The following document contains information on Cypress products. The document has the series name, product name, and ordering part numbering with the prefix “MB”. However, Cypress will offer these products to new and existing customers with the series name, product name, and ordering part number with the prefix “CY”.

How to Check the Ordering Part Number

1. Go to www.cypress.com/pcn.
2. Enter the keyword (for example, ordering part number) in the **SEARCH PCNS** field and click **Apply**.
3. Click the corresponding title from the search results.
4. Download the Affected Parts List file, which has details of all changes

For More Information

Please contact your local sales office for additional information about Cypress products and solutions.

About Cypress

Cypress is the leader in advanced embedded system solutions for the world's most innovative automotive, industrial, smart home appliances, consumer electronics and medical products. Cypress' microcontrollers, analog ICs, wireless and USB-based connectivity solutions and reliable, high-performance memories help engineers design differentiated products and get them to market first. Cypress is committed to providing customers with the best support and development resources on the planet enabling them to disrupt markets by creating new product categories in record time. To learn more, go to www.cypress.com.

AN205255

FR, MB91460, USART

This application note describes the functionality of the USART and their operation modes and gives some examples.

Contents

1 Introduction 1
1.1 Key Features 1
2 The USART with LIN functionality 1
2.1 Block Diagram 2
2.2 Basic Functionality 3
2.3 Registers 4
2.4 Error Conditions 12
2.5 Interface to the BUS 15
3 RX FIFO Behaviour 17
4 USART Examples 18
4.1 Asynchronous Mode (0) without Interrupts 18
4.2 Asynchronous Mode (0) with Interrupts 19
5 Example using RX FIFO 21
6 Example using TX FIFO 23
7 Appendix A 25
7.1 Related Documents 25
8 Additional Information 26
Document History 27

1. Introduction

This application note describes the functionality of the USART and their operation modes and gives some examples.

1.1 Key Features

- Full Duplex
- NRZ/RZ for serial Data Input and Output
- NRZ/RZ for serial Clock Input and Output
- Asynchronous and synchronous Mode
- 7-8 Data Bits and 1-2 Stop Bits for asynchronous mode, even or odd Parity selectable
- Dedicated Baud Rate Generator which consists of 16-bit Reload Counter (with a configurable baud rate of 610 Bits/s up to 4 MBits/s synchronous at 20 MHz Peripheral Clock)
- Baud rate generator can be fed with external clock
- Synchronous master or slave capable
- 4 SPI Clock Modes
- Framing, Overrun, and Parity Error detectable
- LIN Synch Break Detection and Generation (13, 14, 15, 16 Bit Times selectable)
- LIN Synch Field Signal can be fed to Input Capture Unit for Time Measurement
- Start and Stop Bits selectable in synchronous Mode
- Continuous Serial Clock Output selectable in synchronous Mode
- Asynchronous Master-Slave Communication (Address and Data Bit selectable)
- 16 byte FIFO available on USART 4 to 15

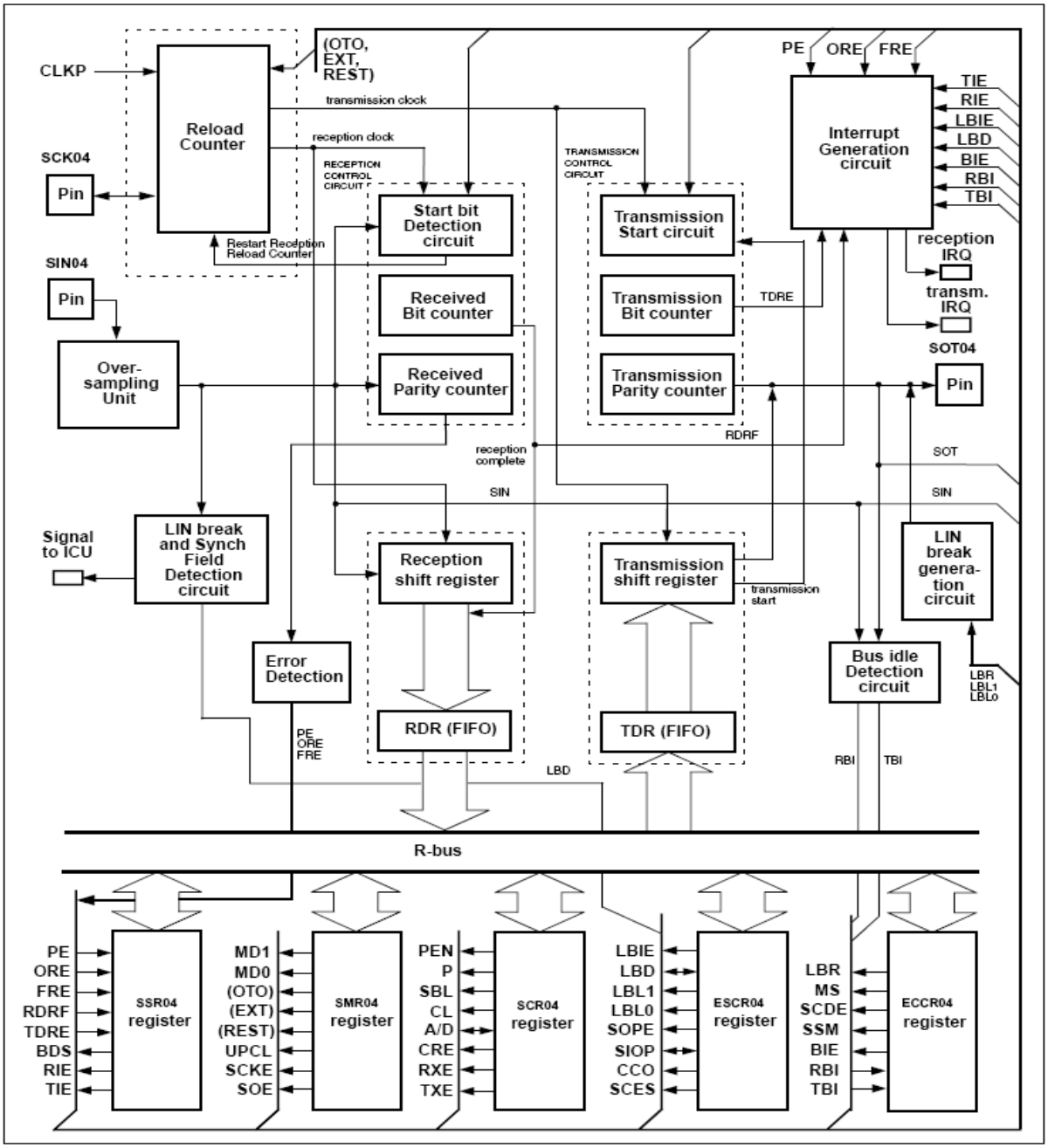
2. The USART with LIN functionality

The basic functionality of the USART with LIN functionality

2.1 Block Diagram

Figure 1 shows the internal block diagram of USART04 exemplarily. The other USARTs can be regarded the same respectively..

Figure 1. USART Block Diagram



2.2 Basic Functionality

The USART has four different operation modes, which are selectable via the MD[1:0] bits in the 2.3.2 Serial Mode Register (SMR)

Table 1. USART Operation Modes

MD1	MD0	Mode	Description
0	0	0	Asynchronous (normal mode)
0	1	1	Asynchronous (Master/Slave Mode)
1	0	2	Synchronous Mode
1	1	3	Asynchronous (LIN Mode)

Please change the mode only when USART reception and transmission is off (SCR:RXE = 0, SCR:TXE = 0). Furthermore it is recommended to reset the USART with the SMR:UPCL bit, after operation mode has changed.

Example:

```
RXE = 1
TXE = 1
```

2.3 Registers

Please note that any changes in the settings of the USART should be done while reception as well as transmission is disabled (RXE = 0, TXE = 0). Otherwise the result of ongoing reception / transmission might be incorrect and the USART might not be initialized correctly.

2.3.1 Serial Control Register (SCR)

Table 2. SCR

Bit No.	Name	Explanation	Value	Operation
7	PEN	Parity Enable*1	0	Parity disabled
			1	Parity enabled
6	P	Parity Even/Odd Selection*1	0	Even Parity enabled
			1	Odd Parity enabled
5	SBL	Stop Bit Length*2	0	1 Stop bit selected
			1	2 Stop bits selected
4	CL	Character Length*3	0	7 Bits
			1	8 Bits
3	AD	Address/Data Bit*4	0	Data Bit
			1	Address Bit
2	CRE	Clear Reception Errors	0	Write: No effect
			1	Write: Errors cleared, Reception is reset
1	RXE	Reception Enable	0	Reception disabled
			1	Reception enabled
0	TXE	Transmission Enable	0	Transmission disabled
			1	Transmission enabled

*1 This function is only available in Mode 0 and 2 if SSM = 1.

*2 This function is only available in Mode 0, 1 and 2 if SSM = 1.

*3 This function is only available in Mode 0 and 1.

*4 This function is only available in Mode 1.

2.3.2 Serial Mode Register (SMR)

Table 3. SMR

Bit No.	Name	Explanation	Value	Operation
7,6	MD1,MD0	Mode Bits	0, 0	Asynchronous Normal Mode
			0, 1	Asynchronous Multiprocessor Mode
			1, 0	Synchronous Mode
			1, 1	Asynchronous LIN-Mode
5	OTO	One-to-One External Clock*5	0	Use external Clock with Baud Rate Generator
			1	Use external Clock as is
4	EXT	External Clock Source	0	Use internal Clock with Baud Rate Generator (Reload Counter)
			1	Use external Clock Source
3	REST	Restart Baud Rate Generator*6	0	Write: No effect
			1	Restart Baud Rate Generator
2	UPCL	USART Programmable Clear*6*7	0	Write: No effect
			1	Write: Reset USART
1	SCKE	Serial Clock Output Enable	0	SCK Pin: Port Function
			1	SCK Pin: Clock Output
0	SOE	Serial Output Enable	0	SOT Pin: Port Function
			1	SOT Pin: Data Output

*5 This function is used, if USART should act as Serial Synchronous Slave Device (Mode 2).

*6 These bits are auto-cleared when set

*7 Reset the USART, only if reception and transmission is disabled (RXE = 0, TXE = 0).

2.3.3 Serial Status register (SSR)

Table 4. SSR

Bit No.	Name	Explanation	Value	Operation
7	PE	Parity Error*8	0	No Parity Error
			1	Parity Error Detected
6	ORE	Overrun Error	0	No Overrun Error
			1	Overrun Error occurred (New Reception, if RDRF = 1)
5	FRE	Framing Error*9	0	No Framing Error
			1	Framing Error Detected (No Stop Bit received)
4	RDRF	Reception Data Register Full	0	No Reception
			1	Reception Data Register is full
3	TDRE	Transmission Data Register Empty	0	Transmission Data Register is full
			1	Transmission Data Register is empty (Ready for Transmission)
2	BDS	Bit Direction*10	0	LSB first
			1	MSB first
1	RIE	Reception Interrupt Enable	0	Interrupt disabled
			1	Interrupt enabled
0	TIE	Transmission Interrupt Enable	0	Interrupt disabled
			1	Interrupt enabled

*8 This flag is only available in Mode 0, 1, and 2 if SSM = 1.

*9 This flag is only available in Mode 0, 1, and 3

*10 This function is not available in Mode 3 (LIN).

2.3.4 Reception Data Register (RDR)

This 8-Bit Register contains the received data. This is indicated by the RDRF flag of the Serial Status Register (SSR). The RDRF flag also generates the reception interrupt, if enabled (RIE = 1).

2.3.5 Transmission Data Register (TDR)

The 8-Bit Register contains the data to be sent. If it is empty (default) the TDRE flag is “1”. Once the data is written to TDR the TDRE flag is cleared (to “0”). Then the data gets shifted to the transmission shift register. Subsequently on the next transmission clock the start bit is transmitted on the bus. After the start bit is transmitted on the subsequent transmission clock the TDRE bit is set to “1” again.

The TDRE flag also generates the transmission interrupt, if enabled (TIE = 1). Note, that it is “1” after Reset.

It should be noted that both TDR and RDR share the same address.

2.3.6 Extended Status/Control Register (ESCR)

Table 5. ESCR

Bit No.	Name	Explanation	Value	Operation
7	LBIE	LIN Break Detection Interrupt enable*11	0	Interrupt disabled
			1	Interrupt enabled
6	LBD	LIN Break Detection Flag/Clear*11	0	Write: Clear LIN Break Detection Interrupt
			1	LIN Break detected
5, 4	LBL1, LBL0	LIN Break Generation Length*11	0, 0	13 Bit Times
			0, 1	14 Bit Times
			1, 0	15 Bit Times
			1, 1	16 Bit Times
3	SOPE	Serial Output Pin Enable	0	SOT Pin is Serial Out
			1	SOT Pin is state of SIOP
2	SIOP	Serial Input/Output Pin	0	Write: Force SOT to “0”, if SOPE=1, Read: State of SIN
			1	Write: Force SOT to “1”, if SOPE=1, Read: State of SIN
1	CCO	Continuous Clock Output*12	0	Continuous Clock disabled
			1	Continuous Clock enabled
0	SCES	Serial Clock Edge Selection (inverted CPOL) *13	0	Sampling on rising clock edge / Normal Clock
			1	Sampling on falling clock edge / Inverted Clock

*11.This bit is only available in Mode 3 (LIN).

*12 This bit is only available in Mode 2. Only reasonable with SSM=1.

*13 This bit is only available in Mode 2.

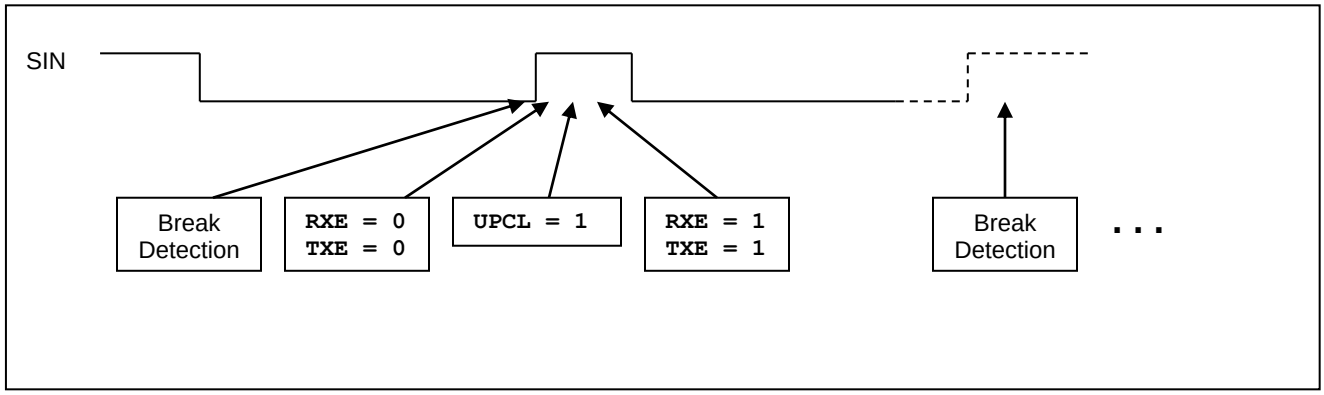
2.3.6.1 LIN Break Detection Consideration

There may be an application that requires LIN Sync Break Detection without Sync Field Detection. In such case, after the LIN break has been detected perform the following steps:

- After detecting the Sync Break, disable the transmission and reception by clearing `TXE` and `RXE` bits of `SCR` register respectively.
- Reset the LIN-USART by setting the UPCL bit of SMR register
- And then re-enable the transmission and reception again by setting TXE and RXE bits of SCR register respectively.

This ensures the correct detection of a possible following break frame.

Figure 2 . LIN Break Detection



2.3.7 Extended Communication Control register (ECCR)

Table 6. ECCR

Bit No.	Name	Explanation	Value	Operation
7	INV	Invert Serial Data*14	0	Serial Data is not inverted (NRZ format)
			1	Serial Data is inverted (RZ format)
6	LBR	Generate LIN Break*15	0	Ignored
			1	Write: Generate LIN Break
5	MS	Synchronous Master/Slave Select*16	0	Synchronous Master Mode
			1	Synchronous Slave Mode
4	SCDE	Synchronous Serial Clock Delay (inverted CPHA) *16	0	No Clock Delay
			1	Clock Delay of half Bit Time
3	SSM	Synchronous Start/Stop Bit Mode*16	0	No start/stop bits in synchronous mode 2
			1	Enable start/stop bits in synchronous mode 2
2	BIE	Bus idle interrupt enable*17	0	Disable bus idle interrupt
			1	Enable bus idle interrupt
1	RBI	Reception Bus Idle*17	0	Activity on SIN, Reception ongoing
			1	Bus Idle, no Reception ongoing
0	TBI	Transmission Bus Idle*17	0	Activity on SOT, Transmission ongoing
			1	Bus Idle, no Transmission

*14 Please note, that setting INV bit causes 0-level at SOT for the first transfer at serial synchronous slave mode, even if TDR was not written to.

*15.This bit is only available in Mode 3 (LIN).

*16.This bit is only available in Mode 2.

*17.This bit cannot be used in synchronous slave Mode 2 (when MS=1).

2.3.8 Baud Rate Generator Register (BGR)

This 16-Bit register contains the divider for the baud rate. The value “v” for the divider can be calculated as:

v = [Φ / b] - 1,

Where, Φ is the peripheral clock CLKP,

b the baud rate

and [ ] gaussian brackets (mathematical rounding function).

2.3.8.1 Minimum and Maximum Ratings

For synchronous operation the minimum divider is 5 (to ensure correct internal signal processing).

Because the USART has an internal 5-times over-sampling unit in mode 0, 1 and 3, it is recommended to use also a divider not less than 5 for asynchronous communication.

The maximum divider in all operation modes is 32768 (v = 32767, due to 15 bit reload counter). If this division factor is insufficient, the Peripheral Clock has to be reduced then.

Table 7. Baud Rate corresponding to CLKP1

Peripheral Clock CLKP1	Minimum Baud Rate (div = 32768)	Maximum Baud Rate (div = 5)
16 MHz	488 Bits/s	3.2 MBits/s
20 MHz	610 Bits/s	4 MBits/s
24 MHz	732 Bits/s	4.8 MBits/s
25 MHz	763 Bits/s	5 MBits/s
48 MHz	1465 Bits/s	9.6 MBits/s

For the relationship between the baud rate and reload values of baud rate generator at different peripheral clock CLKP frequencies please refer the hardware manual.

It should also be noted that there would be a deviation between the desired baud rate and the actual baud rate depending upon the CLKP clock frequency and reload value. This deviation would be further affected if the CLKP is fed from the CLKMOD (clock modulator). Hence the resultant deviation would be dependent on the phase skew of clock modulator which indeed is dependent on configuration such as resolution and modulation degrees at a given PLL frequency (CLKPLL).

2.3.9 FIFO Control Register (FCR)

Table 8. FCR

Bit No.	Name	Explanation	Value	Operation
7 … 4	RXL3-0	Receive Trigger level	0000 … 1111	The Receive trigger level sets the reception FIFO level (number of receptions) after which the reception interrupt should be activated.
3	X	Undefined	0	Always read as 0
2	ERX	Control RX FIFO	0	Disables RX FIFO
			1	Enables RX FIFO
1	ETX	Control TX FIFO	0	Disables TX FIFO
			1	Enables TX FIFO
0	SVD	Select FIFO read status for RX / TX	0	Select reading status from RX FIFO
			1	Select reading status from TX FIFO

RXL[3:0] bits sets up the level of the RX FIFO for the selected LIN USART channel. RXL = n (n = 0..15) will cause the interrupt flag RDRF (Receive Data Register Full) in the SSR (Serial Status Register) of the selected LIN USART channel to be set when the RX FIFO is enabled (ERX = 1) and corresponding number of bytes are received.

There are two different behaviors for the RXL bits :

- **Behavior one:**

If RXL is set to b'0000, then the interrupt would be generated after reception of 1 byte and if RXL is set as b'1111, then the interrupt would be generated after reception of 16 bytes. In general a reception (FIFO) interrupt is triggered if FSR[4:0] > FCR[7:4].

- **Behavior two:**

If RXL is set as b'0000, then the interrupt would be generated immediately and if RXL is set as b'1111, then the interrupt would be generated after reception of 15 bytes. In general a reception (FIFO) interrupt is triggered if FSR[4:0] ≥ FCR[7:4].

The information which device is using which behaviour is listed in the Hardware Manual or Correction Sheet of the Hardware Manual of MB91460 series.

Remark:

For all the devices having behaviour two, a write operation to the FCR register will initialize the FIFO buffer pointers of RX and TX FIFO. When writing FCR register, buffer data which may have been already received into the RX FIFO buffer will no longer be available. Hence FCR register should be configured before starting LIN USART communication to the setting targeted in the application (e.g. which FIFO level should be visible in the FIFO Status Register).

2.3.10 FIFO Status Register (FSR)

Table 9. FSR

Bit No.	Name	Explanation	Value	Operation
7 … 5	X	Undefined	0	Always read as 0
4 … 0	B4-0	FIFO valid data number	00000 … 10000	Indicates the number of stored receptions (SVD=0) or pending transmissions (SVD=1) in the FIFO buffer.

2.4 Error Conditions

2.4.1 Clearing Reception Errors and De-synchronization

CRE bit of SCR register resets reception state machine and next falling edge at SINn input starts reception of new byte. This behavior is shown in Figure 3. Therefore either set CRE bit immediately (within half bit time) after receiving errors to prevent data stream de-synchronization as shown in Figure 4 or wait an application dependent time after receiving errors and set CRE, when SINn input is idle.

Figure 3. CRE Bit Timing

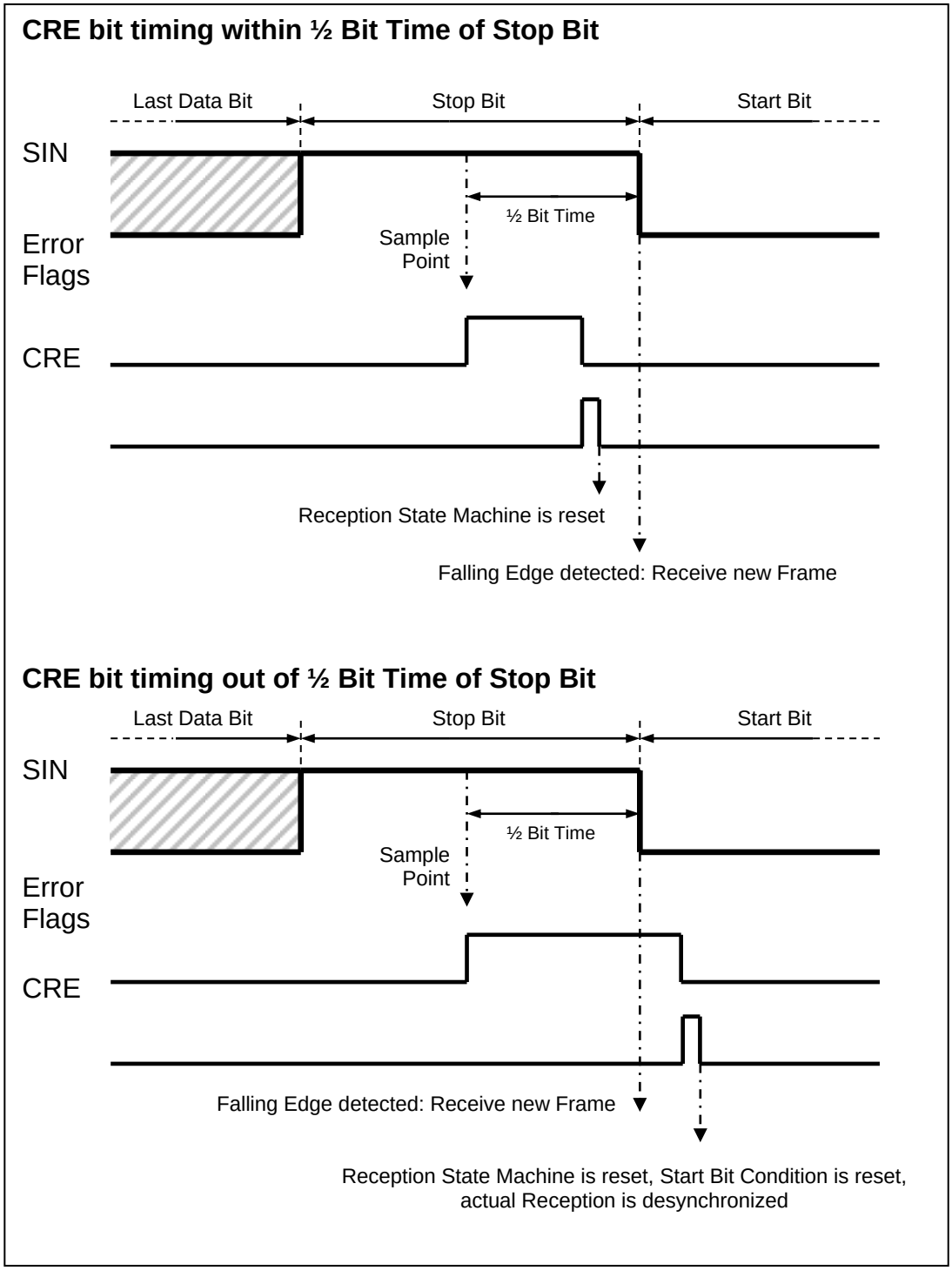
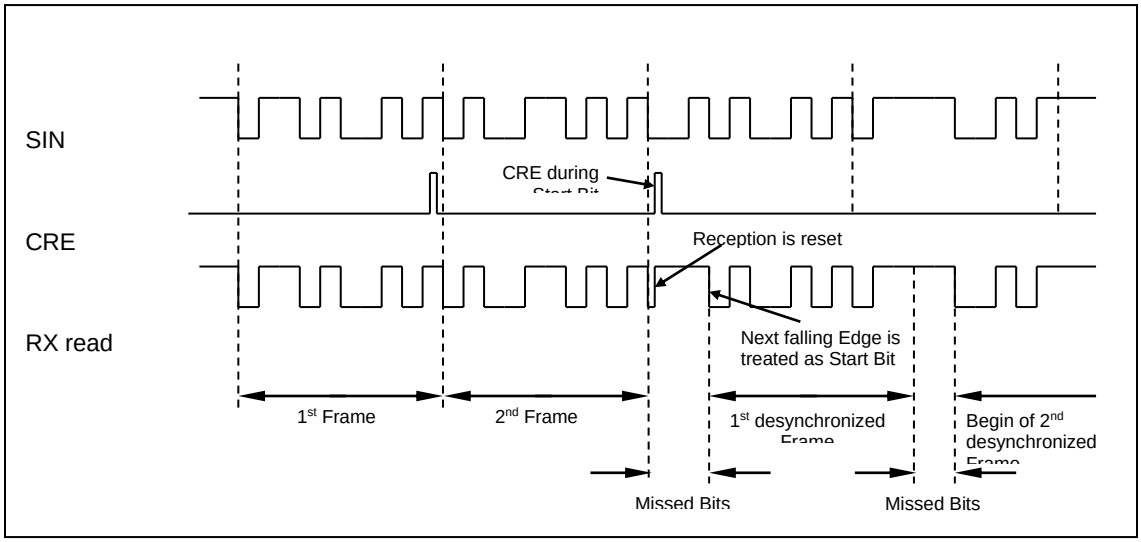


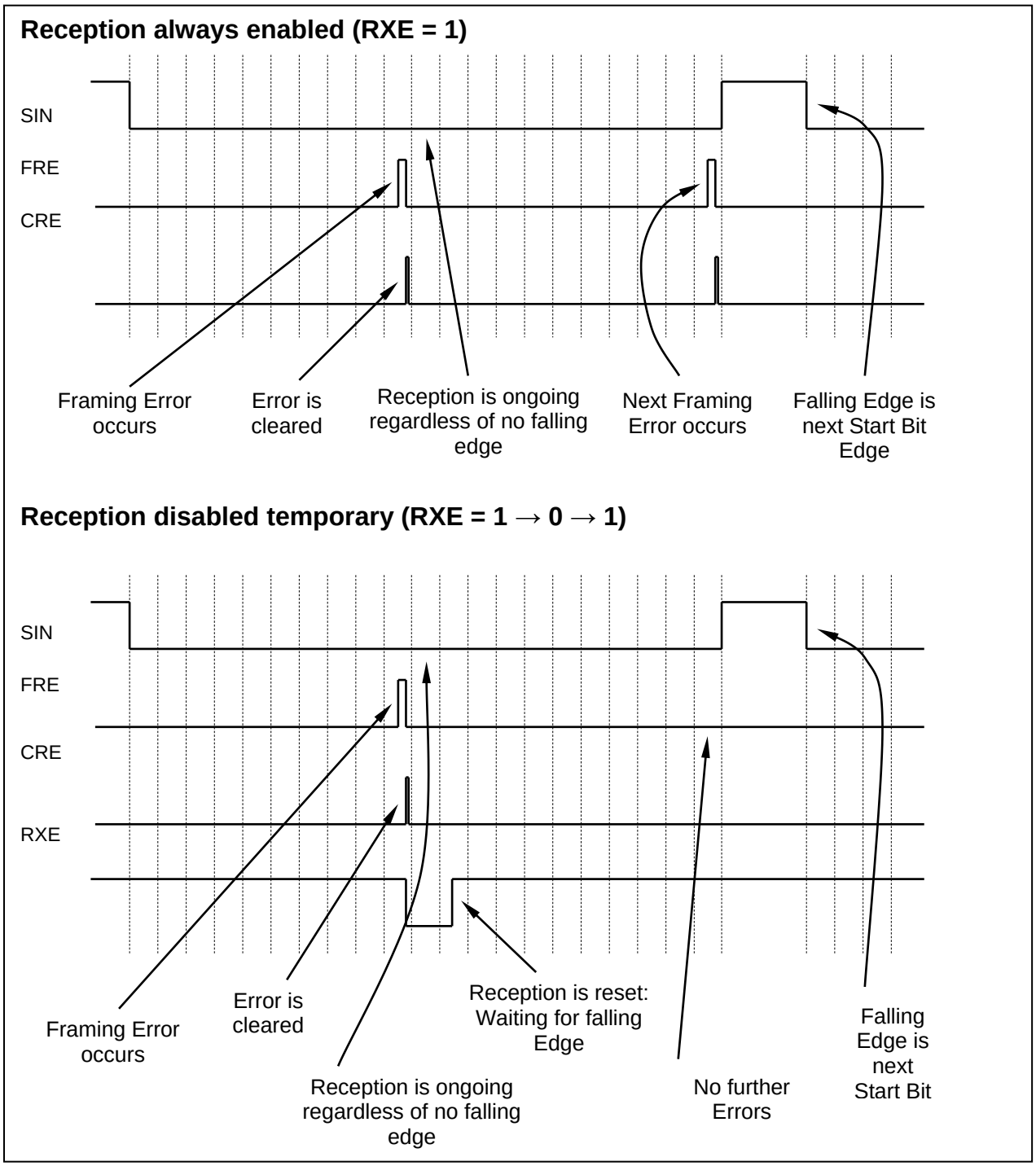
Figure 4. De-synchronization Example



2.4.2 USART Dominant Bus Behavior

Please note that in case a framing error occurred (stop bit: SINn = "0") and next start bit (SINn = "0") follows immediately, this start bit is recognized regardless of no falling edge before as shown in Figure 5. This is used to remain USART synchronized to the data stream and to determine bus always dominant errors by producing next framing errors, if a recessive stop bit is expected. If this behavior is not expected, please disable the reception temporarily (RXE = 1 -> 0 -> 1) after framing error. In this case, reception goes on at next falling edge on SINn.

Figure 5. USART Dominant Bus Behavior



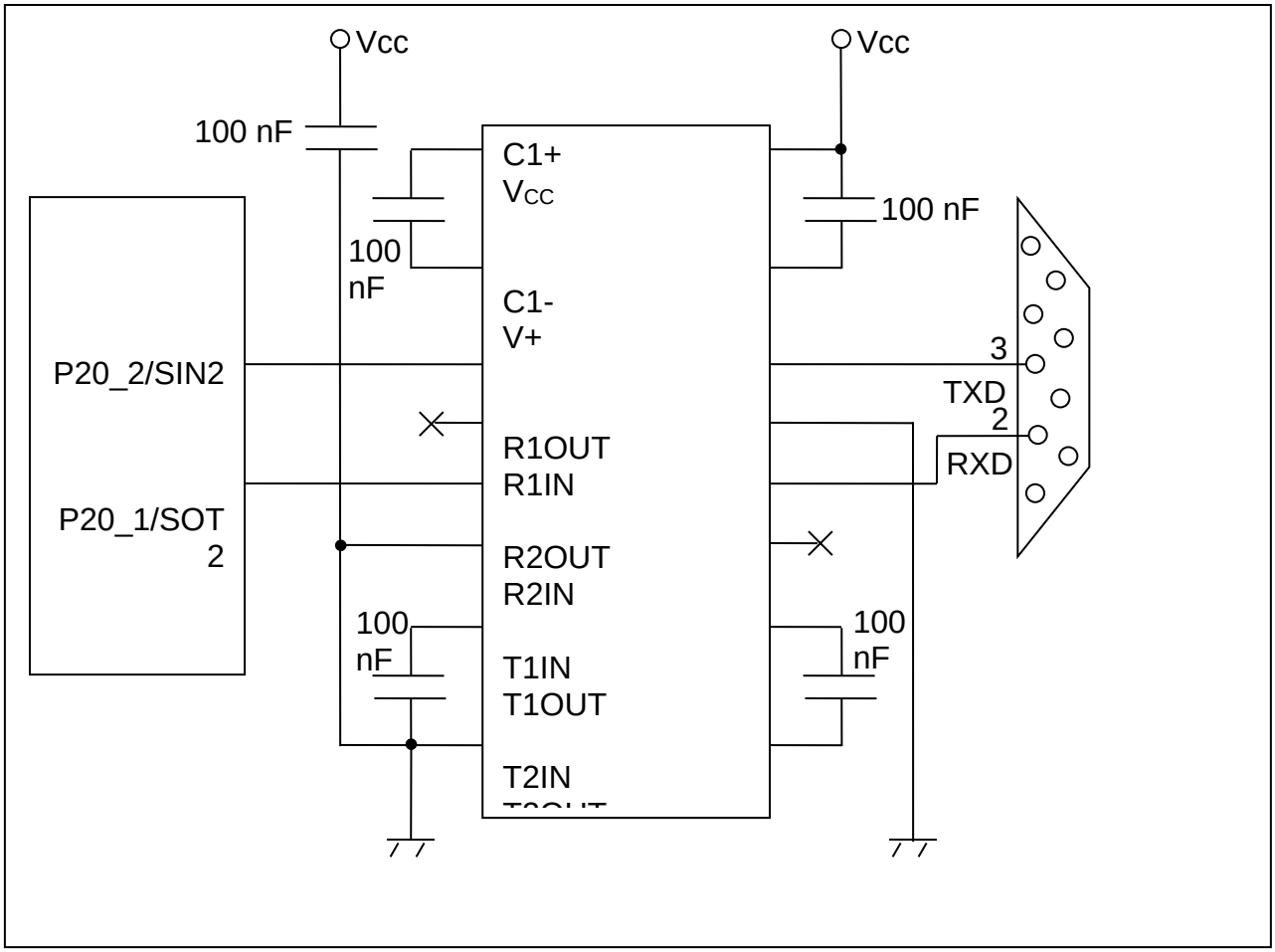
2.5 Interface to the BUS

2.5.1 RS232

USART in asynchronous mode can be interfaced to RS-232 bus via a transceiver. Transceiver provides the ability to receive and transmit the messages over the bus. Figure 6 shows interfacing of MB91467D microcontroller to Transceiver MAX3232CSE. The R1IN input and T1OUT output of the transceiver is connected to TXD and RXD signals of the DB-9 Connector respectively.

The value of the capacitors used is dependent on the supply voltage V_{CC}. Please refer the datasheet of MAX3232CSE for the same.

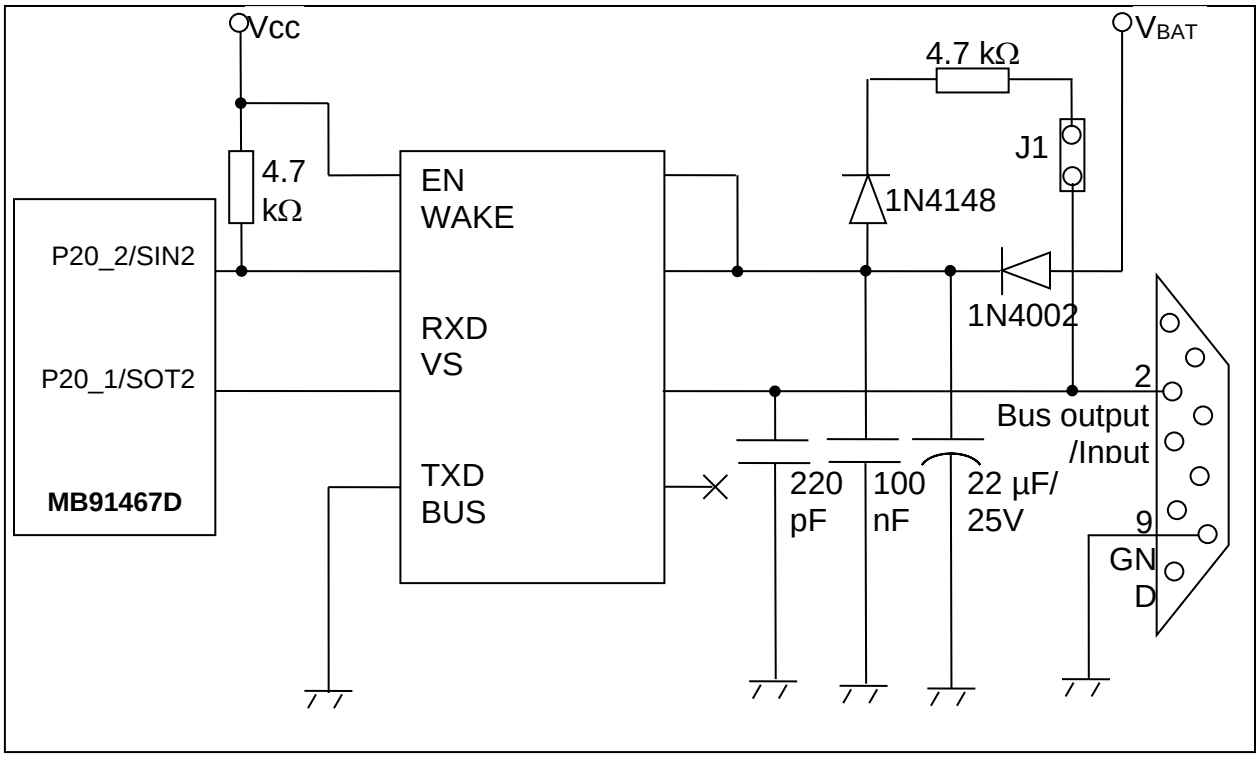
Figure 6. USART Interface to RS-232 Bus



2.5.2 LIN

USART in LIN mode can be interfaced to LIN bus via a transceiver. The transceiver provides the ability to receive and transmit the messages over the bus. Figure 7 shows interfacing of MB91F467D microcontroller to Transceiver TLE7259. The BUS Output/Input of the transceiver is connected to Bus Input/Output signal of a DB-9 Connector, which is the usual connection of the Fujitsu Starter kit Boards. V_{BAT} supply needs to be chosen within a range of 8V to 18V. Jumper J1 needs to be closed in case of LIN-Master and open in case of LIN-Slave.

Figure 7. USART Interface to LIN Bus



3. RX FIFO Behavior

The number of characters which can be received by a given LIN USART channel without overrun depends on the selection of the corresponding FIFO enable/disable flag in the LIN USART FIFO Control Register FCR.

3.1.1 RX FIFO Disabled

When a character is send to the LIN USART channel before the CPU read the character previously received by the same LIN USART channel from the corresponding RDR (Receive Data Register) register the first received character in the RDR register will be lost and the last received character is available in the RDR register.

To notify this situation to the CPU the Overrun Error flag ORE in the Serial Status Register SSR is set. This will cause an interrupt which is passed to the interrupt controller of the MCU if the receive interrupt of the selected LIN USART channel is enabled (the CPU will react on this interrupt if the corresponding ICR register of the MCU is setup appropriately and the global interrupt enable flag in the CPU processor status word is set. For details about interrupt-operation please refer to the hardware manual of the MB91460 Series).

When the Overrun Error flag occurred please use CRE bit in the Serial Control Register (SCR) to clear the Overrun Error flag. Writing 1 to the CRE flag in the SCR will clear the ORE flag (besides two other error flags: FRE (Framing Error) and PE (Parity Error)).

Please be careful while writing to the CRE flag. For more details please refer section 2.4.1 Clearing Reception Errors and De-synchronization.

3.1.2 RX FIFO Enabled

When the RX FIFO buffer of a LIN USART is enabled (`ERX` = 1) the LIN USART is able to read up to 16 characters stored in the FIFO buffer. When the 17th character is received by the LIN USART before the CPU read the first received character from the `RDR` register the first received character in the FIFO is lost (overwritten by the 17th received character). Any further received character will overwrite the 2nd, 3rd, … character in the FIFO.

Occurrence of FIFO Overrun Error is done in the same way as in the case when RX FIFO is not enabled (please refer 3.1.1).

4. USART Examples

Examples for USART

4.1 Asynchronous Mode (0) without Interrupts

```
/*                                    SAMPLE CODE                                    */
/*-------------------------------------------------------------------------------*/

void InitUart2(void)
{
    BGR02 = 1666;     // 9600 baud at 16MHz CLKP
    SCR02 = 0x17;     // 8 bit, clear reception errors, Tx & Rx enabled
    SSR02 = 0x00;     // LSB first, TDRE bit read only default 1 and written 0
    SMR02 = 0x0D;     // Mode 0, Reset Counter, Reset UART, SOT0 enabled

    PFR20_D0 = 1;      // enable SIN2 input
    PFR20_D1 = 1;      // enable resource function
    EPFR20_D1 = 0;     // enable SOT2 output
       }

void Putch2(char TXchr)           // sends a char
{
    while (SSR02_TDRE == 0);     // wait for transmit buffer empty
    TDR02 = TXchr;               // put TXchr into transmission buffer
       }

unsigned char Getch2(void)      // Waits for and returns incoming char
{
    unsigned char RXchr;

    for(;;)
    {
        while(SSR02_RDRF == 0)  // Wait for data received
        {
            __wait_nop();
        }

        RXchr = RDR02;           // Save receive register

        if ((SSR02 & 0xE0) != 0)// Check for errors PE, ORE, FRE
        {
            SCR02_CRE = 1;       // Clear error flags
        }
        else
        {
           return (RXchr);      // Return received character
        }
    }
       }
```

Please note, that the function Getch2() only returns a value, if a byte without any reception errors is received. Otherwise this function will never return.

Please also note that the SIN and SOT Port Pin has to be enabled. Here it is Port20-0 and Port20-1 pins are SIN2 and SOT2 respectively for MB91467D.

4.2 Asynchronous Mode (0) with Interrupts

```
  /*                                  SAMPLE CODE                                  */
  /*-------------------------------------------------------------------------------*/
#define MAXBUF 100;
unsigned char RXbuf[MAXBUF];// Reception Buffer
unsigned char TXbuf[MAXBUF];// Transmission Buffer
unsigned char RXptr = 0;   // Reception Buffer Pointer
unsigned char TXptr = 0;   // Transmission Buffer Pointer
void InitUart2(void)
{
    BGR02 = 1666;           // 9600 baud at 16MHz CLKP
    SCR02 = 0x17;          // 8 bit, clear reception errors, Tx & Rx enabled
    SSR02 = 0x00;         // LSB first, TDRE bit read only default 1 and written 0
    SMR02 = 0x0D;         // Mode 0, Reset Counter, Reset UART, SOT0 enabled
    PFR20_D0 = 1;         // enable SIN2 input
    PFR20_D1 = 1;         // enable resource function
    EPFR20_D1 = 0;       // enable SOT2 output
       }
// Reception Interrupt Service
       __interrupt void RX_USART2(void)
       {
    if ((SSR02 & 0xE0) != 0)// Check for errors PE, ORE, FRE
    {
        SCR02_CRE = 1;      // Clear error flags
    }
    if ( SSR02_RDRF == 1)  // if reception
    {
       if (RXptr < MAXBUF) // Only, if End of Buffer not reached
       {
            Rxbuf[RXptr] = RDR02;  // Fill Reception Buffer
            RXptr++;               // Update Buffer Pointer
       }
    }
}// Transmission Interrupt Service
       __interrupt void TX_USART2(void)
       {
           TDR02 = TXbuf[TXptr];         // Send Buffer Data
           TXptr++;                      // Update Buffer Pointer
           if (TXptr == MAXBUF)          // End of Buffer reached?
           {
               SSR02_TIE = 0;            // Disable Transmission Interrupt
           }
       }
// Main Function
void main(void)
{
    InitIrqLevels();
    __set_il(31);                 // allow all levels
    __EI();                       // globally enable interrupts
    PORTEN = 0x3;                 // Enable I/O Ports
    // Initialize UART2
           InitUart2();
           // Fill Transmission Buffer
           . . .
           // Start communication
           SSR02_RIE = 1;               // Enable Reception Interrupts
           SSR02_TIE = 1;               // Enable Transmission Interrupts
           . . .
       }
```

Please note, that the transmission starts immediately after the command SSR02_TIE = 1; because of the empty Transmission Register (TDRE = 1) causing a transmission interrupt.

Please also note that the corresponding interrupt vectors and levels have to be defined in the vectors.c module of our standard template project.

```
/*                              SAMPLE CODE                                    */
/*-----------------------------------------------------------------------------*/
void InitIrqLevels(void)

 . . .

     ICR21 = 20;    /* USART (LIN) 2 RX             */
                         /* USART (LIN) 2 TX             */
       . . .
}

__interrupt void RX_UART0(void);            /* prototype */
__interrupt void TX_UART0(void);            /* prototype */

 . . .

#pragma intvect DefaultIRQHandler 58    /* USART (LIN) 2 RX             */
#pragma intvect DefaultIRQHandler 59    /* USART (LIN) 2 TX             */
 . . .
```

Please also note that the SIN and SOT Port Pin has to be enabled. Here it is Port20-0 and Port20-1 pins are SIN2 and SOT2 respectively for MB91467D.

5. Example using RX FIFO

This chapter shows a short example using the RX FIFO of a selected LIN USART channel.

The following sample code demonstrates how to configure LIN USART4 RX FIFO to raise an RX interrupt to the CPU when 10 characters have been received into the LIN USART4 RX FIFO buffer.

Necessary steps:

- Initialize the LIN USART 4 at least for reception: Setup baud rate, 8/7 Bits, number of Stop bits, Parity enable). Setup corresponding Port Function Register. These steps are not described here in detail and are encapsulated in the function InitUART4().
- Setup FCR04 (FIFO Control Register of LIN USART channel 4)
- Setup appropriate LIN USART RX ISR (Interrupt Service Routine) for LIN USART channel 4 and add LIN USART RX ISR to the Interrupt Vector Table of the MB91460 Series MCU (for details on interrupt operation please refer to the hardware manual of MB91460 Series).
- Configure ICR25 (interrupt level for LIN USART RX/TX Interrupt).
- Configure Interrupt Level Mask in the CPU processor status word and enable CPU interrupt processing.
- Please make sure the interrupt handler routine UART4_RX_ISR()is included in the corresponding position in the interrupt vector table (for an example please refer to the file vetors.c included in the corresponding MB91460 Softune Workbench template project).

```
 /*                                SAMPLE CODE
    */
 /*------------------------------------------------------------------------------
    */
void InitUart4(void)
{
    BGR04 = 1666;   // 9600 baud at 16MHz CLKP
    SCR04 = 0x17;   // 8 bit, clear reception errors, Tx & Rx enabled
    SSR04 = 0x00;   // LSB first, TDRE bit read only default 1 and written 0
    SMR04 = 0x0D;   // Mode 0, Reset Counter, Reset UART, SOT0 enabled

   // Setup UART4 FIFO.
    FCR04_SVD = 0;    // Display RX FIFO status
    FCR04_ERX = 1;    // 1: enable RX FIFO; 0: disable RX-FIFO
    FCR04_RXL = 9;    // Interrupt after 10 bytes received in FIFO - for
                      // MB91465PA, MB91467TA and MB91469TA. For all the other
                      // variants interrupt after 9 bytes received.

    // Enable RX IRQ for UART4
    SSR04_RIE = 1;
    PFR19_D0 = 1;   // enable SIN4 input
    PFR19_D1 = 1;   // enable SOT4 output
      }
```

```
 /*                                    SAMPLE CODE                                */
 /*-------------------------------------------------------------------------------*/

unsigned char ch[16];

void main(void)
{
    InitIrqLevels();
    __set_il(31);                // allow all levels
    __EI();                      // globally enable interrupts
    PORTEN = 0x3;                // Enable I/O Ports

    // Initialize UART4
    InitUart4();

    while(1)          // Endless loop
    {
       HWWD_CL = 0;  // Clear Watchdog
       /* Do Something */
    }
}

// UART4 RX Interrupt Handler
__interrupt void UART4_RX_ISR(void)
{
    unsigned char number_of_chars;
    unsigned char count = 0;

    // Get the number of characters available in the RX FIFO
    number_of_chars = FSR04;

    // Read characters from the RX FIFO
    while(count < number_of_chars)
    {
      ch[count++] = RDR04;        // Reading RDR sequentially reads
                                  // characters from FIFO
    }
}
```

Please also note that the corresponding interrupt vectors and levels have to be defined in the *vectors.c* module of our standard template project.

```
 /*                                SAMPLE CODE                                  */
 /*-------------------------------------------------------------------------------*/
 void InitIrqLevels(void)

   . . .

     ICR25 = 30;    /* USART (LIN) 4 RX             */
                      /* USART (LIN) 4 TX             */
       . . .
 }

 __interrupt void UART4_RX_ISR (void);         /* prototype */
   . . .

 #pragma intvect DefaultIRQHandler 66    /* USART (LIN) 4 RX             */
   . . .
```

6. Example using TX FIFO

This chapter shows a short example using the TX FIFO a selected LIN USART channel.

The following sample code demonstrates how to configure LIN USART4 TX FIFO so that the transmit interrupt would only be set after transmission of 16 bytes.

Necessary steps:

- Initialize the LIN USART 4 for transmission: Setup baud rate, 8/7 Bits, number of Stop bits, Parity enable). Setup corresponding Port Function Register. These steps are PERFORMED in the function InitUART4().
- Setup FCR04 (FIFO Control Register of LIN USART channel 4)
- Setup appropriate LIN USART TX ISR (Interrupt Service Routine) for LIN USART channel 4 and add LIN USART TX ISR to the Interrupt Vector Table of the MB91460 Series MCU (for details on interrupt operation please refer to the hardware manual of MB91460 Series).
- Configure ICR25 (interrupt level for LIN USART RX/TX Interrupt).
- Configure Interrupt Level Mask in the CPU processor status word and enable CPU interrupt processing.
- Please make sure the interrupt handler routine UART4_TX_ISR()is included in the corresponding position in the interrupt vector table (for an example please refer to the file vetors.c included in the corresponding MB91460 Softune Workbench template project).

```
  /*                                  SAMPLE CODE                                */
  /*-----------------------------------------------------------------------------*/

void InitUart4(void)
{
    BGR04 = 1666;   // 9600 baud at 16MHz CLKP
    SCR04 = 0x17;   // 8 bit, clear reception errors, Tx & Rx enabled
    SSR04 = 0x00;   // LSB first, TDRE bit read only default 1 and written 0
    SMR04 = 0x0D;   // Mode 0, Reset Counter, Reset UART, SOT0 enabled

   // Setup UART4 FIFO
    FCR04_SVD = 1;    // Display TX FIFO status
    FCR04_ETX = 1;    // 1: enable TX FIFO; 0: disable TX-FIFO

    // Enable RX IRQ for UART4
    SSR04_TIE = 1;
    PFR19_D0 = 1;   // enable SIN4 input
    PFR19_D1 = 1;   // enable SOT4 output
       }
```

```
/*                                SAMPLE CODE                                  */
/*-----------------------------------------------------------------------------
      */

unsigned char ch[16] = {"abcdefghijklmnop"};

void main(void)
{
    InitIrqLevels();
    __set_il(31);                // allow all levels
    __EI();                      // globally enable interrupts
    PORTEN = 0x3;                // Enable I/O Ports

    // Initialize UART4
    InitUart4();

    while(1)                      // Endless loop
    {
       HWWD_CL = 0;              // Clear Watchdog
       /* Do Something */
    }
}

// UART4 TX Interrupt Handler
__interrupt void UART4_TX_ISR(void)
{
    unsigned char count = 0;

    // Write characters to the TX FIFO
    while(count < 16)
    {
      TDR04 = ch[count++];        // Writing TDR sequentially writes
                                  // characters to FIFO
    }
}
```

Please also note that the corresponding interrupt vectors and levels have to be defined in the *vectors.c* module of our standard template project.

```
/*                                SAMPLE CODE                                  */
/*-----------------------------------------------------------------------------*/
void InitIrqLevels(void)
{
   . . .

      ICR25 = 30;     /* USART (LIN) 4 RX             */
                      /* USART (LIN) 4 TX             */
. . .
}

__interrupt void UART4_RX_ISR (void);         /* prototype */
   . . .

#pragma intvect DefaultIRQHandler 66    /* USART (LIN) 4 RX             */
   . . .
```

7. Appendix A

Related Documents

7.1 Related Documents

Please find further information in the following documents.

- AN205378 - FR, MB91460, SPI Communication to/from Serial EEPROM (for NM93CS46)

8. Additional Information

Information about Cypress Microcontrollers can be found on the following Internet page:

http://www.cypress.com/cypress-microcontrollers

The software example related to this application note is:

91460_uart_async

91460_uart4_fifo

91460_adc8_uart_async

91460_adc10_uart_async

91460_dma_uart0

It can be found on the following Internet page:

http://www.cypress.com/cypress-mcu-product-softwareexamples

Document History

Document Title: AN205255 - FR, MB91460, USART

Document Number: 002-05255

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	NOFL	05/29/2008	V1.0; First Version; MPi
			06/14/2010	V1.1; CPHA, CPOL logic corrected; MWi
*A	5082062	NOFL	04/06/2016	Converted Spansion Application Note “MCU-AN-300044-E-V11” to Cypress format
*B	5873598	AESATMP9	09/05/2017	Updated logo and copyright.
*C	6058881	NOFL	02/05/2018	Updated hyperlinks across the document. Updated to new template. Completing Sunset Review.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

Cypress Developer Community

Community | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2008-2018. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.