifoThreshold (USART_HandleTypeDef * husart, uint32_t Threshold)**

Function description Set the TXFIFO threshold.

Parameters
- **husart:** USART handle.
- **Threshold:** TX FIFO threshold value This parameter can be one of the following values:
  - USART_TXFIFO_THRESHOLD_1_8
  - USART_TXFIFO_THRESHOLD_1_4
  - USART_TXFIFO_THRESHOLD_1_2
  - USART_TXFIFO_THRESHOLD_3_4
  - USART_TXFIFO_THRESHOLD_7_8
  - USART_TXFIFO_THRESHOLD_8_8

Return values
- **HAL:** status

HAL_USARTEx_SetRxFifoThreshold

Function name **HAL_StatusTypeDef HAL_USARTEx_SetRxFifoThreshold (USART_HandleTypeDef * husart, uint32_t Threshold)**

Function description Set the RXFIFO threshold.

Parameters
- **husart:** USART handle.
- **Threshold:** RX FIFO threshold value This parameter can be one of the following values:
  - USART_RXFIFO_THRESHOLD_1_8
  - USART_RXFIFO_THRESHOLD_1_4
  - USART_RXFIFO_THRESHOLD_1_2
  - USART_RXFIFO_THRESHOLD_3_4
  - USART_RXFIFO_THRESHOLD_7_8
  - USART_RXFIFO_THRESHOLD_8_8

Return values
- **HAL:** status

71.2 USARTEx Firmware driver defines

71.2.1 USARTEx

USARTEx FIFO mode

USART_FIFOMODE_DISABLE FIFO mode disable

USART_FIFOMODE_ENABLE FIFO mode enable

USARTEx RXFIFO threshold level

USART_RXFIFO_THRESHOLD_1_8 RXFIFO FIFO reaches 1/8 of its depth

USART_RXFIFO_THRESHOLD_1_4 RXFIFO FIFO reaches 1/4 of its depth

USART_RXFIFO_THRESHOLD_1_2 RXFIFO FIFO reaches 1/2 of its depth

USART_RXFIFO_THRESHOLD_3_4 RXFIFO FIFO reaches 3/4 of its depth

USART_RXFIFO_THRESHOLD_7_8 RXFIFO FIFO reaches 7/8 of its depth

USART_RXFIFO_THRESHOLD_8_8 RXFIFO FIFO becomes full

USARTEx Synchronous Slave mode

USART_SLAVEMODE_DISABLE USART SPI Slave Mode Enable

USART_SLAVEMODE_ENABLE USART SPI Slave Mode Disable

USARTEx Slave Select Management

USART_NSS_HARD SPI slave selection depends on NSS input pin

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

USARTEx TXFIFO threshold level

USART_TXFIFO_THRESHOLD_1_8 TXFIFO reaches 1/8 of its depth

USART_TXFIFO_THRESHOLD_1_4 TXFIFO reaches 1/4 of its depth

USART_TXFIFO_THRESHOLD_1_2 TXFIFO reaches 1/2 of its depth

USART_TXFIFO_THRESHOLD_3_4 TXFIFO reaches 3/4 of its depth

USART_TXFIFO_THRESHOLD_7_8 TXFIFO reaches 7/8 of its depth

USART_TXFIFO_THRESHOLD_8_8 TXFIFO becomes empty

USARTEx Word Length

USART_WORDLENGTH_7B 7-bit long USART frame

USART_WORDLENGTH_8B 8-bit long USART frame

USART_WORDLENGTH_9B 9-bit long USART frame

72 HAL WWDG Generic Driver

72.1 WWDG Firmware driver registers structures

72.1.1 WWDG_InitTypeDef

Data Fields

- ***uint32_t Prescaler***
- ***uint32_t Window***
- ***uint32_t Counter***
- ***uint32_t EWIMode***

Field Documentation

- ***uint32_t WWDG_InitTypeDef::Prescaler***
  Specifies the prescaler value of the WWDG. This parameter can be a value of ***WWDG_Prescaler***
- ***uint32_t WWDG_InitTypeDef::Window***
  Specifies the WWDG window value to be compared to the downcounter. This parameter must be a number Min_Data = 0x40 and Max_Data = 0x7F
- ***uint32_t WWDG_InitTypeDef::Counter***
  Specifies the WWDG free-running downcounter value. This parameter must be a number between Min_Data = 0x40 and Max_Data = 0x7F
- ***uint32_t WWDG_InitTypeDef::EWIMode***
  Specifies if WWDG Early Wakeup Interupt is enable or not. This parameter can be a value of ***WWDG_EWI_Mode***

72.1.2 WWDG_HandleTypeDef

Data Fields

- ***WWDG_TypeDef * Instance***
- ***WWDG_InitTypeDef Init***

Field Documentation

- ***WWDG_TypeDef* WWDG_HandleTypeDef::Instance***
  Register base address
- ***WWDG_InitTypeDef WWDG_HandleTypeDef::Init***
  WWDG required parameters

72.2 WWDG Firmware driver API description

72.2.1 WWDG specific features

Once enabled the WWDG generates a system reset on expiry of a programmed time period, unless the program refreshes the counter (T[6;0] downcounter) before reaching 0x3F value (i.e. a reset is generated when the counter value rolls over from 0x40 to 0x3F).

- An MCU reset is also generated if the counter value is refreshed before the counter has reached the refresh window value. This implies that the counter must be refreshed in a limited window.
- Once enabled the WWDG cannot be disabled except by a system reset.
- WWDGRST flag in RCC_CSR register informs when a WWDG reset has occurred (check available with __HAL_RCC_GET_FLAG(RCC_FLAG_WWDGRST)).