Ecco le considerazioni:
Negli Example C HITEC la funzione di Trasmissione di un carattere è proposta in due modi:
MODO(1)

```
void  putch(unsigned char byte)  {
      while(!TRMT)      /* set whilst TX in progress */
            continue;
      TXREG = byte;
}
```

OPPURE
MODO(2)

```
void  putch(unsigned char byte)  {
      while(!TXIF)
            continue;
      TXREG = byte;
}
```

Dopo averle usate alternativamente ho riscontrato anomalie varie. Queste anomalie avvenivano se trasmettevo più di un carattere in sequenza come:

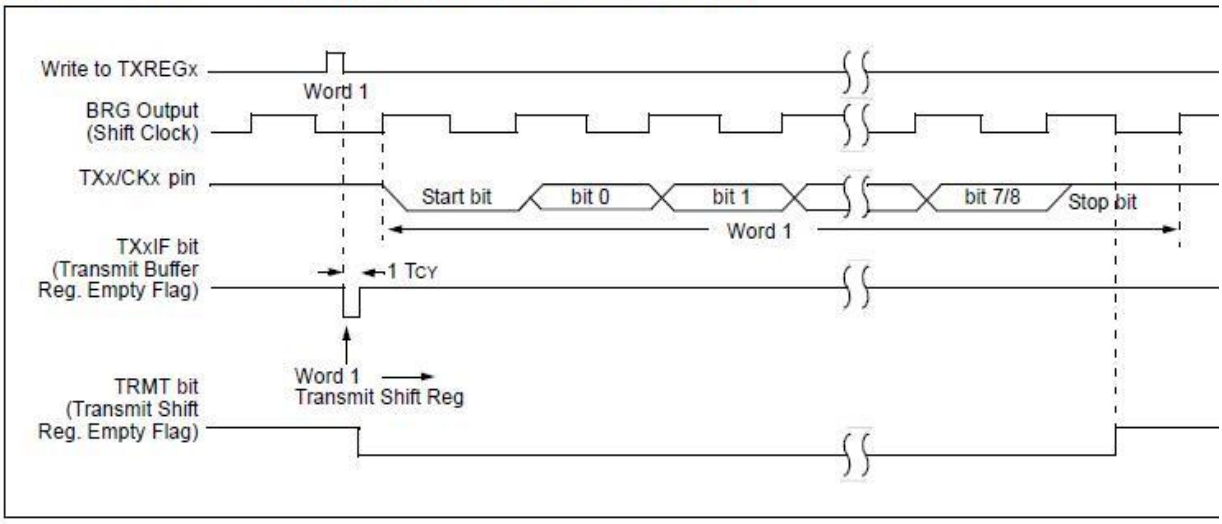
```
putch('A');
putch('B');
```

spesso non funzionava ed era necessario inserire un ritardo (variabile a seconda del BAUD rate usato) del tipo:

```
putch('A');delay(xx);
putch('B');delay(xx);
```

Ho letto con più attenzione il data sheet dei PIC (o almeno spero di aver capito l'inglese) e mi è sorto un dubbio. Vi mostro sotto i diagrammi di trasmissione:

FIGURE 16-3: ASYNCHRONOUS TRANSMISSION



Sia usando la funzione CASO(1) che CASO(1) il dato viene trascritto in TXREG appena TXIF o TMRT diventano bassi. In sostanza la funzione termina appena il dato è scritto in TXREG ma tale dato deve ancora essere trasmesso passando sullo shift REGERTER TRS.
Se il programma sta ancora trasmettendo e io invio un secondo carattere che succede ?

Ecco la modifica proposta per la funzione alla luce dei segnali mostrati dalla figura:

```
void  putch(unsigned char byte)  {
```

```
      while(!TXIF)                        // identico (!TRMT)
            continue;
      TXREG = byte;
      while(!TRMT) continue;              // se TRMT è BASSO (trasmissione non
                                          //terminata) non esce dalla  funzione
}
```

In questo modo la necessità di inserire delay() dopo il putch() è scomparsa e così mi pare anche le anomalie che riscontravo.

Chiedo conforto agli esperti!
Grazie