

```

+-----+
| ME_CBuffer - Buffer circolare software |
+-----+
| Versione: 0.12 Beta |
| Revisione Documento: 2014-10-30 (2013-08-14) |
| Autore: Roberto Gatti |
| Tipo: Componente PSoC |
+-----+
|                                     (c) 2014 Movie Engineering Srl |
+-----+

```

INTRODUZIONE

=====

Implementa un buffer FIFO circolare in cui si mettono e prendono byte senza doversi preoccupare dei contatori di fine e inizio buffer.

Vengono generati errori di overflow e underflow se si preleva da un buffer vuoto o inserisce in uno pieno.

L'array dove vengono memorizzati viene dimensionato in base ad un parametro del componente.

Si vedano i sorgenti della libreria ME_lib_cbuffer.* per informazioni più dettagliate.

PARAMETRI DEL COMPONENTE PSoc

=====

CBUFFER_SIZE : dimensioni dell'array di byte esterno dove vengono memorizzati i dati

DESCRIZIONE API

=====

Il componente può essere usato in più istanze e come sotto-componente.

In presenza di errori il componente deve essere re-inizializzato con _Init per poter essere utilizzato di nuovo.

Funzioni Esportate

```

_Init()           : inizializza il buffer
_PutByte( byte )  : inserisce un byte nel buffer
_GetByte( )       : preleva un byte dal buffer
_AnyData( )       : numero di byte nel buffer (0 se è vuoto)
_Status( )        : ok, underflow, overflow
_ERROR( )         : vera se c'è stato un errore, falsa se tutto ok

```

Macro e altre defines utili

La funzione _Status restituisce lo stato interno del buffer che può assumere i seguenti valori:

```

CBUFFER_NOERROR      (0x00)  Tutto ok
CBUFFER_OVERRUN      (0xFF)   Si è tentato di scrivere in un buffer pieno
CBUFFER_UNDERRUN      (0xEE)   Si è tentato di leggere da un buffer vuoto

```

ESEMPIO DI UTILIZZO

=====

Esempio di buffer circolare usato per memorizzare, in un interrupt di ricezione, tutti i byte in arrivo dalla seriale, senza doversi preoccupare di resettare il buffer di ricezione

perché man mano che i byte vengono letti (FIFO) si libera spazio circolarmente.

```
void `${INSTANCE_NAME}`_InterruptServiceRoutine(void)
{
    BYTE status ;

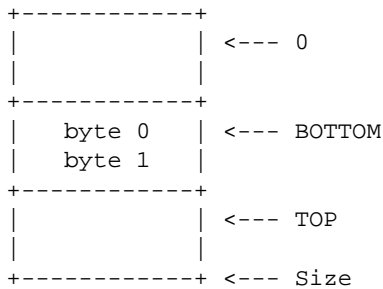
    status = `${INSTANCE_NAME}`_UART_RXSTATUS_REG ;

    if (status & `${INSTANCE_NAME}`_UART_RX_STS_FIFO_NOTEMPTY)
    {
        `${INSTANCE_NAME}`_CBUFFER_PutByte(`${INSTANCE_NAME}`_UART_RXDATA_REG) ;
    }
    else
    {
        `${INSTANCE_NAME}`_serial_errors_count ++ ;
    }
}
```

MECCANISMI O PRINCIPI GENERALI DI FUNZIONAMENTO
=====

Internamente è implementato con due puntatori di top e bottom, quando si inserisce viene incrementato top e quando si preleva viene incrementato bottom. Entrambi i puntatori ciclano a zero una volta raggiunta la fine dell'array.
Top indica sempre la prima posizione vuota, Bottom indica sempre la posizione del byte inserito da più tempo.

Caso semplice:



Caso complicato:

