Mémo pour VxWorks

Quelques commandes shell :

- i : affiche les tâches en cours
- sp(fonction) : crée et démarre une tâche
- ti(nom) : détails sur la tâche, visible dans le browser
- td(nom) : destruction d'une tâche
- ts(nom) : suspend une tâche
- tr(nom) : réveille une tâche
- tw(nom) : indique l'objet sur lequel est en attente une tâche
- [ESC k] : rappel de la dernière commande
- **[CTRL d] : complétion des noms de fonctions + indication des paramètres**
- **[CTRL w] : lance le fichier d'aide associé à la fonction tapée**
- Possibilité de taper du C directement dans le shell :
  - Mot-clés, fonctions, primitives VxWorks avec paramètres séparés par des virgules. Exemple : taskPrioritySet tache, valeur -> modification de la priorité d'une tâche
  - Résultat de la fonction dans le shell
  - Application de la fonction sur la cible (ou simulateur)

Gestion des tâches : #include <taskLib.h>

- tid = taskSpawn(…) : crée et active une tâche
- taskInit(…) : création d'une tâche
- taskActivate(tid) : activation d'une tâche
- taskSuspend(tid) : suspend une tâche
- taskResume(tid) : réveille une tâche
- taskRestart(tid) : relance une tâche avec ses arguments initiaux
- taskDelete(tid) : termine l'exécution d'une tâche et libère la mémoire
- taskSafe(tid) : protège une tâche contre un éventuel effacement
- taskUnsafe(tid) : ré autorise l'effacement de la tâche
- taskLock(tid) : inhibe la préemption
- taskUnlock(tid) : autorise la préemption
- taskPriorityGet(tid) : retourne le niveau de priorité
- taskPrioritySet(tid) : programmation du niveau de priorité (0 à 255)
- taskIdSelf() : retourne le tid de la tâche en cours
- exit : arrête la tâche active et libère la mémoire

Sémaphores : #include <semLib.h>

- semId = semBCreate(SEM_FULL, SEM_Q_FIFO) : sémaphore binaire initialisé à pris
- semId = semMCreate(SEM_Q_PRIORITY || SEM_INVERSION_SAFE) : semaphore d'exclusion mutuelle
- semId = semCCreate(SEM_Q_FIFO, initialCount) : semaphore à compte
- semTake(semId, timeout) : prise du sémaphore
- semGive(semId) : libération du sémaphore
- semFlush(semId) : libère toutes les tâches en attente sur le sémaphore (de type binaire)
- semDelete(semId) : détruit le sémaphore

Manipulation de files de messages : #include <msgQLib.h>

- msgQId = msgQCreate(…) : création d'une file de messages
- msgQSend(…) : envoi d'un message
- msgQReceive(…) : réception d'un message
- msgQNumMsgs(msgQId) : retourne le nombre de messages qu'il y a dans la file
- msgQDelete(msgQId) : détruit la file

Gestion du temps : #include <sysLib.h>

- sysClkRateGet : retourne la fréquence de l'horloge système (ticks par seconde)
- taskDelay(nbTicks) : place la tâche courante en attente de l'écoulement d'un délai exprimé en ticks
- sysClkRateGet() : retourne la fréquence de l'horloge système
- sysClkRateSet(frequence) : modifie la fréquence de l'horloge système

Chiens de garde : #include <wdLib.h>

- wid = wdCreate() : crée un chien de garde
- wdStart(wid, delay, pRoutine, parameters) : (ré)arme le chien de garde qui exécute la routine après le nombre de ticks spécifié
- wdCancel(wid) : désarme le chien de garde
- wdDelete(wid) : détruit un chien de garde

Horloge auxiliaire : #include <sysLib.h>

- sysAuxClkConnect(…) : connexion d'une fonction à l'horloge auxiliaire
- sysAuxClkRateSet(frequence) : programmation de la fréquence de l'horloge auxiliaire (attention !!! les fréquences possibles dépendent de la cible)
- sysAuxClkRateGet() : retourne la fréquence de l'horloge auxiliaire
- sysAuxClkEnable() : active l'horloge auxilaire
- sysAuxClkDisable() : inhibe l'horloge auxiliaire

Interruptions : #include <intLib.h>

- intConnect(…) : connexion d'une fonction à un vecteur d'IT
- intLock : masquage des interruptions jusqu'au niveau défini par intLockLevelSet
- intUnlock : démasquage des interruptions
- intLockLevelSet : positionnnement du niveau de masque des IT (0 à 7)