rbuffer.h, un buffer tournant

Bernard Tatin

2013/2017

Voici un premier essai de *literate programming*, concept inventé par D. Knuth il y a plus de trente ans. À partir de ce seul fichier on génère la documentation et le code. Ici, je reprend du vieux code, cela m'oblige, même s'il est simple, à le repenser et donc, espérons le, à l'améliorer.

Contents

1 rbuffer

C'est un buffer tournant le plus simple possible, capable de gérer des lignes délimitées par *LF* (`'\n'`) mais *CR* (`'\r'`) n'est pas pris en compte.

1.1 premières définitions

Pour limiter les calculs, le code..., la taille du buffer est une puissance de 2 d'où la définition du nombre de bits qui ouvre le bal :

1 ⟨*intro-bits* 1⟩≡
```
#define _RBUFFER_BITS   8
#define RBUFFER_SIZE    (1 << _RBUFFER_BITS)
#define RBUFFER_MASK    (RBUFFER_SIZE - 1)
```

This code is used in chunk 3.
Defines:
_RBUFFER_BITS, never used.
RBUFFER_MASK, never used.
RBUFFER_SIZE, used in chunk 2a.

1.2 la structure

Note: tous les membres de la structure sont définis comme `volatile`. C'est important dans un système embarqué avec des interruptions pouvant manipuler le buffer. Sans `volatile`, une optimisation trop agressive pourrait placer une des valeurs entières dans un registre. En cas d'interruption modifiant cette valeur, le registre, lui, ne bougera pas et des caractères pourraient se perdre.

2a ⟨*tsrbuffer* 2a⟩≡
```
/**
 * @struct TSrbuffer
 * La structure gérant le buffer tournant.
 */
typedef struct {
    volatile int in;
    volatile int out;
    volatile int line_count;
    volatile char buffer[RBUFFER_SIZE];
} TSrbuffer;
```
This code is used in chunk 3.
Defines:
TSrbuffer, never used.
Uses RBUFFER_SIZE 1.

1.2.1 les champs

1.2.2 remarques diverses

On pourrait définir un `VOLATILE` en fonction de l'architecture du type :

2b ⟨*define-volatile* 2b⟩≡
```
#if defined(__with_irqs)
  #define VOLATILE volatile
#else
  #define VOLATILE
#endif
```

Root chunk (not used in this document).

1.3 le fonctionnement

1.3.1 ajout d'un caractère

Le fonctionnement est le suivant pour l'ajout d'un caractère :

- on place le caractère dans le buffer à la position `in`,
- on incrémente `in`,
- si on atteint la limite du buffer, on positionne `in` à 0,
- si le caractère est '\n', on incrémente `line_count`.

1.4 le code final

3 ⟨* 3⟩≡
⟨*intro-bits* 1⟩
⟨*tsrbuffer* 2a⟩

Root chunk (not used in this document).

2 annexes

2.1 extraits de code

⟨* 3⟩ 3
⟨*define-volatile* 2b⟩ 2b
⟨*intro-bits* 1⟩ 1, 3
⟨*tsrbuffer* 2a⟩ 2a, 3

2.2 index