Les CallBacks en C (fonctions de rappel)

Un callback est une fonction d'un programme qui est appelée par un programme extérieur (système d'exploitation, autre programme, …).

Programme applicatif : programme utilisateur qui repose sur un OS.

OS : On considère que l'OS est un programme qui, sur base d'évènements extérieurs et intérieurs au microcontrôleur, va appeler les fonctions du programme applicatif.

Le but final est de créer un mini-OS afin que le programme applicatif ne s'occupe plus des aspects matériels (timers, interruptions, …).

Main.c

```
#include


// VARIABLES GLOBALES


// FONCTIONS DONT ON PASSE LES ADRESSES A callbacks.c  POUR QU'ELLES PUISSENT
// ÊTRE RAPPELEES TOUTES LES X MS
void fonction1(void) ;
void fonction2(void) ;
void fonction3(void) ;


// FONCTION PRINCIPALE
int main(void)
{
        // INITIALISATION HARDWARE
        hardware_Init(void) ;
```

```
        //INITIALISATION CALLBACKS
        Callbacks_Init() ;
        ID_Fct1 = Callbacks_Record_Timer(fonction1, 500) ;
        ID_Fct2 = Callbacks_Record_Timer(fonction3, 100) ;

        Callbacks_Start() ;
        // N'ARRIVE JAMAIS ICI
}

// CONTENU DES FONCTIONS CALLBACKS
void fonction1(void)
{
}
void fonction2(void)
{
}
void fonction.3(void)
{
}

// CONTENU DES AUTRES FONCTIONS
hardware_Init(void) ;
```

Callbacks.h

```
// DEFINE

#define MAX_CALLBACKS          10


// PROTOTYPE DES FONCTIONS EXTERNES

        //INITIALISATION DES CALLBACKS

        void  Callbacks_Init(void) ;

        //ENREGISTRER DES FONCTIONS CALLBACKS LIEES AU TEMPS

        // RETOURNE UN ID ASSOCIE A L'ENREGISTREMENT

        unsigned char Callbacks_Record_Timer(void(*pt_Function)(void), unsigned int Time);


        // RETIRER DES FONCTIONS CALLBACKS LIEES AU TEMPS, PREND L'ID DU CALLBACK

        // COMME ARGUMENT

        void Callbacks_Remove_Timer(unsigned char ID_CB) ;


        //DEMARRAGE DE LA BOUCLE PRINCIPALE

        void callbacks_Start(void) ;
```

Callbacks.c

```
#include


// VARIABLES GLOBALES
unsigned char tmp_int;                    // pour l'interruption du timer
// VARIABLES POUR CALLBACKS TIMER
void (*My_CB[MAX_CALLBACKS])(void);
unsigned int Time_CB[MAX_CALLBACKS];
unsigned int TICK_CB[MAXCALLBACKS];


// DECLARATION DES FONCTIONS INTERNES
void Init_Timer(void)


// CONTENU DES FONCTIONS  EXTERNES
void  Callbacks_Init(void) ;
{
        unsigned char i ;
        // INITIALISATION POUR VARIABLES CALLBACKS TIMER
        for (i = 0 ; i < MAX_CALLBACKS; i++)
        {
                My_CB[i] = 0;
                Time_CB[i] = 0;
        }
// INITIALISATION DU LCD


// CONFIGURATION USART POUR INTERRUPTION RD
}
```

```
// ENREGISTREMENT CALLBACKS TIMER
unsigned char Callbacks_Record_Timer(void(*pt_Function)(void), unsigned int Time)
{
	unsigned int i = 0;
	while (My_CB[i] != 0 && i < MAX_CALLBACKS) i++;
	if (i  < MAX_CALLBACKS)
	{
		My_CB[i] = pt_Function;
		Time_CB[i] = Time;
		TICK_CB[i]= 0;
		return i;
	}
	// IL N'Y A PLUS DE PLACE POUR ENREGISTRER UN CALLBACK
	else return 255;
}
// ENREGISTREMENT CALLBACK USART
void Callbacks_Record_USART(void(*pt_Function)(char))
{
	My_CB_USART = pt_Function;
}

// RETIRER FONCTION DE RAPPEL
void Callbacks_Remove_Timer(unsigned char ID_CB)
{
	if ((ID_CB > 0 && ID_CB < MAX_CALLBACKS);
	{
		My_CB[ID_CB] = 0;
```

```
                Time_CB[ID_CB] = 0;
        }
}
// BOUCLE PRINCIPALE DE L'OS
void callbacks_Start(void)
{
        unsigned char idx;

        // INITIALISATION DE TOUTES LES INTERRUPTIONS

        // CONFIGURATION TIMER
        void Init_Timer() ;

        // BOUCLE INFINIE
        while (1)
        {
                // CHECK LES CONDITIONS POUR RAPPELER LES FONCTIONS LIEES  AUTEMPS
                for (idx = 0 ; idx < MAX_CALLBACKS; idx++)
                {
                        if (My_CB[idx] if (TICK_CB[idx] >= Time_CB[idx])
                                        {
                                                TICK_CB[idx]      = 0;
                                                My_CB[idx]();
                                        }
                }
        }
}
```

```
// CONTENU DES FONCTIONS INTERNES

void Init_Timer()

{



}


// INTERRUPTIONS

// INTERRUPTION TIMER

{

        // AJOURNER TOUS LES TICKS

        for ( tmp_int = 0 ; tmp_int < MAX_CALLBACKS; tmp_int++) TICK_CB[tmp_int]++;

}
```