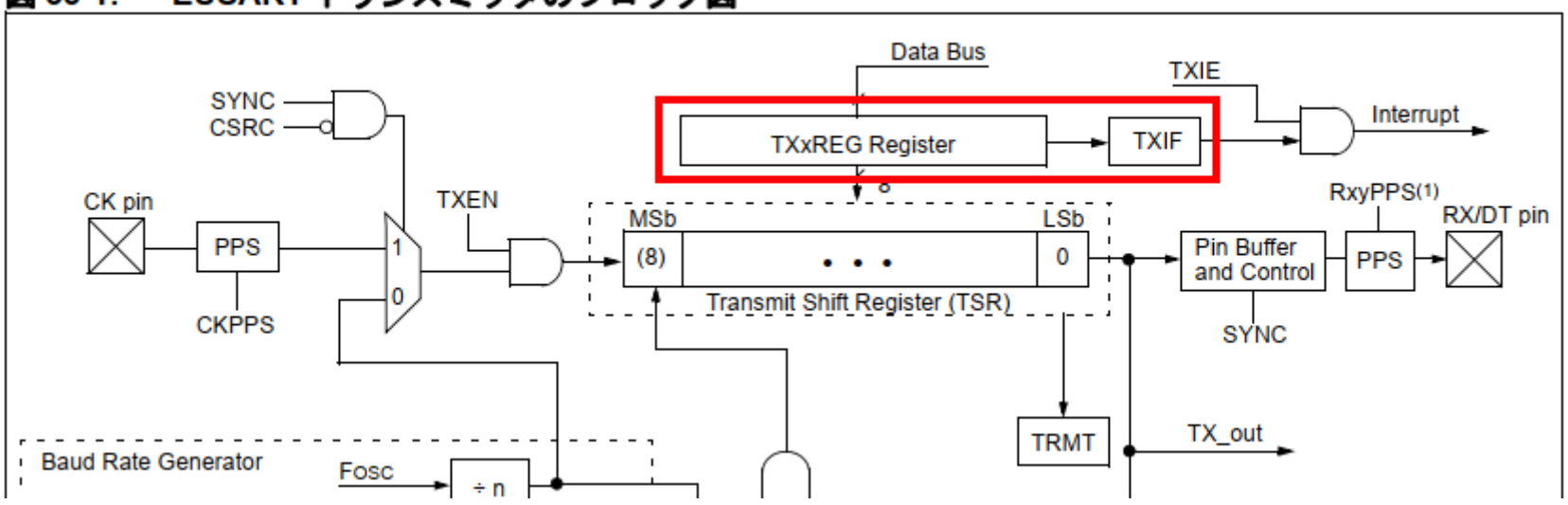
Why？ UART(TX)

TXIF（割り込みフラグ）が Read Only。
TXIE（割り込み許可）の状態に関係なく TXREG（送信バッファ）が空なら TXIF は立ちっぱなし。

図 33-1:　EUSART トランスミッタのブロック図



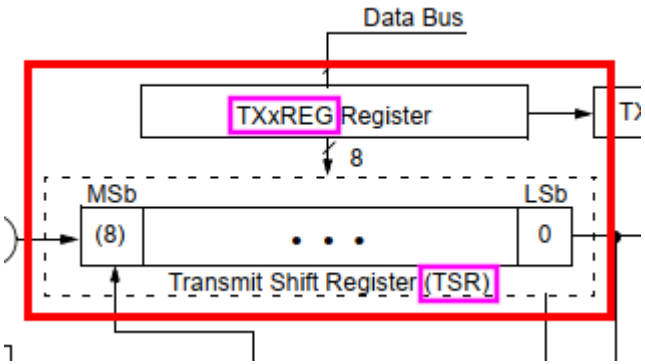
Because UART(TX)

UART(TX)にとってまずい状況とは・・・

TXREGに次に送る１バイトが存在するか、
TXREG→TSRデータ転送中に、
つまりTXREGが空ではないときに

TXREGにデータを書きに行くとまずい。
これ以外の状況ならTXREGにデータを書きにいくのは問題ない。

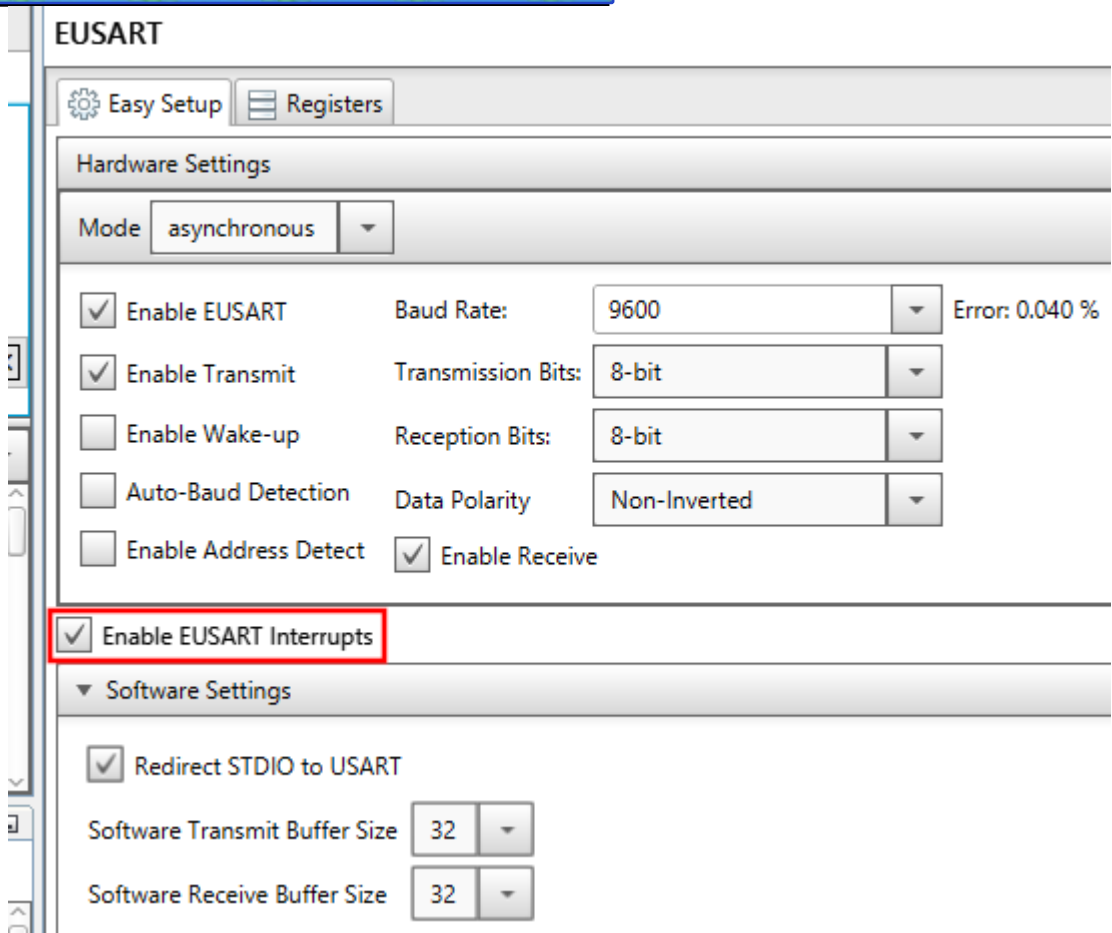
TXREGが空になるタイミングでTXIFが立てば割り込み処理で連続送信ができて便利。



Why？ UART(TX) again

Easy Setup で割り込み送信受信が別々になっていないのは何故か。

受信割り込みのつもりでチェックすると送信も割り込みにされてしまう。（害はないのか）



ソースコードを見に行く（1）

TXREGが空になるタイミングでTXIFが立つので、送信データがある間、TXIEを１にすることでバッファが使える。

TRMT（送信ビジー）を待たないのでメインループの周期に影響が出ない。（割り込みのメリット）

```
173  void EUSART_Write(uint8_t txData)
174  {
175      while(0 == eusartTxBufferRemaining)
176      {   // 用意されたバッファ（デフォルトで 8 byte）の残りバイト数が
177      }   // ゼロならここで処理を止めて待つ
178
179      if(0 == PIE3bits.TXIE)  // データ転送中でない場合は TX1REG に書いていい
180      {                       // 割り込み処理側で残りバイト数全回復で割り込み禁止する
181          TX1REG = txData;    // 転送中でなければレジスタに直書き
182      }                       // メッセージの最初の文字でここに入ることになる
183      else
184      {   // データ転送中（TX1REG から TSR にビット転送の最中）の場合
185          PIE3bits.TXIE = 0;  // 割り込み処理内でバッファにアクセスするので一旦止める
186          eusartTxBuffer[eusartTxHead++] = txData;    // バッファにデータを置く
187          if(sizeof(eusartTxBuffer) <= eusartTxHead)
188          {
189              eusartTxHead = 0;   // データ置き位置の循環処理
190          }                       // どこから置き始めても残りバイト数は変わらない
191          eusartTxBufferRemaining--;  // 残りバイト数の減算
192      }
193      PIE3bits.TXIE = 1;  // この関数が呼ばれたら必ず送信割り込みが発動
194  }
195
```

ソースコードを見に行く（2）

TXREG から TSR への転送が終わるごとに割り込みが発生するのでバッファの残り文字数をチェックして割り込みの継続を決める。

MCC で UART 割り込みを使うにチェックし、TXIF が書き込み可能だと都合が悪い。

```
206  void EUSART_Transmit_ISR(void)
207  {
208      // TX1REG が空になるタイミングで TXIF が立つ
209      // add your EUSART interrupt custom code
210      if(sizeof(eusartTxBuffer) > eusartTxBufferRemaining)
211      {   // 最初の1文字（バッファを使用していない）と
212          // 最後の1文字（バッファが全回復する）を除外している
213          // つまり、バッファが1文字でも使われたらここに入る
214          TX1REG = eusartTxBuffer[eusartTxTail++];
215          if(sizeof(eusartTxBuffer) <= eusartTxTail)
216          {
217              eusartTxTail = 0;
218          }
219          eusartTxBufferRemaining++;
220      }
221      else
222      {   // バッファが使われていない状態になったら割り込み禁止
223          PIE3bits.TXIE = 0;
224      }
225  }
```

ソースコードを見に行く（3）

MCC が出力する printf 用の低レベル関数もこれを呼ぶだけ。TRMT をチェックしない。ユーザー側でこの関数を呼ぶときにも連続して呼べる。

```
199     }
200
201     void putch(char txData)
202     {
203         EUSART_Write(txData);
204     }
205
206     void EUSART_Transmit_ISR(void)
```

大全の低レベル関数はこちら。

```
33      }
34      /**********************************
35       *    低レベル入出力関数の上書き
36       **********************************/
37      void putch(char Data){
38          while(!TX1STAbits.TRMT);     // 送信レディ待ち
39          TX1REG = Data;               // データ送信
40      }
41      char getch(void){
42          while(PIR3bits.RCIF == 0);
43          return(RC1REG);
44      }
45      /**********************************
```