rbuffer.h, un buffer tournant

Bernard Tatin

2013/2017

Abstract

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⟩≡ (3b)

```
#define _RBUFFER_BITS   8
#define RBUFFER_SIZE    (1 « _RBUFFER_BITS)
#define RBUFFER_MASK    (RBUFFER_SIZE - 1)
```

Defines:
_RBUFFER_BITS, never used.
RBUFFER_MASK, never used.
RBUFFER_SIZE, used in chunk 2.

1.2 la structure

On note que tous les membres de la structure sont définis comme `volatile`. C'est important dans un système embarqué avec des interruptions pouvant manipuler le buffer, cela empêche des boucles d'être optimisées au point de ne plus lire la valeur contenue dans la structure pour la stocker dans un registre. Si une interruption modifie une de ces valeurs, une optimisation trop agressive ne permettra pas d'en tenir compte.

2 ⟨*tsrbuffer* 2⟩≡ (3b)

```
/**
 * @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;
```

Defines:
TSrbuffer, never used.
Uses RBUFFER_SIZE 1.

1.2.1 les champs

1.2.2 le fonctionnement

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`.

Pour lire un caractère, c'est un peu plus compliqué, il faut s'assurer qu'il y en a au moins un de présent.

1.2.3 remarques diverses

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

3a ⟨*define-volatile* 3a⟩≡
```
#if defined(__with_irqs)
  #define VOLATILE volatile
#else
  #define VOLATILE
#endif
```

1.3 le code final

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

1.4 exrtaits de code

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

1.5 index