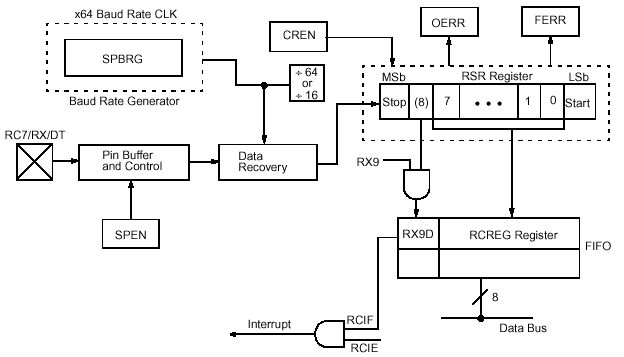
**http://www.picfun.com/graphics/greenball.gifUSARTの使い方**

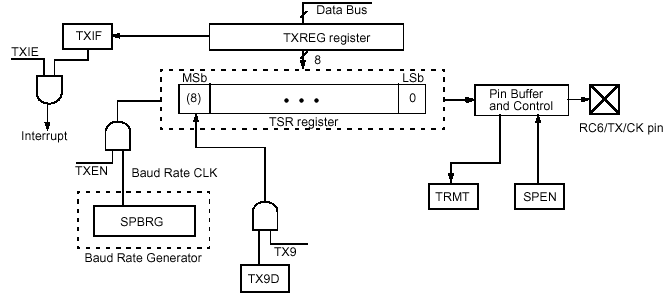
**【USART】**

USARTはシリアル通信を行うためのハードウェアモジュールで、高速のシリアル  
通信を行うことが可能です。  
PIC16シリーズのUSARTと基本的には同じ機能のものですが、PIC18シリーズで  
は１つだけ機能が拡張されました。  
それはアドレス選択機能が付加されたことです。これはもともとPIC16シリーズでも  
あった機能なのですがあまり使い勝手がよくありませんでした。PIC18シリーズに  
なってこの機能が本格的に使用できるようになりました。  
  
USART：Addressable Universal Synchronous　Asynchronous Receiver　Transimitter

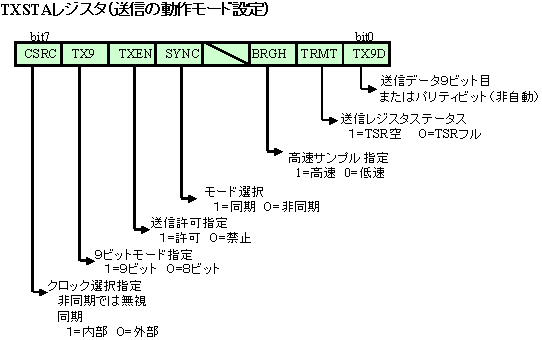
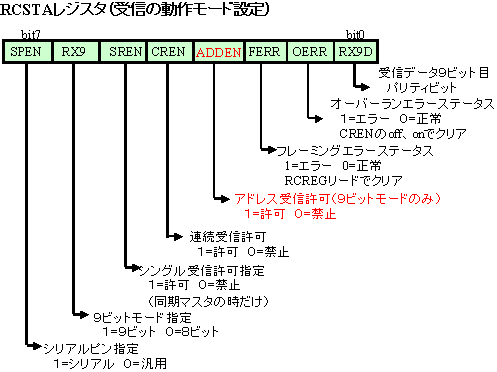
**【USART内部構成】**

USARTの受信部の回路構成は下図のようになっています。ここで特徴的なのは  
９ビットモードの動作です。つまり、TX9ビットが１になっていると９ビットのデータを  
受信するようになります。そして９ビット目の０，１によりその時の８ビットデータが、  
データかアドレスかを区別出来るようになります。  
さらに、ADDENビットを１にしておくと、９ビット目が１の時だけ、つまりアドレスの  
データ受信の時にだけ受信動作をするようになります。このアドレス受信をして  
アドレス一致をソフトウェアで検出し、ADDENを０にしてデータ受信を許可するまで、  
データ受信を受け付けないようになります。



送信部の方の構成も下図のようになっていてやはり９ビットモードのデータ送信  
がやり易くなっています。  
つまり、TX9ビットをセットしておけば常時９ビットモードとなって送信動作を行います。  
９ビット目のデータをTX9Dビットにセットすることで、アドレスモードとデータモードを  
簡単に切り替えられます。  
  


**【制御レジスタの内容】**

USARTを使うためには下記の３個のレジスタの設定が必要です。  
  
(1) TXSTAレジスタ  
　　送信部分の動作モードを設定します。  
(2) RCSTAレジスタ  
　　受信部分の動作モードを設定します。  
(3) SPBRGレジスタ  
　　通信速度を設定します。（ボーレートジェネレータ）  
  
まずTXSTAレジスタは下図のような構成になっていて、PIC16と変更部分はありません。  
非同期モードで使う時にはSYNCを0にします。アドレス選択機能を使う時には、TX9で  
９ビットモードを指定し、TX9Dに９ビット目のデータを設定します。残りの８ビットのデータ  
は上図のTXREGに書き込みます。  
  
  
  
  
  
  
  
受信の方の設定では、非同期にするのはTXSTAで設定されているので連続受信を  
許可します。アドレス選択機能を使う時には、やはりRX9を１にしてやると、受信した  
９ビット目のデータがRX9Dで取り出すことが出来ます。残りの８ビットのデータは  
上図のRCREGから取り出せます。受信中のエラーチェックもRCSTAレジスタのビット  
で確認することが出来ます。  
  
  
  
  
  
  
**SPBRGレジスタの設定方法**　　通信速度を決めるのは、SPBRGレジスタによるボーレートジェネレータが  
　　制御しています。このSPBRGレジスタへに設定する数値とボーレートの  
　　関係は下表のようになります。但し、下表は調歩同期式の場合だけです。  
　　計算式は下記となっています。  
  
　　　X = （Fosc／通信速度／６４）－１　（BRGH=0の時）  
  
　　　X = （Fosc／通信速度／１６）－１　（BRGH=1の時）  
  
  
《表》　**BRGH=0の時の通信速度とSPBRGの値**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| クロック  ボーレート | ４０MHｚ | | ２０MHｚ | | １０MHｚ | |
| SPBRG 設定値 | エラー レイト | SPBRG 設定値 | エラー レイト | SPBRG 設定値 | エラー レイト |
| 1200 |  |  |  |  | 129 | 0.16 |
| 2400 |  |  | 129 | 0.16 | 64 | 0.16 |
| 9600 | 64 | 0.16 | 32 | -1.36 | 15 | 1.73 |
| 19.2K | 32 | 1.36 | 15 | 1.73 | 7 | 1.73 |
| 76.8K | 7 | 1.73 | 3 | 1.73 | 1 | 1.73 |

　　（注）エラーレイトとは、ボーレート周波数のズレの多さを％であらわしたもので  
　　　　　１フレーム内でのズレになりますが、１０ビット（スタートストップ合わせて）  
　　　　　のフレームでのエラーマージンは、１／２ビット幅までのズレを許容すると  
　　　　　すれば、５０％／１０＝５％となります。

**【アドレス選択機能の使い方】**

アドレス選択機能とは、シリアル通信で、１台のマスターに複数のスレーブが同じ線で  
接続されても、マスター側からアドレス指定をすることで、特定のスレーブを指定して  
マスターから送信できるようにする機能です。  
これで、いわゆる「ポーリング方式」により、マスターが中心となって順次スレーブに  
送信するデータがあるかを問い合わせをすれば、複数のスレーブがあっても衝突せず  
にスレーブからマスターへデータを送信することが出来ます。  
  
アドレス選択方式を活用するためには下記手順でスレーブ側を動作させます。  
  
(1) 適当な通信速度になるようSPBRGレジスタを設定する  
(2) 非同期（調歩同期）に設定する、SYNC=0、SPEN=1  
(3) 割込みを使う時は、RCIE=RCIP=１にする。  
(4) ９ビットモードにする。（RX9=1)  
(5) アドレス検出モードにする。（ADDEN=1)  
(6) 連続受信可能にする。(CREN=1)  
(7) RCIFフラグが１になるのをチェックするか、割込みにより受信検知する。  
(8) RCSTAレジスタのRX9が１であることを確認することでアドレス受信を確認する。  
(9) ８ビットのデータを取り出し、アドレス比較をする。  
(10)ADDEN=0にすると次からはデータの受信状態となる。  
(11)以降のデータを受信する。  
(12)一定のバイト数か特定のデータ受信でデータ受信を完了しADDEN=1に戻す。

**【C言語ライブラリ関数】**

MPLAB-C18に用意されているUSARTライブラリ関数は下記となっています。  
全部を使うことは必要無いですが、比較的簡単にUSARTが扱える様になっています。  
  
**(1) ライブラリ関数一覧**

|  |  |  |
| --- | --- | --- |
| 関数名 | 機能と基本形 | 使用例 |
| BusyUSART | 送信中の時１を返す TXSTAレジスタのTRMTフラグ状態を返す | while(BusyUSART( )); |
| char BusyUSART(void); |
| CloseUSART | 送受信を終了し割り込みも禁止する | CloseUSART( ); |
| void CloseUSART(void); |
| DataRdyUSART | 受信バッファに受信完了したデータがあるとき １を返す。 PIRレジスタのRCIFフラグ状態を返す | while(!DataRdyUSART( )); |
| char DataRdyUSART(void); |
| getsUSART | lenで指定された文字数の文字列を連続して受信 しbufferで ポイントされたRAMメモリに格納する。 bufferはlen+1のエリア が必要。 永久に受信完了を待ちタイムアウトは無い。 | char x[10]; getsUSART(x,5); |
| void getsUSART(char \*buffer, unsigned char len); |
| OpenUSART | USARTの使用モードを設定し動作を開始させる。 設定内容は、割込み、通信速度、同期/非同期、 8ビット/９ビット、 マスター/スレーブ、単発/連続受信 | OpenUSART(USART\_TX\_INT\_OFF & USART\_RX\_INT\_OFF & USRT\_ASYNC\_MODE & USART\_EIGHT\_BIT & USART\_CONT\_RX  & USART\_BRGH\_HIGH, 25); |
| void OpenUSART(unsigned char config, char spbrg); |
| putsUSART putsrUSART | RAM、ROMエリアの連続データを送信する。 終了はnullで判定 | char mybuff[20]; putsUSART(mybuff); |
| void putsUSART(char \*data); void putrsUSART(const rom char \*data); |
| ReadUSART | 受信バッファより１個のデータを取り出し返す。 ９ビットモードにも対応し、 USART\_Status.RX\_NINEに格納 | char x; x = ReadUSART( ); |
| char ReadUSART(void); |
| WriteUSART | 送信バッファに１個のデータを書く、９ビットモード にも対応。 ９ビット目のデータは USART\_Status.TX\_NINEにセットする。 | char x; WriteUSART(x); |
| void WriteUSART(char data); |

**(2) OpenUSART用パラメータ**  
　　OpenUSART関数にはUSARTの動作モードを設定するための多くのパラメータが  
　　必要となりますが、これらは下記のようになっています。

|  |  |  |
| --- | --- | --- |
| 設定モード | パラメータ名称 | 意味内容 |
| 割込みの許可禁止 | USART\_INT\_TX\_ON USART\_INT\_TX\_OFF USART\_INT\_RX\_ON USART\_INT\_RX\_OFF | 送信割込みの許可 　　　　　　　　　禁止 受信割込みの許可 　　　　　　　　　禁止 |
| 同期／非同期 | USART\_ASYNCH\_MODE USART\_SYNCH\_MODE | 非同期（調歩）モード 同期モード |
| ８／９ビットモード | USART\_EIGHT\_BIT USART\_NINE\_BIT | ８ビットモード ９ビットモード |
| スレーブ／マスタ | USART\_SYNC\_SLAVE USART\_SYNC\_MASTER | 同期スレーブ 同期マスタ |
| 単発／連続受信 | USART\_SINGLE\_RX USART\_CONT\_RX | 単発受信モード 連続受信モード |
| 高速／低速速度 | USART\_BRGH\_HIGH USART\_BRGH\_LOW | 高速ボーレート 低速ボーレート |

**(3) USARTのステータスのユニオン定義**  
　　USARTのステータス関連の情報はユニオンとして定義されています。その定義  
　　内容は下記となっており、USART\_Status　という変数名となっています。  
  
　　　union USART  
　　　{  
　　　　　unsigned char val;  
　　　　　struct  
　　　　　{  
　　　　　　　unsigned RX\_NINE:1;  
　　　　　　　unsigned TX\_NINE:1;  
　　　　　　　unsigned FRAME\_ERROR:1;  
　　　　　　　unsigned OVERRUN\_ERROR:1;  
　　　　　　　unsigned fill:4;  
　　　　　};  
　　　};

**【プログラム例】**

このUSARTのライブラリ関数を使って実際に作成したプログラム例が下記です。  
この例では、RS232CでパソコンなどとPICを接続し、パソコン側から送られて来た  
１文字のデータをパソコンに送り返すと同時に、液晶表示器に表示します。  
送られて来た文字が「Q」の時だけ、液晶表示器をクリアして「Restrat」という  
メッセージを表示します。同時にパソコン側にも同じメッセージを送信します。  
使用したハードウェアは製作例で紹介しているPIC１８評価基板です。  
  
下記テストプログラムの全体は下記でダウンロードあるいは見る事が出来ます。  
  
　　　[**★ PIC18シリーズUSARTテストプログラム**](http://www.picfun.com/pic18/18ctest5.c)  
  
  
《例》　メインプログラムの部分のみ（漢字スペースを含む）  
  
**//\*\*\*\*\*\* Main Function  
void main(void)  
{  
　　　char Message1[8]="Start!!";  
　　　char Message2[8]="Restart";  
　　　char data;  
  
　　　TRISB = 0;  
　　　OpenXLCD(FOUR\_BIT);  
　　//USARTの初期設定  
　　　OpenUSART(USART\_TX\_INT\_OFF & USART\_RX\_INT\_OFF&   
　　　USART\_ASYNCH\_MODE & USART\_EIGHT\_BIT&  
　　　USART\_CONT\_RX & USART\_BRGH\_LOW, 15); //9600 bps  
　　//開始メッセージ表示、送信  
　　　putsXLCD(Message1);  
　　　putsUSART(Message1);  
  
　　　while(1)  
　　　{  
　//データ受信待ちと返送  
　　　　　　while(!DataRdyUSART());{  
　　　　　　data = ReadUSART();  
　　　　　　WriteUSART(data);  
　　　}  
　//文字がQだったら表示クリア  
　　if(data == 'Q')  
　　　{  
　　　　　　WriteCmdXLCD(0x01); //clear  
　　　　　　while(BusyXLCD());  
　　　　　　putsXLCD(Message2);  
　　　　　　putsUSART(Message2);  
　　　}  
　　　else  
　　　　　　WriteDataXLCD(data);  
　　　}  
}**