DiSEqC – Note d'application concernant la relève du signal DiSEqC

Maquaire Manuel (prof.maquaire@free.fr) – Lycée Marcel Rudloff – Janvier 2005

Constat

Sur la platine du système relatif à la nouvelle parabole, plusieurs choses ont été développées ici et là, dans différents lycées.

Globalement, deux familles de programmes ont été rencontrées :

– Le programme gère le déplacement de la parabole via des ordres envoyés selon la norme DiSEqC, mais il est impossible d'obtenir facilement un signal à l'oscilloscope. Dans la mesure où l'on envoie un signal logique qui représente des valeurs binaires sous forme série, l'oscilloscope ne peut pas se stabiliser correctement. Le *hold off* ne permet pas non plus de palier à ce problème.

– Le programme envoie un signal qui représente un octet unique, de façon continue. Le signal est alors exploitable, mais la parabole nécessite une trame comprise entre trois et cinq octets, donc elle ne tourne pas du tout.

En résumé, soit on fait tourner la parabole, soit on exploite un signal, mais pas les deux.

Suite à ce constat, je me suis penché sur le problème, et une solution très simple a été retenue, solution qui permet de visualiser un octet au choix, sur une trame quelconque.

Il suffit pour cela de générer des signaux annexes sur lesquels l'oscilloscope pourra se déclencher. Un de ces signaux sera à connecter à l'entrée *trigger*, ou sur une voie disponible.

Génération des signaux de synchronisation

Le port A n'étant pas utilisé, c'est celui-ci qui a été retenu pour les signaux annexes. Il faut donc configurer chacune de ses connexions en sortie en ajoutant la ligne de code suivante :

```
DDRA=$FF
```

Lors du début de la procédure d'envoi du premier octet, ajouter la ligne de code suivante :

```
PORTA=%00000010         ' Envoyer un top synchro' oscillo' sur PA1
```

Lors du début de la procédure d'envoi du second octet, ajouter la ligne de code suivante :

```
PORTA=%00000100         ' Envoyer un top synchro' oscillo' sur PA2
```

Etc.

Utilisation

On obtient un signal lié à chaque octet. Sur PA1, par exemple, le signal sera au NL0 constamment, et passera au NL1 le temps de la transmission du premier octet. En synchronisant l'oscilloscope sur PA1, on pourra donc visualiser cet octet, de façon stable.

Réalisation du boîtier

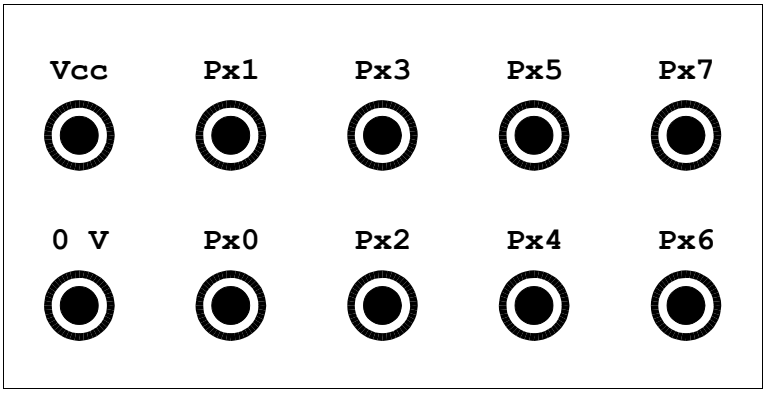
En plus d'ajouter ces quelques lignes de code, il est utile de réaliser un petit boîtier qui fera office de boîtier « gigogne ». En entrée, on connecte le port A ; en sortie, on ne connecte rien (pour notre utilisation) ; des bornes intermédiaires permettent de relier des fiches bananes IP2x pour utiliser tel ou tel signal. Ce boîtier sera donc utile avec bien des manipulations ControlBoy, et non seulement avec la carte DiSEqC.

Pour vous éviter de chercher les éléments dans quinze catalogues, voici une liste qui, envoyée chez Conrad, vous permettra d'avoir l'ensemble de ce qui est nécessaire :

Page	Description	Code	Quantité	PU HT	PT HT
379	Coffret Europa gris	520586-35	1	2,76	2,76
334	Douille 4mm verte	064557-35	10	0,82	8,2
398	Câble en nappe 14 points	609404-35	1	0,79	0,79
351	Fiche mêle HE10 2*5 autodénudant	743500-35	1	1,8	1,8
351	Fiche femelle HE10 2*5 autodénudant	702013-35	1	0,71	0,71
				Total HT :	14,26

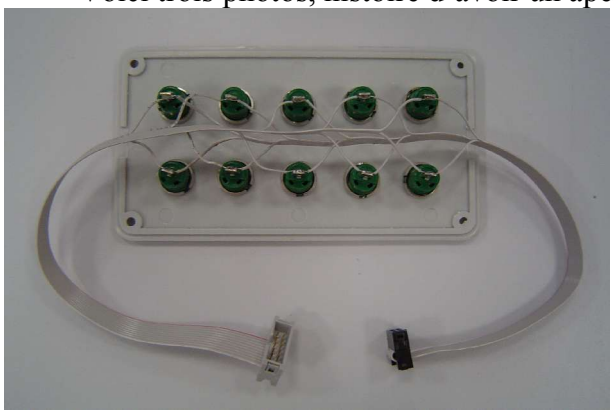
Note : Seulement dix des quatorze brins du câble en nappe sont nécessaires. 50 cm suffisent.

Pour des soucis pratiques, les bornes ont été placées selon l'organisation retenue par ControlLord :

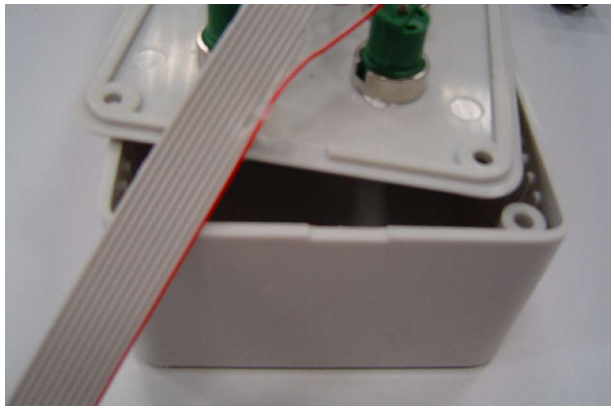


Notes : – « x » est le nom du connecteur du CBoy (pas toujours équivalent au port du 68HC11),
– « Vcc » représente Vcc ou Vpull-up (selon le connecteur).

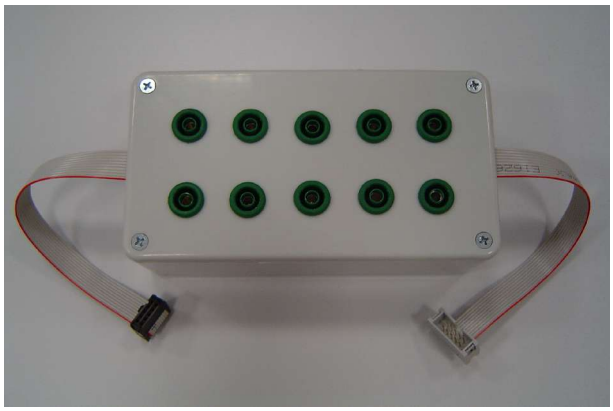
Voici trois photos, histoire d'avoir un aperçu final :



Câblage interne



Rilzan anti-arrachement et limage pour le passage de la nappe



Aspect final

Un programme

Listing de code d'un programme (non optimisé) en Basic11 qui permet d'envoyer de façon continue un ordre de quatre octets, avec gestion des signaux de synchronisation de l'oscilloscope :

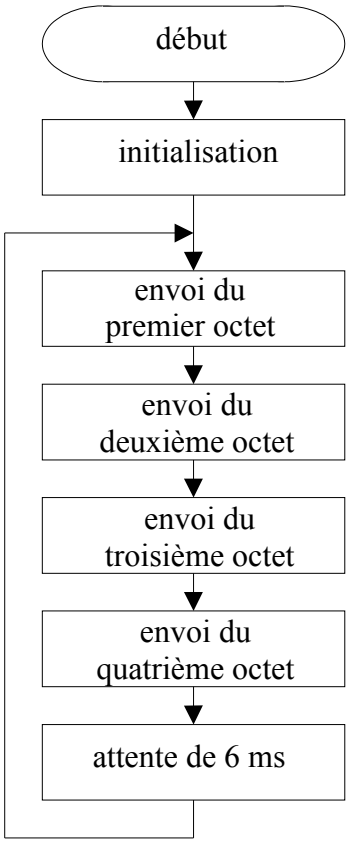
```
' Nom du programme     : SendOrdr.bas (Send order ; envoyer un ordre)
' But du programme     : Envoyer un ordre de commande de façon continue (4 octets)
' Dernière révision    : 17/12/2004
' Auteur               : MAQUAIRE Manuel, Lycée Marcel Rudloff


#include "startcf1.bas"

' Déclaration des variables
       Byte Octet1
       Byte Octet2
```

```
	Byte Octet3
	Byte Octet4
	byte Count, Parity, Mask

	Octet1=$E0
	Octet2=$31
	Octet3=$68                          ' Tourner à l'Est
	Octet4=$01

' Programme principal
	ASM CLI
	DDRA=$FF                            ' Port A en sortie
	do
' Envoi du premier octet
	PORTA=%00000010                     ' Envoyer un top synchro' oscillo' sur PA1
	Mask=%10000000                      ' Initialiser la masque
	Parity=0                            ' Initialiser la variable de parité
	for Count=0 to 7                    ' Envoyer les 8 bits
		if (Octet1 and Mask)=0 then ' Si LSB= NL0, envoyer un 0
			PORTB=$00           ' Activer le signal 22 kHz...
			T1ms()              ' ...pendant 1 ms
			PORTB=$01           ' Désactiver le signal 22 kHz...
			T0ms5()                     ' ...pendant 500 µs
		else                        ' Envoyer un 1
			PORTB=$00           ' Activer le signal 22 kHz...
			T0ms5()                     ' ...pendant 500 µs
			PORTB=$01           ' Désactiver le signal 22 kHz...
			T1ms()              ' ...pendant 1 ms
			Parity=Parity+1     ' Incrémenter le compteur de parité
		end if
		Mask=Mask/2                 ' Décalage à droite du masque
	next
	if (Parity and $01)=1 then          ' Si trame impaire...
		PORTB=$00                   ' Activer le signal 22 kHz...
		T1ms()                      ' ...pendant 1 ms
		PORTB=$01                   ' Désactiver le signal 22 kHz...
		T0ms5()                     ' ...pendant 500 µs
	else                                ' Si trame paire...
		PORTB=$00                   ' Activer le signal 22 kHz...
		T0ms5()                     ' ...pendant 500 µs
		PORTB=$01                   ' Désactiver le signal 22 kHz...
		T1ms()                      ' ...pendant 1ms
	end if
' Envoi du deuxième octet
	PORTA=%00000100
	Mask=%10000000
	Parity=0
	for Count=0 to 7
		if (Octet2 and Mask)=0 then
			PORTB=$00
			T1ms()
			PORTB=$01
			T0ms5()
		else
			PORTB=$00
			T0ms5()
			PORTB=$01
			T1ms()
			Parity=Parity+1
		end if
		Mask=Mask/2
	next
	if (Parity and $01)=1 then
		PORTB=$00
		T1ms()
		PORTB=$01
		T0ms5()
	else
		PORTB=$00
		T0ms5()
		PORTB=$01
		T1ms()
	end if
' Envoi du troisième octet
	PORTA=%00001000
	Mask=%10000000
	Parity=0
	for Count=0 to 7
		if (Octet3 and Mask)=0 then
			PORTB=$00
			T1ms()
			PORTB=$01
			T0ms5()
		else
			PORTB=$00
```

```
                        T0ms5()
                        PORTB=$01
                        T1ms()
                        Parity=Parity+1
                end if
                Mask=Mask/2
        next
        if (Parity and $01)=1 then
                PORTB=$00
                T1ms()
                PORTB=$01
                T0ms5()
        else
                PORTB=$00
                T0ms5()
                PORTB=$01
                T1ms()
        end if
' Envoi du quatrième octet
        PORTA=%00010000
        Mask=%10000000
        Parity=0
        for Count=0 to 7
                if (Octet4 and Mask)=0 then
                        PORTB=$00
                        T1ms()
                        PORTB=$01
                        T0ms5()
                else
                        PORTB=$00
                        T0ms5()
                        PORTB=$01
                        T1ms()
                        Parity=Parity+1
                end if
                Mask=Mask/2
        next
        if (Parity and $01)=1 then
                PORTB=$00
                T1ms()
                PORTB=$01
                T0ms5()
        else
                PORTB=$00
                T0ms5()
                PORTB=$01
                T1ms()
        end if
        T6ms()
        loop

' Sous-programme de temporisation de 500 µs
function T0ms5()
        PSHA
        LDAA    #211
Loop    NOP
        DECA
        BNE     Loop
        PULA
        RTS
end function

'Sous-programme de temporisation de 1 ms
function T1ms()
        T0ms5()
        T0ms5()
end function

'Sous-programme de temporisation de 6 ms
function T6ms()
        T1ms()
        T1ms()
        T1ms()
        T1ms()
        T1ms()
        T1ms()
end function
```

Algorigramme équivalent au programme SendOrdr.bas

Corps du programme :



Expansion de la partie permettant d'envoyer un octet :