DUT informatique, orientation SI, TP n°15-16 2013-14
P. Kauffmann

MODULE TEMPS-REEL EMBARQUEE

Système d'exploitation temps réel : système de base

L'utilisation d'un système d'exploitation temps réel est courante dans les systèmes embarqués à microcontrôleurs car ils rendent le code plus facile à écrire et à maintenir, ce qui contrebalance le ralentissement et la lourdeur induits.

Toutefois, en raison de l'augmentation de la charge du contrôleur et de l'augmentation de la taille du code, les noyaux sont rarement utilisés dans les petites applications.

Nous allons dans cette série de travaux pratiques implémenter deux applications simples : le contrôle d'un petit sonar et l'asservissement en position d'un motoréducteur à courant continu. Dans un premier temps (dans ce TP de 8 h), nous allons mettre en place le système avec une IHM embarquée (clavier et afficheur) et une IHM déportée sur un PC. L'IHM embarquée est celle déjà développée en cours d'année. L'IHM déportée est une application « Visual-Tcl » conforme à ce que nous avons déjà fait lors de la période précédente, reliée au contrôleur via une liaison série asynchrone.

1. Objectifs opérationnels

- Réaliser un cadre pour une application temps réel embarquée comportant une IHM de supervision graphique et une IHM embarquée.
- Maîtriser l'installation et la mise en œuvre du système d'exploitation Itron4.
- Maîtriser les principaux éléments de la bibliothèque du système d'exploitation, notamment :
    - la gestion des tâches,
    - l'ordonnancement en prenant en compte les interruptions,
    - la communication et synchronisation avec des messages et des sémaphores

2. Documentation et matériel nécessaire

Les documentations nécessaires pour la réalisation de ce TP sont :

- polycopié de cours sur les microcontrôleurs,
- notes de cours personnelles,
- documentation électronique de la famille RX62,
- les schémas de la plateforme cible.

Le matériel nécessaire pour la réalisation de ce TP est :

- un ordinateur compatible PC avec l'AGL RX62, un système d'exploitation Itron4 et Tcl/Tk sous Windows,
- un système RX62N avec carte prototypage rapide.
- un potentiomètre linéaire de 1 MΩ avec un secteur gradué en degrés.

3. Organisation matérielle

Le TP sera réalisé par un groupe de deux étudiants. Le TP fera l'objet d'un rapport écrit contenant les réponses aux questions et le code des fonctions demandées, rendu en fin de séance sur papier. De plus, la totalité du code écrit, opérationnel et commenté, sera placé sur la forge de Clermont-Université à l'endroit spécifié par l'enseignant d'encadrement.

4. Manipulations

4.1 Programme embarqué cadre

Nous allons d'abord mettre en place du coté embarqué un programme cadre capable d'exploiter correctement les bibliothèques d'affichage et gestion de clavier. Dans ce but nous allons mettre en place un système temps réel qui contient quatre tâches classiques :

- `start()` qui initialise l'application,
- `affiche()` qui gère l'affichage périodiquement 40 fois par seconde,
- `clavier()` qui gère le clavier 100 fois par seconde et,
- `chargeCPU()`qui fait clignoter les leds indiquant la charge du processeur.

ainsi que le handler cyclique `calcTemps()` qui incrémente le temps (heures, minutes et secondes) à chaque centième de seconde.

La tâche `start()`initialise certains périphériques et objets, puis lance les autres tâches avant de s'autodétruire. Les tâches `affichage()` et `clavier()` sont des reprises et adaptation par héritage des méthodes déjà utilisées de gestion du clavier et de l'afficheur. La tâche `chargeCPU`(), pour sa part, fait clignoter en alternance les deux leds vertes et rouge de la carte mère à la fréquence de 2 Hz lorsque le processeur a une charge nulle. Afin que la fréquence de clignotement diminue avec la charge du processeur, elle sera appelée toutes les millisecondes et aura la priorité la plus faible de toutes les tâches.

N. B. : le code devra reprendre les classes crées dans les TP précédant ; seul du code reliant tous les éléments ensemble devant être écrit.

Le canevas du code à utiliser est celui donné ci-après :

```
#define _GLOBALS_
#include “CppHeader.h”

/*********************************************************
*                       START TASK
*********************************************************/
void start(VP_INT stacd)
{
// Static display
aff.writeStringXY(0, 3, "RTOS ITRON BASE");
```

```
// Start other tasks
act_tsk(ID_affiche);     // display control task
act_tsk(ID_clavier);     // keyboard control task
act_tsk(ID_chargeCPU);   // CPU activity control task
start_cyc(ID_calcTemps); // time calculation cyclic handler

// stop task
ext_tsk();
}
```

Travail à réaliser et à rendre

- Créez l’application minimale juste capable de faire clignoter les leds verte et rouge de la tâche `chargeCPU()`.
- Complétez l'application afin qu'elle affiche sur la première ligne un message fixe et sur la seconde ligne le temps en heures, minutes et seconde depuis le démarrage. Commentez le code afin d'expliquer le rôle de chaque partie en détail. Démontrez le fonctionnement à l'enseignant d'encadrement.
- Ajoutez la saisie d’un nombre entier non signé tenant sur 16 bits à la dernière ligne de l’afficheur si on a préalablement pressé ‘A’. Après la pression de la touche « `entrée` » le nombre (stocké dans une variable à portée globale) sera reformaté et affiché à la même place.
- Montrez le résultat à l’enseignant d’encadrement.

4.2 Position d'un potentiomètre

Le but ici est de numériser comme dans le premier TP d’instrumentation (TP8) 5 fois par seconde la tension générée par un potentiomètre relié à l'entrée analogique `AD4` du microcontrôleur, d'afficher cette valeur sur la cinquième ligne de l’afficheur embarqué, ainsi que de la transmettre à l’IHM déportée sur le PC via un port série asynchrone.

Pour obtenir ce résultat, on va dans l’ordre suivant :

- Réutiliser la fonction d’initialisation du convertisseur en mode "single shot" et 12 bits de précision avec cumul de quatre échantillons (résultat de la mesure sur 14 bits).
- Créer une tâche d’interruption déclenchée par le convertisseur `ad12int()` qui transmet à la tâche `affich()` via les fonctions `(x)snd_dtq()` et `(x)rcv_dtq()` la moyenne de 1024 échantillons sur 16 bits. N.B. : Les cycles de conversion devront être déclenchés 5 fois par seconde par la tâche `affich()`.
- Ajouter le code d’émission sur le port série 5 fois pas seconde dans la tâche `affich()`.
- Utiliser la gestion du clavier pour démarrer et arrêter les conversions à volonté selon qu’on presse 0 ou 1. On utilisera à nouveau les fonctions `(x)snd_dtq()` et `(x)rcv_dtq()` pour transmettre de la tâche `clavier()` à la tâche `affiche()` l’ordre d’arrêter ou démarrer les conversions.

N. B. : le code devra reprendre les classes crées dans les TP précédant adaptées par héritage, notamment les classes `s12ad` et `sci`.

Travail à réaliser et à rendre

- Créez le code dûment commenté selon le cahier des charges qui vient d'être défini et montrez le fonctionnement à l'enseignant d'encadrement.
- Mettez le code relatif aux « data queue » dans votre rapport.

4.3. Réception et exploitation d'une commande venant de l'IHM déportée

Nous aurons besoin dans les TP suivants de pouvoir recevoir des commandes venant du PC de supervision. Notre IHM définie en 4.3 est potentiellement capable d'envoyer des commandes sur le port série. Il ne manque qu'un peu de script pour les envoyer et un peu de code embarqué pour réceptionner ces commandes.

Nous allons écrire maintenant le code embarqué de gestion des commandes du port série en l'organisant de la façon suivante :

- Réalisez une tâche d'interruption de réception du port série `sciReceive()`. Cette fonction recevra la commande sous forme ASCII et la transmettra à la tâche `clavier()` sous la forme d'un message via `(x)snd_mbx()` et `(x)rcv_mbx()` dès qu'elle sera complète (réception du saut de ligne). La tâche `clavier()`activera alors la conversion analogique si la commande reçue est « `start` » ou l'inhibera si la commande reçue est « `stop` ».
- Nous allons aussi utiliser un message `(x)snd_mbx()` et `(x)rcv_mbx()`pour transmettre du handler `calcTemps()` à la tâche `affiche()` le temps (tableau d'heures, minutes et secondes).

N. B. : l'IHM déportée du TP n'existant pas à ce stade du TP, on utilisera une émulation de terminal (par exemple Tera Term) pour générer l'émission des chaînes « start » et « stop ».

Travail à réaliser et à rendre

- Créez le code selon le cahier des charges qui vient d'être défini et montrez le fonctionnement à l'enseignant d'encadrement.
- Mettez dans votre rapport les codes entièrement commentés du code relatif aux « mail box ».

4.4 IHM de contrôle déportée

Le but est ici de réaliser une interface graphique capable de :

- recevoir et stocker des séries de mesures en provenance du port série (c'est à dire du système RX62N),
- afficher simultanément la courbe de ces mesures,
- envoyer des ordres simples à la carte RX62N sous forme de chaînes ASCII.

Chaque mesure reçue du port série se présentera sous la forme d'une chaîne de caractères représentant un entier, terminée par un retour chariot et saut de ligne et respectant la forme suivante : `563\r\n ; le` nombre représentant la mesure comprise entre 0 et 4095,

Le port série sera configuré à la vitesse de 115200 Bauds, données sur 8 bits, pas de parité et 1 bit de stop.

Les données brutes seront sauvegardées de façon séquentielle dans un fichier de type texte nommé `tempo.mes` dans le répertoire courant (une mesure par ligne) lorsque l'affichage des mesures courantes sera activé. Ce fichier sera effacé en quittant l'application, mais on proposera dans une fenêtre modale à ce moment la sauvegarde des données dans un fichier permanent au choix.

L'aspect graphique de l'IHM sera conforme à celle du TP « mesure de tension d'un potentiomètre » de la série des TP sur les capteurs, à savoir :

- affichage des mesures dans un canevas de hauteur 570 pixels et longueur 10000 pixels disposant d'un ascenseur horizontal (un nouveau point sera affiché tous les cinq pixels).
- Présence d'une ligne d'état en bas de la fenêtre de l'application pour indiquer l'état de la connexion au port série, le nom du fichier lu (sans le chemin) ainsi que la valeur de la dernière mesure.
- Présence d'un menu déroulant avec les éléments suivants
    - Fichier
        - Sauvegarder les mesures courantes (prévoir la possibilité du choix du nom et du répertoire, filtrer les extensions .mes et .dat)
        - Charger des mesures enregistrées (prévoir la possibilité du choix du nom et du répertoire, filtrer les extensions .mes et .dat)
        - Quitter l'application
    - Mesures
        - Démarrer les mesures
        - Arrêter les mesures
    - Affichage
        - Effacer l'affichage et revenir à zéro
        - Filtrer l'affichage (case à cocher)
    - Configuration
        - Sélection du port série (`com3` par défaut)
    - Aide
        - A propos
        - Aide en ligne

Travail à réaliser et à rendre

- Ajoutez au code embarqué l'émission des données sur le port série. La tâche

`affich()` activera la fonction d'interruption `sciSend()` en émettant le premier caractère de la chaîne à transmettre. Les caractères suivants à transmettre seront dans une variable globale de type tableau.

- Réalisez la partie graphique de l'IHM en utilisant `Visual Tcl` et soumettez la pour approbation à l'enseignant d'encadrement.
- Codez les deux scripts qui envoient microcontrôleur les commandes « `start` » ou « `stop` » lorsqu'on clique sur « Démarrer les mesures » ou « Arrêter les mesures » et montrez le résultat à l'enseignant d'encadrement.
- Codez les autres procédures nécessaires au bon fonctionnement en Tcl. Montrez le résultat à l'enseignant d'encadrement.
- Ajoutez la procédure de dessin commentée dans le canevas dans votre rapport.

4.5 Amélioration de l'ergonomie

En l'état actuel, notre système est opérationnel, mais il manque quelques éléments pour rendre l'IHM plus conviviale.

Travail à réaliser et à rendre

- Ajoutez :
    1. le quadrillage du canevas avec un trait gris tous les 20 pixels et un trait noir tous les 100 pixels
    2. la mise en place d'une échelle horizontale en secondes et verticale en Volts avec indication des unités
    3. des raccourcis clavier pour les commandes de menu déroulant les plus importantes avec texte d'aide lorsqu'on passe avec la souris
    4. prise en compte à chaque démarrage de
        - de la précédente taille et position de l'application
        - du précédant port série utilisé
        - des précédents répertoires courant pour la sauvegarde et récupération des mesures
    5. un bandeau avec trois icônes de raccourci pour :
        - charger des mesures enregistrées,
        - sauvegarder les mesures courantes,
        - remettre à zéro le fichier de mesures temporaires et l'affichage
    6. un manuel en html avec au moins trois items et images
- Montrez le fonctionnement à chaque étape à l'enseignant d'encadrement.
- Proposez et codez d'autres commandes qui vous paraissent utiles. Expliquez en détail le fonctionnement de ces commandes et pourquoi vous les avez ajoutées.
- Montrez le fonctionnement à l'enseignant d'encadrement.
- Précisez dans votre rapport toutes les procédure spécifiques commentées que vous avez codées (pas les fonctions récupérées dans les corrigés), et ajoutez les à la fin de votre rapport.

5. Conclusion

Durant cette manipulation nous avons mis en place l'ensemble des éléments courants d'une application embarquée. Nous avons aussi vu les principales fonctions qui vont nous être utiles pour la réalisation d'un asservissement.

L'application qui a été créée ici peut donc être considérée comme le prototype d'une application multitâches temps réel simple typique. Son code servira de base et référence pour les manipulations suivantes. Nous pourrons ainsi nous concentrer sur l’application proprement dite sans avoir à écrire à l'avenir beaucoup de code nouveau.