TP Driver de communications CAN sous RTAI

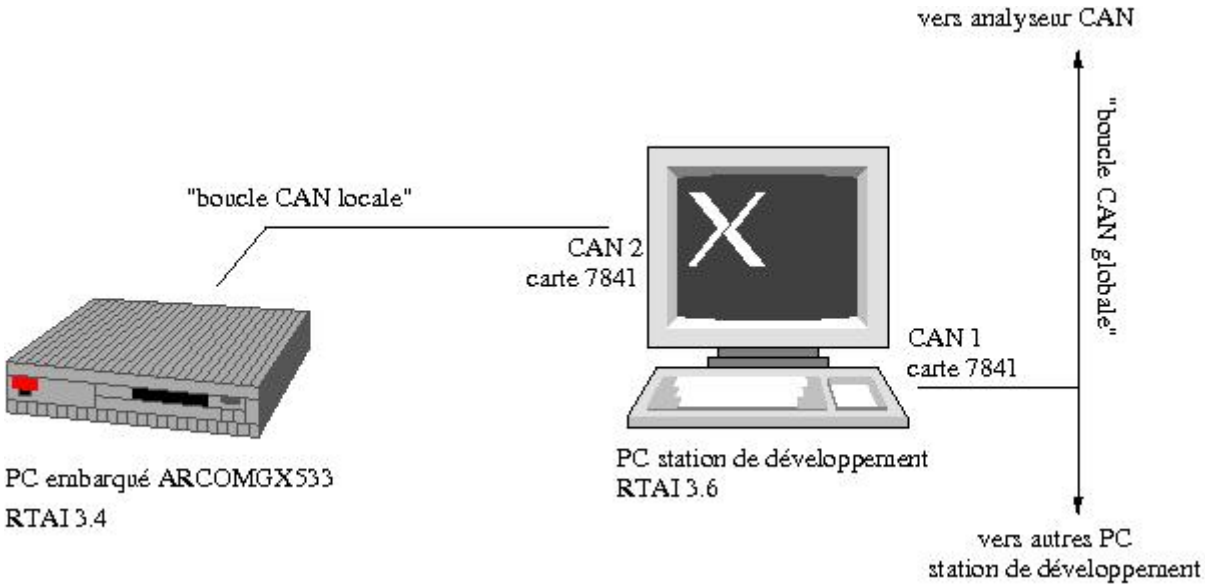
Travail à réaliser

Le but de ce TP est de réaliser et valider les primitives de communications d'accès à un bus CAN sous un environnement de développement temps réel Linux RTAI.

1. Lire complètement ce document avant de commencer les développements. Il vous donne toutes les informations utiles à la réalisation de ce TP.
2. Ecrire un module RTAI réalisant l'envoi périodique d'un message sur le bus CAN.
3. Ecrire une fonction qui affiche dans la console les paramètres d'un message reçu sur le CAN. Pour détecter l'arrivée d'un message, on réalisera une scrutation périodique du bit de réception du SJA 1000 qui gère les communications CAN.
4. Idem question précédente, mais cette fois le sja1000 devra déclencher une interruption à l'arrivée d'un message.

Architecture Matérielle

Chaque groupe disposera sur sa table du matériel suivant : une station de développement de type PC sous Linux RTAI, un PC embarqué sous Linux RTAI. Ces 2 calculateurs sont reliés entre eux par un bus CAN via une carte de communication CAN (boucle locale) . Une boucle CAN globale relie l'ensemble des stations de développement. Sur cette boucle globale est aussi connecté un analyseur CAN.



- Station de développement : elle est composée d'un PC de bureau relié à une connexion Ethernet. L'environnement de travail est Linux RTAI 4. Il dispose d'une carte de communication CAN ADLINK 7841 PCI.
- PC embarqué : C'est une carte ARCOM SBCGX533 qui dispose d'une connexion Ethernet, mais ni écran ni clavier. L'OS installé sur la carte est Linux RTAI 3.4. Les programmes seront développés et compilés sur la station de travail et téléchargés sur la carte via la connexion ethernet. Une carte AIM104 CAN est connectée à cette carte.

- carte communication CAN ADLINK 7841 PCI
  - Cette carte intègre 2 SJA1000 afin de fournir 2 connexions CAN indépendantes.
  - Identification de la carte sur le bus PCI : VENDOR_ID = 0x144A, DEVICE_ID = 0x7841
  - Vitesse de transmission : 125Kb/s, Determination of bit timing parameters SJA1000_BT.ps
  - manuelPCI7841.pdf
  - Data Sheet SJA1000.pdf

- carte communication CAN AIMPC104

    - Cette carte intègre 1 SJA1000.
    - Cette carte connectée au bus PC104 de la carte embarquée est directement mappée dans la mémoire I/O de la carte ARCOMGX533. Le SJA1000 est accessible à l'adresse 0x180 et génère des intérruptions sur l'IRQ 5
    - Vitesse de transmission : 125Kb/s, determination of bit timing parameters SJA1000_BT.ps
    - manuelAIM104.pdf
    - Data Sheet SJA1000.pdf

Architecture logicielle

Tous les développement seront effectués sur la station de développement sur laquelle est installé un OS LINUX RTAI 4. La carte PC embarquée est gérée par un OS Linux RTAI 3.4. Elle ne dispose pas d'outils de compilation. Les compilations seront effectuées par le compilateur gcc qui permettra de compiler du code pour RTAI3.4 ou RTAI 4.
Le comportement de RTAI3.4 et RTAI 4 étant identiques, certains modules pourront être compilés et testés sur la station de travail. Après validation ils seront recompilés pour la carte PC embarqué sur laquelle ils seront chargés via Ethernet et executés.

- RTAI 3.4. (carte PC embarqué) :
  - Le nom de réseau de la carte est arcom#@esiee.fr ou # est le numéro inscrit sur le connecteur ethernet de la carte.
  - Makefile34 pour compiler sur la station un module pour RTAI 3.4, il peut aussi être utilisé pour envoyer les modules compilés sur la cible.
  - transfert de fichiers entre la machine développement et le PC embarqué :
    - *scp fichier_a_transferer arcom@nom_machine.esiee.fr:/home/arcom*
  - il est possible de se logger à distance sur la carte embarquée :
    - ouvrir une fenêtre console
    - *ssh arcom@nom_machine.esiee.fr*
    - login : *arcom* password *arcom*,
  - le chargement des modules peut se faire classiquement avec la commande *insmod* ou à l'aide du script *runarcom*

- RTAI 4 (station de développment)
  - login: *rtai* password: *rtai*
  - Makefile4 pour compiler un module RTAI 4
  - le chargement des modules peut se faire à l'aide du script runPC
  - **Le compte RTAI étant utilisé par tout le monde, archivez vos sources sur votre compte à la fin de chaque séance.** Utilisez pour cela la commande : *scp fichier mon_login@acme1.esiee.fr:~mon_login/*
  - Fichier archive :
    - créer un fichier archive : *tar -zcvf nom_fichier.tgz repertoire_a_archiver/*
    - décompresser un fichier archive : *tar -zxvf fichier.tgz*

Gestion d'interruptions sous RTAI

La gestion des interruptions sous RTAI se fait très simplement à l'aide de quelques fonctions.

- Include : `#include <asm/irq.h>`
- Description du gestionnaire d'interruption:

```
void mon_gestionnaire(void)
{
...
rt_ack_irq(num_irq);/* acquittement de l'interruption */
}
```

- Installation du Handler d'interruption dans le init_module:

```
rt_global_cli(); /* desactivation des IT */
rt_request_global_irq(num_irq,mon_gestionnaire); /*installation
du handler */
                                                  /* sur l'IT
num_irq       */
rt_startup_irq(num_irq); /* activation de la ligne
d'interruption */
rt_global_sti(); /* re-activation des IT */
```

- Désinstallation du handler d'interruption dans le cleanup_module :

```
rt_shutdown_irq(num_irq);/* désactivation de l'IT num_irq */
rt_free_global_irq(num_irq); /* désintallation du handler */
```

Si vous le désirez, vous trouverez des informations complémentaires dans la documentation RTAI, ou dans le Linux_Device_Driver.pdf

Gestion des E/S sous RTAI

Pour écrire et lire dans des zones mémoires allouées à des périphériques, il est nécessaire d'utiliser des fonctions spécifiques. Sous RTAI on pourra utiliser les fonctions suivantes :

- outb(*value*,*address*) : écriture de *value* (byte = 8bits) à *address*
- *value*=inb(*address*) : retourne l'octet contenu à *address*
- outw(value,address) : identique à outb mais on écrit un mot 16bits
- value=inw(address) : identique à inb mais retourne un mot 16bits.

Toutes ces fonctions sont documentées dans linux_device_driver.pdf

Copie de zone mémoire

Il est parfois utile de copier une zone mémoire dans une autre zone mémoire. Sous Linux il existe pour réaliser cette opération la fonction memcpy(&dest,&source,taille) : *dest* est l'adresse de la zone destination, *source* l'adresse de la zone source, et *taille* le nombre d'octets à transférer.

Gestion de la carte CAN ADLINK 7841 PCI sous RTAI

- La carte CAN est connectée au bus PCI. Sur ce type de bus, l'adresse des cartes et les lignes d'interruptions sont fixées au boot par le bios. Pour pouvoir accéder à la carte il faut tout d'abord

"interroger" le bus PCI pour récuperer l'adresse de la carte et la ligne d'irq attribuée. Cette interrogation se fait à partir d'identifier de la carte qui sont le VENDOR_ID = 0x144A et le DEVICE_ID = 0x7841. **Lors de ce TP cette procédure vous est déjà fournie dans le fichier squelette de votre application.**

- On utilisera le CAN à un débit de 125Kb/s : *BTR0 = 0x03 et BTR1=0x1c.* Le output control register sera initialisé à *0xFA*.
- Le CAN 0 (boucle globale) est accessible à l'adresse de base de la carte CAN.
- Le CAN 1 (boucle locale) est accessible à l'adresse de *base+0x80* .

Fichiers Fournis

L'archive tpcan.tgz contient tous les fichiers utiles pour réaliser ce TP . Sur ce 1er TP les développements se feront fait sur le PC sous RTAI 4 avec la carte CAN 7841 sur la "boucle CAN globale" afin de permettre une validation avec l'analyseur logique. Ainsi seuls les fichiers du répertoire PC seront utilies pour ce TP.

- répertoire PC/
  - *squelette_7841.c: Ce fichier est la base de votre module de communication CAN, a vous de le completer.*
  - *Makefile4* : ce makefile est à utiliser si vous voulez compiler un module pour RTAI 4. Il doit être édité :
    - `obj-m := module1_a_compiler.o module2_a_compiler.o ...`
    - Pour compiler, renommez le Makefile4 en Makefile et utiliser la commande *make*
  - *runPC* : chargement et déchargement des modules sur le poste de développement.
    - `./runPC mon_module` chargera les principaux modules rtai et le module mon_module.ko
  - répertoire test_can_7841/ : ce répertoire contient les modules pré-compilés de gestion de la carte can 7841 afin de vous permettre, en cas de doute, de vérifier que la carte CAN fonctionne correctement et est bien reliée au bus CAN. Ces modules utilisent le CAN à 125Kb/s.
    - `./runtest test_CAN_recv` lancera les modules de test pour la réception. La commande `dmesg` vous permettra de visualiser les informations sur les messages reçus.
    - `./runtest test_CAN_send` lancera les modules de test pour l'envoi. Une trame CAN est envoyée périodiquement.
- répertoire ARCOMGX533/
  - *Makefile34* : ce makefile est a utiliser si vous voulez compiler un module pour RTAI 3.4 (carte ARCOM GX533).
    - Il doit être édité :
      - KIT = arcom#@esiee.fr : remplacez le # par le numéro de la carte.
      - `obj-m := module1_a_compiler.o module2_a_compiler.o ...`: spécifie la liste des modules à compiler.
    - Pour compiler il vaut mieux le renommer en Makefile et lancer la commande: *make*
    - Pour envoyer les modules compilés sur la carte ARCOMGX533 : *make send*
    - Pour supprimer les modules compilés : *make clean*
  - *runarcom* : script permettant de charger et décharger les modules sous RTAI 3.4 : *./runarcom mon_module_a_charger (Cette commande est à exécuter sur le PC embarqué).*
  - *3712.h : fichier header gestion de la carte conversion numérique/analogique 3712*
  - *3712.ko: module gestion carte conversion numérique/analogique 3712*
  - squelette_tp2.c : squelette du tp2
  - répertoire test_CAN_AIM104/: ce répertoire contient les modules pré-compilés de gestion de la carte can AIMCANPC104 afin de vous permettre, en cas de doute, de vérifier que la carte CAN fonctionne correctement et est bien reliée au bus CAN. Ces modules utilisent le CAN à 125Kb/s. L'ensemble de ces fichiers doit être copié sur la

carte ARCOMGX533. Les commandes suivantes pourront alors être exécutées sur la carte ARCOMGX533.

- `./runtestarcom test_aim104can_recv` lancera les modules de test pour la réception. La commande `dmesg` vous permettra de visualiser les informations sur les messages reçus.
- `./runtestarcom test_aim104can_send` lancera les modules de test pour l'envoi. Une trame CAN est envoyée périodiquement.

BON COURAGE !!