EF_UART APIs

Generated by Doxygen 1.9.6

Chapter 1

APIs Documentation

1.0.1 EF_UART_enable

```
void EF_UART_enable(
    uint32_t uart_base
)
```

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

Parameters:

- **uart_base** The base memory address of UART registers.

1.0.2 EF_UART_disable

```
void EF_UART_disable(
    uint32_t uart_base
)
```

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

Parameters:

- **uart_base** The base memory address of UART registers.

1.0.3 EF_UART_enableRx

```
void EF_UART_enableRx(
    uint32_t uart_base
)
```

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

Parameters:

- **uart_base** The base memory address of UART registers.

1.0.4 EF_UART_disableRx

```
void EF_UART_disableRx(
    uint32_t uart_base
)
```

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

Parameters:

- **uart_base** The base memory address of UART registers.

1.0.5 EF_UART_enableTx

```
void EF_UART_enableTx(
    uint32_t uart_base
)
```

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

Parameters:

- **uart_base** The base memory address of UART registers.

1.0.6 EF_UART_disableTx

```
void EF_UART_disableTx(
    uint32_t uart_base
)
```

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

Parameters:

- **uart_base** The base memory address of UART registers.

1.0.7 EF_UART_enableLoopBack

```
void EF_UART_enableLoopBack(
    uint32_t uart_base
)
```

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

Parameters:

- **uart_base** The base memory address of UART registers.

1.0.8 EF_UART_disableLoopBack

```
void EF_UART_disableLoopBack(
    uint32_t uart_base
)
```

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

Parameters:

- **uart_base** The base memory address of UART registers.

1.0.9 EF_UART_enableGlitchFilter

```
void EF_UART_enableGlitchFilter(
    uint32_t uart_base
)
```

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

Parameters:

- **uart_base** The base memory address of UART registers.

1.0.10 EF_UART_disableGlitchFilter

```
void EF_UART_disableGlitchFilter(
    uint32_t uart_base
)
```

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

Parameters:

- **uart_base** The base memory address of UART registers.

1.0.11 EF_UART_setCTRL

```
void EF_UART_setCTRL(
    uint32_t uart_base,
    int value
)
```

Parameters:

- **uart_base** The base memory address of UART registers.
- **value** The value of the control register

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

1.0.12 EF_UART_getCTRL

```
int EF_UART_getCTRL(
    uint32_t uart_base
)
```

returns the value of the control register

Parameters:

- **uart_base** The base memory address of UART registers.

Return: control register value

1.0.13 EF_UART_setPrescaler

```
void EF_UART_setPrescaler(
    uint32_t uart_base,
    int prescaler
)
```

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

Parameters:

- **uart_base** The base memory address of UART registers.
- **prescaler** The value of the required prescaler

1.0.14 EF_UART_getPrescaler

```
int EF_UART_getPrescaler(
    uint32_t uart_base
)
```

returns the value of the prescaler

Parameters:

- **uart_base** The base memory address of UART registers.

Return: prescaler register value

1.0.15 EF_UART_setDataSize

```
void EF_UART_setDataSize(
    uint32_t uart_base,
    int value
)
```

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

Parameters:

- **uart_base** The base memory address of UART registers.
- **value** The value of the required data word length

1.0.16 EF_UART_setTwoStopBitsSelect

```
void EF_UART_setTwoStopBitsSelect(
    uint32_t uart_base,
    bool is_two_bits
)
```

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

Parameters:

- **uart_base** The base memory address of UART registers.
- **is_two_bits** bool value, if "true", the stop bits are two and if "false", the stop bit is one

1.0.17 EF_UART_setParityType

```
void EF_UART_setParityType(
    uint32_t uart_base,
    enum parity_type parity
)
```

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

Parameters:

- **uart_base** The base memory address of UART registers.
- **parity** enum parity_type could be "NONE" , "ODD" , "EVEN" , "STICKY_0" , or "STICKY_1"

1.0.18 EF_UART_setTimeoutBits

```
void EF_UART_setTimeoutBits(
    uint32_t uart_base,
    int 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:

- **uart_base** The base memory address of UART registers.
- **value** timeout bits value

1.0.19 EF_UART_setConfig

```
void EF_UART_setConfig(
    uint32_t uart_base,
    int config
)
```

Parameters:

- **uart_base** The base memory address of UART registers.
- **config** The value of the configuration register

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

1.0.20 EF_UART_getConfig

```
int EF_UART_getConfig(
    uint32_t uart_base
)
```

returns the value of the configuration register

Parameters:

- **uart_base** The base memory address of UART registers.

Return: configuration register value

1.0.21 EF_UART_setRxFIFOThreshold

```
void EF_UART_setRxFIFOThreshold(
    uint32_t uart_base,
    int threshold
)
```

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

Parameters:

- **uart_base** The base memory address of UART registers.
- **threshold** The value of the required threshold

1.0.22 EF_UART_getRxFIFOThreshold

```
int EF_UART_getRxFIFOThreshold(
    uint32_t uart_base
)
```

returns the current value of the RX FIFO threshold

Parameters:

- **uart_base** The base memory address of UART registers.

Return: RX FIFO threshold register

1.0.23 EF_UART_setTxFIFOThreshold

```
void EF_UART_setTxFIFOThreshold(
    uint32_t uart_base,
    int threshold
)
```

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

Parameters:

- **uart_base** The base memory address of UART registers.
- **threshold** The value of the required threshold

1.0.24 EF_UART_getTxFIFOThreshold

```
int EF_UART_getTxFIFOThreshold(
    uint32_t uart_base
)
```

returns the current value of the TX FIFO threshold

Parameters:

- **uart_base** The base memory address of UART registers.

Return: TX FIFO threshold register

1.0.25 EF_UART_setFIFOControl

```
void EF_UART_setFIFOControl(
    uint32_t uart_base,
    int value
)
```

Parameters:

- **uart_base** The base memory address of UART registers.
- **config** The value of the FIFO control register

sets the FIFO control register to a certain value where

- bit 0-3: Transmit FIFO Level Threshold
- bit 8-11: Receive FIFO Level Threshold

1.0.26 EF_UART_getFIFOControl

```
int EF_UART_getFIFOControl(
    uint32_t uart_base
)
```

returns the value of the FIFO control register

Parameters:

- **uart_base** The base memory address of UART registers.

Return: FIFO control register value

1.0.27 EF_UART_getTxCount

```
int EF_UART_getTxCount(
    uint32_t uart_base
)
```

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

Parameters:

- **uart_base** The base memory address of UART registers.

Return: TX FIFO level register

1.0.28 EF_UART_getRxCount

```
int EF_UART_getRxCount(
    uint32_t uart_base
)
```

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

Parameters:

- **uart_base** The base memory address of UART registers.

Return: RX FIFO level register

1.0.29 EF_UART_getFIFOStatus

```
int EF_UART_getFIFOStatus(
    uint32_t uart_base
)
```

Parameters:

- **uart_base** The base memory address of UART registers.

Return: FIFO status register value

returns the value of the FIFO status register where

- bit 0-3: Receive FIFO Level
- bit 8-11: Transmit FIFO Level

1.0.30 EF_UART_setMatchData

```
void EF_UART_setMatchData(
    uint32_t uart_base,
    int matchData
)
```

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

Parameters:

- **uart_base** The base memory address of UART registers.
- **matchData** The value of the required match data

1.0.31 EF_UART_getMatchData

```
int EF_UART_getMatchData(
    uint32_t uart_base
)
```

returns the value of the match data register

Parameters:

- **uart_base** The base memory address of UART registers.

Return: match data register value

1.0.32 EF_UART_getRIS

```
int EF_UART_getRIS(
    uint32_t uart_base
)
```

Parameters:

- **uart_base** The base memory address of UART registers.

Return: RIS register 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.

1.0.33 EF_UART_getMIS

```
int EF_UART_getMIS(
    uint32_t uart_base
)
```

Parameters:

- **uart_base** The base memory address of UART registers.

Return: MIS register 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.

1.0.34 EF_UART_setIM

```
void EF_UART_setIM(
    uint32_t uart_base,
    int mask
)
```

Parameters:

- **uart_base** The base memory address of UART registers.
- **mask** The required mask value

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.

1.0.35 EF_UART_getIM

```
int EF_UART_getIM(
    uint32_t uart_base
)
```

Parameters:

- **uart_base** The base memory address of UART registers.

Return: IM register 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.

1.0.36 EF_UART_setICR

```
void EF_UART_setICR(
    uint32_t uart_base,
    int mask
)
```

Parameters:

- **uart_base** The base memory address of UART registers.
- **mask** The required mask value

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.

1.0.37 EF_UART_writeChar

```
void EF_UART_writeChar(
    uint32_t uart_base,
    char data
)
```

transmit a single character through uart

Parameters:

- **uart_base** The base memory address of UART registers.
- **data** The character or byte required to send

1.0.38 EF_UART_writeCharArr

```
void EF_UART_writeCharArr(
    uint32_t uart_base,
    const char * char_arr
)
```

transmit an array of characters through uart

Parameters:

- **uart_base** The base memory address of UART registers.
- **char_arr** An array of characters to send

1.0.39 EF_UART_readChar

```
int EF_UART_readChar(
    uint32_t uart_base
)
```

receive a single character through uart

Parameters:

- **uart_base** The base memory address of UART registers.

Return: the byte received

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 files with brief descriptions:

Chapter 4

Class Documentation

4.1 _EF_UART_TYPE_ Struct Reference

`#include <EF_UART_regs.h>`

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

4.1.1 Member Data Documentation

4.1.1.1 CFG

`__W _EF_UART_TYPE_::CFG`

4.1.1.2 CTRL

`__W _EF_UART_TYPE_::CTRL`

4.1.1.3 GCLK

__W _EF_UART_TYPE_::GCLK

4.1.1.4 IC

__W _EF_UART_TYPE_::IC

4.1.1.5 IM

__RW _EF_UART_TYPE_::IM

4.1.1.6 MATCH

__W _EF_UART_TYPE_::MATCH

4.1.1.7 MIS

__R _EF_UART_TYPE_::MIS

4.1.1.8 PR

__W _EF_UART_TYPE_::PR

4.1.1.9 reserved_0

__R _EF_UART_TYPE_::reserved_0[2]

4.1.1.10 reserved_1

__R _EF_UART_TYPE_::reserved_1[16248]

4.1.1.11 reserved_2

__R _EF_UART_TYPE_::reserved_2[1]

4.1.1.12 reserved_3

__R _EF_UART_TYPE_::reserved_3[57]

4.1.1.13 RIS

__R _EF_UART_TYPE_::RIS

4.1.1.14 RX_FIFO_FLUSH

__W _EF_UART_TYPE_::RX_FIFO_FLUSH

4.1.1.15 RX_FIFO_LEVEL

__R _EF_UART_TYPE_::RX_FIFO_LEVEL

4.1.1.16 RX_FIFO_THRESHOLD

__W _EF_UART_TYPE_::RX_FIFO_THRESHOLD

4.1.1.17 RXDATA

__R _EF_UART_TYPE_::RXDATA

4.1.1.18 TX_FIFO_FLUSH

__W _EF_UART_TYPE_::TX_FIFO_FLUSH

4.1.1.19 TX_FIFO_LEVEL

__R _EF_UART_TYPE_::TX_FIFO_LEVEL

4.1.1.20 TX_FIFO_THRESHOLD

__W _EF_UART_TYPE_::TX_FIFO_THRESHOLD

4.1.1.21 TXDATA

__W _EF_UART_TYPE_::TXDATA

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 0

  Operation succeeded.
- #define EF_DRIVER_ERROR 1

  Unspecified error.
- #define EF_DRIVER_ERROR_BUSY 2

  Driver is busy.
- #define EF_DRIVER_ERROR_TIMEOUT 3

  Timeout occurred.
- #define EF_DRIVER_ERROR_UNSUPPORTED 4

  Operation not supported.
- #define EF_DRIVER_ERROR_PARAMETER 5

  Parameter error.
- #define EF_DRIVER_ERROR_SPECIFIC 6

  Start of driver specific errors.
- #define EF_DRIVER_ERROR_NO_DATA 7

  No data available.

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.

5.1.2 Macro Definition Documentation

5.1.2.1 EF_DRIVER_ERROR

```
#define EF_DRIVER_ERROR 1
```
Unspecified error.

5.1.2.2 EF_DRIVER_ERROR_BUSY

```
#define EF_DRIVER_ERROR_BUSY 2
```
Driver is busy.

5.1.2.3 EF_DRIVER_ERROR_NO_DATA

```
#define EF_DRIVER_ERROR_NO_DATA 7
```
No data available.

5.1.2.4 EF_DRIVER_ERROR_PARAMETER

```
#define EF_DRIVER_ERROR_PARAMETER 5
```
Parameter error.

5.1.2.5 EF_DRIVER_ERROR_SPECIFIC

```
#define EF_DRIVER_ERROR_SPECIFIC 6
```
Start of driver specific errors.

5.1.2.6 EF_DRIVER_ERROR_TIMEOUT

```
#define EF_DRIVER_ERROR_TIMEOUT 3
```
Timeout occurred.

5.1.2.7 EF_DRIVER_ERROR_UNSUPPORTED

```
#define EF_DRIVER_ERROR_UNSUPPORTED 4
```
Operation not supported.

5.1.2.8 EF_DRIVER_OK

```
#define EF_DRIVER_OK 0
```
Operation succeeded.

5.1.3 Typedef Documentation

5.1.3.1 EF_DRIVER_STATUS

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

5.2 EF_Driver_Common.h

Go to the documentation of this file.

```
00001 /*
00002     Copyright 2025 Efabless Corp.
00003
00004
00005     Licensed under the Apache License, Version 2.0 (the "License");
00006     you may not use this file except in compliance with the License.
00007     You may obtain a copy of the License at
00008
00009         www.apache.org/licenses/LICENSE-2.0
00010
00011     Unless required by applicable law or agreed to in writing, software
00012     distributed under the License is distributed on an "AS IS" BASIS,
00013     WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
00014     See the License for the specific language governing permissions and
00015     limitations under the License.
00016
00017 */
00018
00025 #ifndef EF_DRIVER_COMMON_H
00026 #define EF_DRIVER_COMMON_H
00027
00028 /*****************************************************************************
00029 * Includes
00030 *****************************************************************************/
00031 #include <stdint.h>
00032 #include <stdbool.h>
00033 #include <stddef.h>
00034
00035
00036 /*****************************************************************************
00037 * Macros and Constants
00038 *****************************************************************************/
00039 /* General return codes */
00040 #define EF_DRIVER_OK                    0
00041 #define EF_DRIVER_ERROR                 1
00042 #define EF_DRIVER_ERROR_BUSY            2
00043 #define EF_DRIVER_ERROR_TIMEOUT         3
00044 #define EF_DRIVER_ERROR_UNSUPPORTED     4
00045 #define EF_DRIVER_ERROR_PARAMETER       5
00046 #define EF_DRIVER_ERROR_SPECIFIC        6
00047 #define EF_DRIVER_ERROR_NO_DATA         7
00048
00049
00050 /*****************************************************************************
00051 * Typedefs and Enums
00052 *****************************************************************************/
00053
00054 typedef uint32_t EF_DRIVER_STATUS;
00055
00056
00057 /*****************************************************************************
00058 * External Variables
00059 *****************************************************************************/
00060
00061
00062 /*****************************************************************************
00063 * Function Prototypes
00064 *****************************************************************************/
00065
00066
00067 #endif // EF_DRIVER_COMMON_H
00068
00069 /*****************************************************************************
00070 * End of File
00071 *****************************************************************************/
```

5.3 EF_UART.c File Reference

C file for UART APIs which contains the function implmentations.

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

Macros

- #define EF_UART_C

Functions

- 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_enable (EF_UART_TYPE *uart)

  enables using uart by setting "en" bit in the control register to 1
- 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_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_disableRx (EF_UART_TYPE *uart)

  disables using uart RX by clearing uart "rxen" bit in the control register
- 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_disableTx (EF_UART_TYPE *uart)

  disables using uart TX by clearing uart "txen" bit in the control register
- 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_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_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_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_setCTRL (EF_UART_TYPE *uart, uint32_t value)
- 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_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_getPrescaler (EF_UART_TYPE *uart, uint32_t *Prescaler_value)

  returns the value of the prescaler
- 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_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_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_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_setConfig (EF_UART_TYPE *uart, uint32_t value)
- 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_setRxFIFOThreshold (EF_UART_TYPE *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 *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 *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 *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 *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 *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 *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 *uart, uint32_t *MATCH_value)

  returns the value of the match data register
- EF_DRIVER_STATUS EF_UART_getRIS (EF_UART_TYPE *uart, uint32_t *RIS_value)
- EF_DRIVER_STATUS EF_UART_getMIS (EF_UART_TYPE *uart, uint32_t *MIS_value)
- EF_DRIVER_STATUS EF_UART_setIM (EF_UART_TYPE *uart, uint32_t mask)
- EF_DRIVER_STATUS EF_UART_getIM (EF_UART_TYPE *uart, uint32_t *IM_value)
- EF_DRIVER_STATUS EF_UART_setICR (EF_UART_TYPE *uart, uint32_t mask)
- 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_readChar (EF_UART_TYPE *uart, 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_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.
- EF_DRIVER_STATUS EF_UART_charsAvailable (EF_UART_TYPE *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 *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 *uart, uint32_t *parity_mode)

  This function return the parity mode of the UART.
- EF_DRIVER_STATUS EF_UART_busy (EF_UART_TYPE *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 Macro Definition Documentation

5.3.2.1 EF_UART_C

```
#define EF_UART_C
```

5.3.3 Function Documentation

5.3.3.1 EF_UART_busy()

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

This function checks id the UART is busy.

Parameters

in	*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.
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

5.3.3.2 EF_UART_charsAvailable()

```
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.

Parameters

in	*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.
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

5.3.3.3 EF_UART_disable()

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

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

Parameters

in	*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

5.3.3.4 EF_UART_disableGlitchFilter()

```
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

Parameters

in	*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

5.3.3.5 EF_UART_disableLoopBack()

```
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

Parameters

in	*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

5.3.3.6 EF_UART_disableRx()

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

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

Parameters

in	*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

5.3.3.7 EF_UART_disableTx()

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

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

Parameters

in	*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

5.3.3.8 EF_UART_enable()

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

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

Parameters

in	*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

5.3.3.9 EF_UART_enableGlitchFilter()

```
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

Parameters

in	*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

5.3.3.10 EF_UART_enableLoopBack()

```
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

Parameters

in	*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

5.3.3.11 EF_UART_enableRx()

```
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

Parameters

in	*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

5.3.3.12 EF_UART_enableTx()

```
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

Parameters

in	*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

5.3.3.13 EF_UART_getConfig()

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

returns the value of the configuration register

Parameters

in	*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.
out	*CFG_value*	The value of the configuration register

Returns

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

5.3.3.14 EF_UART_getCTRL()

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

returns the value of the control register

Parameters

in	*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.
out	*CTRL_value*	The value of the control register

Returns

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

5.3.3.15 EF_UART_getIM()

```
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

in	*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.
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

5.3.3.16 EF_UART_getMatchData()

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

returns the value of the match data register

Parameters

in	*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.
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

5.3.3.17 EF_UART_getMIS()

```
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

in	*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.
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

5.3.3.18 EF_UART_getParityMode()

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

This function return the parity mode of the UART.

Parameters

in	*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.
out	*parity*	The parity mode of the UART

Returns

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

5.3.3.19 EF_UART_getPrescaler()

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

returns the value of the prescaler

Parameters

in	*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.
out	*Prescaler_value*	The value of the prescaler register

Returns

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

5.3.3.20 EF_UART_getRIS()

```
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

in	*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.
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

5.3.3.21 EF_UART_getRxCount()

```
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)

Parameters

in	*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.
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

5.3.3.22 EF_UART_getRxFIFOThreshold()

```
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

Parameters

in	*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.
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

5.3.3.23 EF_UART_getTxCount()

```
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)

Parameters

in	*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.
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

5.3.3.24 EF_UART_getTxFIFOThreshold()

```
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

Parameters

in	*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.
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

5.3.3.25 EF_UART_readChar()

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

recieve a single character through uart

Parameters

in	*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.
out	*RXDATA_value*	The value of the received character

Returns

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

5.3.3.26 EF_UART_readCharNonBlocking()

```
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.

Parameters

in	*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.
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

5.3.3.27 EF_UART_setConfig()

```
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

in	*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.
in	*config*	The value of the configuration register

Returns

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

5.3.3.28 EF_UART_setCTRL()

```
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

in	*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.
in	*value*	The value of the control register

Returns

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

5.3.3.29 EF_UART_setDataSize()

```
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

Parameters

in	*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.
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

5.3.3.30 EF_UART_setGclkEnable()

```
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

Parameters

in	*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.
in	*value*	The value of the GCLK enable bit

Returns

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

5.3.3.31 EF_UART_setICR()

```
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

in	*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.
in	*mask*	The required mask value

Returns

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

5.3.3.32 EF_UART_setIM()

```
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

in	*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.
in	*mask*	The required mask value

Returns

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

5.3.3.33 EF_UART_setMatchData()

```
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

Parameters

in	*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.
in	*matchData*	The value of the required match data

Returns

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

5.3.3.34 EF_UART_setParityType()

```
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)

Parameters

in	*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.
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

5.3.3.35 EF_UART_setPrescaler()

```
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)

Parameters

in	*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.
in	*prescaler*	The value of the required prescaler

Returns

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

5.3.3.36 EF_UART_setRxFIFOThreshold()

```
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

Parameters

in	*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.
in	*threshold*	The value of the required threshold

Returns

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

5.3.3.37 EF_UART_setTimeoutBits()

```
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

Parameters

in	*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.
in	*value*	timeout bits value

Returns

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

5.3.3.38 EF_UART_setTwoStopBitsSelect()

```
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)

Parameters

in	*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.
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

5.3.3.39 EF_UART_setTxFIFOThreshold()

```
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

Parameters

in	*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.
in	*threshold*	The value of the required threshold

Returns

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

5.3.3.40 EF_UART_spaceAvailable()

```
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.

Parameters

in	*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.
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

5.3.3.41 EF_UART_writeChar()

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

transmit a single character through uart

Parameters

in	*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.
in	*data*	The character or byte required to send

Returns

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

5.3.3.42 EF_UART_writeCharArr()

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

transmit an array of characters through uart

Parameters

in	*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.
in	*char_arr*	An array of characters to send

Returns

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

5.3.3.43 EF_UART_writeCharNonBlocking()

```
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.

Parameters

in	*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.
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)
- #define EF_UART_ERROR_RX_UNAVAILABLE -1
- #define EF_UART_ERROR_TX_UNAVAILABLE 1
- #define EF_UART_SUCCESS 0

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 *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 *uart)

  enables using uart by setting "en" bit in the control register to 1
- 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_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_disableRx (EF_UART_TYPE *uart)

  disables using uart RX by clearing uart "rxen" bit in the control register
- 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_disableTx (EF_UART_TYPE *uart)

  disables using uart TX by clearing uart "txen" bit in the control register
- 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_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_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_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_setCTRL (EF_UART_TYPE *uart, uint32_t value)
- 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_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_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_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_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_setConfig (EF_UART_TYPE *uart, uint32_t config)
- 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_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_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_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_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_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_getMatchData (EF_UART_TYPE *uart, uint32_t *MATCH_value)

  returns the value of the match data register
- 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_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_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_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_getMIS (EF_UART_TYPE *uart, uint32_t *MIS_value)
- EF_DRIVER_STATUS EF_UART_setIM (EF_UART_TYPE *uart, uint32_t mask)
- EF_DRIVER_STATUS EF_UART_getIM (EF_UART_TYPE *uart, uint32_t *IM_value)
- EF_DRIVER_STATUS EF_UART_setICR (EF_UART_TYPE *uart, uint32_t mask)
- 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_writeChar (EF_UART_TYPE *uart, char data)

  transmit a single character through uart
- 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_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.
- 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_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_getParityMode (EF_UART_TYPE *uart, uint32_t *parity_mode)

  This function return the parity mode of the UART.
- EF_DRIVER_STATUS EF_UART_busy (EF_UART_TYPE *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 Macro Definition Documentation

5.4.2.1 EF_UART_CFG_REG_MAX_VALUE

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

5.4.2.2 EF_UART_CFG_REG_TIMEOUT_MAX_VALUE

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

5.4.2.3 EF_UART_CTRL_REG_MAX_VALUE

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

5.4.2.4 EF_UART_DataLength_MAX_VALUE

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

5.4.2.5 EF_UART_DataLength_MIN_VALUE

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

5.4.2.6 EF_UART_ERROR_RX_UNAVAILABLE

```
#define EF_UART_ERROR_RX_UNAVAILABLE -1
```

5.4.2.7 EF_UART_ERROR_TX_UNAVAILABLE

```
#define EF_UART_ERROR_TX_UNAVAILABLE 1
```

5.4.2.8 EF_UART_IC_REG_MAX_VALUE

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

5.4.2.9 EF_UART_IM_REG_MAX_VALUE

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

5.4.2.10 EF_UART_MATCH_REG_MAX_VALUE

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

5.4.2.11 EF_UART_PR_REG_MAX_VALUE

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

5.4.2.12 EF_UART_RX_FIFO_THRESHOLD_REG_MAX_VALUE

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

5.4.2.13 EF_UART_SUCCESS

```
#define EF_UART_SUCCESS 0
```

5.4.2.14 EF_UART_TX_FIFO_THRESHOLD_REG_MAX_VALUE

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

5.4.3 Enumeration Type Documentation

5.4.3.1 parity_type

```
enum parity_type
```

Enumerator

NONE	
ODD	
EVEN	
STICKY_0	
STICKY_1	

5.4.4 Function Documentation

5.4.4.1 EF_UART_busy()

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

This function checks id the UART is busy.

Parameters

in	*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.
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

5.4.4.2 EF_UART_charsAvailable()

```
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.

Parameters

in	*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.
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

5.4.4.3 EF_UART_disable()

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

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

Parameters

in	*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

5.4.4.4 EF_UART_disableGlitchFilter()

```
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

Parameters

in	*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

5.4.4.5 EF_UART_disableLoopBack()

```
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

Parameters

in	*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

5.4.4.6 EF_UART_disableRx()

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

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

Parameters

in	*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

5.4.4.7 EF_UART_disableTx()

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

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

Parameters

in	*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

5.4.4.8 EF_UART_enable()

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

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

Parameters

in	*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

5.4.4.9 EF_UART_enableGlitchFilter()

```
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

Parameters

in	*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

5.4.4.10 EF_UART_enableLoopBack()

```
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

Parameters

in	*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

5.4.4.11 EF_UART_enableRx()

```
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

Parameters

in	*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

5.4.4.12 EF_UART_enableTx()

```
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

Parameters

in	*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

5.4.4.13 EF_UART_getConfig()

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

returns the value of the configuration register

Parameters

in	*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.
out	*CFG_value*	The value of the configuration register

Returns

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

5.4.4.14 EF_UART_getCTRL()

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

returns the value of the control register

Parameters

in	*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.
out	*CTRL_value*	The value of the control register

Returns

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

5.4.4.15 EF_UART_getIM()

```
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

in	*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.
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

5.4.4.16 EF_UART_getMatchData()

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

returns the value of the match data register

Parameters

in	*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.
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

5.4.4.17 EF_UART_getMIS()

```
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

in	*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.
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

5.4.4.18 EF_UART_getParityMode()

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

This function return the parity mode of the UART.

Parameters

in	*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.
out	*parity*	The parity mode of the UART

Returns

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

5.4.4.19 EF_UART_getPrescaler()

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

returns the value of the prescaler

Parameters

in	*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.
out	*Prescaler_value*	The value of the prescaler register

Returns

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

5.4.4.20 EF_UART_getRIS()

```
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

in	*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.
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

5.4.4.21 EF_UART_getRxCount()

```
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)

Parameters

in	*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.
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

5.4.4.22 EF_UART_getRxFIFOThreshold()

```
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

Parameters

in	*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.
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

5.4.4.23 EF_UART_getTxCount()

```
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)

Parameters

in	*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.
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

5.4.4.24 EF_UART_getTxFIFOThreshold()

```
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

Parameters

in	*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.
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

5.4.4.25 EF_UART_readChar()

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

recieve a single character through uart

Parameters

in	*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.
out	*RXDATA_value*	The value of the received character

Returns

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

5.4.4.26 EF_UART_readCharNonBlocking()

```
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.

Parameters

in	*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.
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

5.4.4.27 EF_UART_setConfig()

```
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

in	*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.
in	*config*	The value of the configuration register

Returns

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

5.4.4.28 EF_UART_setCTRL()

```
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

in	*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.
in	*value*	The value of the control register

Returns

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

5.4.4.29 EF_UART_setDataSize()

```
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

Parameters

in	*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.
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

5.4.4.30 EF_UART_setGclkEnable()

```
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

Parameters

in	*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.
in	*value*	The value of the GCLK enable bit

Returns

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

5.4.4.31 EF_UART_setICR()

```
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

in	*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.
in	*mask*	The required mask value

Returns

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

5.4.4.32 EF_UART_setIM()

```
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

in	*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.
in	*mask*	The required mask value

Returns

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

5.4.4.33 EF_UART_setMatchData()

```
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

Parameters

in	*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.
in	*matchData*	The value of the required match data

Returns

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

5.4.4.34 EF_UART_setParityType()

```
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)

Parameters

in	*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.
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

5.4.4.35 EF_UART_setPrescaler()

```
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)

Parameters

in	*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.
in	*prescaler*	The value of the required prescaler

Returns

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

5.4.4.36 EF_UART_setRxFIFOThreshold()

```
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

Parameters

in	*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.
in	*threshold*	The value of the required threshold

Returns

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

5.4.4.37 EF_UART_setTimeoutBits()

```
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

Parameters

in	*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.
in	*value*	timeout bits value

Returns

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

5.4.4.38 EF_UART_setTwoStopBitsSelect()

```
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)

Parameters

in	*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.
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

5.4.4.39 EF_UART_setTxFIFOThreshold()

```
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

Parameters

in	*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.
in	*threshold*	The value of the required threshold

Returns

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

5.4.4.40 EF_UART_spaceAvailable()

```
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.

Parameters

in	*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.
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

5.4.4.41 EF_UART_writeChar()

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

transmit a single character through uart

Parameters

in	*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.
in	*data*	The character or byte required to send

Returns

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

5.4.4.42 EF_UART_writeCharArr()

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

transmit an array of characters through uart

Parameters

in	*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.
in	*char_arr*	An array of characters to send

Returns

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

5.4.4.43 EF_UART_writeCharNonBlocking()

```
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.

Parameters

in	*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.
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.

```
00001 /*
00002     Copyright 2025 Efabless Corp.
00003
00004
00005     Licensed under the Apache License, Version 2.0 (the "License");
00006     you may not use this file except in compliance with the License.
00007     You may obtain a copy of the License at
00008
00009         www.apache.org/licenses/LICENSE-2.0
00010
00011     Unless required by applicable law or agreed to in writing, software
00012     distributed under the License is distributed on an "AS IS" BASIS,
00013     WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
00014     See the License for the specific language governing permissions and
00015     limitations under the License.
00016
00017 */
00018
00019
00026 #ifndef EF_UART_H
00027 #define EF_UART_H
00028
00029
00030 /***************************************************************************
00031 * Includes
00032 ***************************************************************************/
00033 #include "EF_UART_regs.h"
00034 #include "EF_Driver_Common.h"
00035
00036
00037 /***************************************************************************
00038 * Macros and Constants
00039 ***************************************************************************/
```

```
00040 #define EF_UART_CTRL_REG_MAX_VALUE              ((uint32_t)0x0000001F)  // CTRL register only has 5
      bits, and the rest are reserved
00041 #define EF_UART_PR_REG_MAX_VALUE                ((uint32_t)0x0000FFFF)  // PR register only has 16
      bits
00042 #define EF_UART_DataLength_MIN_VALUE            ((uint32_t)0x00000005)  // This UART IP only supports
      data length from 5 to 9 bits
00043 #define EF_UART_DataLength_MAX_VALUE            ((uint32_t)0x00000009)  // This UART IP only supports
      data length from 5 to 9 bits
00044 #define EF_UART_CFG_REG_TIMEOUT_MAX_VALUE       ((uint32_t)0x0000003F)  // The CFG register timeout
      field is 6 bits
00045 #define EF_UART_CFG_REG_MAX_VALUE               ((uint32_t)0x00001FFF)  // The CFG register is 13 bits
00046 #define EF_UART_RX_FIFO_THRESHOLD_REG_MAX_VALUE ((uint32_t)0x0000000F)  // The RX FIFO level register
      is 4 bits
00047 #define EF_UART_TX_FIFO_THRESHOLD_REG_MAX_VALUE ((uint32_t)0x0000000F)  // The TX FIFO level register
      is 4 bits
00048 #define EF_UART_MATCH_REG_MAX_VALUE             ((uint32_t)0x00001FFF)  // The match register is 9
      bits
00049 #define EF_UART_IM_REG_MAX_VALUE                ((uint32_t)0x000003FF)  // The IM register is 10 bits
00050 #define EF_UART_IC_REG_MAX_VALUE                ((uint32_t)0x000003FF)  // The IC register is 10 bits
00051
00052
00053 /*******************************************************************************
00054 * Typedefs and Enums
00055 *******************************************************************************/
00056
00057 enum parity_type {NONE = 0, ODD = 1, EVEN = 2, STICKY_0 = 4, STICKY_1 = 5};
00058
00059
00060
00061 /*******************************************************************************
00062 * Function Prototypes
00063 *******************************************************************************/
00064
00066
00073 EF_DRIVER_STATUS EF_UART_setGclkEnable (EF_UART_TYPE* uart, uint32_t value);
00074
00076
00081 EF_DRIVER_STATUS EF_UART_enable(EF_UART_TYPE* uart);
00082
00083
00085
00090 EF_DRIVER_STATUS EF_UART_disable(EF_UART_TYPE* uart);
00091
00092
00094
00099 EF_DRIVER_STATUS EF_UART_enableRx(EF_UART_TYPE* uart);
00100
00101
00103
00108 EF_DRIVER_STATUS EF_UART_disableRx(EF_UART_TYPE* uart);
00109
00110
00112
00117 EF_DRIVER_STATUS EF_UART_enableTx(EF_UART_TYPE* uart);
00118
00119
00121
00126 EF_DRIVER_STATUS EF_UART_disableTx(EF_UART_TYPE* uart);
00127
00128
00130
00135 EF_DRIVER_STATUS EF_UART_enableLoopBack(EF_UART_TYPE* uart);
00136
00137
00139
00144 EF_DRIVER_STATUS EF_UART_disableLoopBack(EF_UART_TYPE* uart);
00145
00146
00148
00153 EF_DRIVER_STATUS EF_UART_enableGlitchFilter(EF_UART_TYPE* uart);
00154
00155
00157
00162 EF_DRIVER_STATUS EF_UART_disableGlitchFilter(EF_UART_TYPE* uart);
00163
00164
00171
00177 EF_DRIVER_STATUS EF_UART_setCTRL(EF_UART_TYPE* uart, uint32_t value);
00178
00179
00181
00187 EF_DRIVER_STATUS EF_UART_getCTRL(EF_UART_TYPE* uart, uint32_t* CTRL_value);
00188
00189
00191
00197 EF_DRIVER_STATUS EF_UART_setDataSize(EF_UART_TYPE* uart, uint32_t value);
00198
```

```
00199
00201
00207 EF_DRIVER_STATUS EF_UART_setTwoStopBitsSelect(EF_UART_TYPE* uart, bool is_two_bits);
00208
00209
00211
00217 EF_DRIVER_STATUS EF_UART_setParityType(EF_UART_TYPE* uart, enum parity_type parity);
00218
00219
00221
00227 EF_DRIVER_STATUS EF_UART_setTimeoutBits(EF_UART_TYPE* uart, uint32_t value);
00228
00229
00235
00241 EF_DRIVER_STATUS EF_UART_setConfig(EF_UART_TYPE* uart, uint32_t config);
00242
00243
00245
00251 EF_DRIVER_STATUS EF_UART_getConfig(EF_UART_TYPE* uart, uint32_t* CFG_value);
00252
00253
00255
00261 EF_DRIVER_STATUS EF_UART_setRxFIFOThreshold(EF_UART_TYPE* uart, uint32_t threshold);
00262
00263
00265
00271 EF_DRIVER_STATUS EF_UART_getRxFIFOThreshold(EF_UART_TYPE* uart, uint32_t* RX_FIFO_THRESHOLD_value);
00272
00273
00275
00281 EF_DRIVER_STATUS EF_UART_setTxFIFOThreshold(EF_UART_TYPE* uart, uint32_t threshold);
00282
00283
00285
00291 EF_DRIVER_STATUS EF_UART_getTxFIFOThreshold(EF_UART_TYPE* uart, uint32_t* TX_FIFO_THRESHOLD_value);
00292
00293
00294
00296
00302 EF_DRIVER_STATUS EF_UART_setMatchData(EF_UART_TYPE* uart, uint32_t matchData);
00303
00304
00306
00312 EF_DRIVER_STATUS EF_UART_getMatchData(EF_UART_TYPE* uart, uint32_t* MATCH_value);
00313
00314
00316
00322 EF_DRIVER_STATUS EF_UART_getTxCount(EF_UART_TYPE* uart, uint32_t* TX_FIFO_LEVEL_value);
00323
00324
00326
00332 EF_DRIVER_STATUS EF_UART_getRxCount(EF_UART_TYPE* uart, uint32_t* RX_FIFO_LEVEL_value);
00333
00334
00336
00342 EF_DRIVER_STATUS EF_UART_setPrescaler(EF_UART_TYPE* uart, uint32_t prescaler);
00343
00344
00346
00352 EF_DRIVER_STATUS EF_UART_getPrescaler(EF_UART_TYPE* uart, uint32_t* Prescaler_value);
00353
00354
00366
00372 EF_DRIVER_STATUS EF_UART_getRIS(EF_UART_TYPE* uart, uint32_t* RIS_value);
00373
00374
00386
00392 EF_DRIVER_STATUS EF_UART_getMIS(EF_UART_TYPE* uart, uint32_t* MIS_value);
00393
00394
00406
00412 EF_DRIVER_STATUS EF_UART_setIM(EF_UART_TYPE* uart, uint32_t mask);
00413
00414
00426
00432 EF_DRIVER_STATUS EF_UART_getIM(EF_UART_TYPE* uart, uint32_t* IM_value);
00433
00434
00446
00452 EF_DRIVER_STATUS EF_UART_setICR(EF_UART_TYPE* uart, uint32_t mask);
00453
00454
00456
00463 EF_DRIVER_STATUS EF_UART_writeCharArr(EF_UART_TYPE* uart, const char *char_arr);
00464
00465
00467
```

```
00473 EF_DRIVER_STATUS EF_UART_writeChar(EF_UART_TYPE* uart, char data);
00474
00475
00477
00483 EF_DRIVER_STATUS EF_UART_readChar(EF_UART_TYPE* uar, char* RXDATA_value);
00484
00485
00486
00487 // The following functions are not verified yet
00488
      /*****************************************************************************************************************
00489
      /*****************************************************************************************************************
00490
00491 #define EF_UART_ERROR_RX_UNAVAILABLE -1
00492 #define EF_UART_ERROR_TX_UNAVAILABLE 1
00493 #define EF_UART_SUCCESS 0
00494
00495
00497
00504 EF_DRIVER_STATUS EF_UART_readCharNonBlocking(EF_UART_TYPE* uart, char* RXDATA_value, bool*
      data_available);
00505
00507
00514 EF_DRIVER_STATUS EF_UART_writeCharNonBlocking(EF_UART_TYPE* uart, char data, bool* data_sent);
00515
00517
00523 EF_DRIVER_STATUS EF_UART_charsAvailable(EF_UART_TYPE* uart, bool* flag);
00524
00525
00527
00533 EF_DRIVER_STATUS EF_UART_spaceAvailable(EF_UART_TYPE* uart, bool* flag);
00534
00536
00542 EF_DRIVER_STATUS EF_UART_getParityMode(EF_UART_TYPE* uart, uint32_t* parity_mode);
00543
00545
00551 EF_DRIVER_STATUS EF_UART_busy(EF_UART_TYPE* uart, bool* flag);
00552
00553
00554
00555 /*****************************************************************************
00556 * External Variables
00557 *****************************************************************************/
00558
00559
00560 #endif // EF_UART_H
00561
00562 /*****************************************************************************
00563 * End of File
00564 *****************************************************************************/
```

5.6 EF_UART_regs.h File Reference

Classes

- struct _EF_UART_TYPE_

Macros

- #define IO_TYPES
- #define __R volatile const unsigned int
- #define __W volatile unsigned int
- #define __RW volatile unsigned int
- #define EF_UART_CTRL_REG_EN_BIT 0
- #define EF_UART_CTRL_REG_EN_MASK 0x1
- #define EF_UART_CTRL_REG_TXEN_BIT 1
- #define EF_UART_CTRL_REG_TXEN_MASK 0x2
- #define EF_UART_CTRL_REG_RXEN_BIT 2
- #define EF_UART_CTRL_REG_RXEN_MASK 0x4
- #define EF_UART_CTRL_REG_LPEN_BIT 3
- #define EF_UART_CTRL_REG_LPEN_MASK 0x8
- #define EF_UART_CTRL_REG_GFEN_BIT 4
- #define EF_UART_CTRL_REG_GFEN_MASK 0x10

- #define EF_UART_CFG_REG_WLEN_BIT 0
- #define EF_UART_CFG_REG_WLEN_MASK 0xf
- #define EF_UART_CFG_REG_STP2_BIT 4
- #define EF_UART_CFG_REG_STP2_MASK 0x10
- #define EF_UART_CFG_REG_PARITY_BIT 5
- #define EF_UART_CFG_REG_PARITY_MASK 0xe0
- #define EF_UART_CFG_REG_TIMEOUT_BIT 8
- #define EF_UART_CFG_REG_TIMEOUT_MASK 0x3f00
- #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_RX_FIFO_FLUSH_REG_FLUSH_BIT 0
- #define EF_UART_RX_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 EF_UART_TX_FIFO_FLUSH_REG_FLUSH_BIT 0
- #define EF_UART_TX_FIFO_FLUSH_REG_FLUSH_MASK 0x1
- #define EF_UART_TXE_FLAG 0x1
- #define EF_UART_RXF_FLAG 0x2
- #define EF_UART_TXB_FLAG 0x4
- #define EF_UART_RXA_FLAG 0x8
- #define EF_UART_BRK_FLAG 0x10
- #define EF_UART_MATCH_FLAG 0x20
- #define EF_UART_FE_FLAG 0x40
- #define EF_UART_PRE_FLAG 0x80
- #define EF_UART_OR_FLAG 0x100
- #define EF_UART_RTO_FLAG 0x200

Typedefs

- typedef struct _EF_UART_TYPE_ EF_UART_TYPE

5.6.1 Macro Definition Documentation

5.6.1.1 __R

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

5.6.1.2 __RW

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

5.6.1.3 __W

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

5.6.1.4 EF_UART_BRK_FLAG

```
#define EF_UART_BRK_FLAG 0x10
```

5.6.1.5 EF_UART_CFG_REG_PARITY_BIT

```
#define EF_UART_CFG_REG_PARITY_BIT 5
```

5.6.1.6 EF_UART_CFG_REG_PARITY_MASK

```
#define EF_UART_CFG_REG_PARITY_MASK 0xe0
```

5.6.1.7 EF_UART_CFG_REG_STP2_BIT

```
#define EF_UART_CFG_REG_STP2_BIT 4
```

5.6.1.8 EF_UART_CFG_REG_STP2_MASK

```
#define EF_UART_CFG_REG_STP2_MASK 0x10
```

5.6.1.9 EF_UART_CFG_REG_TIMEOUT_BIT

```
#define EF_UART_CFG_REG_TIMEOUT_BIT 8
```

5.6.1.10 EF_UART_CFG_REG_TIMEOUT_MASK

```
#define EF_UART_CFG_REG_TIMEOUT_MASK 0x3f00
```

5.6.1.11 EF_UART_CFG_REG_WLEN_BIT

```
#define EF_UART_CFG_REG_WLEN_BIT 0
```

5.6.1.12 EF_UART_CFG_REG_WLEN_MASK

```
#define EF_UART_CFG_REG_WLEN_MASK 0xf
```

5.6.1.13 EF_UART_CTRL_REG_EN_BIT

```
#define EF_UART_CTRL_REG_EN_BIT 0
```

5.6.1.14 EF_UART_CTRL_REG_EN_MASK

```
#define EF_UART_CTRL_REG_EN_MASK 0x1
```

5.6.1.15 EF_UART_CTRL_REG_GFEN_BIT

```
#define EF_UART_CTRL_REG_GFEN_BIT 4
```

5.6.1.16 EF_UART_CTRL_REG_GFEN_MASK

```
#define EF_UART_CTRL_REG_GFEN_MASK 0x10
```

5.6.1.17 EF_UART_CTRL_REG_LPEN_BIT

```
#define EF_UART_CTRL_REG_LPEN_BIT 3
```

5.6.1.18 EF_UART_CTRL_REG_LPEN_MASK

```
#define EF_UART_CTRL_REG_LPEN_MASK 0x8
```

5.6.1.19 EF_UART_CTRL_REG_RXEN_BIT

```
#define EF_UART_CTRL_REG_RXEN_BIT 2
```

5.6.1.20 EF_UART_CTRL_REG_RXEN_MASK

```
#define EF_UART_CTRL_REG_RXEN_MASK 0x4
```

5.6.1.21 EF_UART_CTRL_REG_TXEN_BIT

```
#define EF_UART_CTRL_REG_TXEN_BIT 1
```

5.6.1.22 EF_UART_CTRL_REG_TXEN_MASK

```
#define EF_UART_CTRL_REG_TXEN_MASK 0x2
```

5.6.1.23 EF_UART_FE_FLAG

```
#define EF_UART_FE_FLAG 0x40
```

5.6.1.24 EF_UART_MATCH_FLAG

```
#define EF_UART_MATCH_FLAG 0x20
```

5.6.1.25 EF_UART_OR_FLAG

```
#define EF_UART_OR_FLAG 0x100
```

5.6.1.26 EF_UART_PRE_FLAG

```
#define EF_UART_PRE_FLAG 0x80
```

5.6.1.27 EF_UART_RTO_FLAG

```
#define EF_UART_RTO_FLAG 0x200
```

5.6.1.28 EF_UART_RX_FIFO_FLUSH_REG_FLUSH_BIT

```
#define EF_UART_RX_FIFO_FLUSH_REG_FLUSH_BIT 0
```

5.6.1.29 EF_UART_RX_FIFO_FLUSH_REG_FLUSH_MASK

```
#define EF_UART_RX_FIFO_FLUSH_REG_FLUSH_MASK 0x1
```

5.6.1.30 EF_UART_RX_FIFO_LEVEL_REG_LEVEL_BIT

```
#define EF_UART_RX_FIFO_LEVEL_REG_LEVEL_BIT 0
```

5.6.1.31 EF_UART_RX_FIFO_LEVEL_REG_LEVEL_MASK

```
#define EF_UART_RX_FIFO_LEVEL_REG_LEVEL_MASK 0xf
```

5.6.1.32 EF_UART_RX_FIFO_THRESHOLD_REG_THRESHOLD_BIT

```
#define EF_UART_RX_FIFO_THRESHOLD_REG_THRESHOLD_BIT 0
```

5.6.1.33 EF_UART_RX_FIFO_THRESHOLD_REG_THRESHOLD_MASK

```
#define EF_UART_RX_FIFO_THRESHOLD_REG_THRESHOLD_MASK 0xf
```

5.6.1.34 EF_UART_RXA_FLAG

```
#define EF_UART_RXA_FLAG 0x8
```

5.6.1.35 EF_UART_RXF_FLAG

```
#define EF_UART_RXF_FLAG 0x2
```

5.6.1.36 EF_UART_TX_FIFO_FLUSH_REG_FLUSH_BIT

```
#define EF_UART_TX_FIFO_FLUSH_REG_FLUSH_BIT 0
```

5.6.1.37 EF_UART_TX_FIFO_FLUSH_REG_FLUSH_MASK

```
#define EF_UART_TX_FIFO_FLUSH_REG_FLUSH_MASK 0x1
```

5.6.1.38 EF_UART_TX_FIFO_LEVEL_REG_LEVEL_BIT

```
#define EF_UART_TX_FIFO_LEVEL_REG_LEVEL_BIT 0
```

5.6.1.39 EF_UART_TX_FIFO_LEVEL_REG_LEVEL_MASK

```
#define EF_UART_TX_FIFO_LEVEL_REG_LEVEL_MASK 0xf
```

5.6.1.40 EF_UART_TX_FIFO_THRESHOLD_REG_THRESHOLD_BIT

```
#define EF_UART_TX_FIFO_THRESHOLD_REG_THRESHOLD_BIT 0
```

5.6.1.41 EF_UART_TX_FIFO_THRESHOLD_REG_THRESHOLD_MASK

```
#define EF_UART_TX_FIFO_THRESHOLD_REG_THRESHOLD_MASK 0xf
```

5.6.1.42 EF_UART_TXB_FLAG

```
#define EF_UART_TXB_FLAG 0x4
```

5.6.1.43 EF_UART_TXE_FLAG

```
#define EF_UART_TXE_FLAG 0x1
```

5.6.1.44 IO_TYPES

```
#define IO_TYPES
```

5.6.2 Typedef Documentation

5.6.2.1 EF_UART_TYPE

```
typedef struct _EF_UART_TYPE_ EF_UART_TYPE
```

5.7 EF_UART_regs.h

Go to the documentation of this file.

```
00001 /*
00002     Copyright 2024 Efabless Corp.
00003
00004     Author: Mohamed Shalan (mshalan@efabless.com)
00005
00006     Licensed under the Apache License, Version 2.0 (the "License");
00007     you may not use this file except in compliance with the License.
00008     You may obtain a copy of the License at
00009
00010         http://www.apache.org/licenses/LICENSE-2.0
00011
00012     Unless required by applicable law or agreed to in writing, software
00013     distributed under the License is distributed on an "AS IS" BASIS,
00014     WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
00015     See the License for the specific language governing permissions and
00016     limitations under the License.
00017
00018 */
00019
00020 #ifndef EF_UARTREGS_H
00021 #define EF_UARTREGS_H
00022
00023 #ifndef IO_TYPES
00024 #define IO_TYPES
00025 #define   __R     volatile const unsigned int
00026 #define   __W     volatile       unsigned int
00027 #define   __RW    volatile       unsigned int
00028 #endif
00029
00030 #define EF_UART_CTRL_REG_EN_BIT 0
00031 #define EF_UART_CTRL_REG_EN_MASK    0x1
00032 #define EF_UART_CTRL_REG_TXEN_BIT   1
00033 #define EF_UART_CTRL_REG_TXEN_MASK  0x2
00034 #define EF_UART_CTRL_REG_RXEN_BIT   2
00035 #define EF_UART_CTRL_REG_RXEN_MASK  0x4
00036 #define EF_UART_CTRL_REG_LPEN_BIT   3
00037 #define EF_UART_CTRL_REG_LPEN_MASK  0x8
00038 #define EF_UART_CTRL_REG_GFEN_BIT   4
00039 #define EF_UART_CTRL_REG_GFEN_MASK  0x10
00040 #define EF_UART_CFG_REG_WLEN_BIT    0
00041 #define EF_UART_CFG_REG_WLEN_MASK   0xf
00042 #define EF_UART_CFG_REG_STP2_BIT    4
00043 #define EF_UART_CFG_REG_STP2_MASK   0x10
```

```
00044 #define EF_UART_CFG_REG_PARITY_BIT  5
00045 #define EF_UART_CFG_REG_PARITY_MASK 0xe0
00046 #define EF_UART_CFG_REG_TIMEOUT_BIT 8
00047 #define EF_UART_CFG_REG_TIMEOUT_MASK    0x3f00
00048 #define EF_UART_RX_FIFO_LEVEL_REG_LEVEL_BIT 0
00049 #define EF_UART_RX_FIFO_LEVEL_REG_LEVEL_MASK    0xf
00050 #define EF_UART_RX_FIFO_THRESHOLD_REG_THRESHOLD_BIT 0
00051 #define EF_UART_RX_FIFO_THRESHOLD_REG_THRESHOLD_MASK    0xf
00052 #define EF_UART_RX_FIFO_FLUSH_REG_FLUSH_BIT 0
00053 #define EF_UART_RX_FIFO_FLUSH_REG_FLUSH_MASK    0x1
00054 #define EF_UART_TX_FIFO_LEVEL_REG_LEVEL_BIT 0
00055 #define EF_UART_TX_FIFO_LEVEL_REG_LEVEL_MASK    0xf
00056 #define EF_UART_TX_FIFO_THRESHOLD_REG_THRESHOLD_BIT 0
00057 #define EF_UART_TX_FIFO_THRESHOLD_REG_THRESHOLD_MASK    0xf
00058 #define EF_UART_TX_FIFO_FLUSH_REG_FLUSH_BIT 0
00059 #define EF_UART_TX_FIFO_FLUSH_REG_FLUSH_MASK    0x1
00060
00061 #define EF_UART_TXE_FLAG    0x1
00062 #define EF_UART_RXF_FLAG    0x2
00063 #define EF_UART_TXB_FLAG    0x4
00064 #define EF_UART_RXA_FLAG    0x8
00065 #define EF_UART_BRK_FLAG    0x10
00066 #define EF_UART_MATCH_FLAG  0x20
00067 #define EF_UART_FE_FLAG 0x40
00068 #define EF_UART_PRE_FLAG    0x80
00069 #define EF_UART_OR_FLAG 0x100
00070 #define EF_UART_RTO_FLAG    0x200
00071
00072 typedef struct _EF_UART_TYPE_ {
00073     __R  RXDATA;
00074     __W  TXDATA;
00075     __W  PR;
00076     __W  CTRL;
00077     __W  CFG;
00078     __R  reserved_0[2];
00079     __W  MATCH;
00080     __R  reserved_1[16248];
00081     __R  RX_FIFO_LEVEL;
00082     __W  RX_FIFO_THRESHOLD;
00083     __W  RX_FIFO_FLUSH;
00084     __R  reserved_2[1];
00085     __R  TX_FIFO_LEVEL;
00086     __W  TX_FIFO_THRESHOLD;
00087     __W  TX_FIFO_FLUSH;
00088     __R  reserved_3[57];
00089     __RW    IM;
00090     __R  MIS;
00091     __R  RIS;
00092     __W  IC;
00093     __W  GCLK;
00094 } EF_UART_TYPE;
00095
00096 #endif
00097
```

5.8 README.md File Reference

Index