F²MC-16FX FAMILY
16-BIT MICROCONTROLLER
ALL SERIES

USART

APPLICATION NOTE
Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Issue</th>
</tr>
</thead>
<tbody>
<tr>
<td>2006-04-28</td>
<td>First Version; MWi</td>
</tr>
<tr>
<td>2006-12-07</td>
<td>V1.1, Reviewed the document and updated with review findings, MPi</td>
</tr>
<tr>
<td>2007-02-21</td>
<td>V1.2, Updated with re-review findings, MPi</td>
</tr>
<tr>
<td>2007-08-29</td>
<td>V1.3, Added schematic, error conditions and updated register related information, MPi</td>
</tr>
<tr>
<td>2007-10-10</td>
<td>V1.4, Timing diagrams updated, MWi</td>
</tr>
<tr>
<td>2008-07-07</td>
<td>V1.5, Add information on PIER; PHu</td>
</tr>
</tbody>
</table>

This document contains 28 pages.
Warranty and Disclaimer

To the maximum extent permitted by applicable law, Fujitsu Microelectronics Europe GmbH restricts its warranties and its liability for all products delivered free of charge (e.g. software include or header files, application examples, target boards, evaluation boards, engineering samples of IC’s etc.), its performance and any consequential damages, on the use of the Product in accordance with (i) the terms of the License Agreement and the Sale and Purchase Agreement under which agreements the Product has been delivered, (ii) the technical descriptions and (iii) all accompanying written materials. In addition, to the maximum extent permitted by applicable law, Fujitsu Microelectronics Europe GmbH disclaims all warranties and liabilities for the performance of the Product and any consequential damages in cases of unauthorised decompiling and/or reverse engineering and/or disassembling. Note, all these products are intended and must only be used in an evaluation laboratory environment.

1. Fujitsu Microelectronics Europe GmbH warrants that the Product will perform substantially in accordance with the accompanying written materials for a period of 90 days form the date of receipt by the customer. Concerning the hardware components of the Product, Fujitsu Microelectronics Europe GmbH warrants that the Product will be free from defects in material and workmanship under use and service as specified in the accompanying written materials for a duration of 1 year from the date of receipt by the customer.

2. Should a Product turn out to be defect, Fujitsu Microelectronics Europe GmbH’s entire liability and the customer’s exclusive remedy shall be, at Fujitsu Microelectronics Europe GmbH’s sole discretion, either return of the purchase price and the license fee, or replacement of the Product or parts thereof, if the Product is returned to Fujitsu Microelectronics Europe GmbH in original packing and without further defects resulting from the customer’s use or the transport. However, this warranty is excluded if the defect has resulted from an accident not attributable to Fujitsu Microelectronics Europe GmbH, or abuse or misapplication attributable to the customer or any other third party not relating to Fujitsu Microelectronics Europe GmbH.

3. To the maximum extent permitted by applicable law Fujitsu Microelectronics Europe GmbH disclaims all other warranties, whether expressed or implied, in particular, but not limited to, warranties of merchantability and fitness for a particular purpose for which the Product is not designated.

4. To the maximum extent permitted by applicable law, Fujitsu Microelectronics Europe GmbH’s and its suppliers’ liability is restricted to intention and gross negligence.

NO LIABILITY FOR CONSEQUENTIAL DAMAGES

To the maximum extent permitted by applicable law, in no event shall Fujitsu Microelectronics Europe GmbH and its suppliers be liable for any damages whatsoever (including but without limitation, consequential and/or indirect damages for personal injury, assets of substantial value, loss of profits, interruption of business operation, loss of information, or any other monetary or pecuniary loss) arising from the use of the Product.

Should one of the above stipulations be or become invalid and/or unenforceable, the remaining stipulations shall stay in full effect.
CONTENTS

1 INTRODUCTION
   1.1 Key Features

2 THE USART WITH LIN FUNCTIONALITY
   2.1 Block Diagram
   2.2 Basic Functionality
   2.3 Registers
      2.3.1 Serial Control Register (SCR)
      2.3.2 Serial Mode Register (SMR)
      2.3.3 Serial Status register (SSR)
      2.3.4 Reception Data Register (RDR)
      2.3.5 Transmission Data Register (TDR)
      2.3.6 Extended Status/Control Register (ESCR)
      2.3.6.1 LIN Break Detection Consideration
      2.3.7 Extended Communication Control register (ECCR)
      2.3.8 Baud Rate Generator Register (BGR)
      2.3.8.1 Minimum and Maximum Ratings
   2.4 Error Conditions
      2.4.1 Clearing Reception Errors and De-synchronization
      2.4.2 USART Dominant Bus Behavior
   2.5 Interface to the BUS
      2.5.1 RS232
      2.5.2 LIN

3 USART EXAMPLES
   3.1 Asynchronous Mode (0) without Interrupts
   3.2 Asynchronous Mode (0) with Interrupts
   3.3 Synchronous Mode (SPI) Master without Interrupts

4 APPENDIX A
   4.1 Related Documents

5 ADDITIONAL INFORMATION
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 In- and Output
- NRZ/RZ for serial Clock In- and Output
- Asynchronous and synchronous Mode
- 7-8 Data Bits and 1-2 Stop Bits for asynchronous mode, even or odd Parity selectable
- Dedicated Reload Counter as Clock Divider for Baud Rate (610 Bits/s up to 4 MBits/s synchronous at 20 MHz Peripheral Clock)
- Reload Counter 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)
2 The USART with LIN functionality

THE BASIC FUNCTIONALITY OF THE USART WITH LIN FUNCTIONALITY

2.1 Block Diagram

Figure 2-1 shows the internal block diagram of the USART.
2.2 Basic Functionality

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

<table>
<thead>
<tr>
<th>MD1</th>
<th>MD0</th>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Asynchronous (normal mode)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Asynchronous (Master/Slave Mode)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>Synchronous Mode</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>Asynchronous (LIN Mode)</td>
</tr>
</tbody>
</table>

Table 2-1: USART Operation Modes

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:

```c
SMR0 = 0x83; // Set USART to Mode 2 (synchronous operation), serial output & serial clock output enabled
SMR0 = 0x87; // Reset USART (UPCL bit#2 is auto-cleared)
```

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

Table 2-2: SCR

Note: When enabling reception, please also enable the corresponding input pin with its Port Input Enable Register (PIER) bit.

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

*Table 2-3: SMR*

Note: If synchronous slave mode is selected, SCK pin input must be enabled using corresponding Port Input Enable Register (PIER).

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

*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).

#### Table 2-4: SSR

### 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. The RDRF flag also generates the reception interrupt, if enabled.

### 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. Note, that it is “1” after Reset.
2.3.6 Extended Status/Control Register (ESCR)

<table>
<thead>
<tr>
<th>Bit No.</th>
<th>Name</th>
<th>Explanation</th>
<th>Value</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>LBI E</td>
<td>LIN Break Detection Interrupt enable*11</td>
<td>0</td>
<td>Interrupt disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Interrupt enabled</td>
</tr>
<tr>
<td>6</td>
<td>LBD</td>
<td>LIN Break Detection Flag/Clear*11</td>
<td>0</td>
<td>Write: Clear LIN Break Detection Interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>LIN Break detected</td>
</tr>
<tr>
<td>5, 4</td>
<td>LBL1, LBL0</td>
<td>LIN Break Generation Length*11</td>
<td>0, 0</td>
<td>13 Bit Times</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0, 1</td>
<td>14 Bit Times</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1, 0</td>
<td>15 Bit Times</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1, 1</td>
<td>16 Bit Times</td>
</tr>
<tr>
<td>3</td>
<td>SOPE</td>
<td>Serial Output Pin Enable</td>
<td>0</td>
<td>SOT Pin is Serial Out</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>SOT Pin is state of SIOP</td>
</tr>
<tr>
<td>2</td>
<td>SIOP</td>
<td>Serial Input/Output Pin</td>
<td>0</td>
<td>Write: Force SOT to “0”, if SOPE=1, Read: State of SIN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Write: Force SOT to “1”, if SOPE=1, Read: State of SIN</td>
</tr>
<tr>
<td>1</td>
<td>CCO</td>
<td>Continuous Clock Output*12</td>
<td>0</td>
<td>Continuous Clock disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Continuous Clock enabled</td>
</tr>
<tr>
<td>0</td>
<td>SCES</td>
<td>Serial Clock Edge Selection (CPOL)*13</td>
<td>0</td>
<td>Clock with Mark Level “1”</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Clock with Mark Level “0”</td>
</tr>
</tbody>
</table>

*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 Break Detection without Synch. Field Detection. In such case, after the LIN break has been detected perform the following steps:

- 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 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.
### 2.3.7 Extended Communication Control register (ECCR)

<table>
<thead>
<tr>
<th>Bit No.</th>
<th>Name</th>
<th>Explanation</th>
<th>Value</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>INV</td>
<td>Invert Serial Data*14</td>
<td>0</td>
<td>Serial Data is not inverted (NRZ format)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Serial Data is inverted (RZ format)</td>
</tr>
<tr>
<td>6</td>
<td>LBR</td>
<td>Generate LIN Break*15</td>
<td>0</td>
<td>Ignored</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Write: Set LIN Break</td>
</tr>
<tr>
<td>5</td>
<td>MS</td>
<td>Synchronous Master/Slave Select*16</td>
<td>0</td>
<td>Synchronous Master Mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Synchronous Slave Mode</td>
</tr>
<tr>
<td>4</td>
<td>SCDE</td>
<td>Synchronous Serial Clock Delay (CPHA)*16</td>
<td>0</td>
<td>No Clock Delay</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Clock Delay of half Bit Time</td>
</tr>
<tr>
<td>3</td>
<td>SSM</td>
<td>Synchronous Start/Stop Bit Mode*16</td>
<td>0</td>
<td>No start/stop bits in synchronous mode 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Enable start/stop bits in synchronous mode 2</td>
</tr>
<tr>
<td>2</td>
<td>BIE</td>
<td>Bus idle interrupt enable</td>
<td>0</td>
<td>Disable bus idle interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Enable bus idle interrupt</td>
</tr>
<tr>
<td>1</td>
<td>RBI</td>
<td>Reception Bus Idle*17</td>
<td>0</td>
<td>Activity on SIN, Reception ongoing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Bus Idle, no Reception ongoing</td>
</tr>
<tr>
<td>0</td>
<td>TBI</td>
<td>Transmission Bus Idle*18</td>
<td>0</td>
<td>Activity on SOT, Transmission ongoing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Bus Idle, no Transmission</td>
</tr>
</tbody>
</table>

*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 flag bit cannot be used in synchronous slave Mode 2.

*18 This flag bit cannot be used in synchronous slave Mode 2 (when MS=1).
2.3.8 Baud Rate Generator Register (BGR)

This 15-Bit register contains the divider for the baud rate. The value “v” for the divider can be calculated as:

\[
v = \lfloor \Phi / b \rfloor - 1,
\]

Where, \( \Phi \) is the peripheral clock CLKP1,

\( b \) the baud rate

and \( \lfloor \rfloor \) gaussian brackets (mathematical rounding function).

\[
\text{Figure 2-2: USART Clock}
\]

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. If this division factor is insufficient, the Peripheral Clock has to be divided then.

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

Table 2-7: Baud Rate corresponding to CLKP1

For the relationship between the baud rate and reload values of baud rate generator at different peripheral clock CLKP1 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 CLKP1 clock frequency and reload value. This deviation would be further affected if the CLKP1 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).

2.4 Error Conditions

2.4.1 Clearing Reception Errors and De-synchronization

\( \text{CRE} \) bit of \( \text{SCR} \) register resets reception state machine and next falling edge at SINn input starts reception of new byte. This behavior is shown in Figure 2-3. Therefore either set \( \text{CRE} \) bit immediately (within half bit time) after receiving errors to prevent data stream de-
synchronization as shown in Figure 2-4 or wait an application dependent time after receiving errors and set CRE, when SINn input is idle.

**Figure 2-3: CRE Bit Timing**

**CRE bit timing within ½ Bit Time of Stop Bit**

- **Last Data Bit** to **Stop Bit** to **Start Bit**
- **SIN**
- **Error Flags**
- **CRE**
- Sample Point
- Reception State Machine is reset
- Falling Edge detected: Receive new Frame

**CRE bit timing out of ½ Bit Time of Stop Bit**

- **Last Data Bit** to **Stop Bit** to **Start Bit**
- **SIN**
- **Error Flags**
- **CRE**
- Sample Point
- Falling Edge detected: Receive new Frame
- Reception State Machine is reset, Start Bit Condition is reset, actual Reception is desynchronized

© Fujitsu Microelectronics Europe GmbH
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 5. This is used to remain UART 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.
Reception always enabled (RXE = 1)

Reception disabled temporary (RXE = 1 → 0 → 1)

---

Figure 2-5: USART Dominant Bus Behaviour
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 2-6 shows interfacing of MB9634x 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 2-6: USART Interface to RS-232 Bus
2.5.2 LIN

USART in LIN mode can be interfaced to LIN bus via a transceiver. Transceiver provides the ability to receive and transmit the messages over the bus. Figure 2-7 shows interfacing of MB9634x 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 Starterkit Boards. $V_{BAT}$ supply needs to 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 2-7: USART Interface to LIN Bus](image)
### 3 USART Examples

#### EXAMPLES FOR USART

#### 3.1 Asynchronous Mode (0) without Interrupts

```c
/* THIS SAMPLE CODE IS PROVIDED AS IS AND IS SUBJECT TO ALTERATIONS. FUJITSU */
/* MICROELECTRONICS ACCEPTS NO RESPONSIBILITY OR LIABILITY FOR ANY ERRORS OR */
/* ELIGIBILITY FOR ANY PURPOSES. */
/* (C) Fujitsu Microelectronics Europe GmbH */
/*---------------------------------------------------------------------------*/

void InitUsart0(void)
{
    PIER08_IE2 = 1; // Enable SINO Port Pin
    BGR0 = 1249;      // 19200 Baud @ 24MHz
    SSR0 = 0x00;      // LSB first, TDRE bit read only default 1 and written 0
    SMR0 = 0x0d;      // enable SOT0, reset, Asynchronous normal mode
    SCR0 = 0x17;      // SN1, clear possible errors, enable RX, TX
}

void Putch0(char TXchr)        // sends a char
{
    while (SSR0_TDRE == 0);    // wait for transmit buffer empty
    TDR0 = TXchr;              // put TXchr into transmission buffer
}

unsigned char Getch0(void) // Waits for and returns incoming char
{
    unsigned char RXchr;
    for(;;)
    {
        while(SSR0_RDRF == 0)  // Wait for data received
        {
            __wait_nop();
        }
        RXchr = RDR0;            // Save receive register
        if ((SSR0 & 0xE0) != 0)// Check for errors PE, ORE, FRE
        {
            SCR0_CRE = 1;        // Clear error flags
        }
        else
        {
            return (RXchr);     // Return received character
        }
    }
}
```

Please note, that the function `Getch0()` 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 Port Pin has to be enabled. Here it is Port08-2 for MB96(F)34x.
3.2 Asynchronous Mode (0) with Interrupts

```c
/* THIS SAMPLE CODE IS PROVIDED AS IS AND IS SUBJECT TO ALTERATIONS. FUJITSU */
/* MICROELECTRONICS ACCEPTS NO RESPONSIBILITY OR LIABILITY FOR ANY ERRORS OR */
/* ELIGIBILITY FOR ANY PURPOSES. */
(C) Fujitsu Microelectronics Europe GmbH

/*---------------------------------------------------------------------------*/
#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 InitUsart0(void)
{
    PIER08_IE2 = 1; // Enable SIN0 Port Pin
    BGRO = 1249; // 19200 Baud @ 24MHz
    SSR0 = 0x00; // LSB first read only default 1 written 0
    SMRO = 0x0d; // enable SOT0, reset, Asynchronous normal mode
    SCR0 = 0x17; // SN1, clear possible errors, enable RX, TX
}

// Reception Interrupt Service
__interrupt void RX_USART0(void)
{
    if ((SSR0 & 0xE0) != 0) // Check for errors PE, ORE, FRE
    {
        SCR0_CRE = 1; // Clear error flags
    }
    if (RXptr < MAXBUF) // Only, if End of Buffer not reached
    {
        Rxbuf[RXptr] = RDR0; // Fill Reception Buffer
        RXptr++; // Update Buffer Pointer
    }
}

// Transmission Interrupt Service
__interrupt void TX_USART0(void)
{
    TDR0 = TXbuf[TXptr]; // Send Buffer Data
    TXptr++; // Update Buffer Pointer
    if (TXptr == MAXBUF) // End of Buffer reached?
    {
        SSR0_TIE = 0; // Disable Transmission Interrupt
    }
}

// Main Function
void main(void)
{
    InitIrqLevels(); // allow all levels
    __set_il(7); // globally enable interrupts
    // Initialize USART0
    InitUsart0();
    // Fill Transmission Buffer
    ...
    // Start communication
    SSR0_RIE = 1; // Enable Reception Interrupts
    SSR0_TIE = 1; // Enable Transmission Interrupts
    ...
}
```
Please note, that the transmission starts immediately after the command `SSR0_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.

```c
/* THIS SAMPLE CODE IS PROVIDED AS IS AND IS SUBJECT TO ALTERATIONS. FUJITSU */
/* MICROELECTRONICS ACCEPTS NO RESPONSIBILITY OR LIABILITY FOR ANY ERRORS OR */
/* ELIGIBILITY FOR ANY PURPOSES. */
/* (C) Fujitsu Microelectronics Europe GmbH */
/*---------------------------------------------------------------------------*/
void InitIrqLevels(void)
{
   . . .
   ICR = (79 << 8) | 6;   // LIN-UART 0 RX of MB96340 Series
   ICR = (80 << 8) | 6;   // LIN-UART 0 TX of MB96340 Series
   . . .

   __interrupt void RX_USART0(void);   // prototype
   __interrupt void TX_USART0(void);   // prototype
   . . .
   #pragma intvect RX_USART0 79        // LIN-UART 0 RX of MB96340 Series
   #pragma intvect TX_USART0 80        // LIN-UART 0 TX of MB96340 Series
   . . .
```

Please also note that the SIN Port Pin has to be enabled. Here it is Port08-2 for MB96(F)34x.
3.3 Synchronous Mode (SPI) Master without Interrupts

The following example shows how to communicate to a NM93CS46 EEPROM. Therefore clock inversion and shift is used (SCES = 1, SCDE = 1). Port PDR09 is used for Chip Select (CS).

/* THIS SAMPLE CODE IS PROVIDED AS IS AND IS SUBJECT TO ALTERATIONS. FUJITSU */
/* MICROELECTRONICS ACCEPTS NO RESPONSIBILITY OR LIABILITY FOR ANY ERRORS OR */
/* ELIGIBILITY FOR ANY PURPOSES. */
/* (C) Fujitsu Microelectronics Europe GmbH */
#define DATASIZE 64   // eeprom memory size in words (16 Bit)
unsigned int data[DATASIZE];  // Data to send to EEPROM
unsigned int readbuffer[DATASIZE]; // Data received from EEPROM

void InitUART(void)
{
PIER08_IE2 = 1; // Enable SIN Port Pin
BGR0 = 15;   // 1M Bit/s @ 16 MHz
ESCR0 = 0x01;   // SCES = 1 => CPOL = 1
ECCR0 = 0x10;   // SCDE = 1 => CPHA = 1
SMR0 = 0x83;   // Mode 2, SCLK enable, SOT enable
SSR0 = 0x04;   // MSB first, no interrupts
SCR0 = 0x03;   // Reception and transmission enable
}

void InitPort09(void)
{
    // Bit#2: CS, Bit#1: PE, Bit#0: PRE
    PDR09 = 0x00;   // All Low
    DDR09 = 0x07;   // CS, PE, PRE to output
}

void read_eeprom(unsigned char adr)
{
    unsigned char din, command, dout;
    PDR09_P2 = 1;   // CS = 1
    TDR0 = 0x01;   // Start-Bit (with leading spaces)
    command = (adr & 0x3F) | 0x80;// Address and Write-Instruction
    dout = command;  // Swap Bits for MSB first??
    while (SSR0_RDRF == 0); // Transmission finished (via reception)?
    din = RDR0;   // Flush reception register
    TDR0 = dout;
    while (SSR0_RDRF == 0); // Transmission finished (via reception)?
    din = RDR0;   // Flush reception register
    SCR0_CRE = 1;   // Clear possible errors, reset reception state
    //   machine
    // NOTE: Make sure, that SCK is "0" while setting SCDE to "0" (ECCR0 = 0x00;)
    // In this case (1M bps) no check is needed. Be careful with slower
    // baud rates!
    ECCR0 = 0x00;   // SCDE = 0 => CPHA = 0 : Needed for special read
    // timing of used EEPROM (may be not necessary
    // for other EEPROM)
    TDR0 = 0x00;   // Set dummy byte to produce SCLK
    while (SSR0_RDRF == 0); // Transmission finished (via reception)?
    din = RDR0;   // MSB
    readBuffer[adr] = (din << 8);
    while (SSR0_TDRE == 0); // Set dummy byte to produce SCLK
    TDR0 = 0x00;
    while (SSR0_RDRF == 0);
void write_eeprom(unsigned char adr)
{
    unsigned char dout, command;
    PDR09_P2 = 1;    // CS = 1
    while (SSR0_TDRE == 0); // Start-Bit (with leading spaces)
    command = (adr & 0x3F) | 0x40; // Address and Write-Instruction
    dout = command;
    while (SSR0_TDRE == 0);
    TDR0 = dout;
    dout = (data[adr] >> 8) & 0xFF; // MSB
    while (SSR0_TDRE == 0);
    TDR0 = dout;
    dout = data[adr] & 0xFF; // LSB
    while (SSR0_TDRE == 0);
    TDR0 = dout;
    while (ECCR0 & 0x01);   // Wait for start of transmission (or ongoing)
    while (!(ECCR0 & 0x01));  // Wait for transmission finished
    PDR09_P2 = 0;    // CS = 0
    wait(1);
    PDR09_P2 = 1;    // CS = 1
    // Next function (waiting for busy release) is made by // polling. Please note, that for the NM93CS46 EEPROM the // wait time can take till 10 ms! I. e. the CPU is then // also busy. For fast application a timer should be used, // which generates an interrupt after 10 ms from here, // so that the CPU can perform other jobs in this time.
    while(ESCR0_SIOP == 1);  // Wait for EEPROM busy
    while(ESCR0_SIOP == 0);  // Wait for EEPROM busy release
    PDR09_P2 = 0;    // CS = 0
}

void write_enable(void)
{
    PDR09_P2 = 1;    // CS = 1
    while (SSR0_TDRE == 0); // Start-Bit (with leading "zeros")
    TDR0 = 0x01;
    while (SSR0_TDRE == 0); // Start-Bit (with leading "zeros")
    TDR0 = 0x01;
    while (ECCR0 & 0x01);   // Wait for start of transmission (or ongoing)
    while (!(ECCR0 & 0x01));  // Wait for transmission finished
    PDR09_P2 = 0;    // CS = 0
}

void write_disable(void)
{
    PDR09_P2 = 1;    // CS = 1
    while (SSR0_TDRE == 0); // Start-Bit (with leading "zeros")
    TDR0 = 0x01;
    while (SSR0_TDRE == 0); // Start-Bit (with leading "zeros")
    TDR0 = 0x01;
    while (ECCR0 & 0x01);   // Wait for start of transmission (or ongoing)
    while (!(ECCR0 & 0x01));  // Wait for transmission finished
    PDR09_P2 = 0;    // CS = 0
}
Please note, that the SIN Port Pin has to be enabled. Here it is Port08-2 for MB96(F)34x.

```c
while (SSR0_TDRE == 0);
TDR0 = 0x00; // WDS command

while (ECCR0 & 0x01); // Wait for start of transm. (or ongoing)
while (!(ECCR0 & 0x01)); // Wait for transmission finished
PDR09_P2 = 0; // CS = 0
```

4 APPENDIX A

RELATED DOCUMENTS

4.1 Related Documents

Please find further information in the following documents.

- **MCU-AN-390088-UART_LIN** Usage of LIN-U(S)ART
- **MCU-AN-390090-UART_LIN_BAUDRATE_CLK** LIN-U(S)ART Baud Rate Generator
- **MCU-AN-300002-SPI** SPI Modes
- **MCU-AN-390105-SPI_ADC** SPI with MAX1286
- **MCU-AN-390104-SPI_EEPROM** SPI with NM93CS46

Note, that the above mentioned documents are written for 16LX MCUs, but also usable for 16FX-USART. Please also note that for 16FX devices the serial data or clock input port pin has to be enabled via PIER register.
5 Additional Information

Information about FUJITSU Microcontrollers can be found on the following Internet page:
http://mcu.emea.fujitsu.com/

The software example related to this application note is:

96340_uart0_async
96340_uart0_async_interrupt
96340_uart1_async
96340_uart2_async
96340_uart3_async
96340_uart_lin_master
96340_uart_lin_slave
96340_uart_sync_spi_nm93cs46
96340_dma_uart0

It can be found on the following Internet page:
http://mcu.emea.fujitsu.com/mcu_product/mcu_all_software.htm
List of Figures

Figure 2-1: USART Block Diagram................................................................. 7
Figure 2-2: USART Clock........................................................................... 13
Figure 2-3: CRE Bit Timing ....................................................................... 14
Figure 2-4: De-synchronization Example.................................................. 15
Figure 2-5: USART Dominant Bus Behaviour ........................................... 16
Figure 2-6: USART Interface to RS-232 Bus............................................. 17
Figure 2-7: USART Interface to LIN Bus .................................................... 18
List of Tables

Table 2-1: USART Operation Modes...................................................................................... 8
Table 2-2: SCR ....................................................................................................................... 8
Table 2-3: SMR ..................................................................................................................... 9
Table 2-4: SSR ..................................................................................................................... 10
Table 2-5: ESCR ................................................................................................................... 11
Table 2-6: ECCR ................................................................................................................... 12
Table 2-7: Baud Rate corresponding to CLKP1.................................................................... 13