My Project

Generated by Doxygen 1.9.6

Chapter 1

API Reference

1.1 Header files

- EF_Driver_Common.h
- EF_UART.h
- EF_UART_regs.h

1.2 File EF_Driver_Common.h

C header file for common driver definitions and types.

1.3 Structures and Types

Type	Name
typedef uint32_t	∗∗EF_DRIVER_STATUS∗∗ *A type that is used to return the status of the driver functions.*

1.4 Macros

Type	Name
define	∗∗EF_DRIVER_ERROR∗∗ 1 *Unspecified error.*
define	∗∗EF_DRIVER_ERROR_BUSY∗∗ 2 *Driver is busy.*
define	∗∗EF_DRIVER_ERROR_NO_DATA∗∗ 7 *No data available.*
define	∗∗EF_DRIVER_ERROR_PARAMETER∗∗ 5 *Parameter error.*
define	∗∗EF_DRIVER_ERROR_SPECIFIC∗∗ 6 *Start of driver specific errors.*

Type	Name
define	∗∗EF_DRIVER_ERROR_TIMEOUT∗∗ 3 *Timeout occurred.*
define	∗∗EF_DRIVER_ERROR_UNSUPPORTED∗∗ 4 *Operation not supported.*
define	∗∗EF_DRIVER_OK∗∗ 0 *Operation succeeded.*

1.5 Structures and Types Documentation

1.5.1 typedef <tt>EF_DRIVER_STATUS</tt>

A type that is used to return the status of the driver functions.

```
typedef uint32_t EF_DRIVER_STATUS;
```

1.6 Macros Documentation

1.6.1 define <tt>EF_DRIVER_ERROR</tt>

Unspecified error.

```
#define EF_DRIVER_ERROR 1
```

1.6.2 define <tt>EF_DRIVER_ERROR_BUSY</tt>

Driver is busy.

```
#define EF_DRIVER_ERROR_BUSY 2
```

1.6.3 define <tt>EF_DRIVER_ERROR_NO_DATA</tt>

No data available.

```
#define EF_DRIVER_ERROR_NO_DATA 7
```

1.6.4 define <tt>EF_DRIVER_ERROR_PARAMETER</tt>

Parameter error.

```
#define EF_DRIVER_ERROR_PARAMETER 5
```

1.6.5 define <tt>EF_DRIVER_ERROR_SPECIFIC</tt>

Start of driver specific errors.

```
#define EF_DRIVER_ERROR_SPECIFIC 6
```

1.6.6 define <tt>EF_DRIVER_ERROR_TIMEOUT</tt>

Timeout occurred.

```
#define EF_DRIVER_ERROR_TIMEOUT 3
```

1.6.7 define <tt>EF_DRIVER_ERROR_UNSUPPORTED</tt>

Operation not supported.

```
#define EF_DRIVER_ERROR_UNSUPPORTED 4
```

1.6.8 define <tt>EF_DRIVER_OK</tt>

Operation succeeded.

```
#define EF_DRIVER_OK 0
```

1.7 File EF_UART.h

C header file for UART APIs which contains the function prototypes.

1.8 Structures and Types

Type	Name
enum	∗∗parity_type∗∗

1.9 Functions

Type	Name
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_busy∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart, bool ∗flag) *This function checks id the UART is busy.*
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_charsAvailable∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart, bool ∗flag) *This function returns a flag indicating whether or not there is data available in the receive FIFO.*
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_disable∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart) *disables using uart by clearing "en" bit in the control register*
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_disableGlitchFilter∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart) *disables glitch filter (filter out noise or glitches on the received signal) by clearing "gfen" bit in the control register*
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_disableLoopBack∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart) *disables loopback (connecting TX to RX signal) by clearing "lpen" bit in the control register*
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_disableRx∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart) *disables using uart RX by clearing uart "rxen" bit in the control register*
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_disableTx∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart) *disables using uart TX by clearing uart "txen" bit in the control register*

Type	Name
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_enable∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart) *enables using uart by setting "en" bit in the control register to 1*
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_enableGlitchFilter∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart) *enables glitch filter (filter out noise or glitches on the received signal) by setting "gfen" bit in the control register to 1*
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_enableLoopBack∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart) *enables loopback (connecting TX to RX signal) by setting "lpen" bit in the control register to 1*
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_enableRx∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart) *enables using uart RX by setting uart "rxen" bit in the control register to 1*
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_enableTx∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart) *enables using uart TX by setting uart "txen" bit in the control register to 1*
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_getCTRL∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart, uint32_t ∗CTRL_↩ value) *returns the value of the control register*
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_getConfig∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart, uint32_t ∗CFG_↩ value) *returns the value of the configuration register*
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_getIM∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart, uint32_t ∗IM_value)
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_getMIS∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart, uint32_t ∗MIS_value)
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_getMatchData∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart, uint32_t ∗↩ MATCH_value) *returns the value of the match data register*
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_getParityMode∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart, uint32_↩ t ∗parity_mode) *This function return the parity mode of the UART.*
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_getPrescaler∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart, uint32_t ∗↩ Prescaler_value) *returns the value of the prescaler*
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_getRIS∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart, uint32_t ∗RIS_value)
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_getRxCount∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart, uint32_t ∗RX_↩ FIFO_LEVEL_value) *returns the current level of the RX FIFO (the number of bytes in the FIFO)*
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_getRxFIFOThreshold∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart, uint32↩ _t ∗RX_FIFO_THRESHOLD_value) *returns the current value of the RX FIFO threshold*
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_getTxCount∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart, uint32_t ∗TX_↩ FIFO_LEVEL_value) *returns the current level of the TX FIFO (the number of bytes in the FIFO)*
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_getTxFIFOThreshold∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart, uint32↩ _t ∗TX_FIFO_THRESHOLD_value) *returns the current value of the TX FIFO threshold*
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_readChar∗∗ (∗∗EF_UART_TYPE∗∗ ∗uar, char ∗RXDATA_↩ value) *recieve a single character through uart*
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_readCharNonBlocking∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart, char ∗RXDATA_value, bool ∗data_available) *This is a non-blocking function that reads a character from the UART receive FIFO if data is available and returns a status code.*
∗∗EF_DRIVER_STATUS∗∗	∗∗EF_UART_setCTRL∗∗ (∗∗EF_UART_TYPE∗∗ ∗uart, uint32_t value)

Type	Name
EF_DRIVER_STATUS	**EF_UART_setConfig** (**EF_UART_TYPE** *uart, uint32_t config)
EF_DRIVER_STATUS	**EF_UART_setDataSize** (**EF_UART_TYPE** *uart, uint32_t value) *sets the Data Size (Data word length: 5-9 bits) by setting the "wlen" field in configuration register*
EF_DRIVER_STATUS	**EF_UART_setGclkEnable** (**EF_UART_TYPE** *uart, uint32_t value) *sets the GCLK enable bit in the UART register to a certain value*
EF_DRIVER_STATUS	**EF_UART_setICR** (**EF_UART_TYPE** *uart, uint32_t mask)
EF_DRIVER_STATUS	**EF_UART_setIM** (**EF_UART_TYPE** *uart, uint32_t mask)
EF_DRIVER_STATUS	**EF_UART_setMatchData** (**EF_UART_TYPE** *uart, uint32_↩ t matchData) *sets the matchData to a certain value at which "MATCH" interrupt will be raised*
EF_DRIVER_STATUS	**EF_UART_setParityType** (**EF_UART_TYPE** *uart, enum **parity_type** parity) *sets the "parity" field in configuration register (could be none, odd, even, sticky 0 or sticky 1)*
EF_DRIVER_STATUS	**EF_UART_setPrescaler** (**EF_UART_TYPE** *uart, uint32_↩ t prescaler) *sets the prescaler to a certain value where Baud_rate = Bus_Clock_↩ Freq/((Prescaler+1)*16)*
EF_DRIVER_STATUS	**EF_UART_setRxFIFOThreshold** (**EF_UART_TYPE** *uart, uint32↩ _t threshold) *sets the RX FIFO threshold to a certain value at which "RXA" interrupt will be raised*
EF_DRIVER_STATUS	**EF_UART_setTimeoutBits** (**EF_UART_TYPE** *uart, uint32_t value) *sets the "timeout" field in configuration register which is receiver timeout measured in number of bits at which the timeout flag will be raised*
EF_DRIVER_STATUS	**EF_UART_setTwoStopBitsSelect** (**EF_UART_TYPE** *uart, bool is_two_bits) *sets the "stp2" bit in configuration register (whether the stop boits are two or one)*
EF_DRIVER_STATUS	**EF_UART_setTxFIFOThreshold** (**EF_UART_TYPE** *uart, uint32↩ _t threshold) *sets the TX FIFO threshold to a certain value at which "TXB" interrupt will be raised*
EF_DRIVER_STATUS	**EF_UART_spaceAvailable** (**EF_UART_TYPE** *uart, bool *flag) *This function returns a flag indicating whether or not the transmit is available, i.e. the transmit FIFO is not full.*
EF_DRIVER_STATUS	**EF_UART_writeChar** (**EF_UART_TYPE** *uart, char data) *transmit a single character through uart*
EF_DRIVER_STATUS	**EF_UART_writeCharArr** (**EF_UART_TYPE** *uart, const char *char_arr) *transmit an array of characters through uart*
EF_DRIVER_STATUS	**EF_UART_writeCharNonBlocking** (**EF_UART_TYPE** *uart, char data, bool *data_sent) *This is a non-blocking function that writes a character to the UART transmit FIFO if the FIFO is not full and returns a status code.*

1.10 Macros

Type	Name
define	**EF_UART_CFG_REG_MAX_VALUE** ((uint32_t)0x00001FFF)
define	**EF_UART_CFG_REG_TIMEOUT_MAX_VALUE** ((uint32_t)0x0000003F)
define	**EF_UART_CTRL_REG_MAX_VALUE** ((uint32_t)0x0000001F)
define	**EF_UART_DataLength_MAX_VALUE** ((uint32_t)0x00000009)
define	**EF_UART_DataLength_MIN_VALUE** ((uint32_t)0x00000005)
define	**EF_UART_ERROR_RX_UNAVAILABLE** -1
define	**EF_UART_ERROR_TX_UNAVAILABLE** 1
define	**EF_UART_IC_REG_MAX_VALUE** ((uint32_t)0x000003FF)
define	**EF_UART_IM_REG_MAX_VALUE** ((uint32_t)0x000003FF)
define	**EF_UART_MATCH_REG_MAX_VALUE** ((uint32_t)0x00001FFF)
define	**EF_UART_PR_REG_MAX_VALUE** ((uint32_t)0x0000FFFF)
define	**EF_UART_RX_FIFO_THRESHOLD_REG_MAX_VALUE** ((uint32_t)0x0000000F)
define	**EF_UART_SUCCESS** 0
define	**EF_UART_TX_FIFO_THRESHOLD_REG_MAX_VALUE** ((uint32_t)0x0000000F)

1.11 Structures and Types Documentation

1.11.1 enum <tt>parity_type</tt>

```
enum parity_type {
    NONE = 0,
    ODD = 1,
    EVEN = 2,
    STICKY_0 = 4,
    STICKY_1 = 5
};
```

1.12 Functions Documentation

1.12.1 function <tt>EF_UART_busy</tt>

This function checks id the UART is busy.

```
EF_DRIVER_STATUS EF_UART_busy (
    EF_UART_TYPE *uart,
    bool *flag
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩
EF_UART_TYPE** is a structure that contains the UART registers.

- `flag` a flag indicating if the UART is busy

Returns:

status A value of type ∗∗EF_DRIVER_STATUS∗∗ : returns a success or error code

1.12.2 function <tt>EF_UART_charsAvailable</tt>

This function returns a flag indicating whether or not there is data available in the receive FIFO.

```
EF_DRIVER_STATUS EF_UART_charsAvailable (
    EF_UART_TYPE *uart,
    bool *flag
)
```

Parameters:

- `uart` An ∗∗EF_UART_TYPE∗∗ pointer, which points to the base memory address of UART registers.∗∗↩ EF_UART_TYPE∗∗ is a structure that contains the UART registers.
- `flag` a flag indicating if there is data available in the receive FIFO

Returns:

status A value of type ∗∗EF_DRIVER_STATUS∗∗ : returns a success or error code

1.12.3 function <tt>EF_UART_disable</tt>

disables using uart by clearing "en" bit in the control register

```
EF_DRIVER_STATUS EF_UART_disable (
    EF_UART_TYPE *uart
)
```

Parameters:

- `uart` An ∗∗EF_UART_TYPE∗∗ pointer, which points to the base memory address of UART registers.∗∗↩ EF_UART_TYPE∗∗ is a structure that contains the UART registers.

Returns:

status A value of type ∗∗EF_DRIVER_STATUS∗∗ : returns a success or error code

1.12.4 function <tt>EF_UART_disableGlitchFilter</tt>

disables glitch filter (filter out noise or glitches on the received signal) by clearing "gfen" bit in the control register

```
EF_DRIVER_STATUS EF_UART_disableGlitchFilter (
    EF_UART_TYPE *uart
)
```

Parameters:

- `uart` An ∗∗EF_UART_TYPE∗∗ pointer, which points to the base memory address of UART registers.∗∗↩ EF_UART_TYPE∗∗ is a structure that contains the UART registers.

Returns:

status A value of type ∗∗EF_DRIVER_STATUS∗∗ : returns a success or error code

1.12.5 function <tt>EF_UART_disableLoopBack</tt>

disables loopback (connecting TX to RX signal) by clearing "lpen" bit in the control register

```
EF_DRIVER_STATUS EF_UART_disableLoopBack (
    EF_UART_TYPE *uart
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩
  EF_UART_TYPE** is a structure that contains the UART registers.

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.6 function <tt>EF_UART_disableRx</tt>

disables using uart RX by clearing uart "rxen" bit in the control register

```
EF_DRIVER_STATUS EF_UART_disableRx (
    EF_UART_TYPE *uart
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩
  EF_UART_TYPE** is a structure that contains the UART registers.

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.7 function <tt>EF_UART_disableTx</tt>

disables using uart TX by clearing uart "txen" bit in the control register

```
EF_DRIVER_STATUS EF_UART_disableTx (
    EF_UART_TYPE *uart
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩
  EF_UART_TYPE** is a structure that contains the UART registers.

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.8 function <tt>EF_UART_enable</tt>

enables using uart by setting "en" bit in the control register to 1

```
EF_DRIVER_STATUS EF_UART_enable (
    EF_UART_TYPE *uart
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩ EF_UART_TYPE** is a structure that contains the UART registers.

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.9 function <tt>EF_UART_enableGlitchFilter</tt>

enables glitch filter (filter out noise or glitches on the received signal) by setting "gfen" bit in the control register to 1

```
EF_DRIVER_STATUS EF_UART_enableGlitchFilter (
    EF_UART_TYPE *uart
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩ EF_UART_TYPE** is a structure that contains the UART registers.

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.10 function <tt>EF_UART_enableLoopBack</tt>

enables loopback (connecting TX to RX signal) by setting "lpen" bit in the control register to 1

```
EF_DRIVER_STATUS EF_UART_enableLoopBack (
    EF_UART_TYPE *uart
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩ EF_UART_TYPE** is a structure that contains the UART registers.

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.11 function <tt>EF_UART_enableRx</tt>

enables using uart RX by setting uart "rxen" bit in the control register to 1

```
EF_DRIVER_STATUS EF_UART_enableRx (
    EF_UART_TYPE *uart
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩
EF_UART_TYPE** is a structure that contains the UART registers.

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.12 function <tt>EF_UART_enableTx</tt>

enables using uart TX by setting uart "txen" bit in the control register to 1

```
EF_DRIVER_STATUS EF_UART_enableTx (
    EF_UART_TYPE *uart
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩
EF_UART_TYPE** is a structure that contains the UART registers.

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.13 function <tt>EF_UART_getCTRL</tt>

returns the value of the control register

```
EF_DRIVER_STATUS EF_UART_getCTRL (
    EF_UART_TYPE *uart,
    uint32_t *CTRL_value
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩
EF_UART_TYPE** is a structure that contains the UART registers.
- `CTRL_value` The value of the control register

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.14 function <tt>EF_UART_getConfig</tt>

returns the value of the configuration register

```
EF_DRIVER_STATUS EF_UART_getConfig (
    EF_UART_TYPE *uart,
    uint32_t *CFG_value
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩ EF_UART_TYPE** is a structure that contains the UART registers.
- `CFG_value` The value of the configuration register

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.15 function <tt>EF_UART_getIM</tt>

```
EF_DRIVER_STATUS EF_UART_getIM (
    EF_UART_TYPE *uart,
    uint32_t *IM_value
)
```

returns the value of the Interrupts Masking Register; which enable and disables interrupts

- bit 0 TXE : Transmit FIFO is Empty.
- bit 1 RXF : Receive FIFO is Full.
- bit 2 TXB : Transmit FIFO level is Below Threshold.
- bit 3 RXA : Receive FIFO level is Above Threshold.
- bit 4 BRK : Line Break; 13 consecutive 0's have been detected on the line.
- bit 5 MATCH : the receive data matches the MATCH register.
- bit 6 FE : Framing Error, the receiver does not see a "stop" bit at the expected "stop" bit time.
- bit 7 PRE : Parity Error; the receiver calculated parity does not match the received one.
- bit 8 OR : Overrun; data has been received but the RX FIFO is full.
- bit 9 RTO : Receiver Timeout; no data has been received for the time of a specified number of bits.

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩ EF_UART_TYPE** is a structure that contains the UART registers.
- `IM_value` The value of the Interrupts Masking Register

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.16 function <tt>EF_UART_getMIS</tt>

```
EF_DRIVER_STATUS EF_UART_getMIS (
    EF_UART_TYPE *uart,
    uint32_t *MIS_value
)
```

returns the value of the Masked Interrupt Status Register

- bit 0 TXE : Transmit FIFO is Empty.
- bit 1 RXF : Receive FIFO is Full.
- bit 2 TXB : Transmit FIFO level is Below Threshold.
- bit 3 RXA : Receive FIFO level is Above Threshold.
- bit 4 BRK : Line Break; 13 consecutive 0's have been detected on the line.
- bit 5 MATCH : the receive data matches the MATCH register.
- bit 6 FE : Framing Error, the receiver does not see a "stop" bit at the expected "stop" bit time.
- bit 7 PRE : Parity Error; the receiver calculated parity does not match the received one.
- bit 8 OR : Overrun; data has been received but the RX FIFO is full.
- bit 9 RTO : Receiver Timeout; no data has been received for the time of a specified number of bits.

Parameters:

- `uart` An ∗∗EF_UART_TYPE∗∗ pointer, which points to the base memory address of UART registers.∗∗↩ EF_UART_TYPE∗∗ is a structure that contains the UART registers.
- `MIS_value` The value of the Masked Interrupt Status Register

Returns:

status A value of type ∗∗EF_DRIVER_STATUS∗∗ : returns a success or error code

1.12.17 function <tt>EF_UART_getMatchData</tt>

returns the value of the match data register

```
EF_DRIVER_STATUS EF_UART_getMatchData (
    EF_UART_TYPE *uart,
    uint32_t *MATCH_value
)
```

Parameters:

- `uart` An ∗∗EF_UART_TYPE∗∗ pointer, which points to the base memory address of UART registers.∗∗↩ EF_UART_TYPE∗∗ is a structure that contains the UART registers.
- `MATCH_value` The value of the match data register

Returns:

status A value of type ∗∗EF_DRIVER_STATUS∗∗ : returns a success or error code

1.12.18 function <tt>EF_UART_getParityMode</tt>

This function return the parity mode of the UART.

```
EF_DRIVER_STATUS EF_UART_getParityMode (
    EF_UART_TYPE *uart,
    uint32_t *parity_mode
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩
EF_UART_TYPE** is a structure that contains the UART registers.
- `parity` The parity mode of the UART

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.19 function <tt>EF_UART_getPrescaler</tt>

returns the value of the prescaler

```
EF_DRIVER_STATUS EF_UART_getPrescaler (
    EF_UART_TYPE *uart,
    uint32_t *Prescaler_value
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩
EF_UART_TYPE** is a structure that contains the UART registers.
- `Prescaler_value` The value of the prescaler register

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.20 function <tt>EF_UART_getRIS</tt>

```
EF_DRIVER_STATUS EF_UART_getRIS (
    EF_UART_TYPE *uart,
    uint32_t *RIS_value
)
```

returns the value of the Raw Interrupt Status Register

- bit 0 TXE : Transmit FIFO is Empty.
- bit 1 RXF : Receive FIFO is Full.
- bit 2 TXB : Transmit FIFO level is Below Threshold.
- bit 3 RXA : Receive FIFO level is Above Threshold.
- bit 4 BRK : Line Break; 13 consecutive 0's have been detected on the line.
- bit 5 MATCH : the receive data matches the MATCH register.

- bit 6 FE : Framing Error, the receiver does not see a "stop" bit at the expected "stop" bit time.
- bit 7 PRE : Parity Error; the receiver calculated parity does not match the received one.
- bit 8 OR : Overrun; data has been received but the RX FIFO is full.
- bit 9 RTO : Receiver Timeout; no data has been received for the time of a specified number of bits.

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩ EF_UART_TYPE** is a structure that contains the UART registers.
- `RIS_value` The value of the Raw Interrupt Status Register

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.21 function <tt>EF_UART_getRxCount</tt>

returns the current level of the RX FIFO (the number of bytes in the FIFO)

```
EF_DRIVER_STATUS EF_UART_getRxCount (
    EF_UART_TYPE *uart,
    uint32_t *RX_FIFO_LEVEL_value
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩ EF_UART_TYPE** is a structure that contains the UART registers.
- `RX_FIFO_LEVEL_value` The value of the RX FIFO level register

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.22 function <tt>EF_UART_getRxFIFOThreshold</tt>

returns the current value of the RX FIFO threshold

```
EF_DRIVER_STATUS EF_UART_getRxFIFOThreshold (
    EF_UART_TYPE *uart,
    uint32_t *RX_FIFO_THRESHOLD_value
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩ EF_UART_TYPE** is a structure that contains the UART registers.
- `RX_FIFO_THRESHOLD_value` The value of the RX FIFO threshold register

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.23 function <tt>EF_UART_getTxCount</tt>

returns the current level of the TX FIFO (the number of bytes in the FIFO)

```
EF_DRIVER_STATUS EF_UART_getTxCount (
    EF_UART_TYPE *uart,
    uint32_t *TX_FIFO_LEVEL_value
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩
EF_UART_TYPE** is a structure that contains the UART registers.
- `TX_FIFO_LEVEL_value` The value of the TX FIFO level register

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.24 function <tt>EF_UART_getTxFIFOThreshold</tt>

returns the current value of the TX FIFO threshold

```
EF_DRIVER_STATUS EF_UART_getTxFIFOThreshold (
    EF_UART_TYPE *uart,
    uint32_t *TX_FIFO_THRESHOLD_value
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩
EF_UART_TYPE** is a structure that contains the UART registers.
- `TX_FIFO_THRESHOLD_value` The value of the TX FIFO threshold register

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.25 function <tt>EF_UART_readChar</tt>

recieve a single character through uart

```
EF_DRIVER_STATUS EF_UART_readChar (
    EF_UART_TYPE *uar,
    char *RXDATA_value
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩
EF_UART_TYPE** is a structure that contains the UART registers.
- `RXDATA_value` The value of the received character

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.26 function <tt>EF_UART_readCharNonBlocking</tt>

This is a non-blocking function that reads a character from the UART receive FIFO if data is available and returns a status code.

```
EF_DRIVER_STATUS EF_UART_readCharNonBlocking (
    EF_UART_TYPE *uart,
    char *RXDATA_value,
    bool *data_available
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩
EF_UART_TYPE** is a structure that contains the UART registers.
- `RXDATA_value` The value of the received character
- `data_available` A flag indicating if data is available in the receive FIFO

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.27 function <tt>EF_UART_setCTRL</tt>

```
EF_DRIVER_STATUS EF_UART_setCTRL (
    EF_UART_TYPE *uart,
    uint32_t value
)
```

sets the control register to a certain value where

- bit 0: UART enable
- bit 1: UART Transmitter enable
- bit 2: UART Receiver enable
- bit 3: Loopback (connect RX and TX pins together) enable
- bit 4: UART Glitch Filer on RX enable

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩
EF_UART_TYPE** is a structure that contains the UART registers.
- `value` The value of the control register

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.28 function <tt>EF_UART_setConfig</tt>

```
EF_DRIVER_STATUS EF_UART_setConfig (
    EF_UART_TYPE *uart,
    uint32_t config
)
```

sets the configuration register to a certain value where

- bit 0-3: Data word length: 5-9 bits
- bit 4: Two Stop Bits Select
- bit 5-7: Parity Type: 000: None, 001: odd, 010: even, 100: Sticky 0, 101: Sticky 1
- bit 8-13: Receiver Timeout measured in number of bits

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩ EF_UART_TYPE** is a structure that contains the UART registers.
- `config` The value of the configuration register

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.29 function <tt>EF_UART_setDataSize</tt>

sets the Data Size (Data word length: 5-9 bits) by setting the "wlen" field in configuration register

```
EF_DRIVER_STATUS EF_UART_setDataSize (
    EF_UART_TYPE *uart,
    uint32_t value
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩ EF_UART_TYPE** is a structure that contains the UART registers.
- `value` The value of the required data word length

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.30 function <tt>EF_UART_setGclkEnable</tt>

sets the GCLK enable bit in the UART register to a certain value

```
EF_DRIVER_STATUS EF_UART_setGclkEnable (
    EF_UART_TYPE *uart,
    uint32_t value
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩ EF_UART_TYPE** is a structure that contains the UART registers.
- `value` The value of the GCLK enable bit

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.31 function <tt>EF_UART_setICR</tt>

```
EF_DRIVER_STATUS EF_UART_setICR (
    EF_UART_TYPE *uart,
    uint32_t mask
)
```

sets the value of the Interrupts Clear Register; write 1 to clear the flag

- bit 0 TXE : Transmit FIFO is Empty.
- bit 1 RXF : Receive FIFO is Full.
- bit 2 TXB : Transmit FIFO level is Below Threshold.
- bit 3 RXA : Receive FIFO level is Above Threshold.
- bit 4 BRK : Line Break; 13 consecutive 0's have been detected on the line.
- bit 5 MATCH : the receive data matches the MATCH register.
- bit 6 FE : Framing Error, the receiver does not see a "stop" bit at the expected "stop" bit time.
- bit 7 PRE : Parity Error; the receiver calculated parity does not match the received one.
- bit 8 OR : Overrun; data has been received but the RX FIFO is full.
- bit 9 RTO : Receiver Timeout; no data has been received for the time of a specified number of bits.

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩ EF_UART_TYPE** is a structure that contains the UART registers.
- `mask` The required mask value

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.32 function <tt>EF_UART_setIM</tt>

```
EF_DRIVER_STATUS EF_UART_setIM (
    EF_UART_TYPE *uart,
    uint32_t mask
)
```

sets the value of the Interrupts Masking Register; which enable and disables interrupts

- bit 0 TXE : Transmit FIFO is Empty.
- bit 1 RXF : Receive FIFO is Full.
- bit 2 TXB : Transmit FIFO level is Below Threshold.
- bit 3 RXA : Receive FIFO level is Above Threshold.
- bit 4 BRK : Line Break; 13 consecutive 0's have been detected on the line.
- bit 5 MATCH : the receive data matches the MATCH register.
- bit 6 FE : Framing Error, the receiver does not see a "stop" bit at the expected "stop" bit time.

- bit 7 PRE : Parity Error; the receiver calculated parity does not match the received one.
- bit 8 OR : Overrun; data has been received but the RX FIFO is full.
- bit 9 RTO : Receiver Timeout; no data has been received for the time of a specified number of bits.

Parameters:

- `uart` An ∗∗EF_UART_TYPE∗∗ pointer, which points to the base memory address of UART registers.∗∗↩ EF_UART_TYPE∗∗ is a structure that contains the UART registers.
- `mask` The required mask value

Returns:

status A value of type ∗∗EF_DRIVER_STATUS∗∗ : returns a success or error code

1.12.33 function <tt>EF_UART_setMatchData</tt>

sets the matchData to a certain value at which "MATCH" interrupt will be raised

```
EF_DRIVER_STATUS EF_UART_setMatchData (
    EF_UART_TYPE *uart,
    uint32_t matchData
)
```

Parameters:

- `uart` An ∗∗EF_UART_TYPE∗∗ pointer, which points to the base memory address of UART registers.∗∗↩ EF_UART_TYPE∗∗ is a structure that contains the UART registers.
- `matchData` The value of the required match data

Returns:

status A value of type ∗∗EF_DRIVER_STATUS∗∗ : returns a success or error code

1.12.34 function <tt>EF_UART_setParityType</tt>

sets the "parity" field in configuration register (could be none, odd, even, sticky 0 or sticky 1)

```
EF_DRIVER_STATUS EF_UART_setParityType (
    EF_UART_TYPE *uart,
    enum parity_type parity
)
```

Parameters:

- `uart` An ∗∗EF_UART_TYPE∗∗ pointer, which points to the base memory address of UART registers.∗∗↩ EF_UART_TYPE∗∗ is a structure that contains the UART registers.
- `parity` enum parity_type could be "NONE" , "ODD" , "EVEN" , "STICKY_0" , or "STICKY_1"

Returns:

status A value of type ∗∗EF_DRIVER_STATUS∗∗ : returns a success or error code

1.12.35 function <tt>EF_UART_setPrescaler</tt>

sets the prescaler to a certain value where Baud_rate = Bus_Clock_Freq/((Prescaler+1)∗16)

```
EF_DRIVER_STATUS EF_UART_setPrescaler (
    EF_UART_TYPE *uart,
    uint32_t prescaler
)
```

Parameters:

- `uart` An ∗∗EF_UART_TYPE∗∗ pointer, which points to the base memory address of UART registers.∗∗↩
EF_UART_TYPE∗∗ is a structure that contains the UART registers.
- `prescaler` The value of the required prescaler

Returns:

status A value of type ∗∗EF_DRIVER_STATUS∗∗ : returns a success or error code

1.12.36 function <tt>EF_UART_setRxFIFOThreshold</tt>

sets the RX FIFO threshold to a certain value at which "RXA" interrupt will be raised

```
EF_DRIVER_STATUS EF_UART_setRxFIFOThreshold (
    EF_UART_TYPE *uart,
    uint32_t threshold
)
```

Parameters:

- `uart` An ∗∗EF_UART_TYPE∗∗ pointer, which points to the base memory address of UART registers.∗∗↩
EF_UART_TYPE∗∗ is a structure that contains the UART registers.
- `threshold` The value of the required threshold

Returns:

status A value of type ∗∗EF_DRIVER_STATUS∗∗ : returns a success or error code

1.12.37 function <tt>EF_UART_setTimeoutBits</tt>

sets the "timeout" field in configuration register which is receiver timeout measured in number of bits at which the timeout flag will be raised

```
EF_DRIVER_STATUS EF_UART_setTimeoutBits (
    EF_UART_TYPE *uart,
    uint32_t value
)
```

Parameters:

- `uart` An ∗∗EF_UART_TYPE∗∗ pointer, which points to the base memory address of UART registers.∗∗↩
EF_UART_TYPE∗∗ is a structure that contains the UART registers.
- `value` timeout bits value

Returns:

status A value of type ∗∗EF_DRIVER_STATUS∗∗ : returns a success or error code

1.12.38 function <tt>EF_UART_setTwoStopBitsSelect</tt>

sets the "stp2" bit in configuration register (whether the stop boits are two or one)

```
EF_DRIVER_STATUS EF_UART_setTwoStopBitsSelect (
    EF_UART_TYPE *uart,
    bool is_two_bits
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩ EF_UART_TYPE** is a structure that contains the UART registers.
- `is_two_bits` bool value, if "true", the stop bits are two and if "false", the stop bit is one

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.39 function <tt>EF_UART_setTxFIFOThreshold</tt>

sets the TX FIFO threshold to a certain value at which "TXB" interrupt will be raised

```
EF_DRIVER_STATUS EF_UART_setTxFIFOThreshold (
    EF_UART_TYPE *uart,
    uint32_t threshold
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩ EF_UART_TYPE** is a structure that contains the UART registers.
- `threshold` The value of the required threshold

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.40 function <tt>EF_UART_spaceAvailable</tt>

This function returns a flag indicating whether or not the transmit is available, i.e. the transmit FIFO is not full.

```
EF_DRIVER_STATUS EF_UART_spaceAvailable (
    EF_UART_TYPE *uart,
    bool *flag
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩ EF_UART_TYPE** is a structure that contains the UART registers.
- `flag` a flag indicating if the transmit FIFO is not full

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.41 function <tt>EF_UART_writeChar</tt>

transmit a single character through uart

```
EF_DRIVER_STATUS EF_UART_writeChar (
    EF_UART_TYPE *uart,
    char data
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩ EF_UART_TYPE** is a structure that contains the UART registers.
- `data` The character or byte required to send

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.42 function <tt>EF_UART_writeCharArr</tt>

transmit an array of characters through uart

```
EF_DRIVER_STATUS EF_UART_writeCharArr (
    EF_UART_TYPE *uart,
    const char *char_arr
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩ EF_UART_TYPE** is a structure that contains the UART registers.
- `char_arr` An array of characters to send

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.12.43 function <tt>EF_UART_writeCharNonBlocking</tt>

This is a non-blocking function that writes a character to the UART transmit FIFO if the FIFO is not full and returns a status code.

```
EF_DRIVER_STATUS EF_UART_writeCharNonBlocking (
    EF_UART_TYPE *uart,
    char data,
    bool *data_sent
)
```

Parameters:

- `uart` An **EF_UART_TYPE** pointer, which points to the base memory address of UART registers.**↩ EF_UART_TYPE** is a structure that contains the UART registers.
- `data` The character or byte required to send
- `data_sent` A flag indicating if the data was sent successfully

Returns:

status A value of type **EF_DRIVER_STATUS** : returns a success or error code

1.13 Macros Documentation

1.13.1 define <tt>EF_UART_CFG_REG_MAX_VALUE</tt>

```
#define EF_UART_CFG_REG_MAX_VALUE ((uint32_t)0x00001FFF)
```

1.13.2 define <tt>EF_UART_CFG_REG_TIMEOUT_MAX_VALUE</tt>

```
#define EF_UART_CFG_REG_TIMEOUT_MAX_VALUE ((uint32_t)0x0000003F)
```

1.13.3 define <tt>EF_UART_CTRL_REG_MAX_VALUE</tt>

```
#define EF_UART_CTRL_REG_MAX_VALUE ((uint32_t)0x0000001F)
```

1.13.4 define <tt>EF_UART_DataLength_MAX_VALUE</tt>

```
#define EF_UART_DataLength_MAX_VALUE ((uint32_t)0x00000009)
```

1.13.5 define <tt>EF_UART_DataLength_MIN_VALUE</tt>

```
#define EF_UART_DataLength_MIN_VALUE ((uint32_t)0x00000005)
```

1.13.6 define <tt>EF_UART_ERROR_RX_UNAVAILABLE</tt>

```
#define EF_UART_ERROR_RX_UNAVAILABLE -1
```

1.13.7 define <tt>EF_UART_ERROR_TX_UNAVAILABLE</tt>

```
#define EF_UART_ERROR_TX_UNAVAILABLE 1
```

1.13.8 define <tt>EF_UART_IC_REG_MAX_VALUE</tt>

```
#define EF_UART_IC_REG_MAX_VALUE ((uint32_t)0x000003FF)
```

1.13.9 define <tt>EF_UART_IM_REG_MAX_VALUE</tt>

```
#define EF_UART_IM_REG_MAX_VALUE ((uint32_t)0x000003FF)
```

1.13.10 define <tt>EF_UART_MATCH_REG_MAX_VALUE</tt>

```
#define EF_UART_MATCH_REG_MAX_VALUE ((uint32_t)0x00001FFF)
```

1.13.11 define <tt>EF_UART_PR_REG_MAX_VALUE</tt>

```
#define EF_UART_PR_REG_MAX_VALUE ((uint32_t)0x0000FFFF)
```

1.13.12 define <tt>EF_UART_RX_FIFO_THRESHOLD_REG_MAX_VALUE</tt>

```
#define EF_UART_RX_FIFO_THRESHOLD_REG_MAX_VALUE ((uint32_t)0x0000000F)
```

1.13.13 define <tt>EF_UART_SUCCESS</tt>

```
#define EF_UART_SUCCESS 0
```

1.13.14 define <tt>EF_UART_TX_FIFO_THRESHOLD_REG_MAX_VALUE</tt>

```
#define EF_UART_TX_FIFO_THRESHOLD_REG_MAX_VALUE ((uint32_t)0x0000000F)
```

1.14 File EF_UART_regs.h

1.15 Structures and Types

Type	Name
typedef struct ∗∗_EF_UART_TYPE_∗∗	∗∗EF_UART_TYPE∗∗
struct	∗∗_EF_UART_TYPE_∗∗

1.16 Macros

Type	Name
define	∗∗EF_UART_BRK_FLAG∗∗ 0x10
define	∗∗EF_UART_CFG_REG_PARITY_BIT∗∗ 5
define	∗∗EF_UART_CFG_REG_PARITY_MASK∗∗ 0xe0
define	∗∗EF_UART_CFG_REG_STP2_BIT∗∗ 4
define	∗∗EF_UART_CFG_REG_STP2_MASK∗∗ 0x10
define	∗∗EF_UART_CFG_REG_TIMEOUT_BIT∗∗ 8
define	∗∗EF_UART_CFG_REG_TIMEOUT_MASK∗∗ 0x3f00
define	∗∗EF_UART_CFG_REG_WLEN_BIT∗∗ 0
define	∗∗EF_UART_CFG_REG_WLEN_MASK∗∗ 0xf
define	∗∗EF_UART_CTRL_REG_EN_BIT∗∗ 0
define	∗∗EF_UART_CTRL_REG_EN_MASK∗∗ 0x1
define	∗∗EF_UART_CTRL_REG_GFEN_BIT∗∗ 4
define	∗∗EF_UART_CTRL_REG_GFEN_MASK∗∗ 0x10
define	∗∗EF_UART_CTRL_REG_LPEN_BIT∗∗ 3
define	∗∗EF_UART_CTRL_REG_LPEN_MASK∗∗ 0x8

Type	Name
define	∗∗EF_UART_CTRL_REG_RXEN_BIT∗∗ 2
define	∗∗EF_UART_CTRL_REG_RXEN_MASK∗∗ 0x4
define	∗∗EF_UART_CTRL_REG_TXEN_BIT∗∗ 1
define	∗∗EF_UART_CTRL_REG_TXEN_MASK∗∗ 0x2
define	∗∗EF_UART_FE_FLAG∗∗ 0x40
define	∗∗EF_UART_MATCH_FLAG∗∗ 0x20
define	∗∗EF_UART_OR_FLAG∗∗ 0x100
define	∗∗EF_UART_PRE_FLAG∗∗ 0x80
define	∗∗EF_UART_RTO_FLAG∗∗ 0x200
define	∗∗EF_UART_RXA_FLAG∗∗ 0x8
define	∗∗EF_UART_RXF_FLAG∗∗ 0x2
define	∗∗EF_UART_RX_FIFO_FLUSH_REG_FLUSH_BIT∗∗ 0
define	∗∗EF_UART_RX_FIFO_FLUSH_REG_FLUSH_MASK∗∗ 0x1
define	∗∗EF_UART_RX_FIFO_LEVEL_REG_LEVEL_BIT∗∗ 0
define	∗∗EF_UART_RX_FIFO_LEVEL_REG_LEVEL_MASK∗∗ 0xf
define	∗∗EF_UART_RX_FIFO_THRESHOLD_REG_THRESHOLD_BIT∗∗ 0
define	∗∗EF_UART_RX_FIFO_THRESHOLD_REG_THRESHOLD_MASK∗∗ 0xf
define	∗∗EF_UART_TXB_FLAG∗∗ 0x4
define	∗∗EF_UART_TXE_FLAG∗∗ 0x1
define	∗∗EF_UART_TX_FIFO_FLUSH_REG_FLUSH_BIT∗∗ 0
define	∗∗EF_UART_TX_FIFO_FLUSH_REG_FLUSH_MASK∗∗ 0x1
define	∗∗EF_UART_TX_FIFO_LEVEL_REG_LEVEL_BIT∗∗ 0
define	∗∗EF_UART_TX_FIFO_LEVEL_REG_LEVEL_MASK∗∗ 0xf
define	∗∗EF_UART_TX_FIFO_THRESHOLD_REG_THRESHOLD_BIT∗∗ 0
define	∗∗EF_UART_TX_FIFO_THRESHOLD_REG_THRESHOLD_MASK∗∗ 0xf
define	∗∗IO_TYPES∗∗
define	∗∗__R∗∗ volatile const unsigned int
define	∗∗__RW∗∗ volatile unsigned int

Type	Name
define	**__W** volatile unsigned int

1.17 Structures and Types Documentation

1.17.1 typedef <tt>EF_UART_TYPE</tt>

```
typedef struct _EF_UART_TYPE_ EF_UART_TYPE;
```

1.17.2 struct <tt>_EF_UART_TYPE_</tt>

Variables:

- **__W** CFG
- **__W** CTRL
- **__W** GCLK
- **__W** IC
- **__RW** IM
- **__W** MATCH
- **__R** MIS
- **__W** PR
- **__R** RIS
- **__R** RXDATA
- **__W** RX_FIFO_FLUSH
- **__R** RX_FIFO_LEVEL
- **__W** RX_FIFO_THRESHOLD
- **__W** TXDATA
- **__W** TX_FIFO_FLUSH

- **__R** TX_FIFO_LEVEL
- **__W** TX_FIFO_THRESHOLD
- **__R** reserved_0
- **__R** reserved_1
- **__R** reserved_2
- **__R** reserved_3

1.18 Macros Documentation

1.18.1 define <tt>EF_UART_BRK_FLAG</tt>

```
#define EF_UART_BRK_FLAG 0x10
```

1.18.2 define <tt>EF_UART_CFG_REG_PARITY_BIT</tt>

```
#define EF_UART_CFG_REG_PARITY_BIT 5
```

1.18.3 define <tt>EF_UART_CFG_REG_PARITY_MASK</tt>

```
#define EF_UART_CFG_REG_PARITY_MASK 0xe0
```

1.18.4 define <tt>EF_UART_CFG_REG_STP2_BIT</tt>

```
#define EF_UART_CFG_REG_STP2_BIT 4
```

1.18.5 define <tt>EF_UART_CFG_REG_STP2_MASK</tt>

```
#define EF_UART_CFG_REG_STP2_MASK 0x10
```

1.18.6 define <tt>EF_UART_CFG_REG_TIMEOUT_BIT</tt>

```
#define EF_UART_CFG_REG_TIMEOUT_BIT 8
```

1.18.7 define <tt>EF_UART_CFG_REG_TIMEOUT_MASK</tt>

```
#define EF_UART_CFG_REG_TIMEOUT_MASK 0x3f00
```

1.18.8 define <tt>EF_UART_CFG_REG_WLEN_BIT</tt>

```
#define EF_UART_CFG_REG_WLEN_BIT 0
```

1.18.9 define <tt>EF_UART_CFG_REG_WLEN_MASK</tt>

```
#define EF_UART_CFG_REG_WLEN_MASK 0xf
```

1.18.10 define <tt>EF_UART_CTRL_REG_EN_BIT</tt>

```
#define EF_UART_CTRL_REG_EN_BIT 0
```

1.18.11 define <tt>EF_UART_CTRL_REG_EN_MASK</tt>

```
#define EF_UART_CTRL_REG_EN_MASK 0x1
```

1.18.12 define <tt>EF_UART_CTRL_REG_GFEN_BIT</tt>

```
#define EF_UART_CTRL_REG_GFEN_BIT 4
```

1.18.13 define <tt>EF_UART_CTRL_REG_GFEN_MASK</tt>

```
#define EF_UART_CTRL_REG_GFEN_MASK 0x10
```

1.18.14 define <tt>EF_UART_CTRL_REG_LPEN_BIT</tt>

```
#define EF_UART_CTRL_REG_LPEN_BIT 3
```

1.18.15 define <tt>EF_UART_CTRL_REG_LPEN_MASK</tt>

```
#define EF_UART_CTRL_REG_LPEN_MASK 0x8
```

1.18.16 define <tt>EF_UART_CTRL_REG_RXEN_BIT</tt>

```
#define EF_UART_CTRL_REG_RXEN_BIT 2
```

1.18.17 define <tt>EF_UART_CTRL_REG_RXEN_MASK</tt>

```
#define EF_UART_CTRL_REG_RXEN_MASK 0x4
```

1.18.18 define <tt>EF_UART_CTRL_REG_TXEN_BIT</tt>

```
#define EF_UART_CTRL_REG_TXEN_BIT 1
```

1.18.19 define <tt>EF_UART_CTRL_REG_TXEN_MASK</tt>

```
#define EF_UART_CTRL_REG_TXEN_MASK 0x2
```

1.18.20 define <tt>EF_UART_FE_FLAG</tt>

```
#define EF_UART_FE_FLAG 0x40
```

1.18.21 define <tt>EF_UART_MATCH_FLAG</tt>

```
#define EF_UART_MATCH_FLAG 0x20
```

1.18.22 define <tt>EF_UART_OR_FLAG</tt>

```
#define EF_UART_OR_FLAG 0x100
```

1.18.23 define <tt>EF_UART_PRE_FLAG</tt>

```
#define EF_UART_PRE_FLAG 0x80
```

1.18.24 define <tt>EF_UART_RTO_FLAG</tt>

```
#define EF_UART_RTO_FLAG 0x200
```

1.18.25 define <tt>EF_UART_RXA_FLAG</tt>

```
#define EF_UART_RXA_FLAG 0x8
```

1.18.26 define <tt>EF_UART_RXF_FLAG</tt>

```
#define EF_UART_RXF_FLAG 0x2
```

1.18.27 define <tt>EF_UART_RX_FIFO_FLUSH_REG_FLUSH_BIT</tt>

```
#define EF_UART_RX_FIFO_FLUSH_REG_FLUSH_BIT 0
```

1.18.28 define <tt>EF_UART_RX_FIFO_FLUSH_REG_FLUSH_MASK</tt>

```
#define EF_UART_RX_FIFO_FLUSH_REG_FLUSH_MASK 0x1
```

1.18.29 define <tt>EF_UART_RX_FIFO_LEVEL_REG_LEVEL_BIT</tt>

```
#define EF_UART_RX_FIFO_LEVEL_REG_LEVEL_BIT 0
```

1.18.30 define <tt>EF_UART_RX_FIFO_LEVEL_REG_LEVEL_MASK</tt>

```
#define EF_UART_RX_FIFO_LEVEL_REG_LEVEL_MASK 0xf
```

1.18.31 define <tt>EF_UART_RX_FIFO_THRESHOLD_REG_THRESHOLD_BIT</tt>

```
#define EF_UART_RX_FIFO_THRESHOLD_REG_THRESHOLD_BIT 0
```

1.18.32 define <tt>EF_UART_RX_FIFO_THRESHOLD_REG_THRESHOLD_MASK</tt>

```
#define EF_UART_RX_FIFO_THRESHOLD_REG_THRESHOLD_MASK 0xf
```

1.18.33 define <tt>EF_UART_TXB_FLAG</tt>

```
#define EF_UART_TXB_FLAG 0x4
```

1.18.34 define <tt>EF_UART_TXE_FLAG</tt>

```
#define EF_UART_TXE_FLAG 0x1
```

1.18.35 define <tt>EF_UART_TX_FIFO_FLUSH_REG_FLUSH_BIT</tt>

```
#define EF_UART_TX_FIFO_FLUSH_REG_FLUSH_BIT 0
```

1.18.36 define <tt>EF_UART_TX_FIFO_FLUSH_REG_FLUSH_MASK</tt>

```
#define EF_UART_TX_FIFO_FLUSH_REG_FLUSH_MASK 0x1
```

1.18.37 define <tt>EF_UART_TX_FIFO_LEVEL_REG_LEVEL_BIT</tt>

```
#define EF_UART_TX_FIFO_LEVEL_REG_LEVEL_BIT 0
```

1.18.38 define <tt>EF_UART_TX_FIFO_LEVEL_REG_LEVEL_MASK</tt>

```
#define EF_UART_TX_FIFO_LEVEL_REG_LEVEL_MASK 0xf
```

1.18.39 define <tt>EF_UART_TX_FIFO_THRESHOLD_REG_THRESHOLD_BIT</tt>

```
#define EF_UART_TX_FIFO_THRESHOLD_REG_THRESHOLD_BIT 0
```

1.18.40 define <tt>EF_UART_TX_FIFO_THRESHOLD_REG_THRESHOLD_MASK</tt>

```
#define EF_UART_TX_FIFO_THRESHOLD_REG_THRESHOLD_MASK 0xf
```

1.18.41 define <tt>IO_TYPES</tt>

```
#define IO_TYPES
```

1.18.42 define <tt>__R</tt>

```
#define __R volatile const unsigned int
```

1.18.43 define <tt>__RW</tt>

```
#define __RW volatile       unsigned int
```

1.18.44 define <tt>__W</tt>

```
#define __W volatile       unsigned int
```

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 _EF_UART_TYPE_ Struct Reference

Public Attributes

- __R **RXDATA**
- __W **TXDATA**
- __W **PR**
- __W **CTRL**
- __W **CFG**
- __R **reserved_0** [2]
- __W **MATCH**
- __R **reserved_1** [16248]
- __R **RX_FIFO_LEVEL**
- __W **RX_FIFO_THRESHOLD**
- __W **RX_FIFO_FLUSH**
- __R **reserved_2** [1]
- __R **TX_FIFO_LEVEL**
- __W **TX_FIFO_THRESHOLD**
- __W **TX_FIFO_FLUSH**
- __R **reserved_3** [57]
- __RW **IM**
- __R **MIS**
- __R **RIS**
- __W **IC**
- __W **GCLK**

The documentation for this struct was generated from the following file:

- EF_UART_regs.h

Chapter 5

File Documentation

5.1 EF_Driver_Common.h File Reference

C header file for common driver definitions and types.

```
#include <stdint.h>
#include <stdbool.h>
#include <stddef.h>
```

Macros

- #define **EF_DRIVER_OK** ((uint32_t)0)
  Operation succeeded.
- #define **EF_DRIVER_ERROR** ((uint32_t)1)
  Unspecified error.
- #define **EF_DRIVER_ERROR_BUSY** ((uint32_t)2)
  Driver is busy.
- #define **EF_DRIVER_ERROR_TIMEOUT** ((uint32_t)3)
  Timeout occurred.
- #define **EF_DRIVER_ERROR_UNSUPPORTED** ((uint32_t)4)
  Operation not supported.
- #define **EF_DRIVER_ERROR_PARAMETER** ((uint32_t)5)
  Parameter error.
- #define **EF_DRIVER_ERROR_SPECIFIC** ((uint32_t)6)
  Start of driver specific errors.

Typedefs

- typedef uint32_t **EF_DRIVER_STATUS**
  A type that is used to return the status of the driver functions.

5.1.1 Detailed Description

C header file for common driver definitions and types.

5.2 EF_Driver_Common.h

Go to the documentation of this file.

```
/*
    Copyright 2025 Efabless Corp.


    Licensed under the Apache License, Version 2.0 (the "License");
    you may not use this file except in compliance with the License.
    You may obtain a copy of the License at

        www.apache.org/licenses/LICENSE-2.0

    Unless required by applicable law or agreed to in writing, software
    distributed under the License is distributed on an "AS IS" BASIS,
    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
    See the License for the specific language governing permissions and
    limitations under the License.

*/

#ifndef EF_DRIVER_COMMON_H
#define EF_DRIVER_COMMON_H

/******************************************************************************
* Includes
******************************************************************************/
#include <stdint.h>
#include <stdbool.h>
#include <stddef.h>


/******************************************************************************
* Macros and Constants
******************************************************************************/
/* General return codes */
#define EF_DRIVER_OK                    ((uint32_t)0)
#define EF_DRIVER_ERROR                 ((uint32_t)1)
#define EF_DRIVER_ERROR_BUSY            ((uint32_t)2)
#define EF_DRIVER_ERROR_TIMEOUT         ((uint32_t)3)
#define EF_DRIVER_ERROR_UNSUPPORTED     ((uint32_t)4)
#define EF_DRIVER_ERROR_PARAMETER       ((uint32_t)5)
#define EF_DRIVER_ERROR_SPECIFIC        ((uint32_t)6)


/******************************************************************************
* Typedefs and Enums
******************************************************************************/

typedef uint32_t EF_DRIVER_STATUS;


/******************************************************************************
* External Variables
******************************************************************************/


/******************************************************************************
* Function Prototypes
******************************************************************************/


#endif // EF_DRIVER_COMMON_H

/******************************************************************************
* End of File
******************************************************************************/
```

5.3 EF_UART.c File Reference

C file for UART APIs which contains the function implmentations.

```
#include "EF_UART.h"
```

Functions

- EF_DRIVER_STATUS EF_UART_setGclkEnable (EF_UART_TYPE_PTR uart, uint32_t value)

  sets the GCLK enable bit in the UART register to a certain value
- EF_DRIVER_STATUS EF_UART_enable (EF_UART_TYPE_PTR uart)

  enables using uart by setting "en" bit in the control register to 1
- EF_DRIVER_STATUS EF_UART_disable (EF_UART_TYPE_PTR uart)

  disables using uart by clearing "en" bit in the control register
- EF_DRIVER_STATUS EF_UART_enableRx (EF_UART_TYPE_PTR uart)

  enables using uart RX by setting uart "rxen" bit in the control register to 1
- EF_DRIVER_STATUS EF_UART_disableRx (EF_UART_TYPE_PTR uart)

  disables using uart RX by clearing uart "rxen" bit in the control register
- EF_DRIVER_STATUS EF_UART_enableTx (EF_UART_TYPE_PTR uart)

  enables using uart TX by setting uart "txen" bit in the control register to 1
- EF_DRIVER_STATUS EF_UART_disableTx (EF_UART_TYPE_PTR uart)

  disables using uart TX by clearing uart "txen" bit in the control register
- EF_DRIVER_STATUS EF_UART_enableLoopBack (EF_UART_TYPE_PTR uart)

  enables loopback (connecting TX to RX signal) by setting "lpen" bit in the control register to 1
- EF_DRIVER_STATUS EF_UART_disableLoopBack (EF_UART_TYPE_PTR uart)

  disables loopback (connecting TX to RX signal) by clearing "lpen" bit in the control register
- EF_DRIVER_STATUS EF_UART_enableGlitchFilter (EF_UART_TYPE_PTR uart)

  enables glitch filter (filter out noise or glitches on the received signal) by setting "gfen" bit in the control register to 1
- EF_DRIVER_STATUS EF_UART_disableGlitchFilter (EF_UART_TYPE_PTR uart)

  disables glitch filter (filter out noise or glitches on the received signal) by clearing "gfen" bit in the control register
- EF_DRIVER_STATUS EF_UART_setCTRL (EF_UART_TYPE_PTR uart, uint32_t value)
- EF_DRIVER_STATUS EF_UART_getCTRL (EF_UART_TYPE_PTR uart, uint32_t ∗CTRL_value)

  returns the value of the control register
- EF_DRIVER_STATUS EF_UART_setPrescaler (EF_UART_TYPE_PTR uart, uint32_t prescaler)

  sets the prescaler to a certain value where Baud_rate = Bus_Clock_Freq/((Prescaler+1)∗16)
- EF_DRIVER_STATUS EF_UART_getPrescaler (EF_UART_TYPE_PTR uart, uint32_t ∗Prescaler_value)

  returns the value of the prescaler
- EF_DRIVER_STATUS EF_UART_setDataSize (EF_UART_TYPE_PTR uart, uint32_t value)

  sets the Data Size (Data word length: 5-9 bits) by setting the "wlen" field in configuration register
- EF_DRIVER_STATUS EF_UART_setTwoStopBitsSelect (EF_UART_TYPE_PTR uart, bool is_two_bits)

  sets the "stp2" bit in configuration register (whether the stop boits are two or one)
- EF_DRIVER_STATUS EF_UART_setParityType (EF_UART_TYPE_PTR uart, enum parity_type parity)

  sets the "parity" field in configuration register (could be none, odd, even, sticky 0 or sticky 1)
- EF_DRIVER_STATUS EF_UART_setTimeoutBits (EF_UART_TYPE_PTR uart, uint32_t value)

  sets the "timeout" field in configuration register which is receiver timeout measured in number of bits at which the timeout flag will be raised
- EF_DRIVER_STATUS EF_UART_setConfig (EF_UART_TYPE_PTR uart, uint32_t value)
- EF_DRIVER_STATUS EF_UART_getConfig (EF_UART_TYPE_PTR uart, uint32_t ∗CFG_value)

  returns the value of the configuration register
- EF_DRIVER_STATUS EF_UART_setRxFIFOThreshold (EF_UART_TYPE_PTR uart, uint32_t value)

  sets the RX FIFO threshold to a certain value at which "RXA" interrupt will be raised
- EF_DRIVER_STATUS EF_UART_getRxFIFOThreshold (EF_UART_TYPE_PTR uart, uint32_t ∗RX_FIFO↩
  _THRESHOLD_value)

  returns the current value of the RX FIFO threshold
- EF_DRIVER_STATUS EF_UART_setTxFIFOThreshold (EF_UART_TYPE_PTR uart, uint32_t value)

  sets the TX FIFO threshold to a certain value at which "TXB" interrupt will be raised
- EF_DRIVER_STATUS EF_UART_getTxFIFOThreshold (EF_UART_TYPE_PTR uart, uint32_t ∗TX_FIFO_↩
  THRESHOLD_value)

returns the current value of the TX FIFO threshold
- EF_DRIVER_STATUS EF_UART_getTxCount (EF_UART_TYPE_PTR uart, uint32_t ∗TX_FIFO_LEVEL_↩ value)

  returns the current level of the TX FIFO (the number of bytes in the FIFO)
- EF_DRIVER_STATUS EF_UART_getRxCount (EF_UART_TYPE_PTR uart, uint32_t ∗RX_FIFO_LEVEL_↩ value)

  returns the current level of the RX FIFO (the number of bytes in the FIFO)
- EF_DRIVER_STATUS EF_UART_setMatchData (EF_UART_TYPE_PTR uart, uint32_t matchData)

  sets the matchData to a certain value at which "MATCH" interrupt will be raised
- EF_DRIVER_STATUS EF_UART_getMatchData (EF_UART_TYPE_PTR uart, uint32_t ∗MATCH_value)

  returns the value of the match data register
- EF_DRIVER_STATUS EF_UART_getRIS (EF_UART_TYPE_PTR uart, uint32_t ∗RIS_value)
- EF_DRIVER_STATUS EF_UART_getMIS (EF_UART_TYPE_PTR uart, uint32_t ∗MIS_value)
- EF_DRIVER_STATUS EF_UART_setIM (EF_UART_TYPE_PTR uart, uint32_t mask)
- EF_DRIVER_STATUS EF_UART_getIM (EF_UART_TYPE_PTR uart, uint32_t ∗IM_value)
- EF_DRIVER_STATUS EF_UART_setICR (EF_UART_TYPE_PTR uart, uint32_t mask)
- EF_DRIVER_STATUS EF_UART_writeChar (EF_UART_TYPE_PTR uart, char data)

  transmit a single character through uart
- EF_DRIVER_STATUS EF_UART_writeCharArr (EF_UART_TYPE_PTR uart, const char ∗char_arr)

  transmit an array of characters through uart
- EF_DRIVER_STATUS EF_UART_readChar (EF_UART_TYPE_PTR uart, char ∗RXDATA_value)

  recieve a single character through uart
- EF_DRIVER_STATUS EF_UART_readCharNonBlocking (EF_UART_TYPE_PTR uart, char ∗RXDATA_value, bool ∗data_available)

  This is a non-blocking function that reads a character from the UART receive FIFO if data is available and returns a status code.
- EF_DRIVER_STATUS EF_UART_writeCharNonBlocking (EF_UART_TYPE_PTR uart, char data, bool ∗data_sent)

  This is a non-blocking function that writes a character to the UART transmit FIFO if the FIFO is not full and returns a status code.
- EF_DRIVER_STATUS EF_UART_charsAvailable (EF_UART_TYPE_PTR uart, bool ∗RXA_flag)

  This function returns a flag indicating whether or not there is data available in the receive FIFO.
- EF_DRIVER_STATUS EF_UART_spaceAvailable (EF_UART_TYPE_PTR uart, bool ∗TXB_flag)

  This function returns a flag indicating whether or not the transmit is available, i.e. the transmit FIFO is not full.
- EF_DRIVER_STATUS EF_UART_getParityMode (EF_UART_TYPE_PTR uart, uint32_t ∗parity_mode)

  This function return the parity mode of the UART.
- EF_DRIVER_STATUS EF_UART_busy (EF_UART_TYPE_PTR uart, bool ∗busy_flag)

  This function checks id the UART is busy.

5.3.1 Detailed Description

C file for UART APIs which contains the function implmentations.

5.3.2 Function Documentation

5.3.2.1 EF_UART_busy()

```
EF_DRIVER_STATUS EF_UART_busy (
            EF_UART_TYPE_PTR uart,
            bool * flag )
```

This function checks id the UART is busy.

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*flag*	a flag indicating if the UART is busy

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.2 EF_UART_charsAvailable()

```
EF_DRIVER_STATUS EF_UART_charsAvailable (
            EF_UART_TYPE_PTR uart,
            bool * flag )
```

This function returns a flag indicating whether or not there is data available in the receive FIFO.

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*flag*	a flag indicating if there is data available in the receive FIFO

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.3 EF_UART_disable()

```
EF_DRIVER_STATUS EF_UART_disable (
            EF_UART_TYPE_PTR uart )
```

disables using uart by clearing "en" bit in the control register

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.4 EF_UART_disableGlitchFilter()

```
EF_DRIVER_STATUS EF_UART_disableGlitchFilter (
            EF_UART_TYPE_PTR uart )
```

disables glitch filter (filter out noise or glitches on the received signal) by clearing "gfen" bit in the control register

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.5 EF_UART_disableLoopBack()

```
EF_DRIVER_STATUS EF_UART_disableLoopBack (
            EF_UART_TYPE_PTR uart )
```

disables loopback (connecting TX to RX signal) by clearing "lpen" bit in the control register

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.6 EF_UART_disableRx()

```
EF_DRIVER_STATUS EF_UART_disableRx (
            EF_UART_TYPE_PTR uart )
```

disables using uart RX by clearing uart "rxen" bit in the control register

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.7 EF_UART_disableTx()

```
EF_DRIVER_STATUS EF_UART_disableTx (
            EF_UART_TYPE_PTR uart )
```

disables using uart TX by clearing uart "txen" bit in the control register

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.8 EF_UART_enable()

```
EF_DRIVER_STATUS EF_UART_enable (
            EF_UART_TYPE_PTR uart )
```

enables using uart by setting "en" bit in the control register to 1

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.9 EF_UART_enableGlitchFilter()

```
EF_DRIVER_STATUS EF_UART_enableGlitchFilter (
            EF_UART_TYPE_PTR uart )
```

enables glitch filter (filter out noise or glitches on the received signal) by setting "gfen" bit in the control register to 1

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.10 EF_UART_enableLoopBack()

```
EF_DRIVER_STATUS EF_UART_enableLoopBack (
            EF_UART_TYPE_PTR uart )
```

enables loopback (connecting TX to RX signal) by setting "lpen" bit in the control register to 1

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.11 EF_UART_enableRx()

```
EF_DRIVER_STATUS EF_UART_enableRx (
            EF_UART_TYPE_PTR uart )
```

enables using uart RX by setting uart "rxen" bit in the control register to 1

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.12 EF_UART_enableTx()

```
EF_DRIVER_STATUS EF_UART_enableTx (
            EF_UART_TYPE_PTR uart )
```

enables using uart TX by setting uart "txen" bit in the control register to 1

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.13 EF_UART_getConfig()

```
EF_DRIVER_STATUS EF_UART_getConfig (
            EF_UART_TYPE_PTR uart,
            uint32_t * CFG_value )
```

returns the value of the configuration register

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*CFG_value*	The value of the configuration register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.14 EF_UART_getCTRL()

```
EF_DRIVER_STATUS EF_UART_getCTRL (
            EF_UART_TYPE_PTR uart,
            uint32_t * CTRL_value )
```

returns the value of the control register

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*CTRL_value*	The value of the control register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.15 EF_UART_getIM()

```
EF_DRIVER_STATUS EF_UART_getIM (
            EF_UART_TYPE_PTR uart,
            uint32_t * IM_value )
```

returns the value of the Interrupts Masking Register; which enable and disables interrupts

- bit 0 TXE : Transmit FIFO is Empty.
- bit 1 RXF : Receive FIFO is Full.
- bit 2 TXB : Transmit FIFO level is Below Threshold.
- bit 3 RXA : Receive FIFO level is Above Threshold.
- bit 4 BRK : Line Break; 13 consecutive 0's have been detected on the line.
- bit 5 MATCH : the receive data matches the MATCH register.
- bit 6 FE : Framing Error, the receiver does not see a "stop" bit at the expected "stop" bit time.
- bit 7 PRE : Parity Error; the receiver calculated parity does not match the received one.
- bit 8 OR : Overrun; data has been received but the RX FIFO is full.
- bit 9 RTO : Receiver Timeout; no data has been received for the time of a specified number of bits.

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*IM_value*	The value of the Interrupts Masking Register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.16 EF_UART_getMatchData()

```
EF_DRIVER_STATUS EF_UART_getMatchData (
            EF_UART_TYPE_PTR uart,
            uint32_t * MATCH_value )
```

returns the value of the match data register

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*MATCH_value*	The value of the match data register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.17 EF_UART_getMIS()

```
EF_DRIVER_STATUS EF_UART_getMIS (
            EF_UART_TYPE_PTR uart,
            uint32_t * MIS_value )
```

returns the value of the Masked Interrupt Status Register

- bit 0 TXE : Transmit FIFO is Empty.
- bit 1 RXF : Receive FIFO is Full.
- bit 2 TXB : Transmit FIFO level is Below Threshold.
- bit 3 RXA : Receive FIFO level is Above Threshold.
- bit 4 BRK : Line Break; 13 consecutive 0's have been detected on the line.
- bit 5 MATCH : the receive data matches the MATCH register.
- bit 6 FE : Framing Error, the receiver does not see a "stop" bit at the expected "stop" bit time.
- bit 7 PRE : Parity Error; the receiver calculated parity does not match the received one.
- bit 8 OR : Overrun; data has been received but the RX FIFO is full.
- bit 9 RTO : Receiver Timeout; no data has been received for the time of a specified number of bits.

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*MIS_value*	The value of the Masked Interrupt Status Register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.18 EF_UART_getParityMode()

```
EF_DRIVER_STATUS EF_UART_getParityMode (
            EF_UART_TYPE_PTR uart,
            uint32_t * parity_mode )
```

This function return the parity mode of the UART.

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*parity*	The parity mode of the UART

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.19 EF_UART_getPrescaler()

```
EF_DRIVER_STATUS EF_UART_getPrescaler (
            EF_UART_TYPE_PTR uart,
            uint32_t * Prescaler_value )
```

returns the value of the prescaler

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*Prescaler_value*	The value of the prescaler register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.20 EF_UART_getRIS()

```
EF_DRIVER_STATUS EF_UART_getRIS (
            EF_UART_TYPE_PTR uart,
            uint32_t * RIS_value )
```

returns the value of the Raw Interrupt Status Register

- bit 0 TXE : Transmit FIFO is Empty.
- bit 1 RXF : Receive FIFO is Full.
- bit 2 TXB : Transmit FIFO level is Below Threshold.
- bit 3 RXA : Receive FIFO level is Above Threshold.
- bit 4 BRK : Line Break; 13 consecutive 0's have been detected on the line.
- bit 5 MATCH : the receive data matches the MATCH register.
- bit 6 FE : Framing Error, the receiver does not see a "stop" bit at the expected "stop" bit time.
- bit 7 PRE : Parity Error; the receiver calculated parity does not match the received one.
- bit 8 OR : Overrun; data has been received but the RX FIFO is full.
- bit 9 RTO : Receiver Timeout; no data has been received for the time of a specified number of bits.

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*RIS_value*	The value of the Raw Interrupt Status Register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.21 EF_UART_getRxCount()

```
EF_DRIVER_STATUS EF_UART_getRxCount (
            EF_UART_TYPE_PTR uart,
            uint32_t * RX_FIFO_LEVEL_value )
```

returns the current level of the RX FIFO (the number of bytes in the FIFO)

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*RX_FIFO_LEVEL_value*	The value of the RX FIFO level register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.22 EF_UART_getRxFIFOThreshold()

```
EF_DRIVER_STATUS EF_UART_getRxFIFOThreshold (
            EF_UART_TYPE_PTR uart,
            uint32_t * RX_FIFO_THRESHOLD_value )
```

returns the current value of the RX FIFO threshold

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*RX_FIFO_THRESHOLD_value*	The value of the RX FIFO threshold register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.23 EF_UART_getTxCount()

```
EF_DRIVER_STATUS EF_UART_getTxCount (
            EF_UART_TYPE_PTR uart,
            uint32_t * TX_FIFO_LEVEL_value )
```

returns the current level of the TX FIFO (the number of bytes in the FIFO)

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*TX_FIFO_LEVEL_value*	The value of the TX FIFO level register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.24 EF_UART_getTxFIFOThreshold()

```
EF_DRIVER_STATUS EF_UART_getTxFIFOThreshold (
            EF_UART_TYPE_PTR uart,
            uint32_t * TX_FIFO_THRESHOLD_value )
```

returns the current value of the TX FIFO threshold

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*TX_FIFO_THRESHOLD_value*	The value of the TX FIFO threshold register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.25 EF_UART_readChar()

```
EF_DRIVER_STATUS EF_UART_readChar (
            EF_UART_TYPE_PTR uar,
            char * RXDATA_value )
```

recieve a single character through uart

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*RXDATA_value*	The value of the received character

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.26 EF_UART_readCharNonBlocking()

```
EF_DRIVER_STATUS EF_UART_readCharNonBlocking (
            EF_UART_TYPE_PTR uart,
            char * RXDATA_value,
            bool * data_available )
```

This is a non-blocking function that reads a character from the UART receive FIFO if data is available and returns a status code.

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*RXDATA_value*	The value of the received character
out	*data_available*	A flag indicating if data is available in the receive FIFO

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.27 EF_UART_setConfig()

```
EF_DRIVER_STATUS EF_UART_setConfig (
             EF_UART_TYPE_PTR uart,
             uint32_t config )
```

sets the configuration register to a certain value where

- bit 0-3: Data word length: 5-9 bits
- bit 4: Two Stop Bits Select
- bit 5-7: Parity Type: 000: None, 001: odd, 010: even, 100: Sticky 0, 101: Sticky 1
- bit 8-13: Receiver Timeout measured in number of bits

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*config*	The value of the configuration register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.28 EF_UART_setCTRL()

```
EF_DRIVER_STATUS EF_UART_setCTRL (
             EF_UART_TYPE_PTR uart,
             uint32_t value )
```

sets the control register to a certain value where

- bit 0: UART enable
- bit 1: UART Transmitter enable
- bit 2: UART Receiver enable
- bit 3: Loopback (connect RX and TX pins together) enable
- bit 4: UART Glitch Filer on RX enable

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*value*	The value of the control register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.29 EF_UART_setDataSize()

```
EF_DRIVER_STATUS EF_UART_setDataSize (
            EF_UART_TYPE_PTR uart,
            uint32_t value )
```

sets the Data Size (Data word length: 5-9 bits) by setting the "wlen" field in configuration register

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*value*	The value of the required data word length

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.30 EF_UART_setGclkEnable()

```
EF_DRIVER_STATUS EF_UART_setGclkEnable (
            EF_UART_TYPE_PTR uart,
            uint32_t value )
```

sets the GCLK enable bit in the UART register to a certain value

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*value*	The value of the GCLK enable bit

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.31 EF_UART_setICR()

```
EF_DRIVER_STATUS EF_UART_setICR (
            EF_UART_TYPE_PTR uart,
            uint32_t mask )
```

sets the value of the Interrupts Clear Register; write 1 to clear the flag

- bit 0 TXE : Transmit FIFO is Empty.
- bit 1 RXF : Receive FIFO is Full.
- bit 2 TXB : Transmit FIFO level is Below Threshold.
- bit 3 RXA : Receive FIFO level is Above Threshold.
- bit 4 BRK : Line Break; 13 consecutive 0's have been detected on the line.
- bit 5 MATCH : the receive data matches the MATCH register.
- bit 6 FE : Framing Error, the receiver does not see a "stop" bit at the expected "stop" bit time.
- bit 7 PRE : Parity Error; the receiver calculated parity does not match the received one.
- bit 8 OR : Overrun; data has been received but the RX FIFO is full.
- bit 9 RTO : Receiver Timeout; no data has been received for the time of a specified number of bits.

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*mask*	The required mask value

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.32 EF_UART_setIM()

```
EF_DRIVER_STATUS EF_UART_setIM (
            EF_UART_TYPE_PTR uart,
            uint32_t mask )
```

sets the value of the Interrupts Masking Register; which enable and disables interrupts

- bit 0 TXE : Transmit FIFO is Empty.
- bit 1 RXF : Receive FIFO is Full.
- bit 2 TXB : Transmit FIFO level is Below Threshold.
- bit 3 RXA : Receive FIFO level is Above Threshold.
- bit 4 BRK : Line Break; 13 consecutive 0's have been detected on the line.
- bit 5 MATCH : the receive data matches the MATCH register.
- bit 6 FE : Framing Error, the receiver does not see a "stop" bit at the expected "stop" bit time.
- bit 7 PRE : Parity Error; the receiver calculated parity does not match the received one.
- bit 8 OR : Overrun; data has been received but the RX FIFO is full.
- bit 9 RTO : Receiver Timeout; no data has been received for the time of a specified number of bits.

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*mask*	The required mask value

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.33 EF_UART_setMatchData()

```
EF_DRIVER_STATUS EF_UART_setMatchData (
            EF_UART_TYPE_PTR uart,
            uint32_t matchData )
```

sets the matchData to a certain value at which "MATCH" interrupt will be raised

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*matchData*	The value of the required match data

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.34 EF_UART_setParityType()

```
EF_DRIVER_STATUS EF_UART_setParityType (
            EF_UART_TYPE_PTR uart,
            enum parity_type parity )
```

sets the "parity" field in configuration register (could be none, odd, even, sticky 0 or sticky 1)

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*parity*	enum parity_type could be "NONE" , "ODD" , "EVEN" , "STICKY_0" , or "STICKY_1"

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.35 EF_UART_setPrescaler()

```
EF_DRIVER_STATUS EF_UART_setPrescaler (
            EF_UART_TYPE_PTR uart,
            uint32_t prescaler )
```

sets the prescaler to a certain value where Baud_rate = Bus_Clock_Freq/((Prescaler+1)∗16)

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*prescaler*	The value of the required prescaler

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.36 EF_UART_setRxFIFOThreshold()

```
EF_DRIVER_STATUS EF_UART_setRxFIFOThreshold (
            EF_UART_TYPE_PTR uart,
            uint32_t threshold )
```

sets the RX FIFO threshold to a certain value at which "RXA" interrupt will be raised

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*threshold*	The value of the required threshold

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.37 EF_UART_setTimeoutBits()

```
EF_DRIVER_STATUS EF_UART_setTimeoutBits (
            EF_UART_TYPE_PTR uart,
            uint32_t value )
```

sets the "timeout" field in configuration register which is receiver timeout measured in number of bits at which the timeout flag will be raised

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*value*	timeout bits value

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.38 EF_UART_setTwoStopBitsSelect()

```
EF_DRIVER_STATUS EF_UART_setTwoStopBitsSelect (
            EF_UART_TYPE_PTR uart,
            bool is_two_bits )
```

sets the "stp2" bit in configuration register (whether the stop boits are two or one)

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*is_two_bits*	bool value, if "true", the stop bits are two and if "false", the stop bit is one

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.39 EF_UART_setTxFIFOThreshold()

```
EF_DRIVER_STATUS EF_UART_setTxFIFOThreshold (
            EF_UART_TYPE_PTR uart,
            uint32_t threshold )
```

sets the TX FIFO threshold to a certain value at which "TXB" interrupt will be raised

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*threshold*	The value of the required threshold

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.40 EF_UART_spaceAvailable()

```
EF_DRIVER_STATUS EF_UART_spaceAvailable (
            EF_UART_TYPE_PTR uart,
            bool * flag )
```

This function returns a flag indicating whether or not the transmit is available, i.e. the transmit FIFO is not full.

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*flag*	a flag indicating if the transmit FIFO is not full

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.41 EF_UART_writeChar()

```
EF_DRIVER_STATUS EF_UART_writeChar (
            EF_UART_TYPE_PTR uart,
            char data )
```

transmit a single character through uart

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*data*	The character or byte required to send

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.42 EF_UART_writeCharArr()

```
EF_DRIVER_STATUS EF_UART_writeCharArr (
            EF_UART_TYPE_PTR uart,
            const char * char_arr )
```

transmit an array of characters through uart

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*char_arr*	An array of characters to send

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.3.2.43 EF_UART_writeCharNonBlocking()

```
EF_DRIVER_STATUS EF_UART_writeCharNonBlocking (
            EF_UART_TYPE_PTR uart,
            char data,
            bool * data_sent )
```

This is a non-blocking function that writes a character to the UART transmit FIFO if the FIFO is not full and returns a status code.

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*data*	The character or byte required to send
out	*data_sent*	A flag indicating if the data was sent successfully

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4 EF_UART.h File Reference

C header file for UART APIs which contains the function prototypes.

```
#include "EF_UART_regs.h"
#include "EF_Driver_Common.h"
```

Macros

- #define **EF_UART_CTRL_REG_MAX_VALUE** ((uint32_t)0x0000001F)
- #define **EF_UART_PR_REG_MAX_VALUE** ((uint32_t)0x0000FFFF)
- #define **EF_UART_DataLength_MIN_VALUE** ((uint32_t)0x00000005)
- #define **EF_UART_DataLength_MAX_VALUE** ((uint32_t)0x00000009)
- #define **EF_UART_CFG_REG_TIMEOUT_MAX_VALUE** ((uint32_t)0x0000003F)
- #define **EF_UART_CFG_REG_MAX_VALUE** ((uint32_t)0x00001FFF)
- #define **EF_UART_RX_FIFO_THRESHOLD_REG_MAX_VALUE** ((uint32_t)0x0000000F)
- #define **EF_UART_TX_FIFO_THRESHOLD_REG_MAX_VALUE** ((uint32_t)0x0000000F)
- #define **EF_UART_MATCH_REG_MAX_VALUE** ((uint32_t)0x00001FFF)
- #define **EF_UART_IM_REG_MAX_VALUE** ((uint32_t)0x000003FF)
- #define **EF_UART_IC_REG_MAX_VALUE** ((uint32_t)0x000003FF)

Enumerations

- enum **parity_type** {
  NONE = 0 , **ODD** = 1 , **EVEN** = 2 , **STICKY_0** = 4 ,
  STICKY_1 = 5 }

Functions

- EF_DRIVER_STATUS EF_UART_setGclkEnable (EF_UART_TYPE_PTR uart, uint32_t value)

  sets the GCLK enable bit in the UART register to a certain value
- EF_DRIVER_STATUS EF_UART_enable (EF_UART_TYPE_PTR uart)

  enables using uart by setting "en" bit in the control register to 1
- EF_DRIVER_STATUS EF_UART_disable (EF_UART_TYPE_PTR uart)

  disables using uart by clearing "en" bit in the control register
- EF_DRIVER_STATUS EF_UART_enableRx (EF_UART_TYPE_PTR uart)

  enables using uart RX by setting uart "rxen" bit in the control register to 1
- EF_DRIVER_STATUS EF_UART_disableRx (EF_UART_TYPE_PTR uart)

  disables using uart RX by clearing uart "rxen" bit in the control register
- EF_DRIVER_STATUS EF_UART_enableTx (EF_UART_TYPE_PTR uart)

  enables using uart TX by setting uart "txen" bit in the control register to 1
- EF_DRIVER_STATUS EF_UART_disableTx (EF_UART_TYPE_PTR uart)

  disables using uart TX by clearing uart "txen" bit in the control register
- EF_DRIVER_STATUS EF_UART_enableLoopBack (EF_UART_TYPE_PTR uart)

enables loopback (connecting TX to RX signal) by setting "lpen" bit in the control register to 1
- EF_DRIVER_STATUS EF_UART_disableLoopBack (EF_UART_TYPE_PTR uart)

  disables loopback (connecting TX to RX signal) by clearing "lpen" bit in the control register
- EF_DRIVER_STATUS EF_UART_enableGlitchFilter (EF_UART_TYPE_PTR uart)

  enables glitch filter (filter out noise or glitches on the received signal) by setting "gfen" bit in the control register to 1
- EF_DRIVER_STATUS EF_UART_disableGlitchFilter (EF_UART_TYPE_PTR uart)

  disables glitch filter (filter out noise or glitches on the received signal) by clearing "gfen" bit in the control register
- EF_DRIVER_STATUS EF_UART_setCTRL (EF_UART_TYPE_PTR uart, uint32_t value)
- EF_DRIVER_STATUS EF_UART_getCTRL (EF_UART_TYPE_PTR uart, uint32_t ∗CTRL_value)

  returns the value of the control register
- EF_DRIVER_STATUS EF_UART_setDataSize (EF_UART_TYPE_PTR uart, uint32_t value)

  sets the Data Size (Data word length: 5-9 bits) by setting the "wlen" field in configuration register
- EF_DRIVER_STATUS EF_UART_setTwoStopBitsSelect (EF_UART_TYPE_PTR uart, bool is_two_bits)

  sets the "stp2" bit in configuration register (whether the stop boits are two or one)
- EF_DRIVER_STATUS EF_UART_setParityType (EF_UART_TYPE_PTR uart, enum parity_type parity)

  sets the "parity" field in configuration register (could be none, odd, even, sticky 0 or sticky 1)
- EF_DRIVER_STATUS EF_UART_setTimeoutBits (EF_UART_TYPE_PTR uart, uint32_t value)

  sets the "timeout" field in configuration register which is receiver timeout measured in number of bits at which the timeout flag will be raised
- EF_DRIVER_STATUS EF_UART_setConfig (EF_UART_TYPE_PTR uart, uint32_t config)
- EF_DRIVER_STATUS EF_UART_getConfig (EF_UART_TYPE_PTR uart, uint32_t ∗CFG_value)

  returns the value of the configuration register
- EF_DRIVER_STATUS EF_UART_setRxFIFOThreshold (EF_UART_TYPE_PTR uart, uint32_t threshold)

  sets the RX FIFO threshold to a certain value at which "RXA" interrupt will be raised
- EF_DRIVER_STATUS EF_UART_getRxFIFOThreshold (EF_UART_TYPE_PTR uart, uint32_t ∗RX_FIFO↩_THRESHOLD_value)

  returns the current value of the RX FIFO threshold
- EF_DRIVER_STATUS EF_UART_setTxFIFOThreshold (EF_UART_TYPE_PTR uart, uint32_t threshold)

  sets the TX FIFO threshold to a certain value at which "TXB" interrupt will be raised
- EF_DRIVER_STATUS EF_UART_getTxFIFOThreshold (EF_UART_TYPE_PTR uart, uint32_t ∗TX_FIFO_↩THRESHOLD_value)

  returns the current value of the TX FIFO threshold
- EF_DRIVER_STATUS EF_UART_setMatchData (EF_UART_TYPE_PTR uart, uint32_t matchData)

  sets the matchData to a certain value at which "MATCH" interrupt will be raised
- EF_DRIVER_STATUS EF_UART_getMatchData (EF_UART_TYPE_PTR uart, uint32_t ∗MATCH_value)

  returns the value of the match data register
- EF_DRIVER_STATUS EF_UART_getTxCount (EF_UART_TYPE_PTR uart, uint32_t ∗TX_FIFO_LEVEL_↩value)

  returns the current level of the TX FIFO (the number of bytes in the FIFO)
- EF_DRIVER_STATUS EF_UART_getRxCount (EF_UART_TYPE_PTR uart, uint32_t ∗RX_FIFO_LEVEL_↩value)

  returns the current level of the RX FIFO (the number of bytes in the FIFO)
- EF_DRIVER_STATUS EF_UART_setPrescaler (EF_UART_TYPE_PTR uart, uint32_t prescaler)

  sets the prescaler to a certain value where Baud_rate = Bus_Clock_Freq/((Prescaler+1)∗16)
- EF_DRIVER_STATUS EF_UART_getPrescaler (EF_UART_TYPE_PTR uart, uint32_t ∗Prescaler_value)

  returns the value of the prescaler
- EF_DRIVER_STATUS EF_UART_getRIS (EF_UART_TYPE_PTR uart, uint32_t ∗RIS_value)
- EF_DRIVER_STATUS EF_UART_getMIS (EF_UART_TYPE_PTR uart, uint32_t ∗MIS_value)
- EF_DRIVER_STATUS EF_UART_setIM (EF_UART_TYPE_PTR uart, uint32_t mask)
- EF_DRIVER_STATUS EF_UART_getIM (EF_UART_TYPE_PTR uart, uint32_t ∗IM_value)
- EF_DRIVER_STATUS EF_UART_setICR (EF_UART_TYPE_PTR uart, uint32_t mask)
- EF_DRIVER_STATUS EF_UART_writeCharArr (EF_UART_TYPE_PTR uart, const char ∗char_arr)

transmit an array of characters through uart
- EF_DRIVER_STATUS EF_UART_writeChar (EF_UART_TYPE_PTR uart, char data)

  transmit a single character through uart
- EF_DRIVER_STATUS EF_UART_readChar (EF_UART_TYPE_PTR uar, char ∗RXDATA_value)

  recieve a single character through uart
- EF_DRIVER_STATUS EF_UART_readCharNonBlocking (EF_UART_TYPE_PTR uart, char ∗RXDATA_value, bool ∗data_available)

  This is a non-blocking function that reads a character from the UART receive FIFO if data is available and returns a status code.
- EF_DRIVER_STATUS EF_UART_writeCharNonBlocking (EF_UART_TYPE_PTR uart, char data, bool ∗data_sent)

  This is a non-blocking function that writes a character to the UART transmit FIFO if the FIFO is not full and returns a status code.
- EF_DRIVER_STATUS EF_UART_charsAvailable (EF_UART_TYPE_PTR uart, bool ∗flag)

  This function returns a flag indicating whether or not there is data available in the receive FIFO.
- EF_DRIVER_STATUS EF_UART_spaceAvailable (EF_UART_TYPE_PTR uart, bool ∗flag)

  This function returns a flag indicating whether or not the transmit is available, i.e. the transmit FIFO is not full.
- EF_DRIVER_STATUS EF_UART_getParityMode (EF_UART_TYPE_PTR uart, uint32_t ∗parity_mode)

  This function return the parity mode of the UART.
- EF_DRIVER_STATUS EF_UART_busy (EF_UART_TYPE_PTR uart, bool ∗flag)

  This function checks id the UART is busy.

5.4.1 Detailed Description

C header file for UART APIs which contains the function prototypes.

5.4.2 Function Documentation

5.4.2.1 EF_UART_busy()

```
EF_DRIVER_STATUS EF_UART_busy (
                EF_UART_TYPE_PTR uart,
                bool * flag )
```

This function checks id the UART is busy.

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*flag*	a flag indicating if the UART is busy

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.2 EF_UART_charsAvailable()

```
EF_DRIVER_STATUS EF_UART_charsAvailable (
            EF_UART_TYPE_PTR uart,
            bool * flag )
```

This function returns a flag indicating whether or not there is data available in the receive FIFO.

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*flag*	a flag indicating if there is data available in the receive FIFO

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.3 EF_UART_disable()

```
EF_DRIVER_STATUS EF_UART_disable (
            EF_UART_TYPE_PTR uart )
```

disables using uart by clearing "en" bit in the control register

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.4 EF_UART_disableGlitchFilter()

```
EF_DRIVER_STATUS EF_UART_disableGlitchFilter (
            EF_UART_TYPE_PTR uart )
```

disables glitch filter (filter out noise or glitches on the received signal) by clearing "gfen" bit in the control register

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.5 EF_UART_disableLoopBack()

```
EF_DRIVER_STATUS EF_UART_disableLoopBack (
            EF_UART_TYPE_PTR uart )
```

disables loopback (connecting TX to RX signal) by clearing "lpen" bit in the control register

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.6 EF_UART_disableRx()

```
EF_DRIVER_STATUS EF_UART_disableRx (
            EF_UART_TYPE_PTR uart )
```

disables using uart RX by clearing uart "rxen" bit in the control register

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.7 EF_UART_disableTx()

```
EF_DRIVER_STATUS EF_UART_disableTx (
            EF_UART_TYPE_PTR uart )
```

disables using uart TX by clearing uart "txen" bit in the control register

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.8 EF_UART_enable()

```
EF_DRIVER_STATUS EF_UART_enable (
            EF_UART_TYPE_PTR uart )
```

enables using uart by setting "en" bit in the control register to 1

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.9 EF_UART_enableGlitchFilter()

```
EF_DRIVER_STATUS EF_UART_enableGlitchFilter (
            EF_UART_TYPE_PTR uart )
```

enables glitch filter (filter out noise or glitches on the received signal) by setting "gfen" bit in the control register to 1

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.10 EF_UART_enableLoopBack()

```
EF_DRIVER_STATUS EF_UART_enableLoopBack (
            EF_UART_TYPE_PTR uart )
```

enables loopback (connecting TX to RX signal) by setting "lpen" bit in the control register to 1

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.11 EF_UART_enableRx()

```
EF_DRIVER_STATUS EF_UART_enableRx (
            EF_UART_TYPE_PTR uart )
```

enables using uart RX by setting uart "rxen" bit in the control register to 1

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.12 EF_UART_enableTx()

```
EF_DRIVER_STATUS EF_UART_enableTx (
            EF_UART_TYPE_PTR uart )
```

enables using uart TX by setting uart "txen" bit in the control register to 1

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.13 EF_UART_getConfig()

```
EF_DRIVER_STATUS EF_UART_getConfig (
            EF_UART_TYPE_PTR uart,
            uint32_t * CFG_value )
```

returns the value of the configuration register

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*CFG_value*	The value of the configuration register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.14 EF_UART_getCTRL()

```
EF_DRIVER_STATUS EF_UART_getCTRL (
            EF_UART_TYPE_PTR uart,
            uint32_t * CTRL_value )
```

returns the value of the control register

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*CTRL_value*	The value of the control register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.15 EF_UART_getIM()

```
EF_DRIVER_STATUS EF_UART_getIM (
            EF_UART_TYPE_PTR uart,
            uint32_t * IM_value )
```

returns the value of the Interrupts Masking Register; which enable and disables interrupts

- bit 0 TXE : Transmit FIFO is Empty.
- bit 1 RXF : Receive FIFO is Full.
- bit 2 TXB : Transmit FIFO level is Below Threshold.
- bit 3 RXA : Receive FIFO level is Above Threshold.
- bit 4 BRK : Line Break; 13 consecutive 0's have been detected on the line.
- bit 5 MATCH : the receive data matches the MATCH register.
- bit 6 FE : Framing Error, the receiver does not see a "stop" bit at the expected "stop" bit time.
- bit 7 PRE : Parity Error; the receiver calculated parity does not match the received one.
- bit 8 OR : Overrun; data has been received but the RX FIFO is full.
- bit 9 RTO : Receiver Timeout; no data has been received for the time of a specified number of bits.

Parameters

`in`	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
`out`	*IM_value*	The value of the Interrupts Masking Register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.16 EF_UART_getMatchData()

```
EF_DRIVER_STATUS EF_UART_getMatchData (
            EF_UART_TYPE_PTR uart,
            uint32_t * MATCH_value )
```

returns the value of the match data register

Parameters

`in`	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
`out`	*MATCH_value*	The value of the match data register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.17 EF_UART_getMIS()

```
EF_DRIVER_STATUS EF_UART_getMIS (
            EF_UART_TYPE_PTR uart,
            uint32_t * MIS_value )
```

returns the value of the Masked Interrupt Status Register

- bit 0 TXE : Transmit FIFO is Empty.
- bit 1 RXF : Receive FIFO is Full.
- bit 2 TXB : Transmit FIFO level is Below Threshold.
- bit 3 RXA : Receive FIFO level is Above Threshold.
- bit 4 BRK : Line Break; 13 consecutive 0's have been detected on the line.
- bit 5 MATCH : the receive data matches the MATCH register.
- bit 6 FE : Framing Error, the receiver does not see a "stop" bit at the expected "stop" bit time.
- bit 7 PRE : Parity Error; the receiver calculated parity does not match the received one.
- bit 8 OR : Overrun; data has been received but the RX FIFO is full.
- bit 9 RTO : Receiver Timeout; no data has been received for the time of a specified number of bits.

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*MIS_value*	The value of the Masked Interrupt Status Register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.18 EF_UART_getParityMode()

```
EF_DRIVER_STATUS EF_UART_getParityMode (
            EF_UART_TYPE_PTR uart,
            uint32_t * parity_mode )
```

This function return the parity mode of the UART.

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*parity*	The parity mode of the UART

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.19 EF_UART_getPrescaler()

```
EF_DRIVER_STATUS EF_UART_getPrescaler (
            EF_UART_TYPE_PTR uart,
            uint32_t * Prescaler_value )
```

returns the value of the prescaler

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*Prescaler_value*	The value of the prescaler register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.20 EF_UART_getRIS()

```
EF_DRIVER_STATUS EF_UART_getRIS (
            EF_UART_TYPE_PTR uart,
            uint32_t * RIS_value )
```

returns the value of the Raw Interrupt Status Register

- bit 0 TXE : Transmit FIFO is Empty.
- bit 1 RXF : Receive FIFO is Full.
- bit 2 TXB : Transmit FIFO level is Below Threshold.
- bit 3 RXA : Receive FIFO level is Above Threshold.
- bit 4 BRK : Line Break; 13 consecutive 0's have been detected on the line.
- bit 5 MATCH : the receive data matches the MATCH register.
- bit 6 FE : Framing Error, the receiver does not see a "stop" bit at the expected "stop" bit time.
- bit 7 PRE : Parity Error; the receiver calculated parity does not match the received one.
- bit 8 OR : Overrun; data has been received but the RX FIFO is full.
- bit 9 RTO : Receiver Timeout; no data has been received for the time of a specified number of bits.

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*RIS_value*	The value of the Raw Interrupt Status Register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.21 EF_UART_getRxCount()

```
EF_DRIVER_STATUS EF_UART_getRxCount (
            EF_UART_TYPE_PTR uart,
            uint32_t * RX_FIFO_LEVEL_value )
```

returns the current level of the RX FIFO (the number of bytes in the FIFO)

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*RX_FIFO_LEVEL_value*	The value of the RX FIFO level register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.22 EF_UART_getRxFIFOThreshold()

```
EF_DRIVER_STATUS EF_UART_getRxFIFOThreshold (
            EF_UART_TYPE_PTR uart,
            uint32_t * RX_FIFO_THRESHOLD_value )
```

returns the current value of the RX FIFO threshold

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*RX_FIFO_THRESHOLD_value*	The value of the RX FIFO threshold register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.23 EF_UART_getTxCount()

```
EF_DRIVER_STATUS EF_UART_getTxCount (
            EF_UART_TYPE_PTR uart,
            uint32_t * TX_FIFO_LEVEL_value )
```

returns the current level of the TX FIFO (the number of bytes in the FIFO)

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*TX_FIFO_LEVEL_value*	The value of the TX FIFO level register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.24 EF_UART_getTxFIFOThreshold()

```
EF_DRIVER_STATUS EF_UART_getTxFIFOThreshold (
            EF_UART_TYPE_PTR uart,
            uint32_t * TX_FIFO_THRESHOLD_value )
```

returns the current value of the TX FIFO threshold

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*TX_FIFO_THRESHOLD_value*	The value of the TX FIFO threshold register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.25 EF_UART_readChar()

```
EF_DRIVER_STATUS EF_UART_readChar (
            EF_UART_TYPE_PTR uar,
            char * RXDATA_value )
```

recieve a single character through uart

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*RXDATA_value*	The value of the received character

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.26 EF_UART_readCharNonBlocking()

```
EF_DRIVER_STATUS EF_UART_readCharNonBlocking (
            EF_UART_TYPE_PTR uart,
            char * RXDATA_value,
            bool * data_available )
```

This is a non-blocking function that reads a character from the UART receive FIFO if data is available and returns a status code.

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*RXDATA_value*	The value of the received character
out	*data_available*	A flag indicating if data is available in the receive FIFO

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.27 EF_UART_setConfig()

```
EF_DRIVER_STATUS EF_UART_setConfig (
            EF_UART_TYPE_PTR uart,
            uint32_t config )
```

sets the configuration register to a certain value where

- bit 0-3: Data word length: 5-9 bits
- bit 4: Two Stop Bits Select
- bit 5-7: Parity Type: 000: None, 001: odd, 010: even, 100: Sticky 0, 101: Sticky 1
- bit 8-13: Receiver Timeout measured in number of bits

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*config*	The value of the configuration register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.28 EF_UART_setCTRL()

```
EF_DRIVER_STATUS EF_UART_setCTRL (
            EF_UART_TYPE_PTR uart,
            uint32_t value )
```

sets the control register to a certain value where

- bit 0: UART enable
- bit 1: UART Transmitter enable
- bit 2: UART Receiver enable
- bit 3: Loopback (connect RX and TX pins together) enable
- bit 4: UART Glitch Filer on RX enable

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*value*	The value of the control register

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.29 EF_UART_setDataSize()

```
EF_DRIVER_STATUS EF_UART_setDataSize (
             EF_UART_TYPE_PTR uart,
             uint32_t value )
```

sets the Data Size (Data word length: 5-9 bits) by setting the "wlen" field in configuration register

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*value*	The value of the required data word length

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.30 EF_UART_setGclkEnable()

```
EF_DRIVER_STATUS EF_UART_setGclkEnable (
             EF_UART_TYPE_PTR uart,
             uint32_t value )
```

sets the GCLK enable bit in the UART register to a certain value

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*value*	The value of the GCLK enable bit

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.31 EF_UART_setICR()

```
EF_DRIVER_STATUS EF_UART_setICR (
             EF_UART_TYPE_PTR uart,
             uint32_t mask )
```

sets the value of the Interrupts Clear Register; write 1 to clear the flag

- bit 0 TXE : Transmit FIFO is Empty.
- bit 1 RXF : Receive FIFO is Full.
- bit 2 TXB : Transmit FIFO level is Below Threshold.
- bit 3 RXA : Receive FIFO level is Above Threshold.
- bit 4 BRK : Line Break; 13 consecutive 0's have been detected on the line.
- bit 5 MATCH : the receive data matches the MATCH register.
- bit 6 FE : Framing Error, the receiver does not see a "stop" bit at the expected "stop" bit time.
- bit 7 PRE : Parity Error; the receiver calculated parity does not match the received one.
- bit 8 OR : Overrun; data has been received but the RX FIFO is full.
- bit 9 RTO : Receiver Timeout; no data has been received for the time of a specified number of bits.

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*mask*	The required mask value

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.32 EF_UART_setIM()

```
EF_DRIVER_STATUS EF_UART_setIM (
            EF_UART_TYPE_PTR uart,
            uint32_t mask )
```

sets the value of the Interrupts Masking Register; which enable and disables interrupts

- bit 0 TXE : Transmit FIFO is Empty.
- bit 1 RXF : Receive FIFO is Full.
- bit 2 TXB : Transmit FIFO level is Below Threshold.
- bit 3 RXA : Receive FIFO level is Above Threshold.
- bit 4 BRK : Line Break; 13 consecutive 0's have been detected on the line.
- bit 5 MATCH : the receive data matches the MATCH register.
- bit 6 FE : Framing Error, the receiver does not see a "stop" bit at the expected "stop" bit time.
- bit 7 PRE : Parity Error; the receiver calculated parity does not match the received one.
- bit 8 OR : Overrun; data has been received but the RX FIFO is full.
- bit 9 RTO : Receiver Timeout; no data has been received for the time of a specified number of bits.

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*mask*	The required mask value

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.33 EF_UART_setMatchData()

```
EF_DRIVER_STATUS EF_UART_setMatchData (
             EF_UART_TYPE_PTR uart,
             uint32_t matchData )
```

sets the matchData to a certain value at which "MATCH" interrupt will be raised

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*matchData*	The value of the required match data

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.34 EF_UART_setParityType()

```
EF_DRIVER_STATUS EF_UART_setParityType (
             EF_UART_TYPE_PTR uart,
             enum parity_type parity )
```

sets the "parity" field in configuration register (could be none, odd, even, sticky 0 or sticky 1)

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*parity*	enum parity_type could be "NONE" , "ODD" , "EVEN" , "STICKY_0" , or "STICKY_1"

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.35 EF_UART_setPrescaler()

```
EF_DRIVER_STATUS EF_UART_setPrescaler (
            EF_UART_TYPE_PTR uart,
            uint32_t prescaler )
```

sets the prescaler to a certain value where Baud_rate = Bus_Clock_Freq/((Prescaler+1)∗16)

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*prescaler*	The value of the required prescaler

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.36 EF_UART_setRxFIFOThreshold()

```
EF_DRIVER_STATUS EF_UART_setRxFIFOThreshold (
            EF_UART_TYPE_PTR uart,
            uint32_t threshold )
```

sets the RX FIFO threshold to a certain value at which "RXA" interrupt will be raised

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*threshold*	The value of the required threshold

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.37 EF_UART_setTimeoutBits()

```
EF_DRIVER_STATUS EF_UART_setTimeoutBits (
            EF_UART_TYPE_PTR uart,
            uint32_t value )
```

sets the "timeout" field in configuration register which is receiver timeout measured in number of bits at which the timeout flag will be raised

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*value*	timeout bits value

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.38 EF_UART_setTwoStopBitsSelect()

```
EF_DRIVER_STATUS EF_UART_setTwoStopBitsSelect (
            EF_UART_TYPE_PTR uart,
            bool is_two_bits )
```

sets the "stp2" bit in configuration register (whether the stop boits are two or one)

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*is_two_bits*	bool value, if "true", the stop bits are two and if "false", the stop bit is one

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.39 EF_UART_setTxFIFOThreshold()

```
EF_DRIVER_STATUS EF_UART_setTxFIFOThreshold (
            EF_UART_TYPE_PTR uart,
            uint32_t threshold )
```

sets the TX FIFO threshold to a certain value at which "TXB" interrupt will be raised

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*threshold*	The value of the required threshold

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.40 EF_UART_spaceAvailable()

```
EF_DRIVER_STATUS EF_UART_spaceAvailable (
            EF_UART_TYPE_PTR uart,
            bool * flag )
```

This function returns a flag indicating whether or not the transmit is available, i.e. the transmit FIFO is not full.

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
out	*flag*	a flag indicating if the transmit FIFO is not full

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.41 EF_UART_writeChar()

```
EF_DRIVER_STATUS EF_UART_writeChar (
            EF_UART_TYPE_PTR uart,
            char data )
```

transmit a single character through uart

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*data*	The character or byte required to send

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.42 EF_UART_writeCharArr()

```
EF_DRIVER_STATUS EF_UART_writeCharArr (
            EF_UART_TYPE_PTR uart,
            const char * char_arr )
```

transmit an array of characters through uart

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*char_arr*	An array of characters to send

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.4.2.43 EF_UART_writeCharNonBlocking()

```
EF_DRIVER_STATUS EF_UART_writeCharNonBlocking (
            EF_UART_TYPE_PTR uart,
            char data,
            bool * data_sent )
```

This is a non-blocking function that writes a character to the UART transmit FIFO if the FIFO is not full and returns a status code.

Parameters

in	*uart*	An EF_UART_TYPE_PTR , which points to the base memory address of UART registers. EF_UART_TYPE is a structure that contains the UART registers.
in	*data*	The character or byte required to send
out	*data_sent*	A flag indicating if the data was sent successfully

Returns

status A value of type EF_DRIVER_STATUS : returns a success or error code

5.5 EF_UART.h

Go to the documentation of this file.

```
/*
    Copyright 2025 Efabless Corp.


    Licensed under the Apache License, Version 2.0 (the "License");
    you may not use this file except in compliance with the License.
    You may obtain a copy of the License at

        www.apache.org/licenses/LICENSE-2.0

    Unless required by applicable law or agreed to in writing, software
    distributed under the License is distributed on an "AS IS" BASIS,
    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
    See the License for the specific language governing permissions and
    limitations under the License.

*/
```

```


#ifndef EF_UART_H
#define EF_UART_H


/******************************************************************************
* Includes
*****************************************************************************/
#include "EF_UART_regs.h"
#include "EF_Driver_Common.h"


/******************************************************************************
* Macros and Constants
*****************************************************************************/
#define EF_UART_CTRL_REG_MAX_VALUE              ((uint32_t)0x0000001F)  // CTRL register only has 5 bits, and the rest are reserved
#define EF_UART_PR_REG_MAX_VALUE                ((uint32_t)0x0000FFFF)  // PR register only has 16 bits
#define EF_UART_DataLength_MIN_VALUE            ((uint32_t)0x00000005)  // This UART IP only supports data length from 5 to 9 bits
#define EF_UART_DataLength_MAX_VALUE            ((uint32_t)0x00000009)  // This UART IP only supports data length from 5 to 9 bits
#define EF_UART_CFG_REG_TIMEOUT_MAX_VALUE       ((uint32_t)0x0000003F)  // The CFG register timeout field is 6 bits
#define EF_UART_CFG_REG_MAX_VALUE               ((uint32_t)0x00001FFF)  // The CFG register is 13 bits
#define EF_UART_RX_FIFO_THRESHOLD_REG_MAX_VALUE ((uint32_t)0x0000000F)  // The RX FIFO level register is 4 bits
#define EF_UART_TX_FIFO_THRESHOLD_REG_MAX_VALUE ((uint32_t)0x0000000F)  // The TX FIFO level register is 4 bits
#define EF_UART_MATCH_REG_MAX_VALUE             ((uint32_t)0x00001FFF)  // The match register is 9 bits
#define EF_UART_IM_REG_MAX_VALUE                ((uint32_t)0x000003FF)  // The IM register is 10 bits
#define EF_UART_IC_REG_MAX_VALUE                ((uint32_t)0x000003FF)  // The IC register is 10 bits


/******************************************************************************
* Typedefs and Enums
*****************************************************************************/

enum parity_type {NONE = 0, ODD = 1, EVEN = 2, STICKY_0 = 4, STICKY_1 = 5};



/******************************************************************************
* Function Prototypes
*****************************************************************************/


EF_DRIVER_STATUS EF_UART_setGclkEnable (EF_UART_TYPE_PTR uart, uint32_t value);


EF_DRIVER_STATUS EF_UART_enable(EF_UART_TYPE_PTR uart);



EF_DRIVER_STATUS EF_UART_disable(EF_UART_TYPE_PTR uart);



EF_DRIVER_STATUS EF_UART_enableRx(EF_UART_TYPE_PTR uart);



EF_DRIVER_STATUS EF_UART_disableRx(EF_UART_TYPE_PTR uart);



EF_DRIVER_STATUS EF_UART_enableTx(EF_UART_TYPE_PTR uart);



EF_DRIVER_STATUS EF_UART_disableTx(EF_UART_TYPE_PTR uart);



EF_DRIVER_STATUS EF_UART_enableLoopBack(EF_UART_TYPE_PTR uart);



EF_DRIVER_STATUS EF_UART_disableLoopBack(EF_UART_TYPE_PTR uart);



EF_DRIVER_STATUS EF_UART_enableGlitchFilter(EF_UART_TYPE_PTR uart);

EF_DRIVER_STATUS EF_UART_disableGlitchFilter(EF_UART_TYPE_PTR uart);



EF_DRIVER_STATUS EF_UART_setCTRL(EF_UART_TYPE_PTR uart, uint32_t value);



EF_DRIVER_STATUS EF_UART_getCTRL(EF_UART_TYPE_PTR uart, uint32_t* CTRL_value);



EF_DRIVER_STATUS EF_UART_setDataSize(EF_UART_TYPE_PTR uart, uint32_t value);



EF_DRIVER_STATUS EF_UART_setTwoStopBitsSelect(EF_UART_TYPE_PTR uart, bool is_two_bits);



EF_DRIVER_STATUS EF_UART_setParityType(EF_UART_TYPE_PTR uart, enum parity_type parity);



EF_DRIVER_STATUS EF_UART_setTimeoutBits(EF_UART_TYPE_PTR uart, uint32_t value);



EF_DRIVER_STATUS EF_UART_setConfig(EF_UART_TYPE_PTR uart, uint32_t config);



EF_DRIVER_STATUS EF_UART_getConfig(EF_UART_TYPE_PTR uart, uint32_t* CFG_value);



EF_DRIVER_STATUS EF_UART_setRxFIFOThreshold(EF_UART_TYPE_PTR uart, uint32_t threshold);



EF_DRIVER_STATUS EF_UART_getRxFIFOThreshold(EF_UART_TYPE_PTR uart, uint32_t* RX_FIFO_THRESHOLD_value);



EF_DRIVER_STATUS EF_UART_setTxFIFOThreshold(EF_UART_TYPE_PTR uart, uint32_t threshold);



EF_DRIVER_STATUS EF_UART_getTxFIFOThreshold(EF_UART_TYPE_PTR uart, uint32_t* TX_FIFO_THRESHOLD_value);




EF_DRIVER_STATUS EF_UART_setMatchData(EF_UART_TYPE_PTR uart, uint32_t matchData);



EF_DRIVER_STATUS EF_UART_getMatchData(EF_UART_TYPE_PTR uart, uint32_t* MATCH_value);



EF_DRIVER_STATUS EF_UART_getTxCount(EF_UART_TYPE_PTR uart, uint32_t* TX_FIFO_LEVEL_value);



EF_DRIVER_STATUS EF_UART_getRxCount(EF_UART_TYPE_PTR uart, uint32_t* RX_FIFO_LEVEL_value);



EF_DRIVER_STATUS EF_UART_setPrescaler(EF_UART_TYPE_PTR uart, uint32_t prescaler);



EF_DRIVER_STATUS EF_UART_getPrescaler(EF_UART_TYPE_PTR uart, uint32_t* Prescaler_value);



EF_DRIVER_STATUS EF_UART_getRIS(EF_UART_TYPE_PTR uart, uint32_t* RIS_value);



EF_DRIVER_STATUS EF_UART_getMIS(EF_UART_TYPE_PTR uart, uint32_t* MIS_value);



EF_DRIVER_STATUS EF_UART_setIM(EF_UART_TYPE_PTR uart, uint32_t mask);



EF_DRIVER_STATUS EF_UART_getIM(EF_UART_TYPE_PTR uart, uint32_t* IM_value);



EF_DRIVER_STATUS EF_UART_setICR(EF_UART_TYPE_PTR uart, uint32_t mask);



EF_DRIVER_STATUS EF_UART_writeCharArr(EF_UART_TYPE_PTR uart, const char *char_arr);



EF_DRIVER_STATUS EF_UART_writeChar(EF_UART_TYPE_PTR uart, char data);



EF_DRIVER_STATUS EF_UART_readChar(EF_UART_TYPE_PTR uar, char* RXDATA_value);



// The following functions are not verified yet
/*****************************************************************************************************************************
/*****************************************************************************************************************************



EF_DRIVER_STATUS EF_UART_readCharNonBlocking(EF_UART_TYPE_PTR uart, char* RXDATA_value, bool*
data_available);


EF_DRIVER_STATUS EF_UART_writeCharNonBlocking(EF_UART_TYPE_PTR uart, char data, bool* data_sent);


EF_DRIVER_STATUS EF_UART_charsAvailable(EF_UART_TYPE_PTR uart, bool* flag);



EF_DRIVER_STATUS EF_UART_spaceAvailable(EF_UART_TYPE_PTR uart, bool* flag);


EF_DRIVER_STATUS EF_UART_getParityMode(EF_UART_TYPE_PTR uart, uint32_t* parity_mode);


EF_DRIVER_STATUS EF_UART_busy(EF_UART_TYPE_PTR uart, bool* flag);



/******************************************************************************
* External Variables
******************************************************************************/


#endif // EF_UART_H

/******************************************************************************
* End of File
******************************************************************************/
```

5.6 EF_UART_regs.h

```
/*
    Copyright 2024 Efabless Corp.

    Author: Mohamed Shalan (mshalan@efabless.com)

    Licensed under the Apache License, Version 2.0 (the "License");
    you may not use this file except in compliance with the License.
    You may obtain a copy of the License at

        http://www.apache.org/licenses/LICENSE-2.0

    Unless required by applicable law or agreed to in writing, software
    distributed under the License is distributed on an "AS IS" BASIS,
    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
    See the License for the specific language governing permissions and
    limitations under the License.

```

```
*/

#ifndef EF_UARTREGS_H
#define EF_UARTREGS_H


/******************************************************************************
* Includes
******************************************************************************/
#include "EF_Driver_Common.h"


/******************************************************************************
* Macros and Constants
******************************************************************************/
#ifndef IO_TYPES
#define IO_TYPES
#define   __R     volatile const uint32_t
#define   __W     volatile       uint32_t
#define   __RW    volatile       uint32_t
#endif

#define EF_UART_CTRL_REG_EN_BIT     ((uint32_t)0)
#define EF_UART_CTRL_REG_EN_MASK    ((uint32_t)0x1)
#define EF_UART_CTRL_REG_TXEN_BIT   ((uint32_t)1)
#define EF_UART_CTRL_REG_TXEN_MASK  ((uint32_t)0x2)
#define EF_UART_CTRL_REG_RXEN_BIT   ((uint32_t)2)
#define EF_UART_CTRL_REG_RXEN_MASK  ((uint32_t)0x4)
#define EF_UART_CTRL_REG_LPEN_BIT   ((uint32_t)3)
#define EF_UART_CTRL_REG_LPEN_MASK  ((uint32_t)0x8)
#define EF_UART_CTRL_REG_GFEN_BIT   ((uint32_t)4)
#define EF_UART_CTRL_REG_GFEN_MASK  ((uint32_t)0x10)
#define EF_UART_CFG_REG_WLEN_BIT    ((uint32_t)0)
#define EF_UART_CFG_REG_WLEN_MASK   ((uint32_t)0xf)
#define EF_UART_CFG_REG_STP2_BIT    ((uint32_t)4)
#define EF_UART_CFG_REG_STP2_MASK   ((uint32_t)0x10)
#define EF_UART_CFG_REG_PARITY_BIT  ((uint32_t)5)
#define EF_UART_CFG_REG_PARITY_MASK ((uint32_t)0xe0)
#define EF_UART_CFG_REG_TIMEOUT_BIT ((uint32_t)8)
#define EF_UART_CFG_REG_TIMEOUT_MASK                    ((uint32_t)0x3f)
#define EF_UART_RX_FIFO_LEVEL_REG_LEVEL_BIT             ((uint32_t)0)
#define EF_UART_RX_FIFO_LEVEL_REG_LEVEL_MASK            ((uint32_t)0xf)
#define EF_UART_RX_FIFO_THRESHOLD_REG_THRESHOLD_BIT     ((uint32_t)0)
#define EF_UART_RX_FIFO_THRESHOLD_REG_THRESHOLD_MASK    ((uint32_t)0xf)
#define EF_UART_RX_FIFO_FLUSH_REG_FLUSH_BIT             ((uint32_t)0)
#define EF_UART_RX_FIFO_FLUSH_REG_FLUSH_MASK            ((uint32_t)0x1)
#define EF_UART_TX_FIFO_LEVEL_REG_LEVEL_BIT             ((uint32_t)0)
#define EF_UART_TX_FIFO_LEVEL_REG_LEVEL_MASK            ((uint32_t)0xf)
#define EF_UART_TX_FIFO_THRESHOLD_REG_THRESHOLD_BIT     ((uint32_t)0)
#define EF_UART_TX_FIFO_THRESHOLD_REG_THRESHOLD_MASK    ((uint32_t)0xf)
#define EF_UART_TX_FIFO_FLUSH_REG_FLUSH_BIT             ((uint32_t)0)
#define EF_UART_TX_FIFO_FLUSH_REG_FLUSH_MASK            ((uint32_t)0x1)

#define EF_UART_TXE_FLAG    ((uint32_t)0x1)
#define EF_UART_RXF_FLAG    ((uint32_t)0x2)
#define EF_UART_TXB_FLAG    ((uint32_t)0x4)
#define EF_UART_RXA_FLAG    ((uint32_t)0x8)
#define EF_UART_BRK_FLAG    ((uint32_t)0x10)
#define EF_UART_MATCH_FLAG  ((uint32_t)0x20)
#define EF_UART_FE_FLAG     ((uint32_t)0x40)
#define EF_UART_PRE_FLAG    ((uint32_t)0x80)
#define EF_UART_OR_FLAG     ((uint32_t)0x100)
#define EF_UART_RTO_FLAG    ((uint32_t)0x200)


/******************************************************************************
* Typedefs and Enums
******************************************************************************/
typedef struct _EF_UART_TYPE_ {
    __R     RXDATA;
    __W     TXDATA;
    __W     PR;
    __W     CTRL;
    __W     CFG;
    __R     reserved_0[2];
    __W     MATCH;
    __R     reserved_1[16248];
    __R     RX_FIFO_LEVEL;
    __W     RX_FIFO_THRESHOLD;
    __W     RX_FIFO_FLUSH;
    __R     reserved_2[1];
    __R     TX_FIFO_LEVEL;
    __W     TX_FIFO_THRESHOLD;
    __W     TX_FIFO_FLUSH;
    __R     reserved_3[57];
    __RW    IM;
    __R     MIS;
    __R     RIS;
    __W     IC;
    __W     GCLK;
} EF_UART_TYPE;


typedef EF_UART_TYPE* EF_UART_TYPE_PTR;



/******************************************************************************
* Function Prototypes
******************************************************************************/



/******************************************************************************
* External Variables
******************************************************************************/




#endif // EF_UARTREGS_H

/******************************************************************************
* End of File
******************************************************************************/
```

Index