Auteur de la documentation

Alexandre Colot, K-Team S.A.
Ch. de Vuasset, CP 111
1028 Préverenges
Suisse

email : info@hemisson.com
Url : www.hemisson.com

Préliminaires :

- Le contenu de ce manuel est susceptible de changer sans préavis.
- Tous les efforts ont été fait afin d'assurer l'exactitude de ce manuel. Si toutes fois vous trouvez une anomalie, merci d'en informer K-Team S.A.
- Malgré les commentaires ci-dessus, K-Team S.A. ne pourra nullement être tenu pour responsable d'eventuelles erreurs dans ce document.

Notations :

- int1 : variable de type bit
- int16 : variable entière 16 bits

Remarque :

- Ce manuel est valable pour tous les HemiOs 1.3X

Table des matières

1 Introduction

Hemisson est équipé d'un processeur Microchip PIC16F877.
Ce processeur a la particularité d'être utilisé dans de très nombreuses applications, ainsi il existe déjà une forte communauté avec de nombreux d'utilisateurs.
Microchip fournit gratuitement un compilateur assembleur pour les PICs, mais ce langage de très bas niveau n'est pas tout à fait adapté au développement rapide de comportement en robotique, il s'agit plutôt d'un langage spécifique pour les couches de bas niveau.
C'est ainsi que des compilateurs pour des langages de plus haut niveau sont apparus sur le marché : Basic, Pascal, C, C++, JAL ...
Nous allons vous présenter dans ce manuel le cas particulier du langage C qui est à notre sens le plus approprié.
En effet, l'ensemble de l'HemiOs (système d'exploitation embarqué dans Hemisson) a été développé dans ce langage et nous vous livrons les codes source sous licence LGPL. C'est à dire que vous pouvez l'utiliser gratuitement et sans contraintes (sauf à des fins commerciales).
Ainsi nous espérons que Hemisson pourra bénéficier des améliorations que chacun d'entre vous aura réalisées et donc progresser rapidement un peu comme cela s'est fait avec Linux.
Nous vous serons donc reconnaissant, de nous faire parvenir par e-mail (hemios@hemisson.com), les améliorations et les corrections que vous aurez apportées au code original.
Vous touverez également sur le site Hemisson (http ://www.hemisson.com) les dernières versions de l'HemiOs à télécharger.
Voyons maintenant plus en détails les aspects de programmation.

PS : Ce document s'adresse à des personnes ayant déjà quelques notions de base de programmation en langage C, il n'est en aucun cas un cours.

2 Le compilateur

Il existe aujourd'hui sur le marché de nombreux compilateur C pour PICs, plus ou moins onéreux mais aussi plus ou moins fiables.
Après avoir testé plusieurs d'entres-eux, nous avons choisi CCSC pour son rapport qualité, support, prix et son ouverture multiplateformes.

Les différentes versions actuelles de CCSC sont :
- PCB : Compilateur C, PICs 12 bits par commande de ligne sous Windows
- PCM : Compilateur C, PICs 14 bits par commande de ligne sous Windows
- PCH : Compilateur C, PIC18 par commande de ligne sous Windows
- PCW : IDE + PCB + PCM sous Windows
- PCWH : IDE + PCB + PCM + PCH sous Windows
- PCBL : Compilateur C, PICs 12 bits par commande de ligne sous Linux
- PCML : Compilateur C, PICs 14 bits par commande de ligne sous Linux
- PCHL : Compilateur C, PIC18 par commande de ligne sous Linux

Suite à des accords commerciaux avec CCS, vous avez à présent la possibilité d'obtenir une version particuliére de PCW (limitée aux PICs 16F877 et 16F876) à un prix exceptionnel, sur le site web hemisson dans la section logiciels (Url : http ://www.hemisson.com/French/logiciels.html).

Nous ne fournissons pour le moment aucun support pour un autre compilateur C. Si vous disposez déjà d'un autre compilateur C pour PIC, il vous faudra adapter le code de l'HemiOs.

3 Utilisation de CCSC

Les étapes suivantes sont nécessaires à l'installation de CCSC :

- Créez un répertoire PICC dans le répertoire Program Files.
- Copiez dans ce répertoire les fichiers de licence que vous avez recus (.crg).
- Exécutez le fichier d'installation (.exe) et suivez les instructions à l'écran.

Lorsque l'installation est terminée, pour modifier et compiler HemiOs :

- Téléchargez sur le site web Hemisson la dernière version de l'HemiOs.
- Dézzippez les fichiers recus.
- Démarrez PICC (CCS) (vous trouverez un raccourci sur le bureau).
- Cliquez sur Project puis Open et sélectionnez le fichier .pjt de l'HemiOs.

Attention, lors de la toute première utilisation :

- Après avoir ouvert le projet, cliquez dans projet puis Include Dirs.
- Vérifiez qu'aucun fichier ou répertoire n'apparaît. Si ce n'est pas le cas, effacez-les avec le bouton Delete.
- Cliquez sur le bouton Close.
- Cliquez sur le bouton OK, de la boîte de dialoque vous demandant si vous souhaitez garder ces paramètres pour tout vos projets.

Après avoir incorporé votre code, appuyez sur la touche F9 de votre clavier pour compiler. Si il n'y a pas d'erreurs dans votre code, CCS aura généré un fichier .hex qui sera le fichier à uploader dans Hemisson.

4 Utilisation de Hemisson Uploader

Une fois votre projet compilé, localisez le fichier *.hex* et consultez la documentation sur *Hemisson Uploader*, disponible sur le site web Hemisson dans la section Assistance.

5 HemiOs

5.1 Qu'est ce que c'est ?

Pour illustrer ce qu'est l'HemiOs, on peut dire que " HemiOs est pour Hemisson, ce qu'est Windows ou Linux pour votre PC ".
En effet, l'HemiOs se charge de gérer au mieux les ressources d'Hemisson et il vous permet d'accéder au matériel embarqué.

5.2 Sa structure

HemiOs est constitué de plusieurs fichiers :

- main.c : fichier principal qui contient le main, c'est dans ce fichier que vous déclarerez vos variables, vos fonctions et que vous écrirez votre code.
- main.h : paramètres d'Hemisson (fréq. du quartz, baudrate...) (*)
- 16f877.h : adresses des registres du processeur (*)
- hemisson.c : fonctions de gestion et d'accès au matériel (*)
- hemisson.h : prototypes des fonctions que vous pouvez utiliser (*)
- variables.c : variables internes (*)
- constantes.h : définitions des constantes (*)
- versions.txt : descriptif des modifications réalisées à chaque version d'HemiOs

(*) : ces fichiers ne doivent pas être édités pas les utilisateurs "ordinaires" qui programment des comportements sur Hemisson, mais seulement par les utilisateurs "experts" qui veulent modifier et améliorer l'HemiOs.

D'autre fichiers seront ajoutés par la suite, ils serviront de librairies pour l'utilisation des modules d'extension.

6 Les fonctions

6.1 Définition

Une petite précision s'impose avant de passer en revue l'ensemble des fonctions disponible dans l'HemiOs.
Dans la notation " a = f(b) ; "

- a est la valeur de renvoi
- f est la fonction
- b est le paramètre

6.2 Fonctions de configuration

void hemisson_init(void)
But : Initialiser Hemisson
Argument : Aucun
Renvoi : Aucun
Exemple : hemisson_init() ;
Rem. : Cette fonction doit figurer au début de votre programme

void hemisson_config_low_voltage_detect(int1)
But : Configurer la détection du bas niveau de la batterie
Argument : Enable ou Disable(par défaut)
Renvoi : Aucun
Exemple : hemisson_config_low_voltage_detect(Enable) ;
Rem. : Cette fonction doit absolument être appelée avant hemisson_init() ;

void hemisson_config_auto_refresh_sensors(int1)
But : Configurer la méthode de lecture des capteurs IRs.
Argument : Manual ou Refresh(par défaut)
Renvoi : Aucun
Exemple : hemisson_config_auto_refresh_sensors(Manual) ;

void hemisson_config_refresh_speed(int1)
But : Configurer la fréquence de rafraîchissement des capteurs Irs.
Argument : Fast ou Normal(par défaut)
Renvoi : Aucun
Exemple : hemisson_config_refresh_speed(Fast) ;

void hemisson_config_auto_refresh_tv_remote(int1)
But : Configurer le rafraîchissement du récepteur de télécommande TV.
Argument : Manual ou Refresh(par défaut)
Renvoi : Aucun
Exemple : hemisson_config_auto_refresh_tv_remote(Manual) ;

void hemisson_config_rs232_control(int1)
But : Activer ou désactiver le mode "Serial Remote Control"
Argument : Disable ou Enable(par défaut)
Renvoi : Aucun
Exemple : hemisson_config_rs232_control(Disable) ;
Rem. : Attention, si ce mode est Enable vous pouvez utiliser la fonction printf mais vous ne pouvez plus rien lire depuis le port série (pas de getc, gets ...)

void hemisson_config_tv_remote_control(int1)
But : Activer ou désactiver le mode "TV Remote Control"
Argument : Disable ou Enable(par défaut)
Renvoi : Aucun
Exemple : hemisson_config_tv_remote_control(Disable) ;

6.3 Fonctions de lecture de flags

int1 hemisson_flag_sensors_refreshed(void)
But : Savoir si la valeur des Irs à été rafraîchie
Argument : Aucun
Renvoi : Le bit vaut 1 lorsque les capteurs viennent d'être rafraîchis
Exemple : i = hemisson_flag_sensors_refreshed() ;
Rem. : Vous devez ensuite appeler la fonction hemisson_flag_sensors_reset pour remettre le flag à zéro

void hemisson_flag_sensors_reset(void)
But : Mettre à zéro le flag de rafraîchissement des capteurs Irs
Argument : Aucun
Renvoi : Aucun
Exemple : hemisson_flag_sensors_reset() ;

int1 hemisson_flag_rs232_filtering(void)
But : Savoir si le mode "Serial Remote Control" est actif
Argument : Aucun
Renvoi : Le bit vaut 1 si le mode est actif, 0 s'il est inactif
Exemple : i = hemisson_flag_rs232_filtering() ;

int1 hemisson_flag_tv_data_refreshed(void)
But : Savoir si la valeur du récepteur de télécommande TV à été rafraîchie
Argument : Aucun
Renvoi : Le bit vaut 1 si des données ont été reçues
Exemple : i = hemisson_flag_tv_data_refreshed() ;
Rem. : Vous devez ensuite appeler la fonction hemisson_flag_tv_data_reset pour remettre le flag à zéro

void hemisson_flag_tv_data_reset(void)
But : Mettre à zéro le flag de rafraîchissement du récepteur de télécommande TV
Argument : Aucun
Renvoi : Auncun
Exemple : hemisson_flag_tv_data_reset() ;

6.4 Fonctions d'accès aux I-O's

unsigned int hemisson_get_proximity(int)
But : Obtenir la mesure de proximité d'un capteur IR
Argument : Front, FrontLeft, FrontRight, Left, Right, Rear, GroundLeft, GroundRight
Renvoi : Valeur entière 8 bits (0(rien) à 255(obstacle proche))
Exemple : i = hemisson_get_proximity(Front) ;

unsigned int hemisson_get_brightness(int)
But : Obtenir la mesure de luminosité d'un capteur IR
Argument : Front, FrontLeft, FrontRight, Left, Right, Rear, GroundLeft, GroundRight
Renvoi : Valeur entière 8 bits (0(bcp de lumière) à 255(peu de lumière))
Exemple : i = hemisson_get_brightness(Rear) ;

int1 hemisson_get_switch_state(int)
But : Lire l'état d'un interrupteur
Argument : 0, 1, 2 ou 3 (numéro de l'interrupteur)
Renvoi : 0 = interrupteur ouvert , 1 = interrupteur fermé
Exemple : i = hemisson_get_switch_state(2) ;

unsigned int hemisson_get_tv_data(void)
But : Obtenir les donnés lues par le récepteur de TV
Argument : Aucun
Renvoi : Valeur entière 8 bits (0 à 255)
Exemple : i = hemisson_get_tv_data() ;

void hemisson_set_speed(int , int)
But : Régler la vitesse des moteurs gauche (première valeur) et droit (deuxième valeurs)
Argument : -15 à 15 (0 = Stop)
Renvoi : Aucun
Exemple : hemisson_set_speed(5 , -5) ;

void hemisson_beep(int1)
But : Produire un son continu
Argument : On ou Off
Renvoi : Aucun
Exemple : hemisson_beep(On) ;

void hemisson_led_frontleft(int1)
But : Allumer ou éteindre la Led avant-gauche
Argument : On ou Off
Renvoi : Aucun
Exemple : hemisson_led_frontleft(On) ;

void hemisson_led_frontright(int1)
But : Allumer ou éteindre la Led avant-droite

Argument : On ou Off
Renvoi : Aucun
Exemple : hemisson_led_frontright(On) ;

void hemisson_led_pgmexec(int1)
But : Allumer ou éteindre la Led Pgm-Exec
Argument : On ou Off
Renvoi : Aucun
Exemple : void hemisson_led_pgmexec(On) ;

void hemisson_led_onoff(int1)
But : Allumer ou éteindre la Led On-Off
Argument : On ou Off
Renvoi : Aucun
Exemple : hemisson_led_onoff(On) ;

void hemisson_manual_refresh_sensors(void)
But : Rafraîchir manuellement une zone de capteurs IRs
Argument : FrontZone, GroundZone ou RearZone
Renvoi : Aucun
Exemple : hemisson_manual_refresh_sensors(FrontZone) ;
Rem. : Utilisez ensuite l'une des 2 fonctions de lecture des capteurs IRs (hemisson_get_proximity ou hemisson_get_brightness)

6.5 Fonctions de délai

void hemisson_delay_ms(int16)

But : Insérer un délai d'attente
Argument : Temps d'attente en millisecondes (0 à 65535)
Renvoi : Aucun
Exemple : hemisson_delay_ms(2000) ;

void hemisson_delay_us(int16)

But : Insérer un délai d'attente
Argument : Temps d'attente en microsecondes (0 à 65535)
Renvoi : Aucun
Exemple : hemisson_delay_us(200) ;

6.6 Fonctions d'accès externe

int1 hemisson_ext_read_PINB0(void)
But : Lecture de la pin B0
Argument : Aucun
Renvoi : 0 ou 1
Exemple : i = hemisson_ext_read_PINB0() ;

int1 hemisson_ext_read_PINB6(void)
But : Lecture de la pin B6
Argument : Aucun
Renvoi : 0 ou 1
Exemple : i = hemisson_ext_read_PINB6() ;

int1 hemisson_ext_read_PINB7(void)
But : Lecture de la pin B7
Argument : Aucun
Renvoi : 0 ou 1
Exemple : i = hemisson_ext_read_PINB7() ;

void hemisson_ext_write_PINB0(int1)
But : Ecriture sur la pin B0
Argument : 0 ou 1
Renvoi : Aucun
Exemple : hemisson_ext_write_PINB0(0) ;

void hemisson_ext_write_PINB6(int1)
But : Ecriture sur la pin B6
Argument : 0 ou 1
Renvoi : Aucun
Exemple : hemisson_ext_write_PINB6(1) ;

void hemisson_ext_write_PINB7(int1)
But : Ecriture sur la pin B7
Argument : 0 ou 1
Renvoi : Aucun
Exemple : hemisson_ext_write_PINB7(0) ;

6.7 Fonctions spécifiques à CCSC

Le compilateur CCS contient également un très grand nombre de fonctions, comme les accès I2C, la génération de nombre aléatoires, délais etc...
Vous pouvez trouver toutes les infos dans l'aide de CCS.

7 Exemples

7.1 Moteurs

```
#include <hemisson.h>

// Add your variables here
// ...

void main( void )
{
      hemisson_init();                    // Initialisation d'Hemisson
      while(1)
      {
            hemisson_set_speed(5,5);      // Hemisson avance tout droit
            hemisson_delay_ms(1000);      // Attente de 1000 ms (1 s)
            hemisson_set_speed(-5,5);     // Hemisson tourne à gauche
            hemisson_delay_ms(1000);      // Arrêt
      }
}
```

Commentaires : ce programme fait avancer Hemisson pendant une seconde, le fait tourner sur place à gauche et recommence.

7.2 LEDs

```
#include <hemisson.h>

// Add your variables here
// ...

void main( void )
{
      hemisson_init();                    // Initialisation d'Hemisson
      while(1)
      {
            hemisson_led_frontleft(1);    // On allume la LED avant gauche
            hemisson_delay_ms(1000);      // Attente de 1000 ms (1 s)
            hemisson_led_frontleft(0);    // On éteint la LED avant gauche
            hemisson_delay_ms(1000);      // Arrêt
      }
}
```

Commentaires : ce programme fait clignoter une LED.

7.3 Buzzer

```
#include <hemisson.h>

// Add your variables here
// ...

void main( void )
{
      hemisson_init();                    // Initialisation d'Hemisson
      while(1)
      {
            hemisson_beep(1);             // Buzzer On
            hemisson_delay_ms(1000);      // Attente de 1000 ms (1 s)
            hemisson_beep(0);             // Buzzer Off
            hemisson_delay_ms(1000);      // Arrêt
      }
}
```

Commentaires : ce programme fait beeper le buzzer.

7.4 Interrupteurs

```
#include <hemisson.h>

// Add your variables here
// ...

void main( void )
{
      hemisson_init();                              // Initialisation d'Hemisson
      while(1)
      {
            if( hemisson_get_switch_state(0) == 1 )  // Test la position de l'interrupteur 0
            {
                  hemisson_led_frontleft(1);        // On allume la LED avant gauche
            }
            else
            {
                  hemisson_led_frontleft(0);        // On éteint la LED avant gauche
            }
      }
}
```

Commentaires : ce programme lit l'état d'un interrupteur et allume ou éteint une LED en fonction sa position.

7.5 Communication sérielle

```
#include <hemisson.h>

// Add your variables here
// ...

void main( void )
{
      char reponse = 0x00;
      hemisson_init();                                  // Initialisation d'Hemisson
      hemisson_config_rs232_control(Disable);           // Annule le contrôle sériel par l
      while(1)
      {
            printf("1: Buzzer On\r\n");                 // Affichage des choix
            printf("2: Buzzer Off\r\n");
            printf("Entrez votre choix: \r\n")
            reponse = getc();                           // Récupération du choix
            switch( reponse )
            {
                  case '1':
                        hemisson_beep(1);
                        break;
                  case '2':
                        hemisson_beep(0);
                        break;
            }
      }
}
```

Commentaires : ce programme envoie des caractères via la liaison sérielle puis attend une réponse. Suivant la réponse donnée, il produit un son ou éteint le buzzer. Pour tester ce programme, il vous faut utiliser l'hyperterminal.

8 Conclusion

Nous espérons avoir suffisamment décrit la méthode de programmation du robot Hemisson en langage C pour que vous puissiez commencer à réaliser vos propres algorithmes comportementaux.
Si toutefois des points paticuliers n'étaient pas clairs, n'hésitez pas à poser vos questions sur le forum Hemisson.