Document d'accompagnement du programme
DiSEqC06.bas
(version 04.05.05)

Auteurs

LOPES Léonel, lycée de Bischheim : mouvements de base, gestion du clavier, gestion de la mémorisation.

MAQUAIRE Manuel, lycée Marcel Rudloff, Strasbourg : ajout de quelques fonctions de déplacement, de la gestion afficheur, de la génération des signaux de synchronisation pour un oscilloscope externe ; création du fichier d'accompagnement.

But

Ce programme permet la gestion du déplacement de la parabole HH100 (STAB) *via* une carte ControlBoy F1. Un clavier de seize touches est mis en œuvre, ainsi que qu'un afficheur LCD de deux lignes de seize caractères.

En outre, des signaux de synchronisation sont générés sur le port A, qui permettent de stabiliser l'écran d'un oscilloscope sur un octet quelconque du signal DiSEqC.

Utilisation du programme

Le programme `DiSEqC06.bas` *nécessite deux autres fichiers (à placer dans le même répertoire) :*

- `StartCF1.bas` (fourni par ControlLord) ;
- `Driver2.bas` (fichier `Driver.bas` fourni par ControlLord, légèrement modifié)

Lors du lancement, le programme se place dans un état d'attente d'une commande, *via* le clavier.

Pointer au sud

0 : positionnement de la parabole au sud (position zéro). Une fois cet ordre demandé, on ne peut pas l'annuler.

Rotation à l'est

1 : demande de mouvement angulaire minimum de la parabole (pas-à-pas).

4 : rotation pendant une seconde. Si cette touche est maintenue enfoncée, on a alors un mouvement continu de la parabole.

7 : rotation pendant dix secondes. Cette touche permet d'avoir le temps de faire des mesures sur le système, sans presser une touche de façon continue.

Rotation à l'ouest

2 : demande de mouvement angulaire minimum de la parabole (pas-à-pas).

5 : rotation pendant une seconde. Si cette touche est maintenue enfoncée, on a alors un mouvement continu de la parabole.

8 : rotation pendant dix secondes. Cette touche permet d'avoir le temps de faire des mesures sur le système, sans presser une touche de façon continue.

Arrêt

3, 6 et 9 : arrêt du mouvement du positionneur, sauf suite à une demande de position zéro.

Mémorisation et rappel de positions

A puis 1 à 9 : mémorisation de la position actuelle du positionneur.

B puis 1 à 9 : rappel d'une position mémorisée.

C : annulation de la mémorisation ou du rappel.

Utilisation des signaux de synchronisation de l'oscilloscope

Sur le port A, quatre signaux sont générés :

– PA1 est au NL1 lors de l'envoi du premier octet d'une trame DiSEqC.

– PA2 est au NL1 lors de l'envoi du deuxième octet d'une trame DiSEqC.

– PA3 est au NL1 lors de l'envoi du troisième octet d'une trame DiSEqC.

– PA4 est au NL1 lors de l'envoi du quatrième octet d'une trame DiSEqC (ce quatrième octet n'existe pas pour tous les ordres).

Il suffit donc de connecter par exemple la voie A d'un oscilloscope sur le signal DiSEqC (ou son homologue filtré : RFilt), la voie B sur PA2, et de synchroniser l'oscilloscope sur cette dernière voie pour visualiser le deuxième octet de la trame, de façon stable.

Remarques techniques

Le programme est écrit en Basic11, selon la décision prise au lors des GAR 2004/ 2005.

Principe général : on lit une touche, et on l'interprète en vérifiant toutes les possibilités, de façon linéaire.

Un *flag* est utilisé au sein de ce programme ; un *flag* (drapeau) est une variable dont la valeur représente un état du programme ou de son environnement, à un instant *t*. Ici, le *flag* peut prendre plusieurs valeurs, explicitées au sein de la section suivante.

Lors de l'interprétation de chacune des touches : un `if...then...[else...]` permet de rafraîchir ou non l'écran, afin de supprimer le scintillement qui apparaît lors de l'appui continu sur une touche

Lors de l'interprétation de la touche 0 (aller en position zéro / pointer au sud) : l'ordre est envoyé en même temps que la gestion de l'affichage. Ceci car, de tous les ordres traités ici, c'est le seul qui soit mémorisé par le circuit récepteur de la carte DiSEqC. En n'appliquant pas cette méthode, l'ordre serait lancé plusieurs fois lors d'un appui long. C'est aussi pourquoi on

attend 150 ms en cas de répétition.

Lors de l'interprétation des touches [1] ou [2] (déplacement vers l'est ou l'ouest, pas-à-pas) : en cas d'appui continu, une temporisation de 150 ms est intercalée entre les envois, afin d'avoir un mouvement pas-à-pas visible (et non pseudo-continu).

Mémorisation des positions : le HH100 (STAB) peut mémoriser la position de quarante-neuf satellites. Pour simplifier le programme et son utilisation, on ne gère ici que neuf positions différentes (touches [1] à [9]) ; la touche [0] est réservée à la position zéro (pointage au sud).

Sous-programme `SendMessage()` : l'ordre d'arrêt nécessite trois octets, contre quatre pour les autres ordres utilisés au sein du programme. Le dernier paramètre de cette fonction est placé à $FFF si l'on ne désire pas l'envoyer.

Flag

Un *flag* est utilisé pour la gestion de l'afficheur, afin de supprimer le scintillement visible lors de l'envoi d'une même donnée de façon cyclique. Les valeurs qu'il peut prendre et les états correspondants sont listés ci-dessous :

0	En attente depuis moins d'une seconde et demi.
10	Dernière commande = Est (pas-a-pas).
20	Dernière commande = Ouest (pas-à-pas).
30	Dernière commande = Arrêt.
40	Dernière commande = Est (pendant une seconde).
50	Dernière commande = Ouest (pendant une seconde).
70	Déplacement pendant dix secondes vers l'est en cours (reste entre neuf et dix secondes).
71	Déplacement pendant dix secondes vers l'est en cours (reste moins de neuf secondes).
80	Déplacement pendant dix secondes vers l'ouest en cours (reste entre neuf et dix secondes).
81	Déplacement pendant dix secondes vers l'ouest en cours (reste moins de neuf secondes).
100	Dernière commande = Demande de position initiale.
110	Procédure de mémorisation en cours (état inutilisé ici).
130	Procédure de rappel en cours (état inutilisé ici).
170	Aucune activité du clavier depuis environ une seconde et demi.