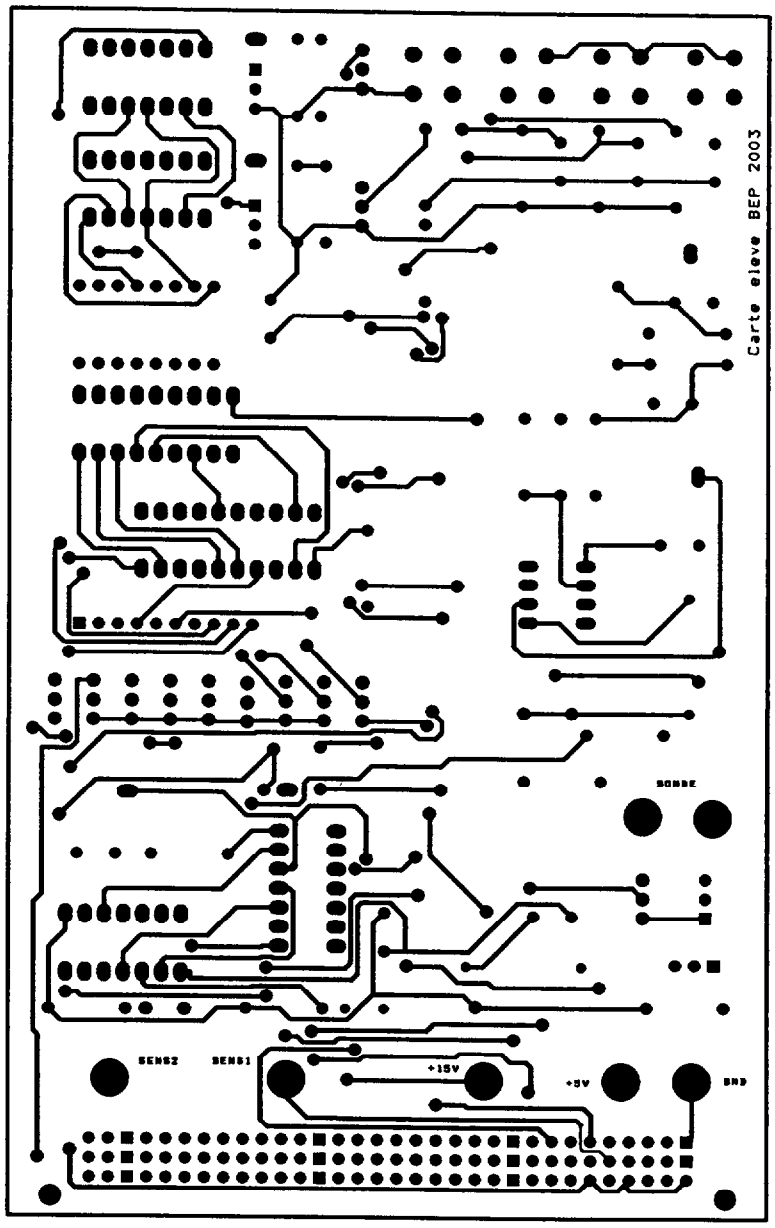
EXEMPLE DE CORRECTION

COTE COMPOSANT

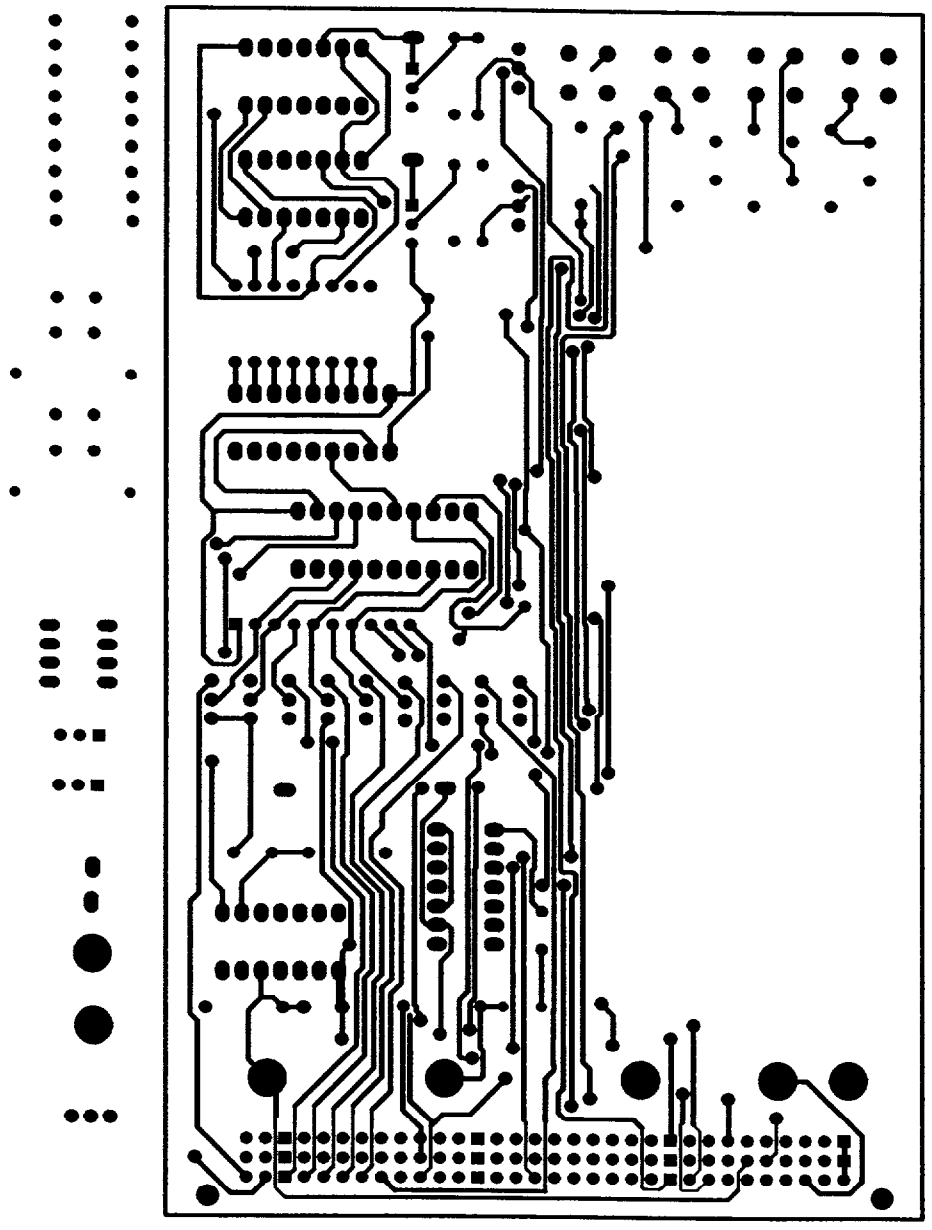


Tripode : Carte eleve

39/137

CARTE ELEVE A COMPLETER

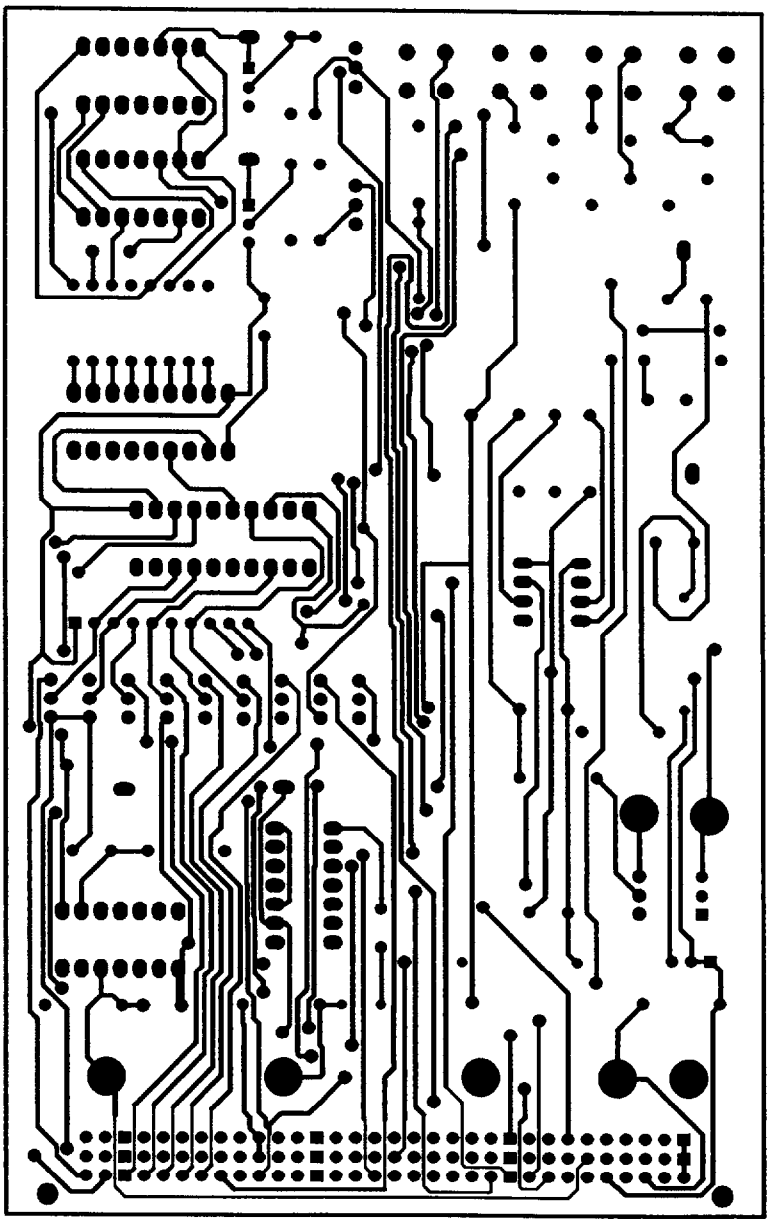
COTE CUIVRE



Tripode : Carte eleve

EXEMPLE DE CORRECTION

COTE CUIVRE



Tripode : Carte eleve

3. ETUDE LOGICIELLE

- Fonctionnement général
  - Mise en situation et schéma de raccordement
  - Algorigramme
  - Programme
- Traitement Température
  - Mise en situation et schéma simplifié
  - Algorithme
  - Programme
  - Travail demandé

FONCTIONNEMENT GENERAL

Mise en situation de la carte élève avec l'Objet Technique.

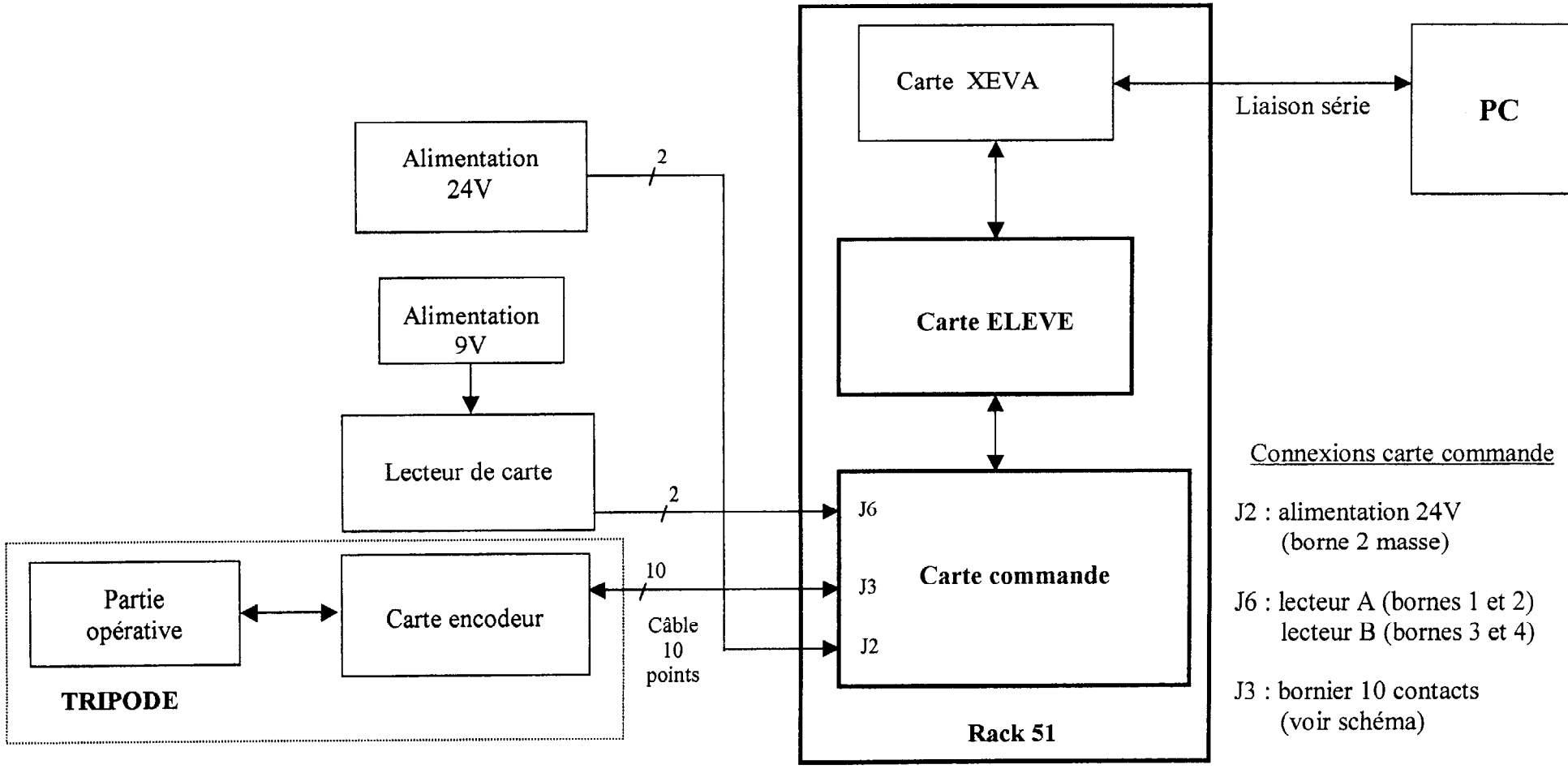
L'objectif final de cette activité est de mettre en œuvre un programme qui permet de tester la carte élève en fonctionnement normal avec le tripode.

Le projet à charger avec le logiciel "Ride" est Gestion_Tripode.prj associé au programme Tripode_main.c écrit en langage C. Le fonctionnement de ce programme est décrit par l'algorigramme « Gestion_Tripode » .

Pour ce mode de fonctionnement, la carte commande et la carte élève doivent être insérées dans le rack 51. Toutes les connexions à effectuer sont décrites sur le schéma de raccordement de la page suivante.

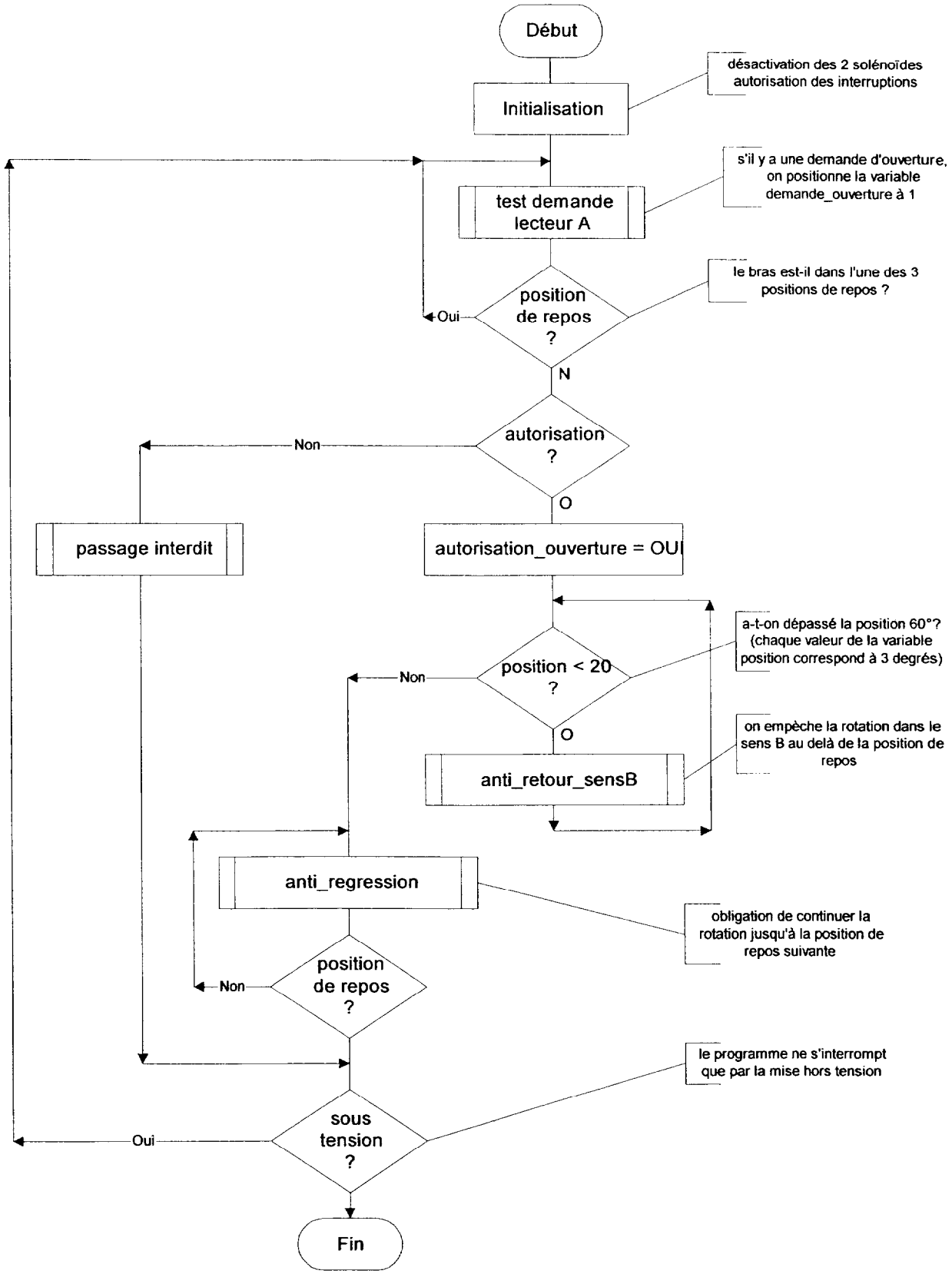
Sur la carte élève, tous les cavaliers JP0 à JP11 doivent être dans la position 1-2.

SCHEMA DE RACCORDEMENT GENERAL



Remarques : - toutes les liaisons sur la carte de commande se font en face avant (nez de carte)
- la liaison série PC ←→ XEVA se fait par l'arrière du Rack 51
- les cartes Elève et commande sont présentes en même temps dans le rack 51

ALGORIGRAMME DU PROGRAMME DE GESTION DU TRIPODE



```
/*****************************************************************************
            Programme principal de gestion du tripode
*****************************************************************************
Projet                          : Gestion_Tripode.prj
*****************************************************************************/
/* Définition des variables et constantes
*****************************************************************************/
#include <reg552.h>
#define INT1 2                          //INT1 est l'interruption n°2
#define active 0
#define inactive 1
#define allume 0
#define eteint 1
at 0xC2 sbit UD  ;                      //Signal U/D donnant le sens de la rotation en entrée sur P4.2
at 0xC6 sbit switch2;                   //switch2 en entrée sur P4.6
at 0xC5 sbit switch3;                   //switch3 en entrée sur P4.5
at 0xC4 sbit switch4;                   //switch4 en entrée sur P4.4
at 0x8000 char xdata status;            //adresse de lecture de sens 3 et de demande ouverture
at 0x0095 sbit SOL_A ;                  //Adresse de la commande du Solénoïde 1 en P1.5
at 0x0092 sbit  SOL_B ;                 //Adresse de la commande du Solénoïde 1 en P1.2
at 0x0093 sbit Led_Rouge;               //Adresse de la commande de la Led Rouge en P1.3
at 0X0094 sbit Led_Verte;               //Adresse de la commande de la Led Verte en P1.4
/*****************************************************************************/
/* Définition des variables globales ( Valables pour toutes les fonctions) */
int position = 0;                   /* Variable indiquant la position du tripode en degrés
                                    si position = 0, position repos
                                    si position > 0, rotation sens A
                                    si position < 0, rotation sens B */
bit autorisation_ouverture = 0;  //bit indiquant si une demande du lecteur A a été reçue
//*****************************************************************************
/* Déclaration des prototypes de fonctions  */
void test_demande_lecteurA(void);
void passage_interdit(void);
void anti_regression(void);
void anti_retour_sensB (void);
void initialisation (void);
/*****************************************************************************
Fonction principale
Gère le fonctionnement du tripode dans le sens de passage A suivant le fonctionnement décrit dans
la documentation technique.
Interdit systématiquement le passage dans le sens B.
*****************************************************************************/
void main (void)
{
        initialisation();           //initialisation convertisseur, timer, interruptions, etc ...
        do                          //boucle infinie
        {
                do
                {
                        test_demande_lecteurA();
                        position = 0;
                }
        while((status & 0x04) == 0);
```

```
        while((status & 0x08) == 0)
        {
                passage_interdit();
        }

        if ( autorisation_ouverture )
        {
                Led_Rouge = eteint;
                autorisation_ouverture = 0;
                do
                {
                        anti_retour_sensB();
                }
                while (position != 20);
                do
                {
                        anti_regression();
                }
                while( (status & 0x04)!= 0 );
        }
        else
        {
                passage_interdit();
        }
        position = 0;
        Led_Verte = eteint;
        Led_Rouge = allume;
        }
        while(1);
}
/* Fin de fonction principale  */
//*****************************************************************************
//Fonction interruption
/* Permet d'effectuer une incrémentation ou décrémentation de la variable position en fonction
 du signal U/D à chaque interruption sur INT1    -> Incrémentation si U/D = 1
                                                 -> Décrémentation si U/D à 0.
*****************************************************************************/
void comptage_rotation () interrupt INT1
{
        if ((status & 0x04) == 0 )
        {
                position = 0;
        }
        else
        {
                if (UD == 1)
                {
                        position++;
                }
                else
                {
                        position--;
                }
        }
}
```

```
/*****************************************************************************
Fonction test_demande_lecteurA.
 La fonction vient tester le signal AUTOR présent à l'adresse status (0x8000)en bit 3.
Si autor est à 1, mémorisation de la demande dans la variable autorisation_ouverture.
*****************************************************************************/
void test_demande_lecteurA(void)
{
        if ( (status & 0x08) == 0)
        {
                autorisation_ouverture = 1;
                Led_Verte = allume;
                Led_Rouge = eteint;
        }
}
/*****************************************************************************
Fonction Passage_Interdit
Active SOL_A tant que la position du tripode est supérieure à 10° ou SOL_B tant que la position
du tripode est inférieure à -10°.
*****************************************************************************/
void passage_interdit()
{
        do
        {
                while( position > 3)
                        {
                        SOL_A = active;
                        }
                while( position < -3)
                        {
                        SOL_B = active;
                        }
                SOL_A = inactive;
                SOL_B = inactive;
        }
        while( (status & 0x04) != 0 );
}
/*****************************************************************************
Fonction anti_retour_sensB.
Active le solénoïde B tant que la position du tripode est inférieure ou égale à -10°.
Sortie de la fonction des que la position est supérieure à -10°.
*****************************************************************************/
void anti_retour_sensB (void)
{
        while( position <= -3 )
        {
                SOL_B = active;
        }
        SOL_B = inactive;
}
```

```
/*****************************************************************************
Fonction anti_regression.
*****************************************************************************/
void anti_regression(void)
{
        while ( position < 20 )
        {
                SOL_B = active;
        }
        SOL_B = inactive;
}
/*****************************************************************************
Fonction Initialisation.
*****************************************************************************/
void    initialisation (void)
{
        Led_Verte = eteint;
        Led_Rouge = allume;
        SOL_A = inactive;               //Solénoïde 1 au repos
        SOL_B = inactive;               //Solénoïde 2 au repos
        EA = 1;                         //Autorisation des interruptions
        IT1 = 0;                        //Prise en compte de int1 sur un niveau 0
        EX1 = 1;                        //Autorisation de la prise en compte de int1
}
```

TRAITEMENT TEMPERATURE

Mise en situation.

L'objectif de cette activité est de :

- Réaliser l'étude logicielle mettant en œuvre les outils logiciels associés au kit développement à microcontrôleur
- Vérifier le fonctionnement de la carte élève
- Valider l'étude qualitative et quantitative des fonctions FP4 et FP5.

Pour ce mode de fonctionnement, la carte commande et la carte élève doivent être insérées dans le rack 51. Aucun élément extérieur n'est nécessaire.

Sur la carte élève, tous les cavaliers JP0 à JP10 doivent être dans la position 1-2.

Le projet à charger avec le logiciel ride est Temp_7seg.prj associé au programme Température_7seg_main.c écrit en langage C.

Cavalier JP11 en position 1-2

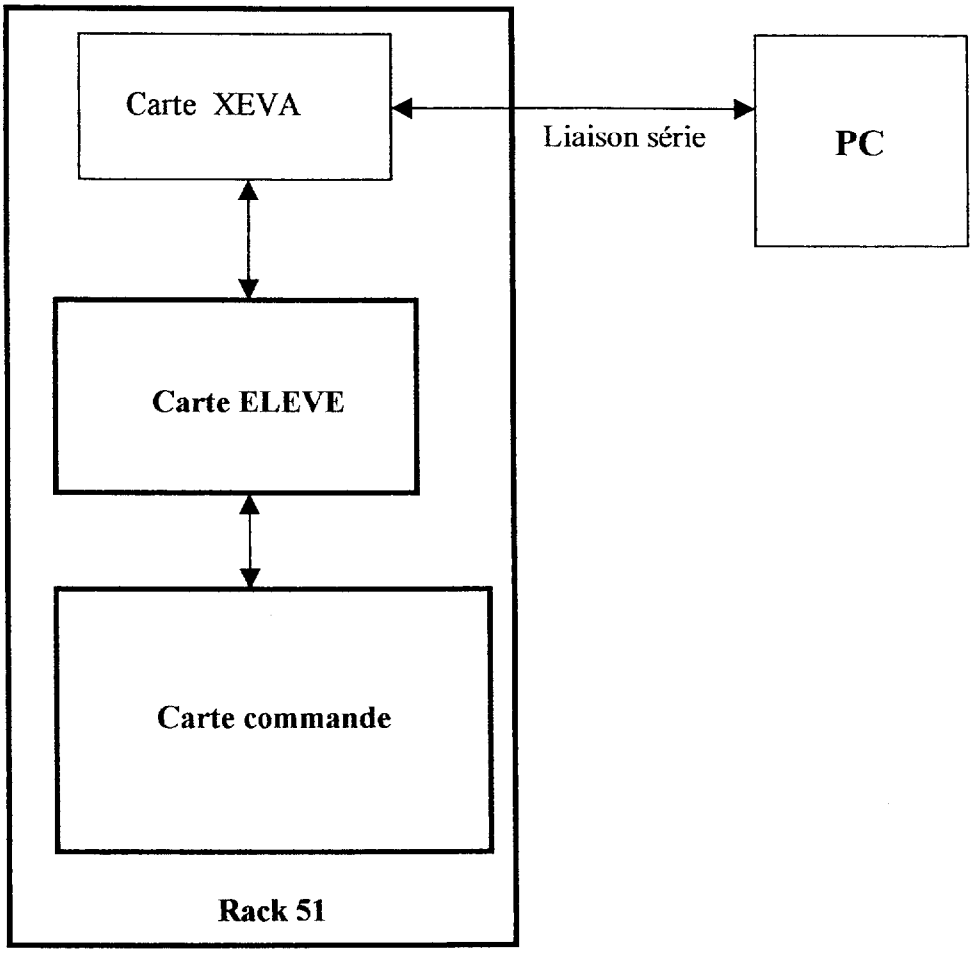
Le programme permet d'afficher sur les deux afficheurs la température présente au niveau du capteur de température KTY81-210 présent sur la carte élève.

Cavalier JP11 en position 2-3

Le capteur étant déconnecté, une boite à décade connectée entre les douilles J22 et J23 permettra de simuler des mesures de températures précises.

Attention : il ne doit pas y avoir de carte Xéva-démo dans le rack en même temps que les cartes commande et élève. (Problèmes de fonctionnement du convertisseur analogique/numérique).

Schéma de raccordement



Remarques : - la liaison série PC ←→ XEVA se fait par l'arrière du Rack 51
- les cartes Elève et Commande sont présentes en même temps dans le rack 51
- les cavaliers JP0 à JP10 doivent être placés en position 1-2

Algorithme de fonctionnement du programme

Température_7seg_main.C

Début

Déclaration et initialisation des variables
Initialisation du timer 0 en mode 01
Tant que « sous tension »
initialisation du convertisseur analogique/numérique
Tant que « convertisseur non prêt »
Fin de tant que
lancement de la conversion
Tant que « conversion non fine »
Fin de tant que
lecture du résultat de la conversion
calcul de la température à afficher
extraction des dizaines
extraction des unités
répéter
arrêt affichage
recherche du code correspondant aux unités
envoi du code vers les afficheurs
commande afficheur droit
temporisation 5 ms
arrêt affichage
recherche du code correspondant aux dizaines
envoi du code vers les afficheurs
commande afficheur gauche
temporisation 5 ms
incrémentation de la variable comptant le nombre d'affichages
jusqu'à « nombre d'affichages = 50 »
remise à zéro du nombre d'affichages
Fin de tant que

Fin

Le programme Température_7seg_main.c permet d'afficher sur les deux afficheurs 7 segments la température présente au niveau de la sonde KTY10 sur la carte élève. Le signal V_{TEMP} est envoyé sur l'entrée de conversion $P_{5.4}$ du micro contrôleur. La conversion analogique/numérique est effectuée sur 8 bits. Le résultat de la conversion est alors traité par le calcul suivant : Température = (résultat – 148) / 0.68).
Il y a ensuite séparation dans deux variables des unités et des dizaines puis envoi vers les afficheurs des codes correspondant aux valeurs à afficher et affichage de ces valeurs par un affichage multiplexé. La durée d'affichage est de 5 ms sur chaque afficheur. Il y a une mesure de température toutes les 500 ms.

```
/*****************************************************************************
        Programme de gestion de la température
*****************************************************************************
Projet          : Température_7seg.prj
*****************************************************************************/
#include <reg552.H>     /* Fichier de déclaration des SFR du 80C552*/

char at 0xD000  xdata   commande_afficheurs;    //adresse de la commande des afficheurs DISP0 et DISP1
char at 0xE000  xdata afficheur;                //adresse des afficheurs (data à afficher)

#define droit  0x40     //équivalence permettant de commander l'afficheur droit
#define gauche 0x80     //équivalence permettant de commander l'afficheur gauche
#define arret  0xC0     //équivalence permettant de commander l'arrêt des deux afficheurs

void temporisation (void);

unsigned char Table_caractere[10] = {   0x3F, 0x06, 0x5B, 0x4F,  //Tableau des codes permettant
                                        0x66, 0x6D, 0x7D, 0x07,  //d'afficher 0 à 9 sur les
                                        0x7F, 0x6F};             //afficheurs 7 segments
unsigned int Nb_Affichage = 0;
unsigned char resultat;
unsigned char temperature;
unsigned char dizaine;
unsigned char unite;
unsigned char code_unite;
unsigned char code_dizaine;
/* Fonction principale */
void main(void)
{
        TMOD = (TMOD | 0x01);//Timer 0 en mode 1
        TMOD = (TMOD & 0xFD);
        while(1)                    //Boucle infinie
        {
                EAD = 0;            //désactive la prise en compte de l'interruption de fin de conversion
                ADCON = (ADCON & 0xC8);     //mise à 0 des bits 1,2,4,5 de ADCON
                ADCON = (ADCON | 0x04);     //ADDR0 = 0, ADDR1 = 0, ADDR2 = 1
                                            //--sélection de l'adresse 4 (P5.4)--
                                            //ADEX = 0, lancement de la conversion par logiciel
                                            //ADCI = 0, mise à 0 du bit de fin de conversion
                while( ADCON & 0x18 ) //test si ADCI et ADCS = 0 --attente convertisseur prêt --
                {
                }
                ADCON = (ADCON | 0x0C);     //Déclenchement de la conversion
                while( !( ADCON & 0x10))    //Attente de fin de conversion
                {
                }
                resultat = ADCH;                        //Lecture du résultat de la conversion
                temperature =  ((resultat - 148) / 0.68);   //Calcul de la température en fonction du résultat
                                                        // de la conversion
                dizaine = temperature / 10;             //Extraction des dizaines
                unite = temperature - (dizaine * 10 );  //Extraction des unités
                do
                {
                        commande_afficheurs = arret;            //Arrêt de l'affichage
                        code_unite = Table_caractere[unite];    //Transfert du code des unités
                        afficheur = code_unite;                 //Envoi du code unité vers les afficheurs
                        commande_afficheurs = droit;            //Commande de l'afficheur droit
                        temporisation();                        //Temporisation de 5 milisecondes
                        commande_afficheurs = arret;            //Arrêt de l'affichage
                        code_dizaine = Table_caractere[dizaine]; //Transfert du code dizaine
                        afficheur = code_dizaine;               //Envoi du code dizaine vers les afficheurs
```

```
                commande_afficheurs = gauche;        //Commande de l'afficheur gauche
                temporisation();                     //Temporisation de 5 milisecondes
                Nb_Affichage ++;                     //Incrémentation de la variable Nb_Affichage
            }
            while (Nb_Affichage < 50);       //Bouclage tant que le nombre d'affichages est inférieur à 50
            Nb_Affichage = 0;                //Remise à zéro du nombre d'affichages
        }
}           //Fin du main
/*****************************************************************************************
Fonction Temporisation
*****************************************************************************************/
void temporisation(void)
{
        TR0 = 0;                //Arrêt du timer 0
        TF0 = 0;
        TH0 = 0xEC;             :/Chargement du registre de poids forts du timer 0
        TL0 = 0x77;             //Chargement du registre de poids faibles du timer 0
        TR0 = 1;                //démarage du timer 0
  while (TF0 == 0);             //Attente de la fin du comptage
}
```

Travail demandé.

Pré requis : Donner l'expression de V_{temp} (d.d.p. avant conversion) en fonction de R_{KTY}

1) Etablir l'équation de la courbe représentative de la tension (après conversion) en fonction de la température (cette courbe sera assimilée à une droite, la valeur convertie est exprimée en décimal).

   Pour les questions 2 et 3, remplacer le capteur par une résistance dont la valeur correspond à une température de 25°C.

2) Déterminer la valeur (après conversion) de la d.d.p. à 25°C .

   Vérifier dans le logiciel cette valeur de d.d.p.

   Vérifier dans le logiciel la valeur de la température correspondant à cette d.d.p.

3) Rechercher dans le logiciel les variables correspondant aux unités et aux dizaines et comparer les valeurs de ces variables à la température affichée

4. ETUDE STRUCTURELLE

- Etude de FP5
  - Calculs
  - Simulation
- Etude de FS22 (comptage)
  - Simulation et chronogramme

ETUDE DE FP5
« SURVEILLANCE DE LA TEMPERATURE »

Analyse quantitative.

Cette étude permet d'élaborer la formule utilisée dans le projet « temp_7seg.prj » servant à afficher la température. Vous trouverez, dans les pages suivantes, une simulation de FP5 qui valide l'étude structurelle qui suit.
Au cours de ces calculs, il est très important de ne pas faire d'approximations car les écarts en fin de calcul seraient significatifs.

ATTENTION : SUR LE SCHEMA STRUCTUREL FOURNI PAR LE CONSTRUCTEUR, LA RESISTANCE R47 DEVRAIT ETRE MARQUEE 22k

Expression V_{T1} en fonction V_{REF} et R_{KTY}

$$V_{REF} = \frac{V_{T1} \times R_{43}}{R_{43} + R_{KTY}} \qquad \text{et} \qquad V_{T1} = \frac{V_{REF} \times (R_{43} + R_{KTY})}{R_{43}}$$

$$V_{T1} = \frac{V_{REF} \times (10 + R_{KTY})}{10}$$

Expression V_{T2} en fonction V_{REF} et R_{KTY}

La d.d.p. V_3 est présente sur la broche 3 de U17A

$$V_3 = \frac{V_{T1} \times R_{42}}{R_{42} + R_{45}} = 0{,}99 \times V_{T1} \qquad \text{donc} \qquad V_3 = \frac{V_{REF} \times (10 + R_{KTY})}{10} \times 0{,}99$$

par le théorème de superposition, on peut écrire :

$$V_3 = \frac{V_{T2} \times R_{41}}{R_{41} + R_{46}} + \frac{V_{REF} \times R_{46}}{R_{41} + R_{46}}$$

$$V_{T2} = \frac{V_3 \times (R_{41} + R_{46}) - V_{REF} \times R_{46}}{R_{41}}$$

$$V_{T2} = \frac{\frac{V_{REF}}{10} \times (10 + R_{KTY}) \times 0{,}99 \times (R_{41} + R_{46}) - V_{REF} \times R_{46}}{R_{41}}$$

$$V_{T2} = V_{REF}(0{,}5643 \times R_{KTY} + 0{,}943)$$