// -----------------------------------------------------------------------------

// Nom Fichier : can.c

// Fonction : Programme d'émission et réception de trames CAN sur bus CAN

// avec bibliotheque INFINEON AP292201

// Auteur : Pascal Berthou & Agnan de Bonneval

// Date : janvier 2010

// -----------------------------------------------------------------------------

// -------------------AVANT TOUTE CHOSE, VOUS DEVEZ IMPERATIVEMENT : -----------

// 1) INDIQUER VOS

// - NOMS

// - Numéro de COMPTE

// - DATE

// 2) RENOMMER ce fichier en : can.c

// 3) L'ADAPTER selon les besoins de la manipulation.

// -----------------------------------------------------------------------------

// Ces includes seront à déplacer ou supprimer selon la structuration en fichier

// choisie pour les programmes d'application

#include <c167.h>

#include <gnutrap.h>

#include <can\_16x.h>

#include "commc167.h"

// ----Petit lexique

// - MO : Message OBJET

//

// ---- Constantes GENERIQUES pour l'initialisation du bus CAN -------

#define CAN\_BAUDRATE 500 // Baud rate (vitesse du bus) : 500 Kbps

#define CAN\_IE 0 // Interruptions CAN inactives

// ---- Constantes GENERIQUES (potentiellement) pour initialisation des MO -----

#define CAN\_XTD\_BIT 0 // Taille des identifiants CAN : sur 11 bits

#define CAN\_TXIE\_BIT 0 // Génération d'interruptions d'EMISSION : NON

#define CAN\_RXIE\_BIT 0 // Génération d'interruptions de RECEPTION : NON

// ---- Constantes spécialisees pour chaque Message Objet particulier ----------

#define M01\_CAN\_ID 0x002 // Valeur de l'IDENTIFIFANT CAN choisi pour

// Message Objet particulier (ex. : ici, MO n°1)

#define M01\_CAN\_DIR\_BIT 0 // "DIRECTION" du M0 :

// - RECEPTION : 0 EMISSION : 1

#define M01\_CAN\_DLC 0 // Nombre d'octets de DONNES dans la trame CAN

// (DLC : Data Lenght Code)

// ---- Les fonctions de la bibliotheque a developper --------------------------

// (0) CAN\_init : initialisation du module CAN et des MO

// (1) CAN\_send : EMISSION d'une trame de DONNEES

// (2) CAN\_receive : RECEPTION d'une trame de DONNEES

// (3) CAN\_reqremote : EMISSION d'une trame de REQUETTE

// (4) CAN\_setremote : "RECEPTION" d'une trame de REQUETTE

// ET "RENVOI" d'une trame de DONNEES

// -----------------------------------------------------------------------------

// ------ La fonction CAN\_init() ----------------------------------------------

//

// Initialise le module CAN et les MO: vitesse, ..., parametres des M0, ...

// Remarque : pour la partie definition des MO, cette fonction depend du role

// de chaque noeud CAN (elle donc specialisee pour chaque noeud) -> il vaut

// mieux ne pas l'inclure dans la bibliotheque

// -----------------------------------------------------------------------------

void CAN\_init()

{

// Initialisation du module CAN

init\_can\_16x(CAN\_BAUDRATE,CAN\_IE,0,0);

// Initialisation des Messages Objets

// - ces initialisations sont SPECIFIQUES a l'abonne considere

// - iL vaut mieux les placer dans le code de chaque abonne

}

// ------ La fonction CAN\_send(int NumMO, char \*p, int size) ------------------

//

// Emettre une trame CAN de DONNEES via un MO : Charge une donnée a émettre

// puis envoie une trame de DONNEES.

// - paramètres : NumMO [in]: numéro du message objet utilisé,

// p [in]: pointe sur la donnée à envoyer,

// size [in]: longueur du message (nbr d'octets de données)

// - valeur rendue : 1 si la transmission a été effectuée,

// -1 si erreur

// -----------------------------------------------------------------------------

int CAN\_send(int NumMO, char \*p, int size)

{

void def\_mo\_16x(unsigned char nr, unsigned char xtd, unsigned long id,

unsigned char dir, unsigned char dlc, unsigned char txie,

unsigned char rxie);

/\* Define message object: MO-number, xtd, id, dir, dlc, TXIE, RXIE

/\* incan16x.c \*/

void ld\_modata\_16x(unsigned char nr, unsigned char \*upl\_data\_ptr);

/\* load data bytes of a message object (1..14) \*/

/\* rdm1516x.c \*/

return 1024 ;

}

while(1);

// ------ La fonction CAN\_receive(int NumMO, char \*p) -------------------------

//

// Recevoir une trame CAN de DONNEES via un MO : VERIFIE si une trame de DONNEES

// est arrivée, et si oui, retourne un pointeur sur les DONNEES recues.

// Attention cete fonction est non blocante

// - paramètres : NumMO [in] : numéro du message objet utilisé,

// p [out]: pointe sur la donnée recue

// - valeur rendue : la longeur du message (longueur qui est >0),

// 0 si pas de message,

// -1 si erreur

// -----------------------------------------------------------------------------

/\*

int CAN\_receive(int NumMO, char \*p)

{

return 1024 ;

}

// ------ La fonction CAN\_reqremote(int NumMO) -------------------------------

//

// Emettre une trame CAN de REQUETTE via un MO : envoie une demande

// pour provoquer, chez un autre noeud CAN, une transmission automatique :

// - paramètres : NumMO [in]: numéro du message objet utilisé,

// - valeur rendue : 1 si la transmission a été effectuée,

// -1 si erreur

// REMARQUES :

// - Le MO doit être positioné en RECEPTION

// - un message "vide" doit être émis comme requête

// -----------------------------------------------------------------------------

int CAN\_reqremote(int NumMO)

{

return 1024 ;

}

// ------ CAN\_setremote(int NumMO, char \*p, int size)---------------------------

//

// Recevoir une trame CAN de REQUETTE via un MO : charge la donnee à émettre

// en transmission automatique quand une trame de requête arrivera

// - paramètres : NumMO [in]: numéro du message objet utilisé,

// p [in]: pointe sur la donnée à envoyer,

// size [in]: longueur du message (nbr d'octets de données)

// - valeur rendue : 1 si le chargement a été effectué,

// -1 si erreur

// REMARQUES :

// - Le MO doit être positionee en EMISSION

// -----------------------------------------------------------------------------

int CAN\_setremote(int NumMO, char \*p, int size)

{

return 1024 ;

}

\*/