ACADEMIE DE LYON
BACCALAUREAT STI GENIE ELECTRONIQUE

EPREUVE DE CONSTRUCTION ELECTRONIQUE

JUIN 2011

SYSTEME D'ECLAIRAGE DE SPECTACLE A COMMANDE DMX

DOSSIER TECHNIQUE

SOMMAIRE

1 - DESCRIPTION DU SYSTEME D'ECLAIRAGE

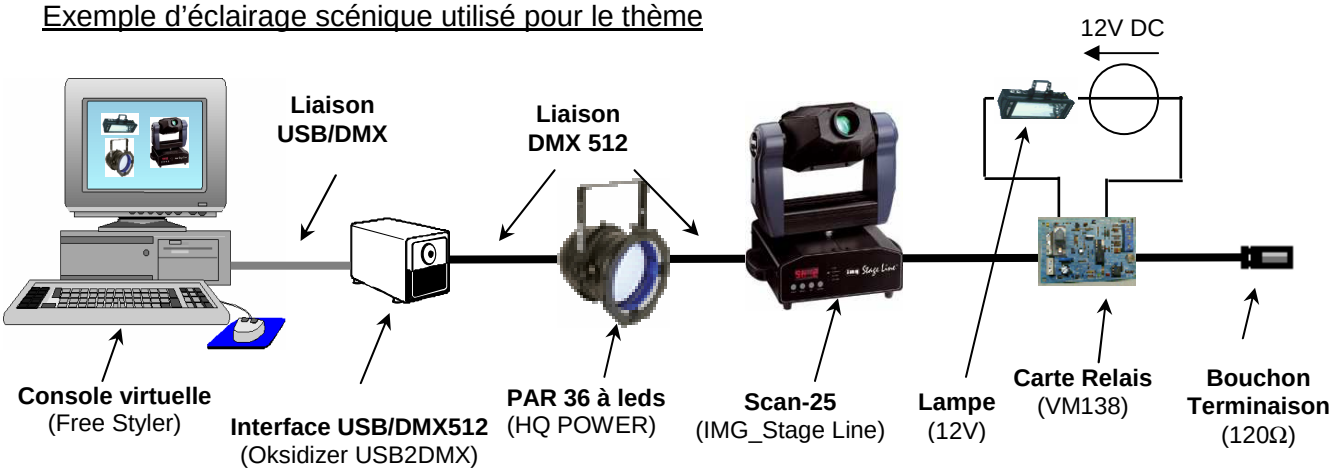
1-1) Organisation du système d'éclairage scénique utilisé pour le thème

Il est composé :

- De la lyre TWIST-25
- D'un projecteur à LEDS PAR36
- D'une commande par relais VM138
- D'une console virtuelle (réalisée à partir d'une application sur un ordinateur)
- D'une interface USB/DMX512

Tous les éléments sont interconnectés par l'intermédiaire d'un bus qui utilise le protocole DMX 512 pour communiquer les informations à chacun des dispositifs d'éclairage.

Exemple d'éclairage scénique utilisé pour le thème



1-2) Présentation du protocole DMX

Le DMX512 est un protocole de communication utilisé principalement pour commander des éclairages de scène.(D comme données et MX comme multiplexées)

Le DMX comporte 512 canaux multiplexés. Chacun canal était initialement prévu pour commander des niveaux de luminosité. On peut imaginer 512 potentiomètres sur un pupitre de commande reliés à 512 dispositifs d'éclairage. La position de chaque potentiomètre est envoyée par la liaison de données sous forme d'un octet série ayant une valeur équivalente décimale comprise entre 0 et 255. La valeur 0 correspond à une ampoule complètement éteinte et la valeur 255 à une ampoule complètement allumée.

Les progrès dans la conception d'éclairages, ont permis d'utiliser les mêmes valeurs de 0 à 255 pour commander la rotation d'un disque de gobos ou d'un disque de couleurs, du pan ou du tilt d'un miroir ou de l'ouverture d'une lentille.

Le protocole DMX est très simple et fiable.

Mise en place du câblage DMX

Chaque éclairage dans un système DMX est branché sur le suivant au moyen d'un cordon spécial DMX équipé généralement d'une fiche XLR à 3 ou 5 broches. Chaque éclairage possède une fiche d'entrée (XLR mâle) et une fiche de sortie (XLR femelle).

Une commande est installée à une extrémité de la chaîne et une résistance de 120 Ohms sert de « bouchon » à l'autre extrémité. Ce bouchon absorbe la puissance du signal qui serait autrement renvoyé dans le câble et perturberait la transmission de données. Cela dit, il est possible de faire fonctionner un petit nombre d'appareils sans bouchon de fin.

Chaque éclairage dans une installation DMX requiert un certain nombre de canaux pour fonctionner.

Ainsi, la lyre Stage Line TWIST-25 DMX a besoin de 5 canaux.

- Le premier canal utilisé pour l'effet tremblement et correspond à l'adresse du récepteur.
- Le second canal est utilisé pour le choix du gobo
- Le troisième canal est utilisé pour le choix de la couleur
- Le quatrième canal est utilisé pour le PAN
- Le cinquième canal est utilisé pour le TILT.

Dans une installation d'éclairage DMX, le premier effet de lumière sera réglé sur le canal 1. S'il a besoin de 4 canaux, cela veut dire que le prochain canal disponible sera le canal 5. Si un autre récepteur est ajouté, comme par exemple un scanner à 6 canaux, le prochain canal disponible sera le canal 11. Vous pouvez continuer ainsi à ajouter des récepteurs jusqu'à ce que tous les 512 canaux soient utilisés.

Selon le jeu de lumière utilisé, il est possible de régler l'adresse de différentes manières :

- Soit directement dans un menu accessible par bouton de choix et un affichage du numéro de canal (c'est le cas de la lyre TWIST 25)

- Soit à partir de commutateurs DIP permettant de fixer la valeur du canal en fonction de leur position (ON, OFF)

Les Commandes DMX

Les commandes servent à obtenir l'effet désiré comme par exemple le positionnement de la tête du projecteur, la sélection d'un gobo, d'une couleur pour chaque jeu de lumière.

L'effet désiré peut être obtenu manuellement en effectuant le réglage de chaque canal.

L'effet désiré peut aussi être mémorisé dans la commande et ensuite les réglages peuvent être modifiés pour créer un autre effet qui sera à son tour mémorisé.

Lorsque tous les effets ont été réglés, il est possible, selon les commandes, de faire défiler les effets dans l'ordre pour créer un show de lumière.

Les différents modèles de commande offrent un nombre différent de séquences et d'étapes par séquence. Ils peuvent contrôler la vitesse avec laquelle les effets changent et certains modèles possèdent un microphone ou une entrée pour une activation par la musique. Les modèles plus évolués ont des touches pour commuter rapidement entre les différents programmes et offrent la possibilité d'insérer ou d'éliminer des lumières individuelles ou des groupes de lumière d'une séquence.

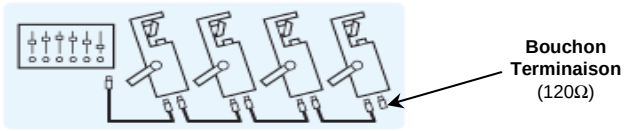
Ils permettent même le contrôle manuel de chaque lumière.

Comme le DMX est un standard reconnu mondialement, on peut utiliser des récepteurs de plusieurs fabricants dans un même système.

Câbles DMX & Bouchons de fin

Bien que les cordons DMX ressemblent à de simples câbles de microphone, ils sont en fait fabriqués à partir de câbles spéciaux appairés et torsadés qui garantissent une bonne transmission des signaux DMX. Dans une installation typique, chaque effet est relié au suivant en commençant par la commande DMX et en finissant par un bouchon de fin. La plupart des effets DMX présentent une entrée et une sortie pour faciliter cette configuration.

Il est recommandé de monter un bouchon de terminaison à la fin de la boucle DMX.



2 - DESCRIPTION DE LA LYRE

La lyre Stage Line TWIST 25 est un jeu de lumière destiné aux semi professionnels.

La lyre peut être commandée via une interface DMX512, une télécommande ou un micro intégré.

Les domaines d'utilisation sont: scène, discothèques, loisirs. L'appareil possède un pupitre de commande avec affichage.

Pour un montage au plafond des lyres, l'affichage à LED peut être inversé de 180°.

La lyre DMX comporte les éléments suivants :

- 5 fonctions de commande DMX
- Une gestion par la musique via le micro intégré
- Un mode Master / Slave avec 2 programmes Show courts

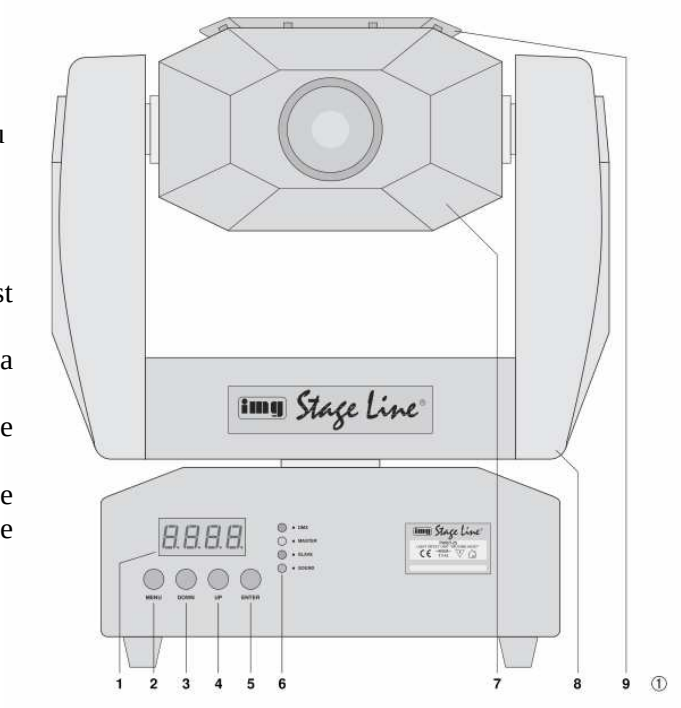
Remarque : La commande peut être géré par la mini télécommande LC-3 ou FSC-3, disponible en option.

2-1) Caractéristiques de la lyre

- Résolution horizontale et verticale 255 paliers respectivement
- Angle de rotation horizontal 540°, vertical 270°
- Roue de couleurs avec 11 couleurs plus blanc
- Roue de gobos avec 14 gobos plus spot et shutter, effet strobe / effet gobo switch avec vitesse variable (1-7 éclairs / seconde)
- Compteur des heures de fonctionnement
- Lampe halogène correspondante (24 V/250 W/GX5,3 : par exemple (HLG-24/250MRL)
- Possibilité d'éteindre la lampe par le DMX
- Livrée avec oeillet pour un montage au plafond
- Alimentation : 230 V˜/50 Hz/300W
- Dimensions : 33 x 38 x 29 cm
- Poids : 11 kg

2-2) Description des fonctionalités de la lyre

Face avant :



(1) Affichage

(2) Touche MENU pour appeler le menu de réglage

(3) Touche DOWN pour sélectionner un réglage dans le menu

(4) Touche UP pour sélectionner un réglage dans le menu

(5) Touche ENTER pour confirmer un réglage de menu

(6) LEDs de contrôle, témoin du mode de fonctionnement :
DMX = brille si le signal de commande DMX est présent à l'entrée DMX IN (17)
MASTER = mode Master : commande via la télécommande LC-2 ou le micro intégré
SLAVE = mode Slave (esclave) : commande via une seconde TWIST-25
SOUND = brille brièvement lorsque l'appareil modifie la couleur, le modèle ou le mouvement via un signal de musique

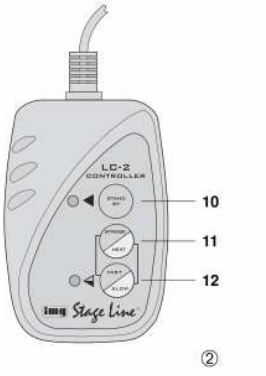
(7) Tête orientable

(8) Bras de rotation

(9) Couvercle pour le compartiment lampe

Télécommande :

La télécommande est disponible en option et n'est pas livrée avec la lyre TWIST-25.



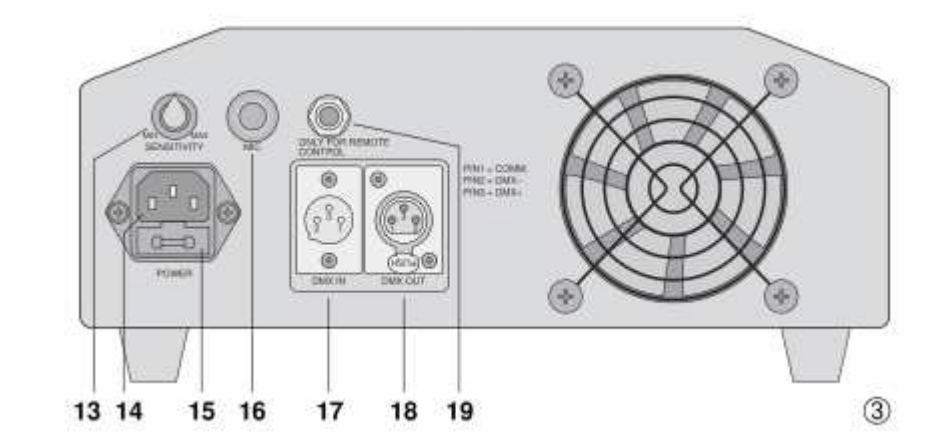
(10) Touche STAND BY pour la fonction Blackout (tête orientable sur la position de départ et lumière éteinte)

(11) Touche STROBE/NEXT : en mode Fast (rapide) pour activer un effet stroboscope (tant que la touche est enfoncée)

(12) Touche FAST/SLOW pour commuter entre les modes Fast et Slow (en mode Slow, la LED brille)

Conseil : Pour une commande via la télécommande, aucun signal DMX ne doit être présent à l'entrée DMX IN (17).

Face arrière :



(13) Réglage de la sensibilité du micro ; en tournant le réglage dans le sens des aiguilles d'une montre, la sensibilité augmente.
(14) Prise POWER pour relier l'appareil à une prise secteur 230 V~/50 Hz via le cordon de liaison livré
(15) Porte-fusible ; tout fusible fondu doit être remplacé par un fusible de même type.
(16) Microphone pour la commande via la musique
(17) Entrée DMX (1 = masse, 2 = DMX-, 3 = DMX+)
(18) Sortie DMX (1 = masse, 2 = DMX-, 3 = DMX+)
(19) Prise de branchement pour la télécommande LC-2

2-3) Consignes de sécurité sur l'utilisation de la lyre

Cet appareil n'est conçu que pour une utilisation en intérieur. Protégez-le de tout type de projections d'eau, des éclaboussures, d'une humidité élevée et de la chaleur (plage de température de fonctionnement autorisée : 0 - 40 °C).

Pendant le fonctionnement, la tête orientable (7) chauffe fortement. Pour éviter toute brûlure, ne touchez pas la tête pendant le fonctionnement de l'appareil, laissez-la refroidir quelques minutes après l'arrêt de l'appareil avant de la toucher.

Ne faites rien tomber dans les ouïes de ventilation, vous pourriez vous électrocuter.

Ne faites pas fonctionner l'appareil ou débranchez-le immédiatement du secteur lorsque :

1. des dommages apparaissent sur l'appareil ou sur le cordon secteur,
2. après une chute ou un cas similaire si vous avez un doute sur l'état de l'appareil,
3. des défaillances apparaissent.

Attention !

L'appareil est alimenté par une tension dangereuse 230 V~. Ne touchez jamais l'intérieur de l'appareil car en cas de mauvaise manipulation, vous pourriez subir une décharge électrique mortelle.

Ne débranchez jamais l'appareil en tirant sur le cordon secteur, tenez-le toujours par la prise.

Lorsque vous transportez la lyre, prenez-la toujours par le socle. En aucun cas vous ne devez la porter en la tenant par la tête orientable (7) ou le bras (8).

Utilisation via un contrôleur

La TWIST-25 dispose de 5 canaux DMX pour être utilisée via un contrôleur.

1. canal : activation du changement continu des couleurs et modèles ou de l'effet tremblement
2. canal : commande des modèles
3. canal : commande de la couleur
4. canal : rotation de la tête orientable
5. canal : inclinaison de la tête orientable

1) Reliez la prise DMX IN (17) via un cordon XLR 3 à la sortie DMX du contrôleur.
2) Reliez la prise DMX OUT (18) à l'entrée DMX de l'appareil suivant, puis reliez la sortie de ce dernier à l'entrée de l'appareil suivant et ainsi de suite jusqu'à ce que l'ensemble des jeux de lumière soit relié, la prise DMX OUT du dernier appareil sera reliée au bouchon de terminaison.
3) Réglez l'adresse de démarrage DMX qui correspond au canal du contrôleur DMX512 prévu pour la commande du canal 1 de la TWIST-25 :

a) Enfoncez la touche MENU (2) une fois jusqu'à ce que l'affichage indique Addr (adresse).

b) Enfoncez la touche ENTER (5). L'affichage clignote.

c) Avec la touche DOWN (3) ou UP (4), réglez l'adresse entre 1 et 512.

d) Enfoncez ensuite la touche ENTER. Si elle n'est pas enfoncée dans les 8 secondes, l'appareil revient au réglage précédent.

2-4) Notice simplifiée de l'utilisation de la lyre

Addr Réglage de l'adresse de démarrage

001
...
512

ShOd Mode show

Sh 1 Show 1
Sh 2 Show 2

SLOd Mode Slave

no Normal
2LSh 2 light show

tPAn Mouvement rotation (pan)

no Normal
YES Rotation inversée

tiLt Mouvement inclinaison (tilt)

no Normal
YES Mouvement inversé

diSP Affichage

dSP Normal
dSP1 Inversé de 180° (pour montage au plafond)

Avec la touche ENTER, commutez le réglage.
L'affichage clignote pendant 8 secondes. Ensuite, le menu s'efface.

FAdJ Position fixe de la tête orientable

La tête orientable peut être réglée de manière fixe sur 5 positions différentes :
Regardant vers le haut (bas), avant, gauche, arrière ou droit. Par plusieurs pressions sur la touche ENTER, sélectionnez la position fixe. Pour revenir au mode de fonctionnement précédent, enfoncez la touche MENU.
Le point suivant du menu clignote pendant 8 secondes. Ensuite, le menu s'efface.

tESt Auto test

Une fois la touche ENTER enfoncée, un test interne est effectué. Pour le terminer, enfoncez la touche MENU.
Le point suivant du menu clignote pendant 8 secondes. Ensuite, le menu s'efface.

FHrS Compteur heures de fonctionnement (fixture hours)

Une fois la touche ENTER enfoncée, l'affichage indique le nombre d'heures de fonctionnement. Pour revenir au mode de fonctionnement précédent, enfoncez la touche MENU.
Le point suivant du menu clignote pendant 8 secondes. Ensuite, le menu s'efface.

rESt Reset (réinitialisation)

Par une pression sur la touche ENTER, l'appareil est réinitialisé : La tête orientable va brièvement sur la position de départ et la lampe halogène s'éteint brièvement. Ensuite l'appareil revient sur le mode de fonctionnement précédent.

3 - ORGANISATION DE LA MAQUETTE PEDAGOGIQUE

La lyre est décomposée en plusieurs cartes.

Le principe de reproduction des effets lumineux (à l'aide des disques GOBOS et COULEURS) est similaire à celui du système réel.

La détection de la position d'origine se fait grâce à un aimant solidaire de chaque disque et d'un capteur à effet hall monté sur chaque carte moteur.

Le mouvement de la tête de la lyre (PAN et TILT) étant difficile à mettre en œuvre, seuls les moteurs sont commandés et entraînent 2 bras qui matérialisent l'angle de déplacement.

La détection de la position d'origine se fait grâce à la forme du bras et du capteur fourche optique monté sur chaque carte moteur.

La gestion des canaux est différente de celle sur la lyre réelle, elle est organisée comme suit :

- Canal 1 : Adresse du récepteur (seulement)
- Canal 2 : Choix des GOBOS (12 gobos fixes ou changeants)
- Canal 3 : Choix des COULEURS (12 couleurs fixes ou changeantes)
- Canal 4 : Choix position PAN
- Canal 5 : Choix position TILT
- Canal 6 : Choix de l'intensité lumineuse.

La maquette pédagogique est décomposée en groupe de 4 élèves.

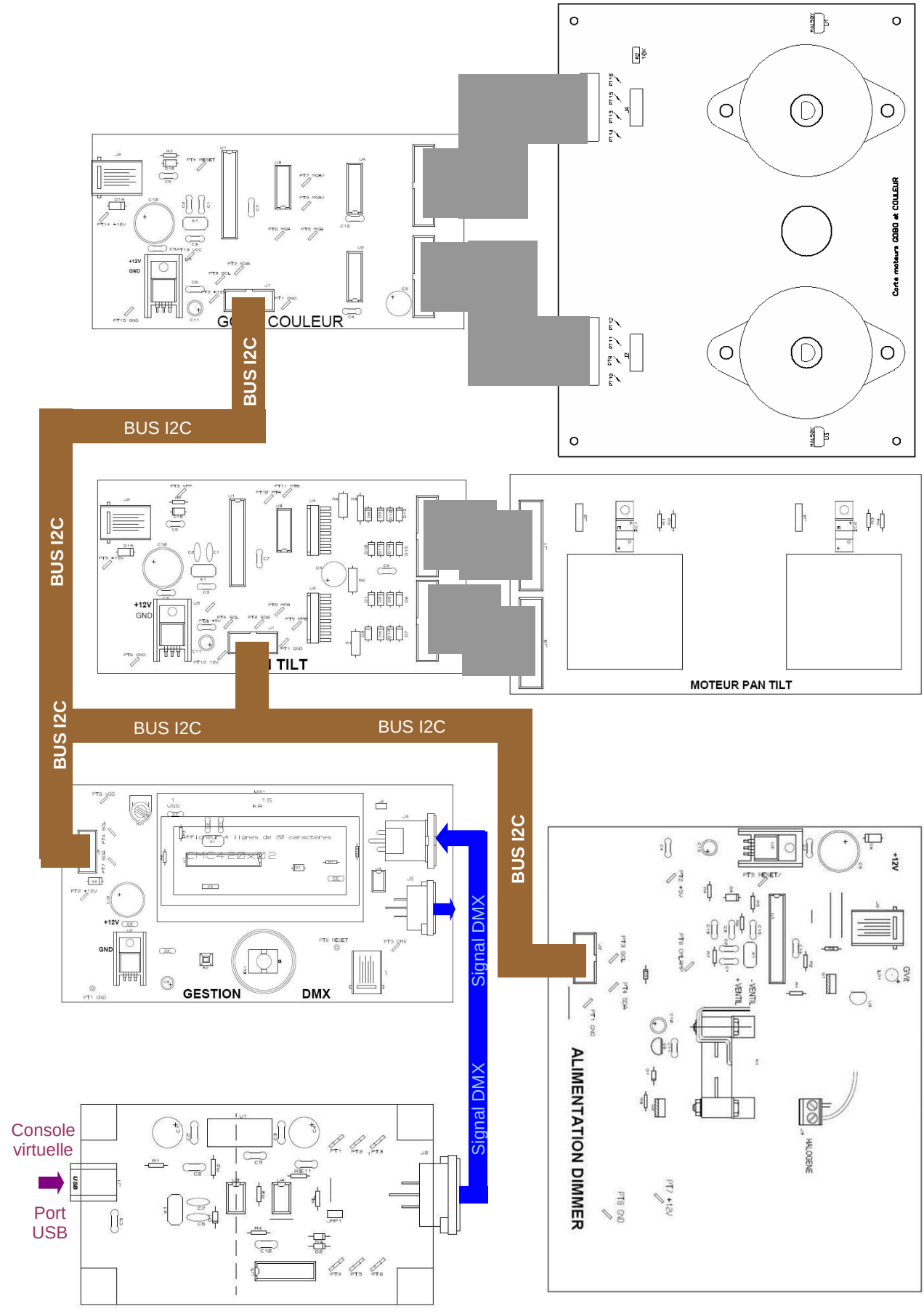
Répartition des cartes à réaliser par les élèves :

- 1 Carte Gestion du protocole DMX (réalisation 1)
- 1 Carte Commande GOBOS et COULEURS + 1 carte moteurs Gobos et couleurs (réalisation 2)
- 1 Carte Commande PAN et TILT + 1 carte moteurs PAN et TILT (réalisation 3)
- 1 Carte Alimentation DIMMER (réalisation 4)

Carte réalisée par le professeur :

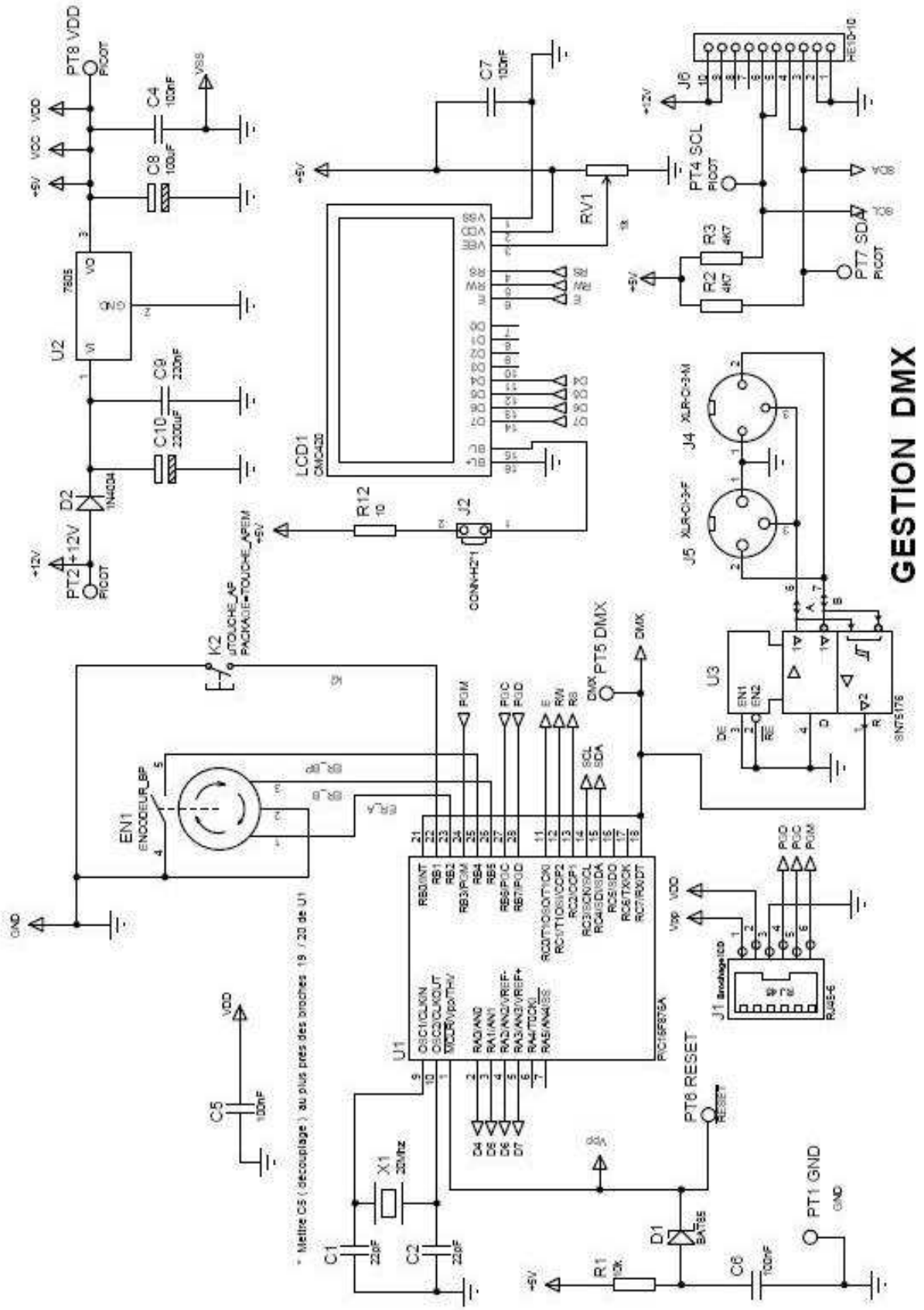
- 1 Carte interface USB/DMX qui permet de commander la maquette à partir des signaux DMX issus de la console virtuelle grâce aux interfaces (cpp_test ou FreeStyler).

4 - INTERCONNEXION DES CARTES DE LA MAQUETTE PEDAGOGIQUE



5 - REALISATION ELEVE 1 : GESTION MODES DE FONCTIONNEMENT

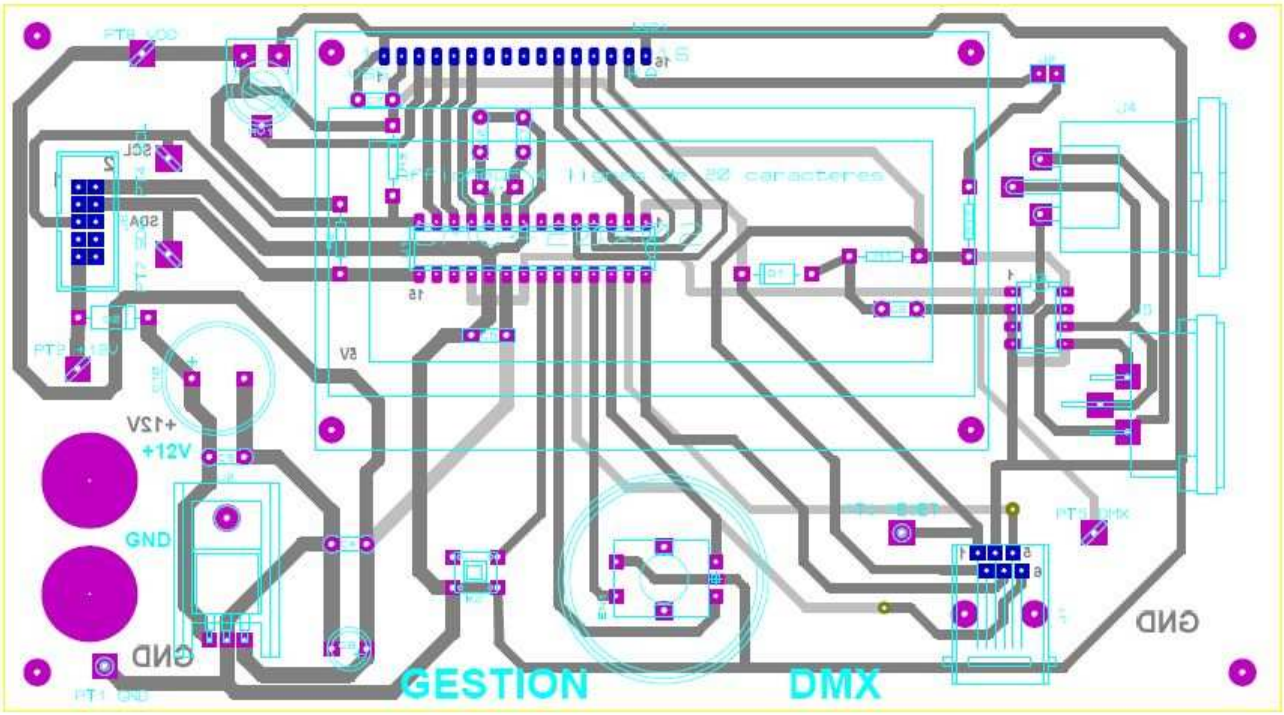
5-1) Schéma structurel



5-2) Nomenclature des composants de la carte Gestion DMX

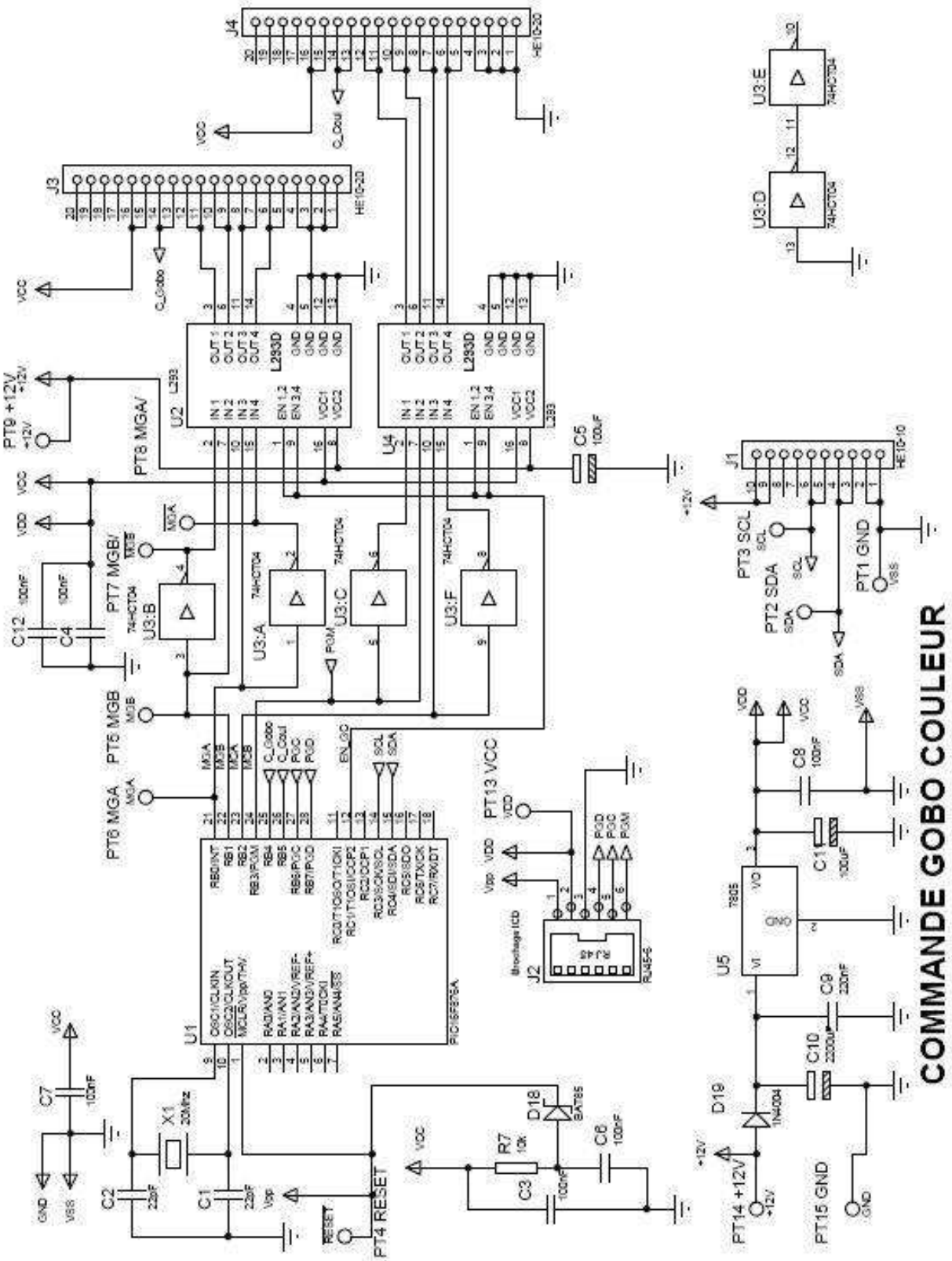
QTY	PART-REFS	VALUE	
Resistors			
1	R1	10k	
2	R2, R3	4K7	
1	R12	10	
Capacitors			
2	C1, C2	22pF	(2pas)
4	C4-C7	100nF	(2pas)
1	C8	100uF	(2pas)
1	C9	220nF	(2pas)
1	C10	2200uF	(3pas)
Integrated Circuits			
1	U1	PIC16F876A	(+tulip 28b)
1	U2	7805	(+dissipateur ML26)
1	U3	SN75176	(+tulip 8b)
Diodes			
1	D1	BAT85	
1	D2	1N4004	
Miscellaneous			
1	EN1	11CTV1Y22LFACF	(ENCODEUR_BP rotatif)
1	J1	RJ45-6	(Embase CI F)
1	J2	CONN-H2*1	(+ jumper)
1	J4	XLR-CI-3-M	
1	J5	XLR-CI-3-F	
1	J6	HE10-10	(Afficheur 4x20)
1	K2	PHAP3303	(MICRO-TOUCHE_AP)
1	LCD1	CMC420	
7	PT1... PT8	Cosses poignard	
1	RV1	1k	(2pasx4pas)
1	X1	20Mhz	(2pas)
2	dbl puits	Douilles châssis sécurité	

5-3) Schéma d'Implantation des composants de la carte Gestion DMX

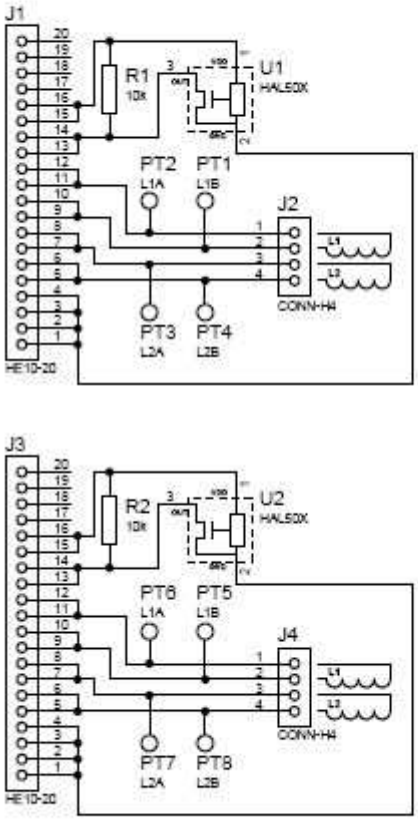


6 - Réalisation élève 2 : Commande + Moteurs GOBOS et COULEURS

6-1) Schéma structurel de la carte Commande GOBOS et COULEURS



6-2) Schéma structurel de la carte Moteurs GOBOS et COULEURS



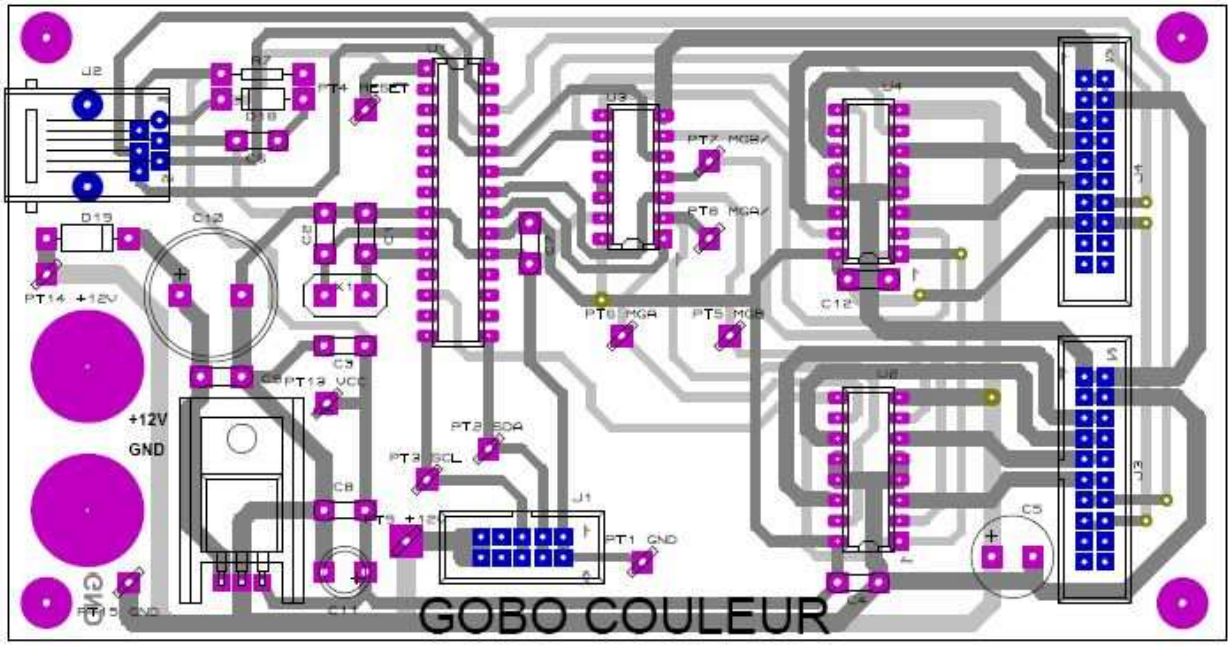
6-3) Nomenclature des composants de la carte Commande GOBOS et COULEURS

QTY	PART-REFS	VALUE	
Resistors			
1	R1	10k	
Capacitors			
2	C1, C2	22pF	(2pas)
6	C3, C4, C6-C8, C12	100nF	(2pas)
2	C5, C11	100uF	(2pas)
1	C9	220nF	(2pas)
1	C10	2200uF	(3pas)
Integrated Circuits			
1	U1	PIC16F876A	(+sup tulip 28)
2	U2, U4	L293	(+sup tulip 16)
1	U3	74HCT04	(+sup tulip 14)
1	U5	7805	(+dissipateur ML26)
Diodes			
1	D3	BAT85	
1	D2	1N4004	
Miscellaneous			
1	J1	HE10-10	
1	J2	RJ45-6	
2	J3, J4	HE10-20	
12	PT1	Cosses poignard	
1	X1	20Mhz	Quartz (2pas)
2	DBL puits	douille sécurité	

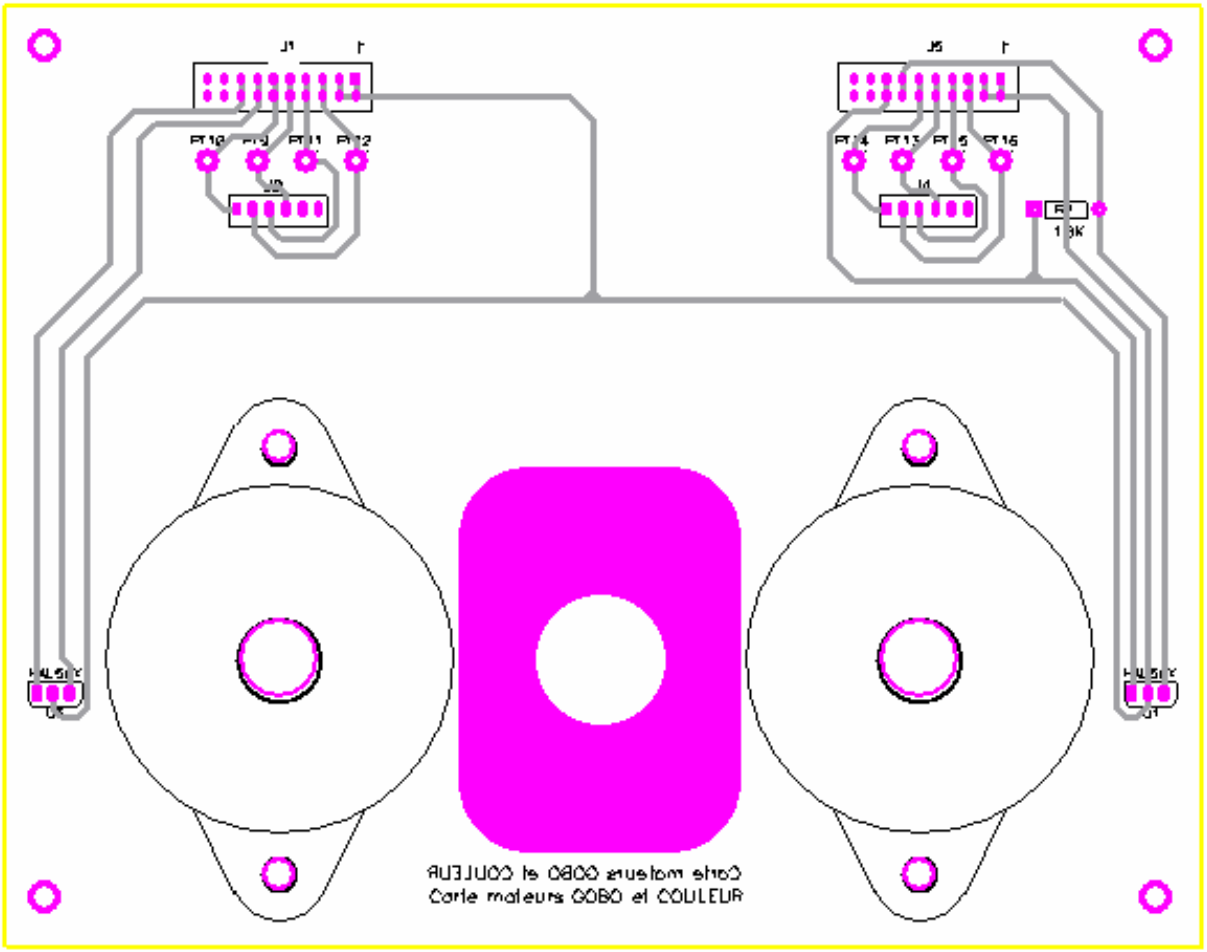
6-4) Nomenclature des composants de la carte Moteurs GOBOS et COULEURS

QTY	PART-REFS	VALUE
Resistors		
2	R1, R2	10k
Integrated Circuits		
2	U1, U2	HAL50X
Miscellaneous		
2	J1, J3	HE10-20
2	J2, J4	CONN-H4
8	PT	Cosses poignard
2	Moteur	Moteur pas à pas

6-5) Schéma d'Implantation des composants de la carte Commande GOBOS et COULEURS

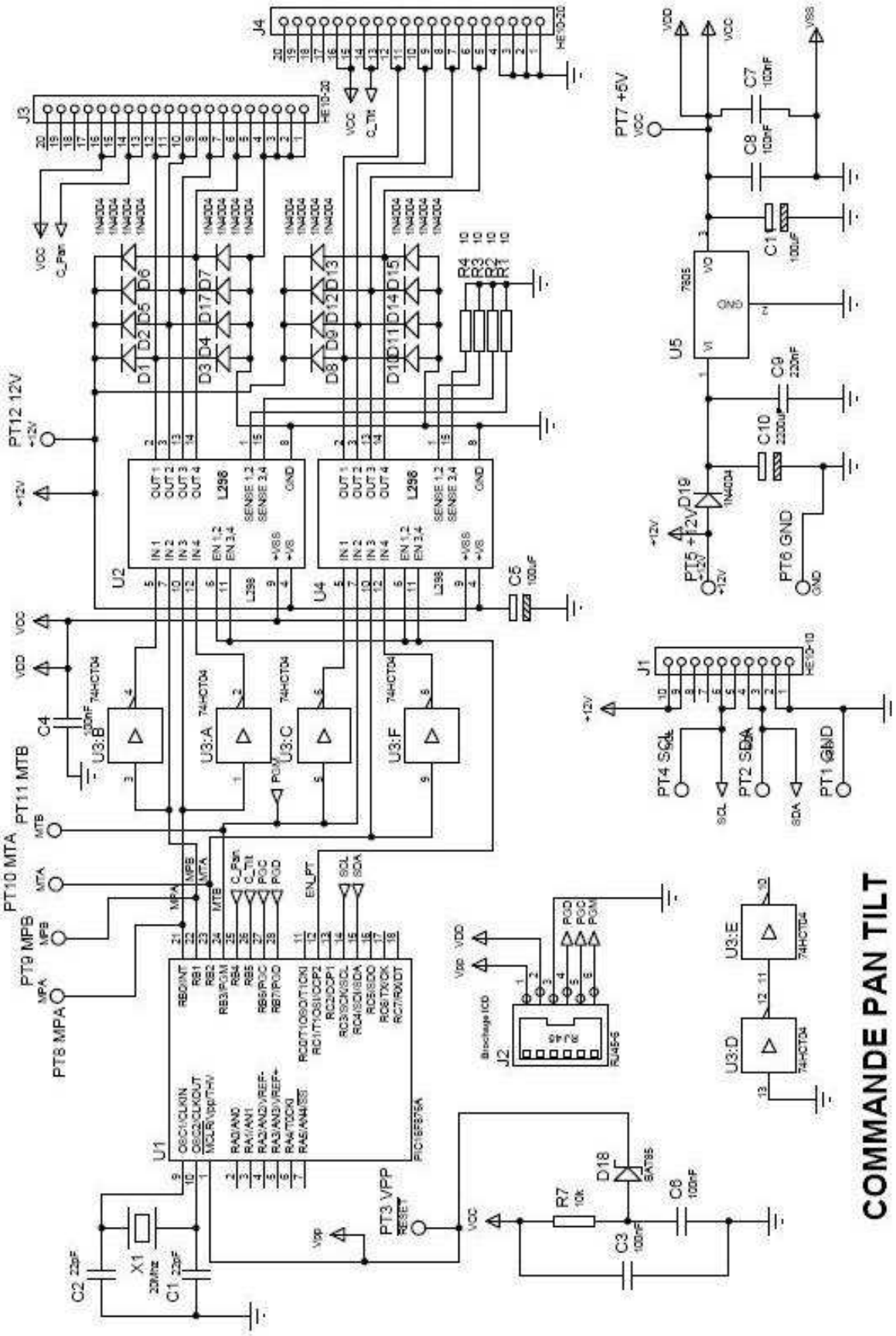


6-6) Schéma d'Implantation des composants de la carte Moteurs GOBOS et COULEURS

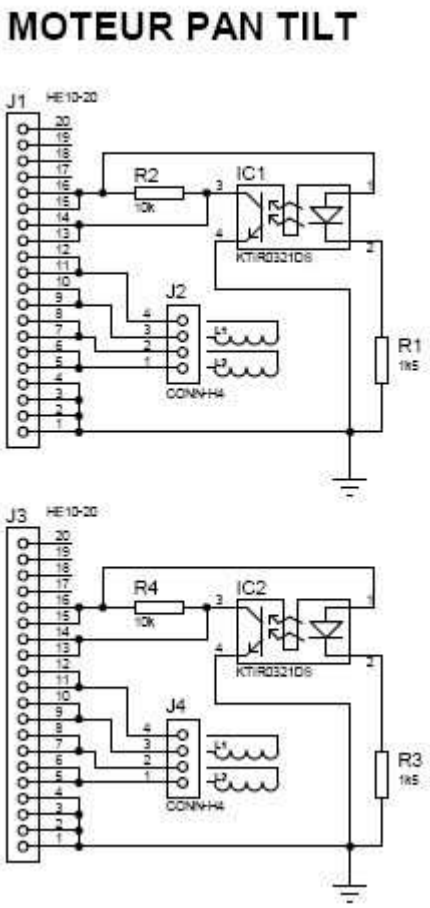


7 - Réalisation élève 3 : Commande + Moteurs PAN et TILT

7-1) Schéma structurel de la carte Commande PAN et TILT



7-2) Schéma structurel de la carte Moteurs PAN et TILT



7-3) Nomenclature des composants de la carte Commande PAN ET TILT

QTY	PART-REFS	VALUE	
---	---------	-----	
Resistors			

4	R1-R4	10	
1	R7	10k	
Capacitors			

2	C1, C2	22pF	(2pas)
5	C3, C4, C6-C8	100nF	(2pas)
2	C5, C11	100uF	(2pas)
1	C9	220nF	(2pas)
1	C10	2200uF	(3pas)
Integrated Circuits			

1	U1	PIC16F876A	(+sup tulip 28)
2	U2, U4	L298	(+sup tulip 14)
1	U3	74HCT04	
1	U5	7805	(+dissipateur ML26)
Diodes			

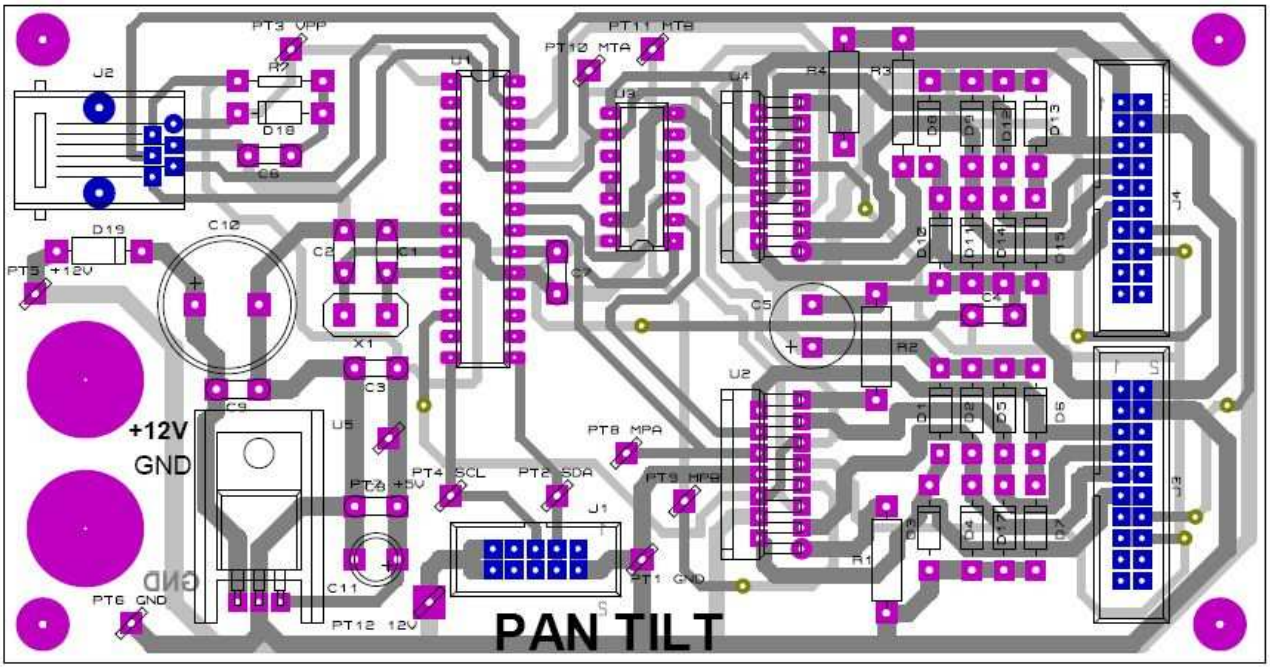
17	D1-D15, D17, D19	1N4004	
1	D18	BAT85	

QTY	PART-REFS	VALUE
Miscellaneous		
1	J1	HE10-10
1	J2	RJ45-6
2	J3, J4	HE10-20
11	PT	Cosses poignard
1	X1	20 MHz
2	DBL PUITS	douille sécurité

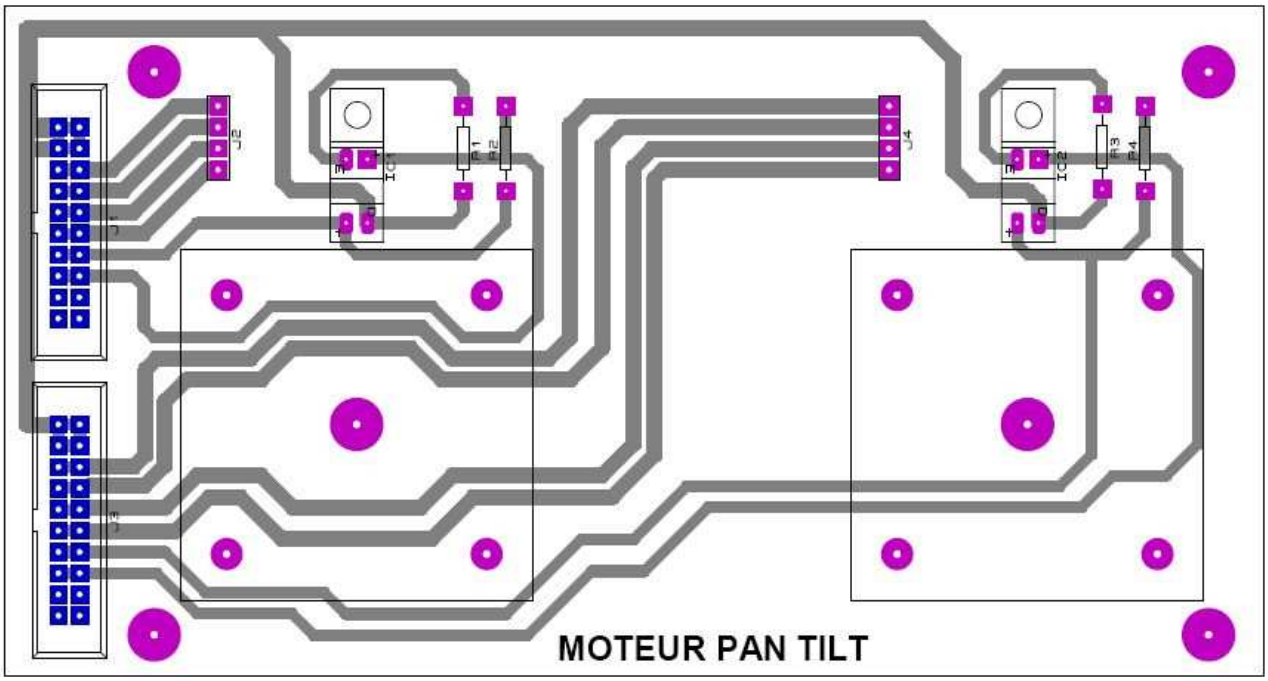
7-4) Nomenclature des composants de la carte Moteurs PAN ET TILT

QTY	PART-REFS	VALUE
Resistors		
2	R1, R3	1k5
2	R2, R4	10k
Miscellaneous		
2	IC1, IC2	KTIR0321DS
2	J1, J3	HE10-20
2	J2, J4	CONN-H4
2	Fourche opt	KTRIR0321
2	Moteur	Moteur pas à pas

7-5) Schéma d'Implantation des composants de la carte Commande PAN ET TILT

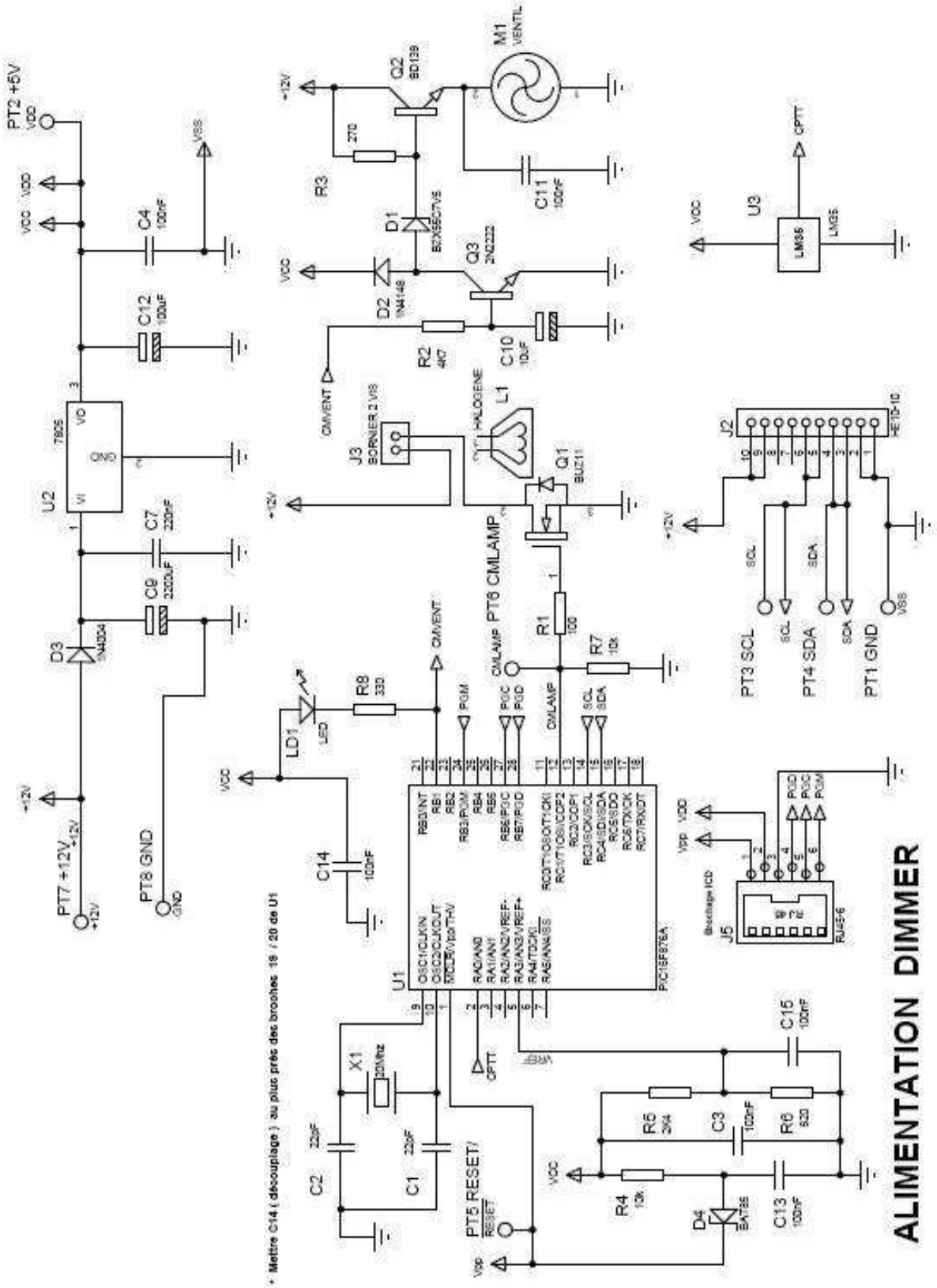


7-6) Schéma d'Implantation des composants de la carte Moteurs PAN ET TILT



8 - Réalisation élève 4 : Alimentation Dimmer

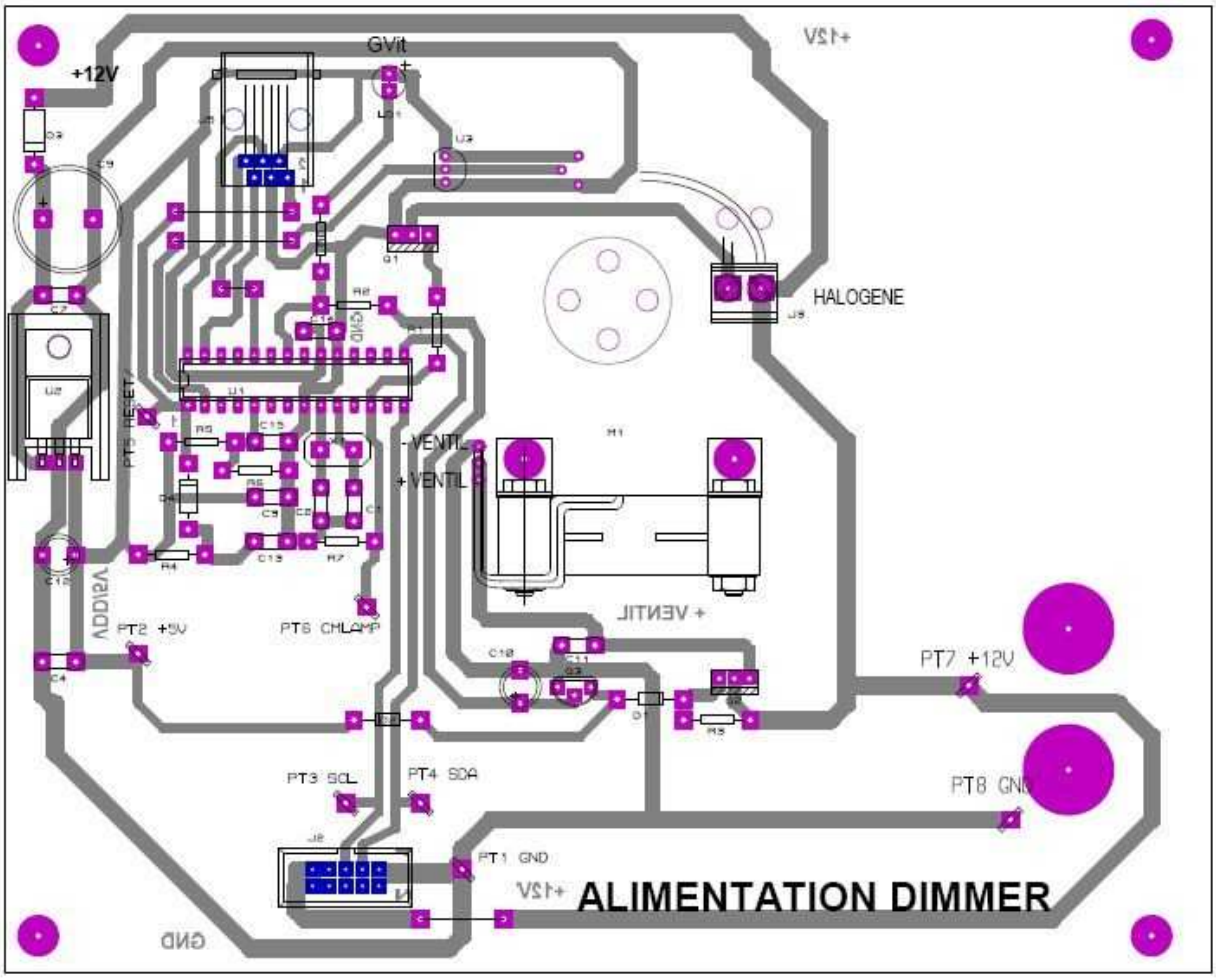
8-1) Schéma structurel de la carte alimentation Dimmer



8-2) Nomenclature des composants de la carte alimentation Dimmer

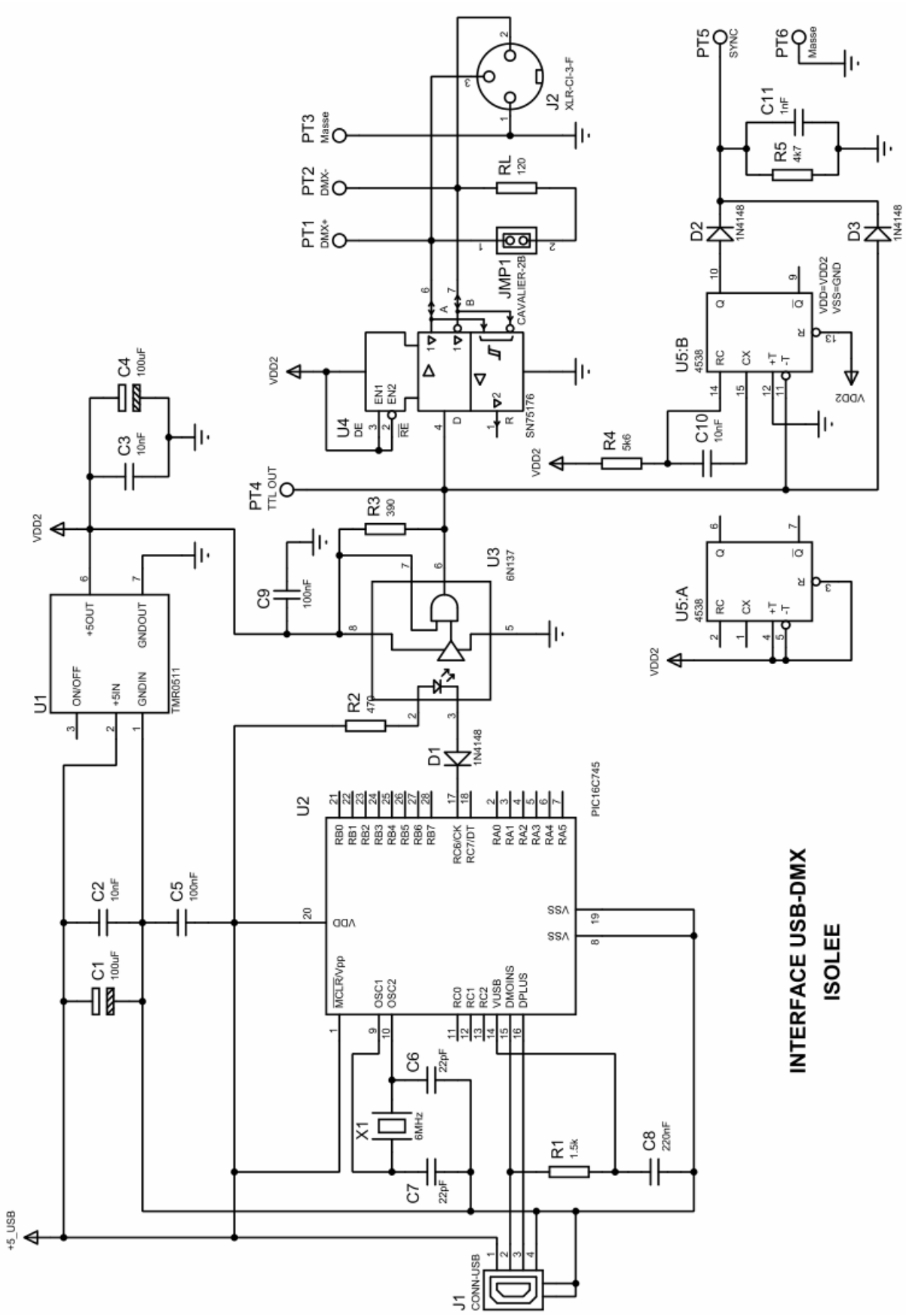
QTY	PART-REFS	VALUE	
Resistors			
1	R1	100	
1	R2	4K7	
1	R3	270	
2	R4, R7	10k	
1	R5	2K4	
1	R6	820	
1	R8	330	
Capacitors			
2	C1, C2	22pF	(2pas)
6	C3, C4, C11, C13-C15	100nF	(2pas)
1	C7	220nF	(2pas)
1	C9	2200uF	(2pas)
1	C10	10uF	(3pas)
1	C12	100uF	(2pas)
Integrated Circuits			
1	U1	PIC16F876A	(+sup. tuli28)
1	U2	7805	(+dissipateur ML26)
1	U3	LM35	
Transistors			
1	Q1	BUZ11	
1	Q2	BD139	
1	Q3	2N2222	
Diodes			
1	D1	BZX55C7V5	
1	D2	1N4148	
1	D3	1N4004	
1	D4	BAT85	
Miscellaneous			
1	J2	HE10-10	
1	J3	BORNIER 2 VIS	
1	J5	RJ45-6	
1	L1	HALOGENE MR11 12V/20W	
1	Support	Embase halogène	
1	LD1	LED	
8	PT	Cosses poignard	
1	X1	20Mhz	
1	M1	VENTIL	
2	dble puit	Douille sécurit	

8-3) Schéma d'Implantation des composants de la carte alimentation Dimmer



9 - Interface USB/DMX512

9-1) Schéma structurel de l'interface USB/DMX



9-2) Nomenclature des composants

QTY	PART-REFS	VALUE
---	---------	-----
Resistors		

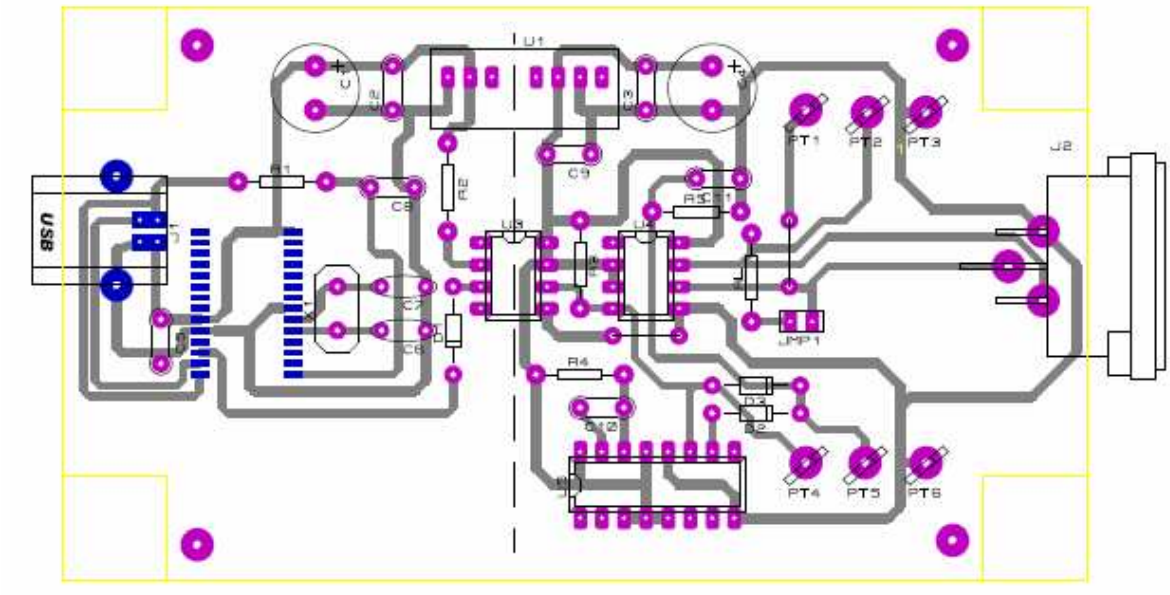
1	R1	1.5k
1	R2	470
1	R3	390
1	R4	5k6
1	R5	4k7
Capacitors		
----------	(Tous 2 pas)	
2	C1, C4	100uF
3	C2, C3, C10	10nF
2	C5, C9	100nF
2	C6, C7	22pF
1	C8	220nF
1	C11	1nF
Integrated Circuits		

1	U1	TMR0511
1	U2 (CMS)	PIC16C745
1	U3 (+sup tulip 8)	6N137
1	U4 (+sup tulip 8)	SN75176
1	U5 (+sup tulip 16)	4538
Diodes		

3	D1-D3	1N4148
Miscellaneous		

1	J1	CONN-USB
1	J2	XLR-CI-3-F
1	JMP1	CAVALIER-2B
6	PT	Cosses poignard
1	RL	120
1	X1	CRYSTAL

9-3) Schéma d'Implantation des composants



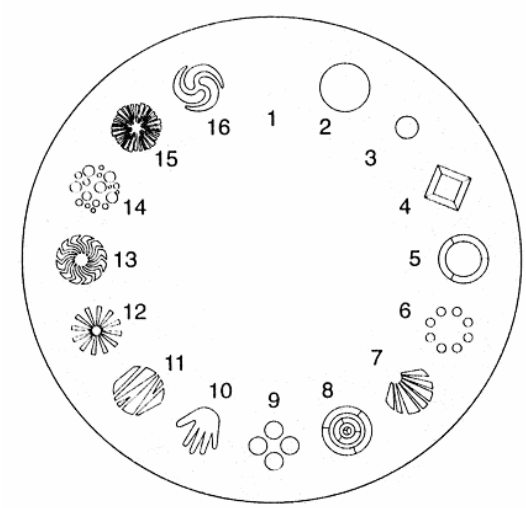
10 - Partie opérative

10-1) Codes DMX utilisés par la Lyre Twist-25 (système réel)

Canal 1 : Effets

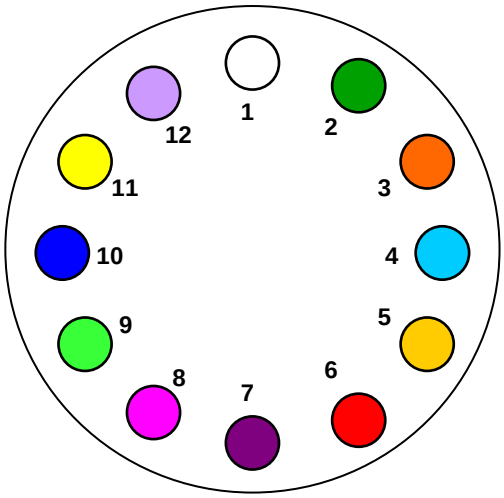
Valeur DMX	Fonction
0 à 15	Pas d'effet
16 à 91	Changement continu de 2 motifs (lent à rapide)
92 à 135	Changement continu de 2 couleurs (lent à rapide)
136 à 195	Changement continu de 2 motifs et de 2 couleurs (lent à rapide)
196 à 255	Effet de tremblement (lent à rapide) (une consigne comprise entre 8 et 127 doit être envoyée sur le canal 2)

Canal 2 : Motifs (*gobos*)



Canal DMX	N°de Gobo
0 à 7	Gobo n°1 (obturation blackout)
8 à 15	Gobo n°2
16 à 23	Gobo n°3
24 à 31	Gobo n°4
32 à 39	Gobo n°5
40 à 47	Gobo n°6
48 à 55	Gobo n°7
56 à 63	Gobo n°8
64 à 71	Gobo n°9
72 à 79	Gobo n°10
80 à 87	Gobo n°11
88 à 95	Gobo n°12
96 à 103	Gobo n°13
104 à 111	Gobo n°14
112 à 119	Gobo n°15
120 à 127	Gobo n°16
128 à 255	Changement continu des gobos (lent à rapide).

Canal 3 : couleurs



Valeur DMX	Couleur
0 à 10	Blanc
11 à 21	Vert
22 à 30	Orange
33 à 43	Bleu clair
44 à 54	Ambre
55 à 65	Rouge
66 à 76	Violet
77 à 87	Rose
88 à 98	Vert clair
99 à 109	Bleu
110 à120	Jaune
121 à 126	Magenta
128 à 255	Changement continu des couleurs (lent à rapide)

Canal 4 : Rotation de la tête (*Pan*) : 0 à 255.

Canal 5 : Inclinaison de la tête (*Tilt*) : 0 à 255.

10-2) Tableau de correspondance des codes DMX sur le système didactisé

Correspondance des canaux entre la lyre Twist-25 et la lyre didactisée (système pédagogique) .

Canal DMX	Potentiomètre virtuel	Lyre Stage Line Twist-25	Maquette Lyre didactisée
1	Pot. 1	Effets (tremblement)	/
2	Pot. 2	Motifs GOBOS	Motifs GOBOS
3	Pot. 3	Couleurs	Couleurs
4	Pot. 4	Mvt Panoramique PAN	Mvt Panoramique PAN
5	Pot. 5	Mvt Roulis TILT	Mvt Roulis TILT
6	Pot. 6	/	Intensité lumineuse
7	Pot. 7	/	non utilisé
8	Pot. 8	/	non utilisé
31	Pot. « 31 »	/	non utilisé
32	Pot. « 32 »	/	non utilisé
511	Pot. « 511 »	/	non utilisé
512	Pot. « 512 »	/	non utilisé

Remarque :

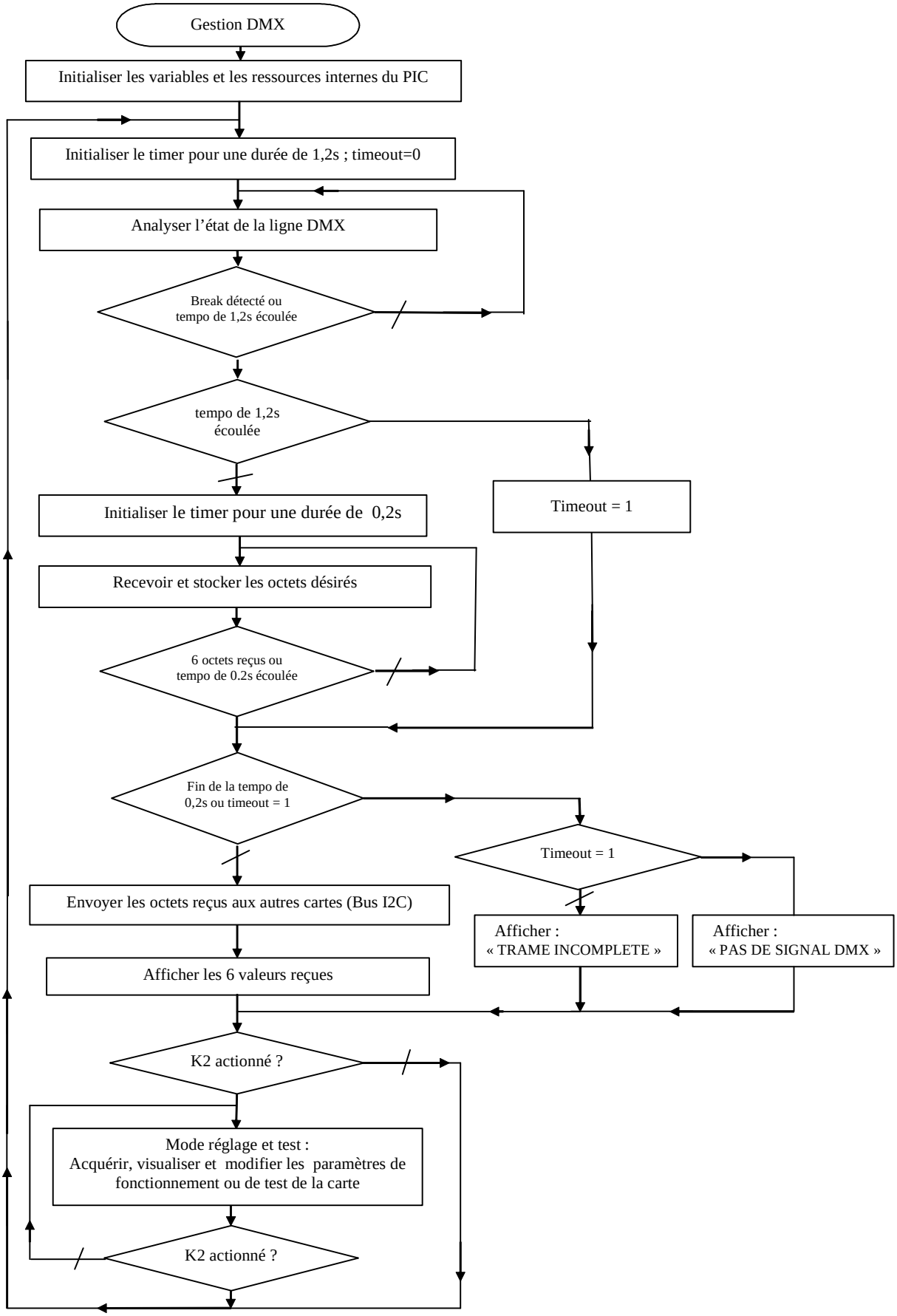
- la fonction variation de lumière (gradateur) absente sur la lyre Stage Line Twist-25, est intégrée à la lyre didactisée ; elle utilise le canal 6.

Connexion au système réel :

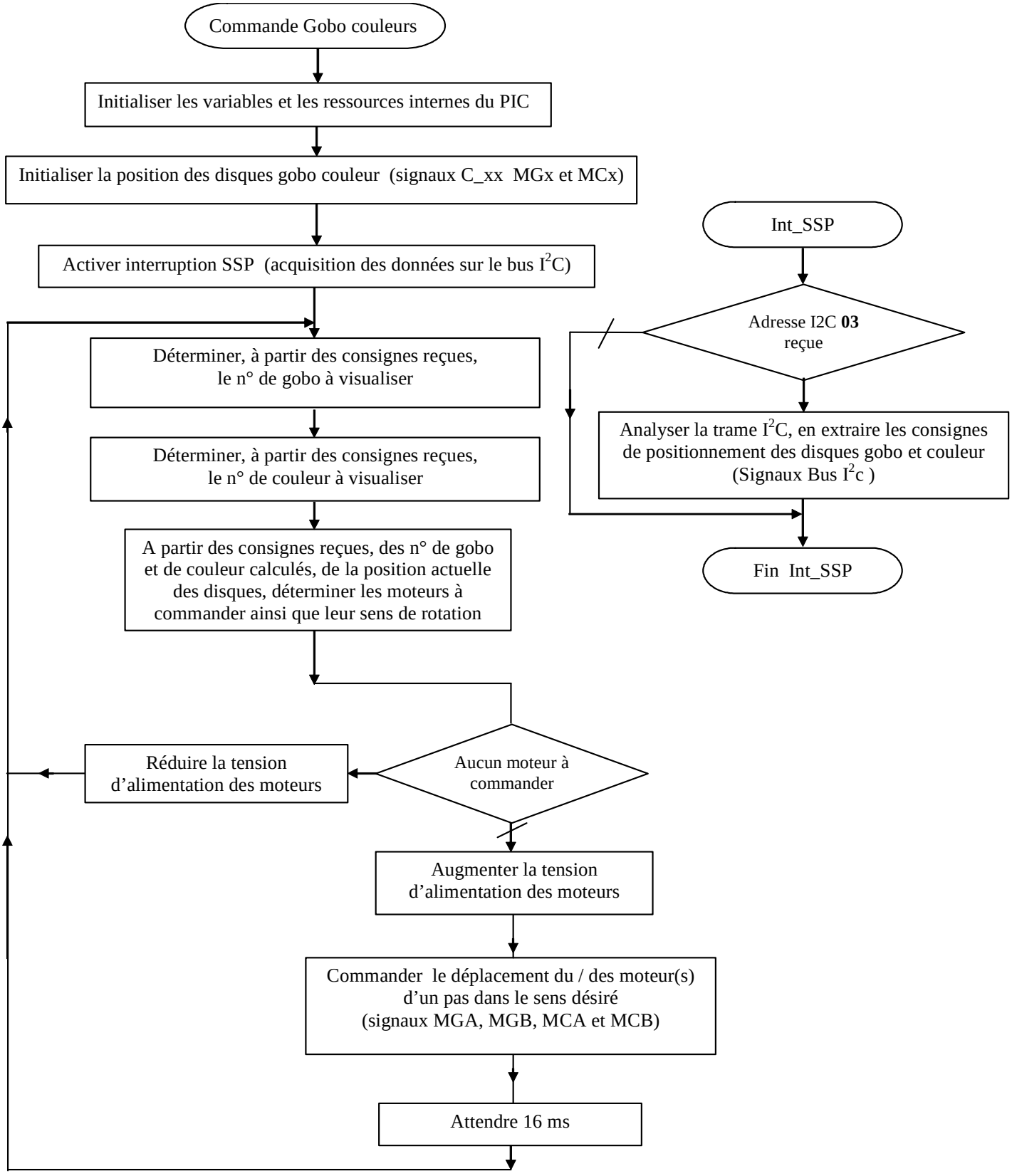
En installant deux commutateurs manuels à 4 pôles , il est possible de brancher la commande des moteurs Pan_Tilt de la maquette sur les moteurs de la lyre elle-même.
C'est intéressant car cela permet de piloter la vraie partie opérative à partir de nos cartes.
ATTENTION :il est impératif de ne pas alimenter la lyre qui risquerait alors de rechercher la position d'origine de ses moteurs. Cette position d'origine n'est pas gérée par la maquette. Le débattement mécanique de la lyre étant limitée, il y risque de destruction du circuit intégré de sortie.

11 - Algorigrammes des cartes

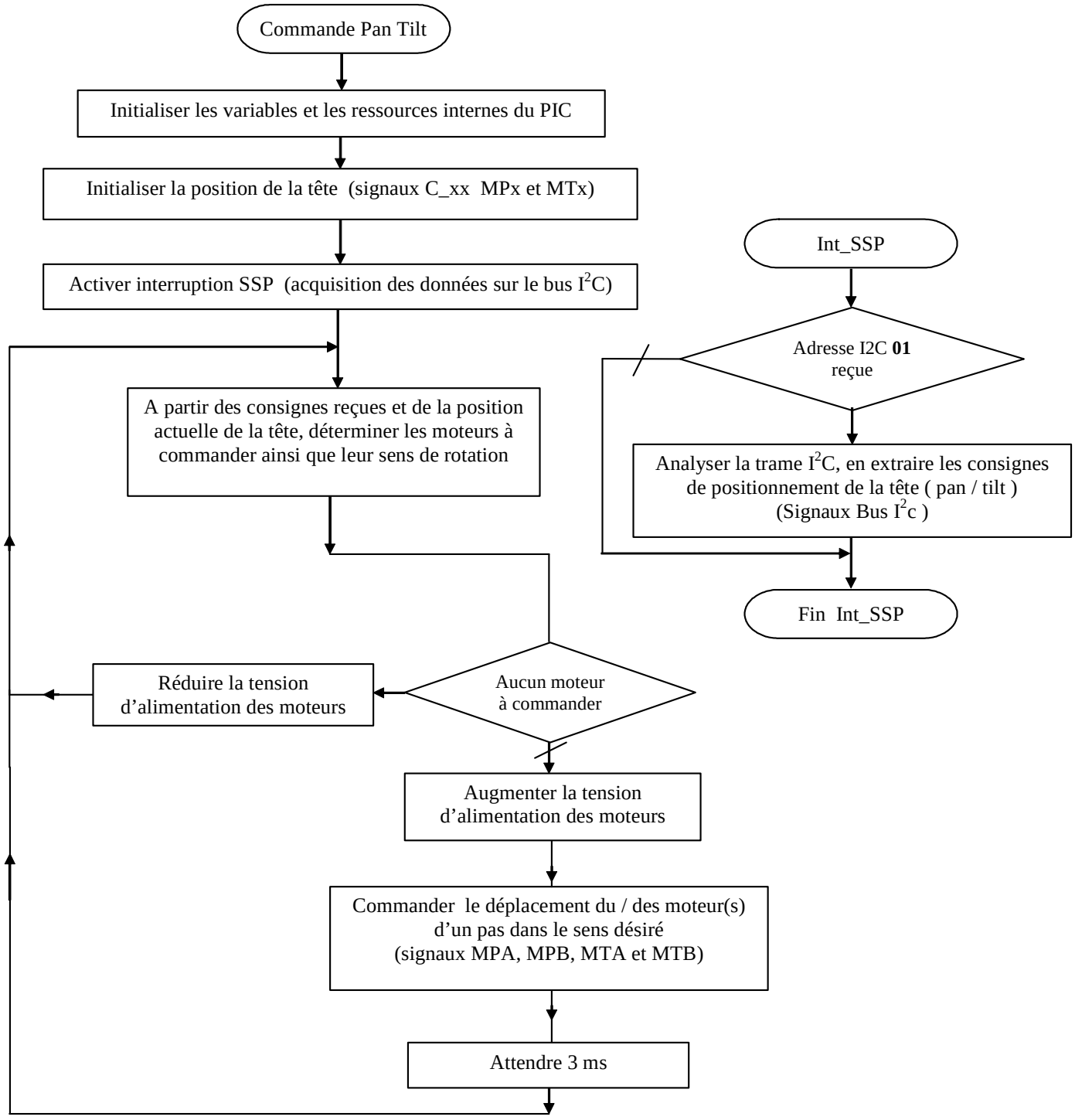
11-1) Algorigramme : Carte Gestion DMX



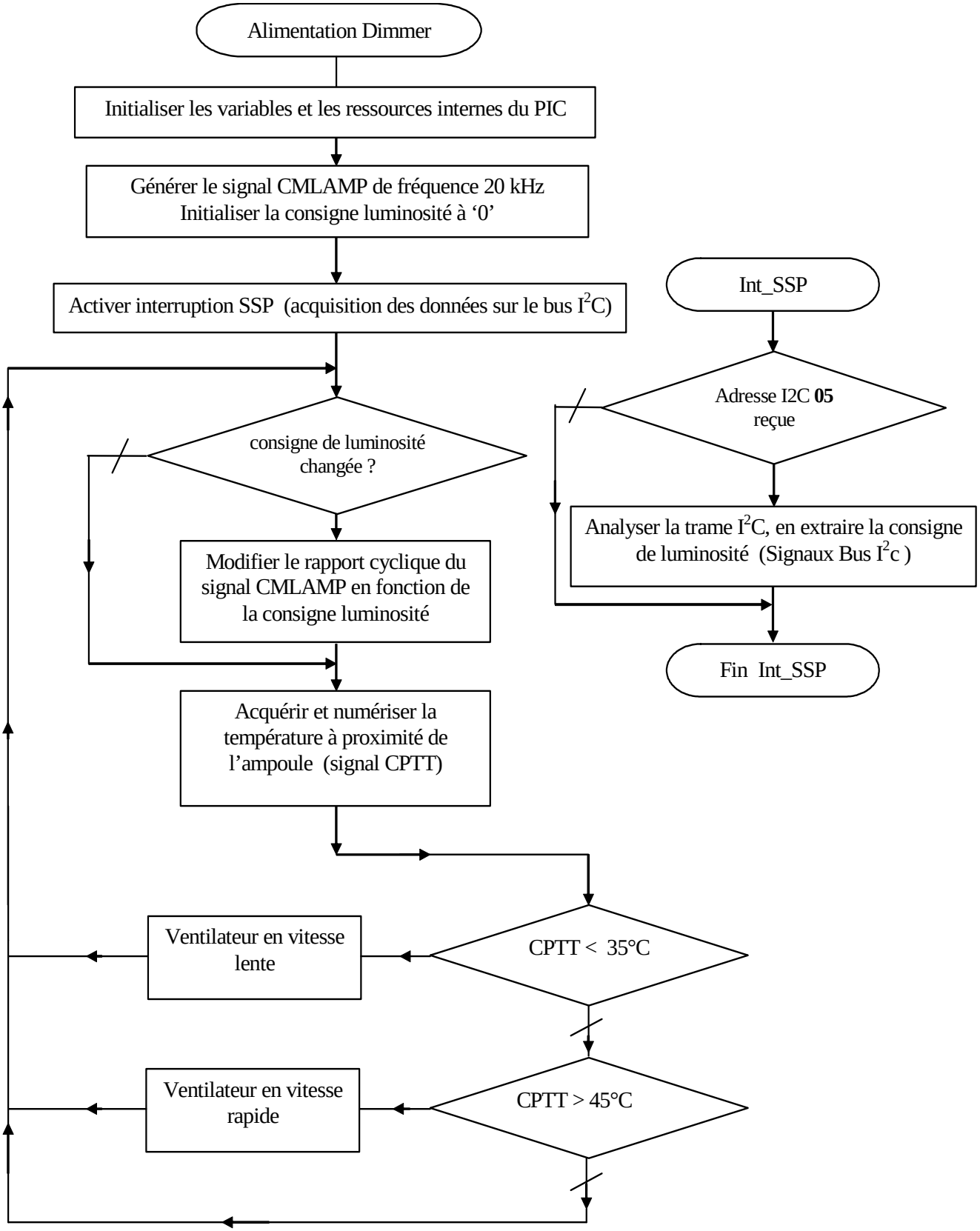
11-2) Algorigramme : Carte Commande Gobos couleurs



11-3) Algorigramme : Carte Commande Pan Tilt



11-4) Algorigramme : Carte Alimentation Dimmer


Alimentation Dimmer
Initialiser les variables et les ressources internes du PIC
Générer le signal CMLAMP de fréquence 20 kHz
Initialiser la consigne luminosité à '0'
Activer interruption SSP (acquisition des données sur le bus I2C)
consigne de luminosité changée ?
Modifier le rapport cyclique du signal CMLAMP en fonction de la consigne luminosité
Acquérir et numériser la température à proximité de l'ampoule (signal CPTT)
CPTT < 35°C
Ventilateur en vitesse lente
CPTT > 45°C
Ventilateur en vitesse rapide
Int_SSP
Adresse I2C 05 reçue
Analyser la trame I2C, en extraire la consigne de luminosité (Signaux Bus I2c)
Fin Int_SSP

12 - Programmes des cartes

12-1) Code source 'c' : Carte gestion DMX

```
//  prog Gestion DMX    Version5

#include <16F876A.h>
#device adc=8
#fuses NOWDT,HS, NOPUT, NOPROTECT, NODEBUG, BROWNOUT, noLVP, NOCPD, NOWRT
#use delay(clock=20000000)
#define D4   PIN_A0
#define D5   PIN_A1
#define D6   PIN_A2
#define D7   PIN_A3
#define ER_B  PIN_B5
#define ER_A  PIN_B2
#define ER_BP PIN_B4
#define k2    PIN_B1
#define ad_pan_tilt     1
#define ad_gobo_coul    3
#define ad_alim_dimmer  5
#use rs232(baud=250000,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)   // 8  bits
#use i2c(Master,slow,sda=PIN_C4,scl=PIN_C3,force_hw)
#use fast_io(A)
#use fast_io(B)
#use fast_io(C)
#NOLIST
#include "lcd420_lyre.C"
#LIST
#byte STATUS=0x03
#byte RCSTA=0x18
#byte RCREG=0x1A
#byte PIR1=0x0C
#byte PIE1=0x8C
#byte SSPCON2=0x91
#byte SSPSTAT=0x94
#byte SSPBUF=0x13
#bit  TMR1_IF=PIR1.0  // à  1 indique le débordement du timer1
#bit  DMX=0x06.0      // bit 0 du port B
#bit  RCIF=PIR1.5
#bit  RP0=STATUS.5
#bit  RP1=STATUS.6
#bit  CREN=RCSTA.4
#bit  FERR=RCSTA.2
#bit  SPEN=RCSTA.7
#bit  SEN=SSPCON2.0
#bit  ACKSTAT=SSPCON2.6
#bit  PEN=SSPCON2.2
#bit  R_W=SSPSTAT.2
int8 mess,adr;
int8 cptdt_dmx ; //compteur durée entre break
int1 flg_menu_chg,flg_val_chg,num_val;
int1 ea,eb,ebp,lb,lbp;
int8 bufout[4]; // 4 octets utiles
int8 can1,can2,can3,can4,can5,can6;
int1 flg_dmx,reglage; // à 1 si présence de la trame DMX
int1 fin_recp;  // à 1 indique le fin de l arécption des 6 octets DMX
int8 numreg,valreg,valreg_1,valreg_2,valreg_3,valreg_4;
char const menu [6][8]=  {              // en ROM
                "Adr DMX",
                "Pan    ",
                "Tilt   ",
                "Gobo   ",
                "Couleur",
                "Dimmer ",
                 };

envoi_i2c(int8 adi2c, int8 nboct); // prototypes
int8 encodeur(void);

void main() {
//*****  initialiser les variables et les ressources internes du pic.
   int8 i;
   adr = read_EEPROM (1);  // lecture de l'adresse DMX de départ dans l'eerom
   port_b_pullups(TRUE);
   setup_adc_ports(NO_ANALOGS);
   setup_adc(ADC_OFF);
```

```
setup_timer_0(RTCC_INTERNAL|RTCC_DIV_2); //res 0.4µs of 102µs / div by 256-> res 51.2µs overflow 13.1ms
setup_timer_1(T1_INTERNAL|T1_DIV_BY_8); //res 1.6µs of 104,8ms / div by 2 -> résolution .4µs overflow 26,2ms
setup_timer_2(T2_DISABLED,0,1);
setup_comparator(NC_NC_NC_NC);
setup_vref(FALSE);
SET_TRIS_A(0x30); //RA0 à RA3 en sortie
SET_TRIS_B(0xFF); //tout en entrée
SET_TRIS_C(0xf8); //RC0 à RC2 en sortie
delay_ms(100);  // ?????
lcd_init();
mess =1;
lcd_putc("\fGESTION DMX 2009 \nlecture can DMX V4\n\n41204120313231303038");
delay_ms(1400);
reglage = 0;
CREN=0;
while (true)
{
   int1 flg_tmin=0;
   int1 flg_tmax=1;
   int8 savstat;
   int8 x;
   while ( !(flg_tmin && flg_tmax))  //attendre le break
   {
      cptdt_dmx = 0 ;  // compteur durée entre 2 pulses de break
      set_timer1(0000);  // le réglage du débordement du timer1 est de 104ms
      TMR1_IF=0;       // mise à 0 du flag débordement du timer1
      while (DMX)   // on attend le passage à zéro de la ligne DMX   (exit si plus de 1,2s)
       {
         if (TMR1_IF)
            {
            TMR1_IF=0;
            set_timer1(0000);
            ++cptdt_dmx;
            }
        if (cptdt_dmx>12)
            goto abs_trame;             //break; ???
         }
     set_timer1(0000);  // le réglage du débordement du timer1 est de 104ms
     TMR1_IF=0;       // mise à 0 du flag débordement du timer1
     cptdt_dmx = 0 ;
     flg_tmin=0;
     flg_tmax=1;
     while(!DMX)     // on attend le passage à 1 de la ligne (fin de break)
        {
        if (get_timer1() > 52)   // état bas de plus de 88 µs
           flg_tmin=1;
        if (TMR1_IF)
           {
           TMR1_IF=0;
           set_timer1(0000);
           ++cptdt_dmx;
           }
        if (cptdt_dmx>2)
           flg_tmax=0;
        if (cptdt_dmx>12)
           goto abs_trame;
        }
   }
   flg_dmx=1;
   fin_recp=0;
//*****  zone en assembleur : réception des 6 octets à partir de l'octet 'adr'
   #asm
   movf STATUS,w
   movwf savstat
   bcf  RP0
   bcf  RP1 //bank0
   bsf  CREN
   movf adr,w
   movwf x
            CLRF   0x0F // mise à Zéro du timer1
   CLRF   0x0E //     TMR1_IF=0;       // mise à 0 du flag débordement du timer1
   BCF    TMR1_IF //     TMR1_IF=0;       // mise à 0 du flag débordement du timer1
 bclx:
   btfsc TMR1_IF   // 100ms max pour arriver à l' octet "adr"
   goto  time_out  //
   btfss RCIF  //PIR1,RCIF
   GOTO bclx
   movf RCREG,w  // RCREG dans W
   decf x,f
```

```
    btfss STATUS,2 // bit Z
    GOTO bclx
    CLRF   0x0F // mise à Zéro du timer1
    CLRF   0x0E //     TMR1_IF=0;       // mise à 0 du flag débordement du timer1
    BCF    TMR1_IF //     TMR1_IF=0;       //100 ms maxi pour lire les 6 octets
bcl1:
    btfsc TMR1_IF
    goto  time_out //
    btfss RCIF  //PIR1,RCIF
    GOTO bcl1
    movf RCREG,w  // RCREG dans W
    movwf can1

bcl2:
            btfsc TMR1_IF
    goto  time_out //
            btfsc TMR1_IF
    goto  time_out //
    btfss RCIF  //PIR1,RCIF
    GOTO bcl2
    movf RCREG,w  // RCREG dans W
    movwf can2
bcl3:
            btfsc TMR1_IF
    goto  time_out //
    btfss RCIF  //PIR1,RCIF
    GOTO bcl3
    movf RCREG,w  // RCREG dans W
    movwf can3
bcl4:
            btfsc TMR1_IF
    goto  time_out //
    btfss RCIF  //PIR1,RCIF
    GOTO bcl4
    movf RCREG,w  // RCREG dans W
    movwf can4
bcl5:
            btfsc TMR1_IF
    goto  time_out //
    btfss RCIF  //PIR1,RCIF
    GOTO bcl5
    movf RCREG,w  // RCREG dans W
    movwf can5
bcl6:
            btfsc TMR1_IF
    goto  time_out //
    btfss RCIF  //PIR1,RCIF
    GOTO bcl6
    movf RCREG,w  // RCREG dans W
    movwf can6
    bsf fin_recp
time_out:
    bcf  CREN
    movf savstat,w
    movwf STATUS
    #endasm
            goto suite;
    abs_trame:  //
    flg_dmx = 0;     // alors plus de signal DMX
suite:
    if ( flg_dmx == 0 )  //&& mess!=54)
      {
      lcd_putc("\f\n PAS DE SIGNAL DMX\n ");
      mess = 54;
      }
    if ( flg_dmx == 1 && mess!=59 && fin_recp == 0)
      {
      lcd_putc("\f\n Break DMX OK\n  trame incomplete ");
      mess = 59;
      }
    if (fin_recp == 1)
    {
      fin_recp =0;
      mess=41;
            printf(lcd_putc,"\fGobo :%3u Coul :%3u\nPan  :%3u Tilt :%3u\nGrad :%3u Effet:%3u ",can2,can3,can4,can5,can6,can1);
      printf(lcd_putc,"\nAdr:%3u     reg: BP2",adr);
    // envoi des 6 octets sur bus I2C
    bufout[0] = can4;
    bufout[1] = can5;
```

```
  envoi_i2c(ad_pan_tilt,2);  //
  bufout[0] = can6;
  envoi_i2c(ad_alim_dimmer,1); //
  bufout[0] = can1;
  bufout[1] = can2;
  bufout[2] = can3;
  envoi_i2c(ad_gobo_coul,3);
  }
  if (!input(k2))
     {
     reglage=1;
     while (!input(k2))
        delay_ms (20);
     numreg=0;
     valreg=0;
     lb = input(ER_B);
     lbp= input(ER_BP);
     flg_menu_chg = 1;
     flg_val_chg = 0;
     num_val=0;   // à zéro indique réglage du num de canal / à 1 réglage valeur
     }
  while (reglage)
  {
     //delay_ms (20);
     if (mess != 12)
        {
        printf(lcd_putc,"\fChoisir et  valider\n Reg :\n ");
        printf(lcd_putc,"\nReg: ENC Sortir: BP2");
        mess =12;
        }
        if (flg_menu_chg && (!num_val))
           {
           lcd_gotoxy(8,2);
           printf(lcd_putc," %s",menu[numreg]);
           printf(lcd_putc,"\n                  ");
           flg_menu_chg = 0;
           }
        delay_ms(05);
        i = encodeur();
        if ((i==1)&&(!num_val))
           {
           flg_menu_chg = 1;
           ++ numreg;
           if (numreg == 6)
              numreg =0;
           }
        if ((i==0xFF)&&(!num_val))
           {
           flg_menu_chg = 1;
           -- numreg;
           if (numreg == 0xff)
              numreg =5;
           }
        if ((i==0x80)&&(!num_val))
           {
           flg_val_chg = 1;
           num_val=1;
           i=0;
           if (numreg==0)
              valreg = adr;
           }
         if ((i==1)&&(num_val))
           {
           flg_val_chg = 1;
           ++ valreg;
           }
        if ((i==0xFF)&&(num_val))
           {
           flg_val_chg = 1;
           -- valreg;
           }
        if ((i==0x80)&&(num_val))
           {
           flg_menu_chg = 1;
           num_val=0;
           i=0;
           if (numreg==0)
              {
              adr = valreg;
```

```
            write_eeprom(1,adr);
            }
          }
        if (flg_val_chg && (numreg == 0))
          {
          lcd_gotoxy(15,3);
          printf(lcd_putc," %3u",valreg);
          flg_val_chg = 0;
          }
          if (flg_val_chg && (numreg != 0))
          {
          lcd_gotoxy(15,3);
          printf(lcd_putc," %3u",valreg);
          flg_val_chg = 0;

          if (numreg==5)
            {
             bufout[0] = valreg;
             envoi_i2c(ad_alim_dimmer,1); //
            }
          if (numreg==1)// pan
            {
            bufout[0] = valreg;
            bufout[1] = 0;
            valreg_1 = valreg;
            envoi_i2c(ad_pan_tilt,2);
            }
          if (numreg==// pan + tilt
            {
            valreg_2 = valreg;
            bufout[0] = valreg_1;//valeur précédemment réglée pour pan
            bufout[1] = valreg_2;//valeur réglée pour tilt
            envoi_i2c(ad_pan_tilt,2);  //
            }
            ////////////////////////////////////////////////////////
          if (numreg==3)// gobo
            {
            bufout[0] = 0;
            bufout[1] = valreg;
            bufout[2] = 0;
            envoi_i2c(ad_gobo_coul,3);
            valreg_3 = valreg;//mise en mémoire de la position souhaitée pour gobo.
            }
           if (numreg==4)//coul+gobo
            {
            valreg_4 = valreg;
            bufout[0] = 0;
            bufout[1] = valreg_3;
            bufout[2] = valreg_4;
            envoi_i2c(ad_gobo_coul,3);//mise en position du réglage demandé
            }
            ////////////////////////////////////////////////////////
          }

      if (!input(K2))
        {
        reglage = 0;
        while (!input(k2))
          delay_ms (20);
       }
    }
  }
}

int8 encodeur()
  {
  int8 x=0;
  ea = input(ER_A);
  eb = input(ER_B);
  ebp= input(ER_BP);
  if ( ebp == lbp)
    {
     if (eb == lb)
       x=0;
     else
      {
      if ((eb && ea) || ((!eb)&&(!ea)))
        x=1;
      else
```

```
          x=0xff;
        lb=eb;
        }
      }
    else
      {
      lbp=ebp;
      if (!ebp)
        x=0x80;
      }
    return x;
    }

envoi_i2c(int8 adi2c, int8 nboct)
{
    int8 i,vx;
    vx= adi2c * 2;
   // big_I2C_start();
    while (R_W); // transfert en cours .. on attend la fin d'émission
    while ((SSPCON2 & 0x1f)!=0);
    SEN=1;
    //big_I2C_send(vx);
    while (R_W);
    while ((SSPCON2 & 0x1f)!=0);
    SSPBUF = vx ;
    // big_I2C_check();
    while (R_W);
    while ((SSPCON2 & 0x1f)!=0);
    if (ACKSTAT)
      {
      delay_cycles(40); // no acq non traité
      }
    for (i=0;i<nboct;i++)
      {
      // big_I2C_send(bufout[i]);
      while (R_W);
      while ((SSPCON2 & 0x1f)!=0);
      SSPBUF = bufout [i];
      //big_I2C_check();
      while (R_W);
      while ((SSPCON2 & 0x1f)!=0);
      if (ACKSTAT)
      {
      delay_cycles(40);
      }
      }
    delay_cycles(4);
    //big_I2C_stop();
    while (R_W);
    while ((SSPCON2 & 0x1f)!=0);
    PEN=1;
}
```

12-2) CODE SOURCE 'C' : CARTE COMMANDE GOBO COULEUR

```
// // *****  Carte Gobos couleurs   adresse I2C: 3

#define Couleur   // à suprimer pour carte uniquement Gobo (sans couleur)
#include <16F876A.h>
#device adc=8
#FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, BROWNOUT, NOLVP, NOCPD, NOWRT
#use delay(clock=20000000)
const int8 ALIM_LP = 30 ; //
const int8 ALIM_HP = 250 ; //
const int8 v_mot = 16 ;
#use i2c(Slave,Slow,sda=PIN_C4,scl=PIN_C3,force_hw,address=0x06)  // bug adresse i2c *2
#use fast_io(B)
#use fast_io(C)
#byte PortB = 0x06
#byte SSPSTAT=0x94
#byte SSPBUF=0x13
#byte PIR1=0x0C
#byte PIE1=0x8C
#bit  TMR1_IF=PIR1.0  // à  1 indique le débordement du timer1
#bit  D_A= SSPSTAT.5
#bit  MGA = PortB.0
#bit  MGB = PortB.1
#bit  MCA = PortB.2
#bit  MCB = PortB.3
#bit  C_Gobo  = PortB.4
#bit  C_Coul  = PortB.5
int1  flg_gobo_horaire,flg_gobo_trigo,flg_coul_horaire,flg_coul_trigo;
int8  can_effet, can_gobo, can_coul; // valeur DMX des canaux 1 2 3
int8  n_gobo,n_coul; // numéro de gobo de 0 à 15 et de coul de 0 à 11
int8  pgobd,pcoud; // position gobo au couleur calculée
int8  tmp_gobo,tmp_coul; // tempo
int8  pos_gobo,pos_coul; //position réel des moteurs gobo et coul
int8  ptr;  // pointeur pour le tableau
int8  bufin[6]; // 3 octets utiles

//********   gestion INT_SSP
#int_SSP
SSP_isr() {
   int8 ssptemp;
   ssptemp = SSPSTAT & 0b00101101;
   if (!(ssptemp ^ 0b00001001))  {
      ptr = SSPBUF; //(pour lecture buffer)
      ptr=0;
      }
   else if (!(ssptemp ^ 0b00101001)){
      bufin[ptr] = SSPBUF;
      ++ptr;
      }
}

void rotation (int1 gobo_h, int1 gobo_t, int1 coul_h, int1 coul_t); //déclaration

void main() {
//********   initialiser les ressources internes du pic
   int8 i;
   int1 fgtim1 = 0;
   int1 fgtim2 = 0;
   setup_adc_ports(NO_ANALOGS);
   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
   setup_timer_1(T1_INTERNAL|T1_DIV_BY_4); // div by 4 -> résolution .8µs overflow 52.4ms
   setup_timer_2(T2_DIV_BY_1,255,1);  // deb 51,2µs
   setup_ccp2(CCP_PWM);   // mise en route du module PWM
   setup_comparator(NC_NC_NC_NC);
   setup_vref(FALSE);
   SET_TRIS_B(0xF0); //RB0 à RB3 sortie
   SET_TRIS_C(0xFD); //RC1 sortie
   bufin [1] = 0 ;  // gobo = 0
   bufin [2] = 0 ;  // coul =0
   set_pwm2_duty(ALIM_HP);
   flg_gobo_trigo=1;
   flg_coul_trigo=1;
   tmp_gobo = 1;
   tmp_coul = 1;
//********   initialiser la postition des disques gobo et  couleur
#ifdef Couleur    // compilation conditionnelle (si il n'y a pas de moteur 'couleur' suprimer le define du début)
```

```
  while (C_Gobo || C_Coul ){
#else
  while (C_Gobo){
#endif
    if (!C_Gobo)
      flg_gobo_trigo=0;
    if (!C_Coul)
      flg_coul_trigo=0;
    rotation (0,flg_gobo_trigo,0,flg_coul_trigo);
    delay_ms(v_mot);
    }
  pos_gobo = 0;
  pos_coul = 0;
  set_pwm2_duty(ALIM_LP);
//********   activer interruption SSP
  i = SSPBUF;
  enable_interrupts(INT_SSP);
  enable_interrupts(GLOBAL);
  n_gobo = 0;
  n_coul = 0;
//*****************************************************************************//
  while (true)
    {
    can_effet = bufin[0];
    can_gobo = (bufin[1]) ;  //
    can_coul = bufin[2]   ;  //
//********   déterminer le n°de gobo à visualiser
    if (can_gobo < 128)
      {   // on déduit e le N°de gobo (0 à15)
      n_gobo = can_gobo / 8;
      }
    else
      {  // on inc le N°de Gobo aprés tempo
      i = ((~can_gobo)>>2)+1 ; // & 0x7f;
      if (i < tmp_gobo)
        tmp_gobo = i ;
      if (TMR1_IF)
        {
        --  tmp_gobo;
        set_timer1(0000);  // le réglage du débordement du timer1 est de 52ms
        TMR1_IF=0;       // mise à 0 du flag débordement du timer1
        fgtim1 =1;
        }
      if (fgtim2)
        {
        --tmp_gobo;
        fgtim2 =0;
        }
      if (tmp_gobo ==0)
        {
        n_gobo++;
        tmp_gobo = i ;
        }
      if (n_gobo == 16)
        n_gobo = 0 ;
      }
    pgobd = n_gobo * 3;
//*******    déterminer le n°de couleur à visualis er
    if (can_coul < 128){   // on déduit e le N°d e couleur  (0 à 11)
      n_coul= can_coul / 11;
      }
    else {  // on inc le N°de couleur aprés la   tempo
      i = ((~can_coul )>>2) +1  ;
      if (i < tmp_coul)
        tmp_coul = i ;
      if (TMR1_IF)
        {
        --  tmp_coul;
        set_timer1(0000);  // le réglage du débordement du timer1 est de 52ms
        TMR1_IF=0;       // mise à 0 du flag débordement du timer1
        fgtim2 =1;
        }
      if (fgtim1)
        {
        --tmp_coul;
        fgtim1 =0;
        }
      if (tmp_coul ==0){
        n_coul++;
```

```
          tmp_coul = i ;
          }
        if (n_coul == 12)
          n_coul = 0 ;
        }
      pcoud = n_coul * 4;
////////////////////////////////////////////////////////////////////////////////
      //  à ajouter gestion des effets.....
////////////////////////////////////////////////////////////////////////////////
//  pgobd et pcoud contiennent les positions gobo et couleurs désirées de 0 à 48
//********   déterminier les moteurs à commander ainsi que leur sens de rotation
// pour le disque des gobos
      flg_gobo_horaire = 0 ;
      flg_gobo_trigo = 0 ;
      if ( pos_gobo != pgobd)
        {
        if (pos_gobo > pgobd)
          {
          if ((pos_gobo - pgobd) <24)
            {
            flg_gobo_trigo=1;
            --pos_gobo;
            }
          else
            {
            flg_gobo_horaire=1;
            ++pos_gobo;
            }
          }
        else
          {
          if ((pgobd - pos_gobo) <24)
            {
            flg_gobo_horaire=1;
            ++pos_gobo;
            }
          else
            {
            flg_gobo_trigo=1;
            --pos_gobo;
            }
          }
        }
      if (pos_gobo == 49)
      pos_gobo = 0;
      if (pos_gobo == 255)
      pos_gobo = 48;
// idem disque couleur
      flg_coul_horaire = 0 ;
      flg_coul_trigo = 0 ;
      if ( pos_coul != pcoud)
        {
        if (pos_coul > pcoud)
          {
          if ((pos_coul-pcoud) <24)
            {
            flg_coul_trigo=1;
            --pos_coul;
            }
          else
            {
            flg_coul_horaire=1;
            ++pos_coul;
            }
          }
        else
          {
          if ((pcoud - pos_coul) <24)
            {
            flg_coul_horaire=1;
            ++pos_coul;
            }
          else
            {
            flg_coul_trigo=1;
            --pos_coul;
            }
          }
        }
```

```
    if (pos_coul == 49)
      pos_coul = 0;
    if (pos_coul == 255)
      pos_coul = 48;
//*****************************************
    if (flg_gobo_horaire || flg_gobo_trigo || flg_coul_horaire || flg_coul_trigo)
      set_pwm2_duty(ALIM_HP); // augmenter la tension d'alimentation des moteurs
    else
      set_pwm2_duty(ALIM_LP); // réduire la tension d'alimentation des moteurs
//********  commander le déplacement du / des moteurs
    rotation (flg_gobo_horaire,flg_gobo_trigo,flg_coul_horaire,flg_coul_trigo);
//********  attendre 16ms
    delay_ms(v_mot);
  }
}

void rotation (gobo_h, gobo_t, coul_h, coul_t){
  if (gobo_h){  // si flag gobo_h à 1 rotation d'un pas du moteur gobo dans le sens Horaire
    if (MGA){
      if (MGB)
        MGB=0;
      else
        MGA=0;
      }
    else {
      if (MGB)
        MGA=1;
     else
        MGB=1;
      }
   }
  if (gobo_t){   // si flag gobo_t à 1 rotation d'un pas du moteur gobo dans le sens Trigo
    if (MGA){
      if (MGB)
        MGA=0;
      else
        MGB=1;
      }
    else {
      if (MGB)
        MGB=0;
      else
        MGA=1;
      }
  }
  if (coul_h){  // si flag coul_h à 1 rotation d'un pas du moteur coul dans le sens Horaire
    if (MCA){
      if (MCB)
        MCB=0;
     else
        MCA=0;
      }
   else {
      if (MCB)
        MCA=1;
      else
        MCB=1;
     }
  }
  if (coul_t){   // si flag coul_t à 1 rotation d'un pas du moteur coul dans le sens Trigo
    if (MCA){
      if (MCB)
        MCA=0;
      else
       MCB=1;
      }
    else {
      if (MCB)
        MCB=0;
     else
        MCA=1;
     }
  }
}
```

12-3) CODE SOURCE 'C' : CARTE COMMANDE PAN TILT

```
// *****  Carte Pan Tilt  adresse I2C: 1

#define Tilt   // à supprimer pour carte à moteur Pan seul (sans tilt)
#include <16F876A.h>
#device adc=8
#FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, BROWNOUT, NOLVP, NOCPD, NOWRT
#use delay(clock=20000000)
const int8 ALIM_LP = 30 ; //
const int8 ALIM_HP = 250 ; //
const int8 v_mot = 5 ;//
#use i2c(Slave,Slow,sda=PIN_C4,scl=PIN_C3,force_hw,address=0x02)  // bug adresse i2c *2
#use fast_io(B)
#use fast_io(C)
#byte PortB = 0x06
#byte SSPSTAT=0x94
#byte SSPBUF=0x13
#byte PIR1=0x0C//
#byte PIE1=0x8C//
#bit  D_A= SSPSTAT.5
#bit  MPA = PortB.0
#bit  MPB = PortB.1
#bit  MTA = PortB.2
#bit  MTB = PortB.3
#bit  C_Pan   = PortB.4
#bit  C_Tilt  = PortB.5

int1   fph,fpt,fth,ftt;
int16  can_p, can_t; // valeur DMX des canaux 4 et 5 ( si adresse lyre pédagogique = 1 )
int16  pos_p,pos_t;   //position réel des moteurs pan et tilt
int8   ptr;  // pointeur pour le tableau
int8   bufin[4]; // 2/3 octets utiles


//********   gestion INT_SSP
#int_SSP
SSP_isr() {
   int8 ssptemp;
   ssptemp = SSPSTAT & 0b00101101;
   if (!(ssptemp ^ 0b00001001))  {
      ptr = SSPBUF; //(pour lecture buffer)
      ptr=0;
      }
   else if (!(ssptemp ^ 0b00101001)){
      bufin[ptr] = SSPBUF;
      ++ptr;
      }
}

void rotation (int1 Flg_pan_h, int1 Flg_pan_t,  int1 Flg_tilt_h, int1 Flg_tilt_t); //déclaration

void main() {
//********   initialiser les ressources internes du pic

   int8 i;
   setup_adc_ports(NO_ANALOGS);
   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DIV_BY_1,255,1);  // deb 51,2µs
   setup_ccp2(CCP_PWM);   // mise en route du module PWM
   setup_comparator(NC_NC_NC_NC);
   setup_vref(FALSE);
   SET_TRIS_B(0xF0); //RB0 à RB3 sortie
   SET_TRIS_C(0xFD); //RC1 sortie
   bufin [0] = 0 ;  // pan = 0
   bufin [1] = 0 ;  // tilt =0
   set_pwm2_duty(ALIM_HP);
   fpt=1;
   ftt=1;
//********   initialiser la postition de la tête

#ifdef tilt    // compilation conditionnelle (si il n'y a pas de moteur 'Tilt' suprimer le define du début)
   while (fpt||ftt){             //tant que les 2 zéros ne sont pas détectés
#else
   while (! (C_Pan)){                    //ne pas tenir compte de cette ligne avec 2 moteurs
```

```
#endif                          // les moteurs tournent....
    if (C_Pan)                    // c_pan ==1 si le moteur pan s'arrête
      fpt=0;                     // initialisation de fpt à 0 : mémorisation position d'origine pour moteur pan
    if (C_Tilt)                   // c__tilt ==1 si le moteur tilt s'arrête
      ftt=0;                     // initialisation de ftt à 0 : mémorisation position d'origine pour moteur tilt
    rotation (0,fpt,0,ftt);         // recherche position d'origine
    delay_ms(v_mot);
              }
    pos_p = 0;                    // fait coincider la variable "Position Réélle Pan" avec  "position d'origine" pour pan
    pos_t = 0;                    // fait coincider la variable "Position Réélle tilt" avec  "position d'origine" pour tilt
    set_pwm2_duty(ALIM_LP);
//********   activer interruption SSP

    i = SSPBUF;
    enable_interrupts(INT_SSP);
    enable_interrupts(GLOBAL);

  while (true)
        {
                        //********   déterminer les moteurs à commander ainsi que leur sens de rotation
   can_p = (bufin[0]) * 2;                  // chargement position curseur pan x 2
   can_t = bufin[1] * 2;                    //chargement position curseur tilt x 2
    fph = 0 ;                             //raz fph ( sens horaire pour pan )
    fpt = 0 ;                             //raz fph ( sens trigo pour pan )
    if ( pos_p != can_p)                  //compare pos_p et can_p ( position désirée )
      {
      if (pos_p > can_p) {
        fpt=1;
        --pos_p;
                    }
      else{
        fph=1;
        ++pos_p;
         }
      }
    fth = 0 ;
    ftt = 0 ;
    if ( pos_t != can_t)
      {
      if (pos_t > can_t) {
        ftt=1;
        --pos_t;
                    }
      else {
        fth=1;
        ++pos_t;
          }
      }

    if (fph || fpt || fth || ftt)//ces flags indiquent le sens de rotation souhaité pour chaque moteur.
      set_pwm2_duty(ALIM_HP);// augmenter la tension d'alimentation des moteurs
    else
      set_pwm2_duty(ALIM_LP);// réduire la tension d'alimentation des moteurs
//********   commander le déplacement du ou des moteurs
    rotation (fph,fpt,fth,ftt);
//********   attendre 3ms
    delay_ms(v_mot);
  }
}
//rotation d'un pas
void rotation (int1 Flg_pan_h, int1 Flg_pan_t, int1 Flg_tilt_h, int1 Flg_tilt_t) {
  if (Flg_pan_h){  // si flag pan_h à 1 rotation d'un pas du moteur pan dans le sens Horaire
    if (MPA){
      if (MPB)
        MPB=0;
      else
        MPA=0;
       }
    else {
      if (MPB)
        MPA=1;
     else
        MPB=1;
      }
          }
  if (Flg_pan_t){   // si flag pan_t à 1 rotation d'un pas du moteur pan dans le sens Trigo
    if (MPA){
      if (MPB)
        MPA=0;
```

```
      else
        MPB=1;
         }
   else {
      if (MPB)
        MPB=0;
      else
        MPA=1;
       }
            }
if (Flg_tilt_h){  // si flag tilt_h à 1 rotation d'un pas du moteur tilt dans le sens horaire
   if (MTA){
      if (MTB)
        MTB=0;
    else
        MTA=0;
       }
  else {
      if (MTB)
        MTA=1;
      else
        MTB=1;
       }
              }
if (Flg_tilt_t){   // si flag tilt_t à 1 rotation d'un pas du moteur tilt dans le sens Trigo
   if (MTA){
      if (MTB)
        MTA=0;
      else
       MTB=1;
     }
   else {
      if (MTB)
        MTB=0;
     else
        MTA=1;
       }
            }
 }
```

12-4) Code source 'c' : Carte Alimentation Dimmer

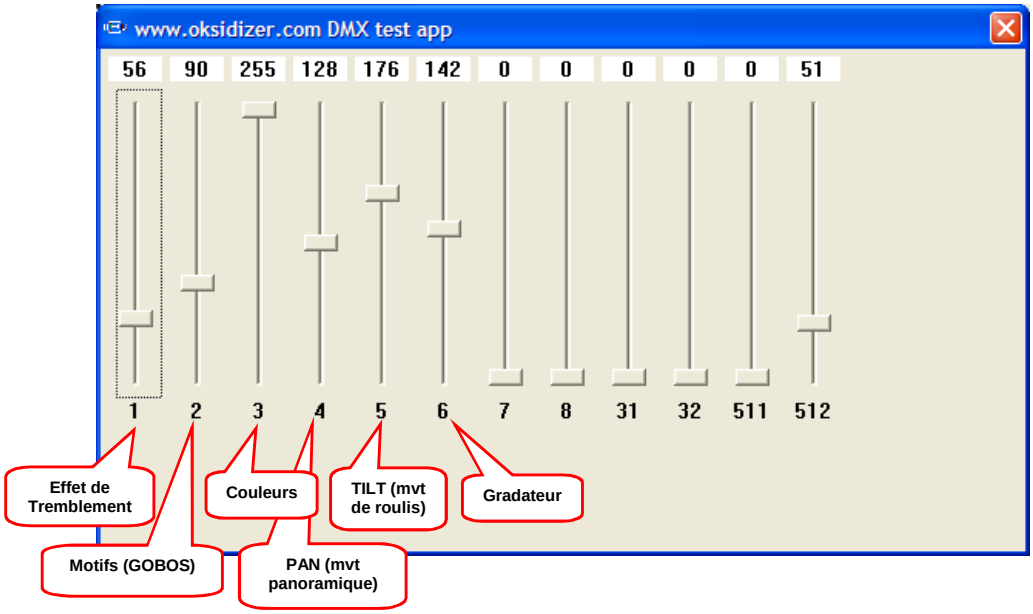
```
// *****  Carte Alim  dimmer adresse I2C: 5
#include <16F876A.h>
#device adc=8
#FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, BROWNOUT, NOLVP, NOCPD, NOWRT
#use delay(clock=20000000)
//#define CMLAMP   PIN_C2
//#define CPTT     PIN_A0
#define VREF     PIN_A3
#use i2c(Slave,Slow,sda=PIN_C4,scl=PIN_C3,force_hw,address=0x0a)  // bug adresse i2c *2
#use fast_io(B)
#use fast_io(C)
#byte SSPSTAT=0x94
#byte SSPBUF=0x13
#bit  D_A= SSPSTAT.5
#bit  CMVENT=0x06.1   // port B bit 1
int8   can5,ptr;
int8   bufin[4];
int8   val_teta;
const int8 TETA_OFF = 70 ; // Pour 35°C (Vref=An3)
const int8 TETA_ON = 90 ; // pour 45 °C

//********   gestion INT_SSP
#int_SSP
SSP_isr() {
   int8 ssptemp;
   ssptemp = SSPSTAT & 0b00101101;
   if (!(ssptemp ^ 0b00001001))
      {
      ptr = SSPBUF; //(pour lecture buffer)
      ptr=0;
      }
   else if (!(ssptemp ^ 0b00101001)){
      bufin[ptr] = SSPBUF;
      ++ptr;                                          }
}

void main() {
//********   initialiser les variables et les ressources internes du pic
   int16 i;  //(16 pour tempo)
   setup_adc_ports(AN0_AN1_VSS_VREF);
   setup_adc(ADC_CLOCK_INTERNAL);
   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DIV_BY_1,255,1);  // deb 51,2µs
   setup_ccp2(CCP_PWM);   // mise en route du module PWM
   setup_comparator(NC_NC_NC_NC);
   setup_vref(FALSE);
   SET_TRIS_B(0xFD); //RB1 sortie
   SET_TRIS_C(0xFD); //RC1 sortie
   set_adc_channel(0); //
   can5 = 0 ;
   bufin [0] = 75 ; // lampe à 30%
   CMVENT = 1 ; // ventilo au ralenti
//********   activer interruption SSP
   enable_interrupts(INT_SSP);
   enable_interrupts(GLOBAL);
   while (true)
   {
//********   actualiser la consigne de luminosité
      if ( bufin[0] != can5)
         {
         can5 = bufin[0];
         set_pwm2_duty(can5);                  // rapport cyclique à 50%
         }
//******** gestion de la température - on compare la moyenne de 4 mesures et les ref
      val_teta =0 ;
      for (i=0;i<4;i++)
         val_teta = val_teta + (read_adc() / 4) ;
      if (val_teta < TETA_OFF) //35°C mettre le ventilo au ralenti
         CMVENT = 1 ;
      if (val_teta > TETA_ON)  //45°C mettre le ventilo au max
         CMVENT = 0 ;
   }
}
```

13 - Description sommaire du logiciel Test_cpp

Test_cpp se présente comme une fenêtre Windows comportant 12 potentiomètres virtuels destinés à définir les niveaux des canaux DMX 1 à 8, et des canaux 31, 32, 511 et 512.

Pour chaque canal, le niveau varie entre 0 et 255 selon la position du curseur.

Si la lyre est configurée à l'adresse DMX n°1, alors les effets commandés sont ceux figurant dans les bulles de l'illustration suivante.



* L'effet tremblement n'existe que sur la lyre réelle et le gradateur n'existe que sur le système didactique.

14 - Description sommaire du logiciel Freestyler

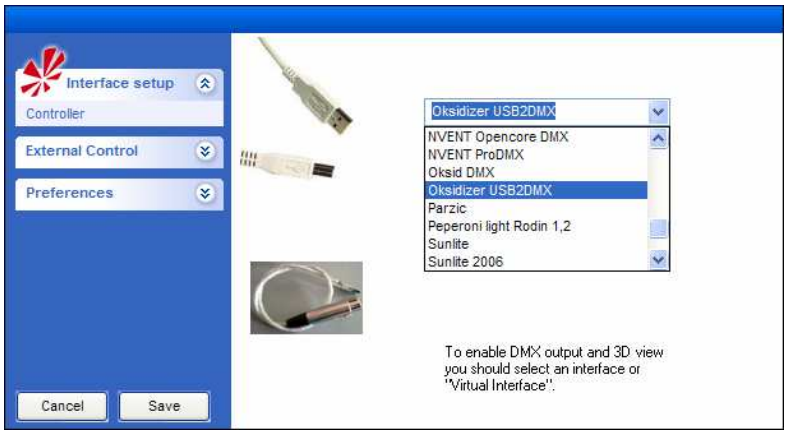
Lors de la première utilisation, il faut sélectionner l'interface utilisé (ici sur la ligne USB) ainsi que les projecteurs et effets à commander.

Une fois la configuration sauvegardée, il sera possible de passer directement au pilotage des effets.

14-1) Démarrage de Freestyler, et choix des « fixtures »

Dans le Menu Setup :

Aller à « FreeStyler Setup et choisir l'interface « Oksidizer USB2DMX »



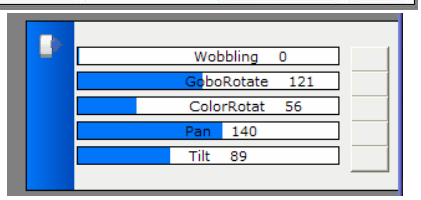
Aller ensuite à « Add/Remove fixtures » ;
pour configurer la lyre IMG Stageline ,choisir ce constructeur dans la liste et sélectionner le projecteur SCAN-25.

Le logiciel propose d'installer **1** projecteur,
et de lui donner l'adresse DMX de départ : **1**

Accepter ces choix.

Le projecteur apparaît en haut à gauche dans la fenêtre principale (il s'agit ici d'un projecteur à miroir asservi, mais qui possède les mêmes commandes que la lyre).

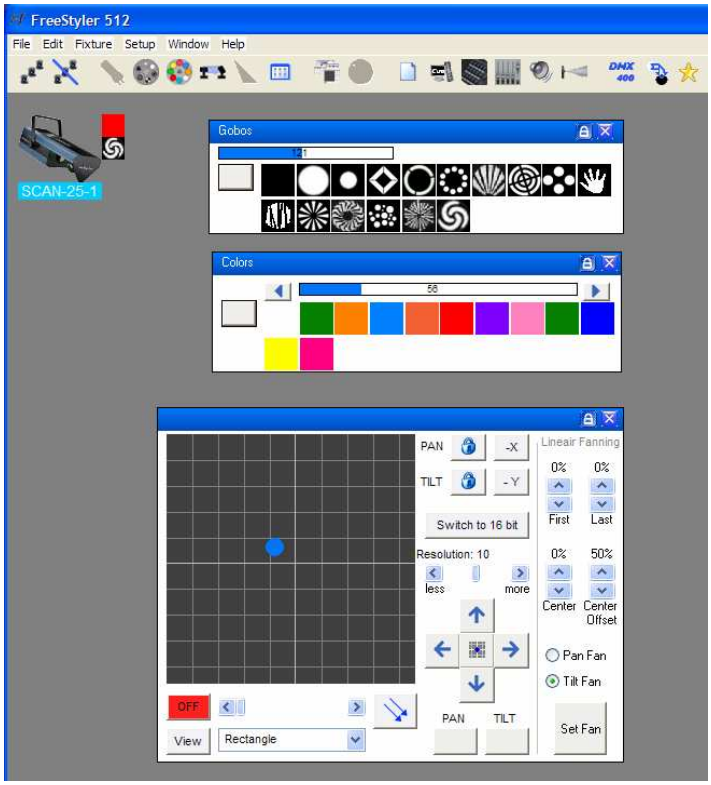
En sélectionnant le projecteur (cliquer dessus), une tirette sur la droite permet d'ouvrir une fenêtre qui montre les valeurs courantes des codes DMX envoyés au projecteur.



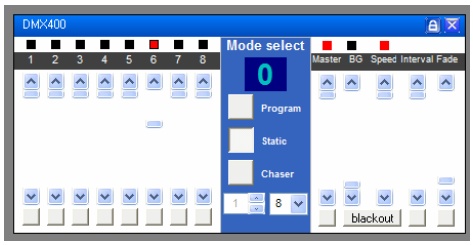
Vous pouvez enregistrer la configuration « File, Save Locations ».

14-2) Pilotage des effets avec Freestyler

En cliquant sur les boutons correspondants sous la ligne de menus, on fait apparaître des fenêtres de contrôle pour chaque effet. A noter le plan 2D qui permet de gérer avec un seul pointage les mouvements PAN et TILT.



Le bouton console DMX400 vous permet de commander directement les effets par potentiomètres (de manière statique), comme le gradateur de la maquette élève, à l'adresse DMX 6.



Avec la console DMX400 vous pouvez aussi programmer des scènes (ensemble de réglages) que vous faites défiler ensuite de manière dynamique.

Pour de plus amples renseignements, voir le fichier d'aide en français FREESTYLER-FR.hlp

15 - Annexe 1 : Le Protocole DMX-512

Le DMX-512 est un protocole mis au point en 1986 et 1990 par l'USITT (United Institute of Theater Technology) ; il définit une norme de transmission de données pour les techniques d'éclairage.

Cette norme est libre de droits et sa mise en œuvre est simple et économique : une liaison filaire issue du pupitre de commande relie l'un à l'autre tous les récepteurs et permet de véhiculer l'ensemble des informations qui seront traduites en intensité lumineuse, couleur, mouvement, etc...

Ce protocole entièrement numérique s'inspiré d'une technique éprouvée, les bus de liaison RS 485, utilisés en informatique et dans l'industrie.

Le DMX-512 simplifie notablement le pilotage d'un système d'éclairage complexe.

15-1) - Principe

Le protocole DMX fixe un standard pour la transmission d'informations entre une commande et des récepteurs déportés. Tout repose sur des trains d'impulsions numériques composés de signaux rectangulaires transmis de façon cyclique à une fréquence de 250 kHz. Le contenu des trames quand à lui, reste identique sur tout le cheminement du bus.

Émetteur :

La liaison est un bus de transmission unidirectionnel. Il y a un émetteur exclusif et pas de retour d'information. Les données sont recopiées par chaque récepteur, la plupart du temps de façon passive.

Récepteurs :

1 à 32 récepteurs peuvent être connectés en file derrière un émetteur. Le nombre de récepteurs dépend de l'ensemble des caractéristiques électriques que présente un réseau.

Terminaison :

Il est nécessaire de brancher une résistance de terminaison en fin de ligne (vulgairement appelée bouchon DMX). Son rôle est d'assurer la bonne circulation du courant entre les conducteurs actifs. Elle empêche des réflexions de trames déjà transmises qui perturberaient la validité des signaux.

Longueur d'une ligne :

S'appuyant sur couche physique RS-485, la norme DMX autorise des liaisons jusqu'à 1000 mètres, à condition d'utiliser un câble, spécialisé pour la transmission de données, adapté à la norme RS-485.

Un système d'adressage permet aux récepteurs de ne prendre en compte que les valeurs des canaux qui leur sont affectés. Il devient alors très facile d'intégrer un nouvel appareil dans la chaîne, chaque récepteur recevant l'intégralité des trames transmises.

Canaux :

512 canaux peuvent être transmis sur une même trame.

Précautions :

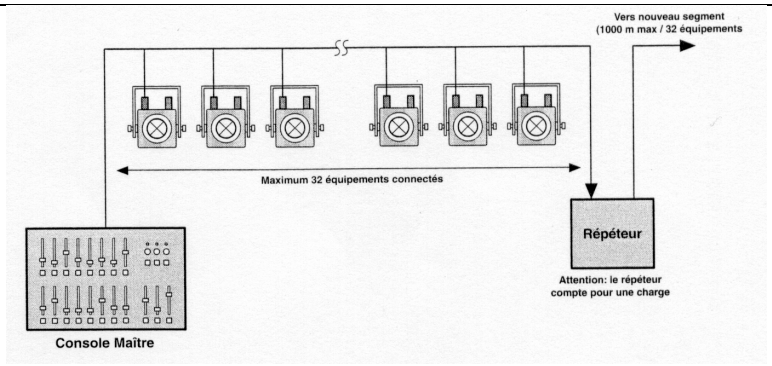
Les branchements en Y sont interdits. Un nouvel appareil sera obligatoirement inséré dans la liaison série existante.

Extension de réseaux :

Si l'on envisage de brancher plus de 32 récepteurs derrière sa console, de parcourir des distances importantes, de partir dans plusieurs directions, etc… on aura recours aux appareils suivants :

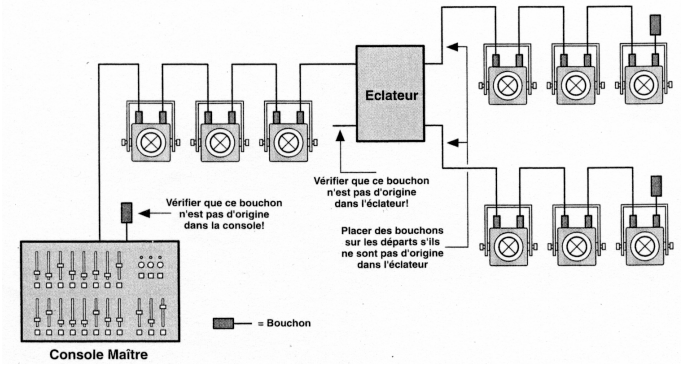
Le répétiteur :

Cet appareil rafraîchit ou remet en forme les impulsions d'un signal DMX. Il est intercalé sur le bus, pour fractionner les longues distances. Il permet aussi l'ajout de 32 récepteurs supplémentaires. Les terminaisons sont toujours nécessaires sur les fins de lignes.



L'éclateur ou *splitter* :

Le splitter est un répétiteur possédants plusieurs départs vers 32 récepteurs ; les terminaisons sont là, encore nécessaires, y compris sur les sorties inutilisées.



15-2) Le multiplexage des données

Principe :

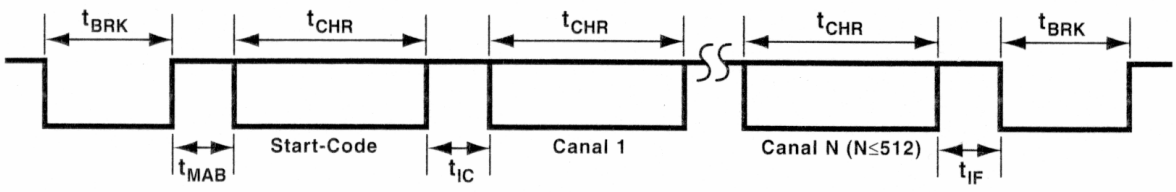
Les données d'un bus DMX 512 sont transmises sous la forme d'une succession d'octets. Un octet est composé de huit bits et peut avoir 256 états pour représenter une valeur d'intensité, de couleur, de position, etc…)

Le principe de commande est très simple : on transmet successivement sur la liaison série des octets, dont la valeur correspond à la consigne de niveau de chaque canal.

Structure d'une trame DMX :

Un cycle commence par une Initialisation : Break + MAB (Mark After Break), suivi d'un Start-code et des octets utiles (consignes pour les canaux DMX).

Les récepteurs détectent et se synchronisent sur ce début de cycle ; ils comptent alors les « octets » ou caractères transmis et enregistrent celui (ou ceux) qui correspond(ent) à leur(s) numéro(s) de canal. Les autres octets sont ignorés. L'ensemble de la trame est transmise (souvent de façon passive) au récepteurs suivants.



t_{BRK} = Temps de Break = 88 µs min à 1 s max
t_{MAB} = Temps de Mark = 4 µs (DMX512 - 1986) / 8 µs (DMX512 - 1990)
t_{IC} = Temps intercaractère = 0 µs min à 1 s max
t_{IF} = Temps intertrame = 0 µs min à 1 s max
t_{CHR} = Durée d'un caractère (1 bit de Start + 8 bits de donnée + 2 bits de stop = 11 bits)
(Durée nominale d'un bit = 4 µs / 3,92 µs min / 4,08 µs max) soit 44 µs typique

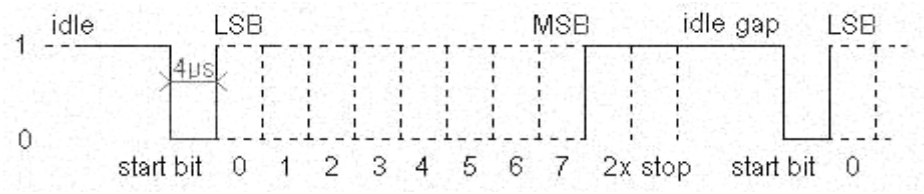
- Le **break** (niveau 0) de 88 µs minimum correspond à une interruption d'émission pendant la durée d'au moins deux octets ; c'est un état atypique facilement détectable par les récepteurs. Attention, certains équipements tolèrent mal les durée trop importantes (au delà de 200 ms).

- L'impulsion **Mark After Break** (état 1) a une durée de 8 µs minimum (durée deux bits).
- Le **start-code** généralement à 0 indique la nature des informations transmises (0 pour les données linéaires sur 8 bits) ; d'autres valeurs de start-code sont réservés pour un usage futur.
- Les **données** sont présentées sur le bus de façon sérielle. Un bit a une durée de 4 µs ± 2%.
- Un **temps de repos** (idle) caractérisé par état haut de la ligne peut être intercalé entre les paquets de données.

Composition d'un caractère :

Les octets sont constitués de 11 bits (1 bit de start +8 bits de données + 2 bits de stop). La durée d'un octet est de 44 µs minimum (rappel : 11 fois la durée 1 bit 4 µs).

Des temps de pause peuvent être intercalés entre deux octets.



- Un **start-bit**, état bas, précède la transmission de l'octet
- **LSB > MSB** : le bit de poids le plus faible jusqu'au bit de poids le plus fort
- **Deux bits de stop**, état haut, après la fin de l'octet
- Certaines consoles ne transmettent pas les données de façon continue et peuvent intercaler un temps de pause **idle** avant le start-bit suivant.

Tableau des durées :

Limites des temps de transmission :

- L'intervalle séparant deux impulsions de Mark (remise à zéro) doit être:
- d'au moins 1 196 µs (≅ 2 ms ; durée de la transmission de 24 codes DMX)
- au maximum d'une seconde (temps de repos inclus).
- En cas d'absence de signal, le récepteur doit maintenir son dernier état au minimum pendant cette durée d'une seconde.

appellations	durée type (µs)	duree min. (µs)	duree max. (µs)	valeur utilisée dans le présent système
break	88	88	200 *	160
MAB (mark after break)	8	8	10^6	25
1 bit	4	3,92	4,08	4
1 caractère 11 bits	44	43,12	44,88	44
entre deux trames	22 668	2 ms	1 s	11 ms

Les durées minimales sont données pour un système à 24 canaux DMX.

15-3) Composition de la période de la trame :

(Trame émise par la console via le logiciel CPP)

Dans la description qui suit, la période est calée de telle sorte que les nombres représentants les temps soient les plus simples possibles.
La période dure 77 ms (4ms + 73 ms)

De 0 à 4 ms :

De 0 à 0,4 ms :

44µs (pour le dernier canal) (voir composition de chaque canal*)
160 µs pour le TIF (temps intercalaire final) (niveau haut)
112 µs pour le break** (niveau bas)
40 µs pour le MAB (mark after break) (niveau haut)
44µs pour le start code (canal zéro). Quand il contient une valeur nulle, cela signifie que les commandes sont proportionnelles.

De 0,4 à 4ms :

TICo (temps inter canal initial).
Pendant ce temps, la ligne est au niveau haut (pseudo repos)
C'est le temps entre le start-code et le canal 1.
C'est ce temps qui apparaît nettement (trace au niveau haut), quand on visualise une période complète de la trame.

De 4 à 77ms :

Pendant chaque milliseconde est émis
-1 paquet de 7 canaux (7 x 44µs) avec un TIC (temps intercalaire) nul, entre chaque canal d'un paquet courant(n°1 à n° 511)
- suivi d'un temps inter paquet (TIP) de 692 µs (au niveau haut)
soit en tout 73 paquets.

*Description de la composition d'un canal :

Il comporte 11 bits en tout (chaque bit dure 4 µs) donc la durée totale d'un canal est de 44µs
-Un bit de start (niveau bas)
-D0 (le bit zéro de la donnée démultiplexée) (LSB)
-D1
-D2
-D3
-D4
-D5
-D6
-D7 (le dernier bit de la donnée démultiplexée) (MSB)
-2 bits de stop (au niveau haut)

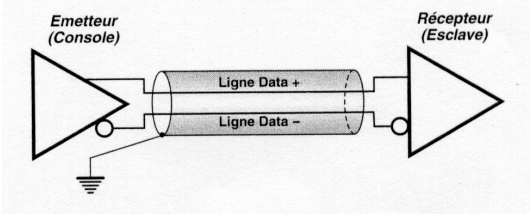
** Les récepteurs sont synchronisés sur le break, qui initialise un compte à rebours pour chaque récepteur. Ce qui permet à chaque récepteur de n'enregistrer que le segment qui lui est utile dans chaque période de trame.
Il aurait été plus logique de commencer la description de la trame par le break, mais le découpage dans le temps étant plus compliqué, cette possibilité n'a pas été retenue.

15-4) Caractéristiques électriques

La définition des caractéristiques électriques suivantes assure la compatibilité entre les différents appareils reliés en réseaux.

Les liaisons :

Les signaux sont transmis au moyen d'une liaison symétrique. Deux fils véhiculent les signaux en opposition de phase et l'étage d'entrée du récepteur détecte les différences d'amplitude entre ces deux conducteurs. De cette façon un parasite induit sur les deux fils sera ramené à un potentiel nul grâce à l'utilisation d'un amplificateur différentiel.



Amplitude des signaux :

La tension entre les deux conducteurs actifs doit être au minimum de 200 mV. Les valeurs limites sont fixées de - 7 V min. à + 12 V max. par rapport à la masse.

Liaison série :

La transmission est de type liaison série asynchrone ; chaque unité d'information est composée de plusieurs bits qui sont transmis les uns après les autres dans un ordre défini.

La transmission est unidirectionnelle, aucun acquittement des récepteurs vers l'émetteur n'est possible ; la fiabilité du DMX est simplement assurée par la répétition constante des cycles

Vitesse de transmission :

Sa vitesse de transmission est de 250 kbauds. La durée d'un bit est donc de 4 µs. A noter que de par leur forme les signaux occupent une bande passante atteignant les 2,5 MHz.

Quantification des données :

Le codage des informations se fait sur 8 bits par adresse soit 255 valeurs possibles. Un bit a une résolution de 100 / 255 = 0,39 % de la valeur pleine échelle. L'utilisation possible de deux canaux DMX permet la gestion de paramètres sur 16 bits.

Impédances des appareils:

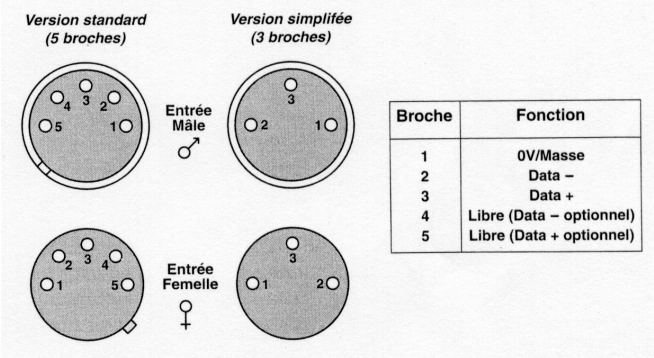
Celle d'un récepteur doit être supérieure à 12 k Ω et celle de la ligne supérieure à 60 Ω.

Caractéristiques des liaisons :

Pour les liaisons, deux paires en 2x2x 0.22 mm min. et protégées contre les rayonnements extérieurs (feuille en aluminium + blindage) sont préconisées. L'impédance caractéristique entre les conducteurs actifs est de 80 pF/m et entre conducteur et feuille de masse de 150 pF/m.

15-4) Connectique

La norme prévoit l'usage de connecteurs de type XLR à 5 broches ; en fait, on utilise plus souvent des connecteurs XLR à 3 broches, fréquemment utilisés en audio et donc moins chers.



Broche	Fonction
1	0V/Masse
2	Data –
3	Data +
4	Libre (Data – optionnel)
5	Libre (Data + optionnel)

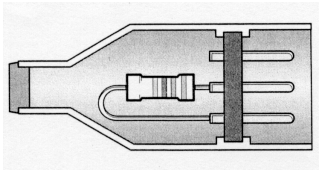
Les sorties (console, interface maître) doivent être équipées de connecteurs femelles, les entrées (projecteurs, gradateurs, …) de connecteurs mâles ; ces derniers possèdent généralement un second connecteur femelle pour la reprise de liaison vers les autres équipements esclaves.

Emetteur	châssis femelle (console)
Récepteur	châssis mâle (gradateur)
Recopie en sortie du récepteur	châssis femelle (gradateur)
Terminaison	fiche mâle (dernier récepteur)

Il ne doit y avoir aucune liaison entre la masse et l'un des points chauds (Data+ et Data-).

Réalisation du *bouchon terminal* :

Il suffit de souder une résistance de 120 Ohms entre les points chauds (2 et 3) à l'intérieur d'une fiche mâle.



Application : il vous faudra réaliser un câble DMX de quelques mètres pour la liaison entre la console et la lyre, et un « bouchon » de terminaison. Veillez à acquérir des fiches XLR 3 broches et du câble adéquat.

16 - Annexe 2 : Le bus I2C

16-1) Introduction

Le bus I2C (Inter Integrated Circuit Bus) est le bus historique, devenu standard, développé par Philips dans les années 80, pour permettre de relier facilement à un microprocesseur divers circuits intégrés (spécialisés dans le stockage et l'affichage de données, dans l'exécution de fonction numériques ou analogiques diverses), en particulier dans un téléviseur.

Il existe d'innombrables périphériques exploitant ce bus, dans les appareils TV et vidéo (récepteur télécommande, réglages ampli BF, tuner, horloge, gestion péritel…), mais aussi dans les systèmes audio et radio, postes téléphoniques, systèmes électroniques automobiles, appareils électroménagers, etc.

16-2) Caractéristiques générales

Le bus I2C permet de faire communiquer entre eux des composants électroniques très divers grâce à seulement trois fils :

- Un signal de donnée (SDA),
- Un signal d'horloge (SCL),
- Un signal de référence électrique (Masse).

Ceci permet de réaliser des équipements ayants des fonctionnalités très puissantes en conservant un circuit imprimé très simple, par rapport un schéma classique (8 bits de données, 16 bits d'adresse + les bits de contrôle).

D'autre part, ce bus est multi-maître (plusieurs circuits peuvent prendre le contrôle du bus) et sa longueur peut atteindre 3 à 4 mètres à condition que la charge capacitive n'excède pas 400 pF.

Les données sont transmises en série à 100 kbits/s en mode standard et jusqu'à 400 kbits/s en mode rapide. Ce qui ouvre la porte de cette technologie à toutes les applications où la vitesse n'est pas primordiale.

De nombreux fabricants ayant adopté le système, la variété des circuits disponibles disposant d'un port I2C est énorme : Ports d'E/S bidirectionnels, Convertisseurs A/N et N/A, mémoires (RAM, EPROM, EEPROM, etc...), Circuits Audio (Egaliseur, Contrôle de volume, ...) et autre drivers (LED , LCD , ...).

16-3) Fonctionnement du bus I2C

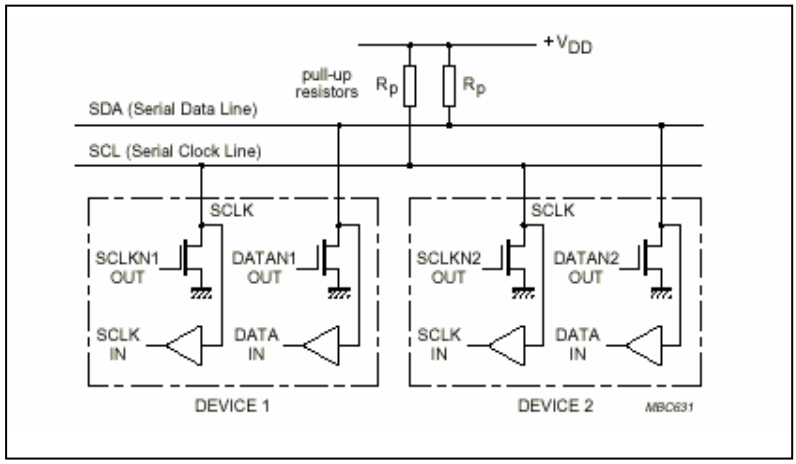
Caractéristiques électriques

Afin de d'éviter les conflits électriques les Entrées/Sorties, SDA et SCL sont de type "Collecteur Ouvert" (ou "Drain Ouvert").

Les sorties étant à collecteur (ou drain) ouvert la tension de sortie à l'état haut est la tension « ramenée » par les résistances de rappel (qui sont connectées à la ligne d'alimentation des circuits « VDD »).

Cela permet ainsi la présence de plusieurs maîtres sur le bus.

Cela permet aussi la communication entre dispositifs réalisés dans des technologies différentes et utilisant éventuellement des tensions d'alimentation différentes.



Pour les dispositifs fonctionnant sous une tension de 5V ±10%, les niveaux d'entrée sont : V_{ILmax} = 1,5V et V_{IHmin} = 3V . Dans tous les cas la tension de sortie à l'état bas est V_{OLmax} = 0,4V

Terminologie

Emetteur : Unité qui envoie les données sur le bus.
Récepteur : Unité qui reçoit les données du bus.
Maître : Unité qui démarre un transfert, génère des signaux d'horloge et met fin au transfert .
Esclave : Unité adressée par le maître.
Multi-maître : Possibilité pour plusieurs maîtres de tenter de prendre le contrôle du bus en même temps, sans altérer le message.
Arbitrage : Procédure permettant de résoudre les conflits d'accès des maîtres au bus et d'éviter l'altération du message. Le contrôle du bus n'est accordé qu'à un maître à la fois .
SDA : Ligne des signaux de données.
SCL : Ligne des signaux d'horloge.

Protocole de transmission .

Le protocole de communication doit prendre en compt les règles suivantes :
- une distinction entre les circuits maîtres (ceux qui décident du dialogue) et les circuits esclaves,
- une identification des circuits,
- un acquittement des transferts (confirmation par les circuits de la bonne réception des informations qui leur ont été transmises),
- un système de priorité en cas de conflit.

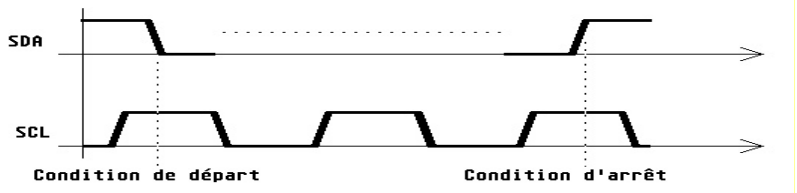
Dans les applications courantes les modes les plus utilisés sont les suivants :

Le maître veut lire une information contenue dans un circuit esclave.

Le déroulement de la séquence va être :

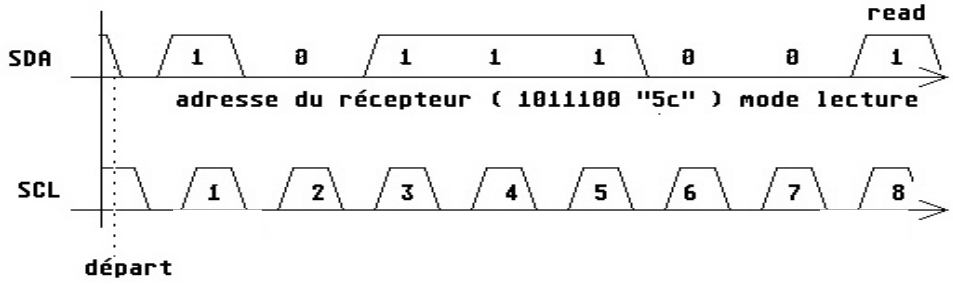
«1 » Condition de départ .

Un front descendant est appliqué sur la ligne SDA quand SCL est au niveau haut :



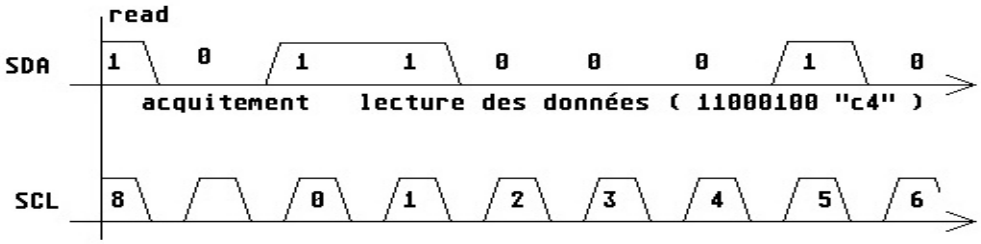
« 2 » Transmission de l'adresse (premier octet) .

A chaque niveau haut de SCL correspond un bit de donnée sur SDA (en commençant par les bits de poids fort). Ce premier octet contient l'adresse du circuit désiré (bit 7 … 1 ce qui autorise 128 possibilités) puis l'information de sens de transfert (bit 0, de niveau un en lecture et de niveau 0 en écriture) .



« 3 » Acquittement .

Après chaque octet reçu, le récepteur doit générer un signal d'acquittement. Pour cela, durant le niveau haut de SCL qui suit la transmission de l'octet, le récepteur (l'esclave dans ce cas) met la ligne SDA au niveau bas.



« 4 » Transmission du deuxième octet (lecture) .

Durant chaque niveau haut sur SCL, l'esclave (ici émetteur) envoie un des 8 bits de la donnée (en commençant toujours par le bit de poids fort).
Après une bonne réception, c'est le circuit maître qui doit générer le signal d'acquittement en maintenant la ligne SDA au niveau bas durant le niveau haut de SCL.
Dans le cas où le récepteur n'est pas en mesure de recevoir les données, il doit mettre la ligne SCL au niveau bas pour mettre l'émetteur en attente.

« 5 » Condition d'arrêt .

Un front montant est appliqué sur la ligne SDA quand SCL est au niveau haut.

Le maître veut écrire une information dans un circuit esclave.

Les étapes **1**, **2**, **3** et **5** sont identiques au cas précédent. Au niveau de l'étape 4 c'est le maître qui va émettre de la même façon qu'il transmet les adresses.

Le mode multi-maître.

L'horloge (SCL) résultant de ce mode, est le produit des différentes horloges générées par l'ensemble des maîtres. C'est ainsi qu'est générée l'horloge SCL synchronisée, dont la durée du niveau bas est déterminée par le dispositif ayant la période d'horloge au niveau bas la plus longue, et le niveau haut par le dispositif ayant la période d'horloge au niveau haut la plus courte.
En cas de conflit le maître prioritaire est celui qui envoie un niveau bas sur la ligne SDA.
Le maître qui tente d'envoyer un niveau haut sur SDA, et qui cependant ne peut y lire qu'un niveau bas, arrête son transfert et passe en mode récepteur esclave (au cas où ce soit à lui que l'on veuille s'adresser).

Conclusion

De part sa conception, ce système permet donc un dialogue extrêmement souple (mode multi-maître) et des réalisations très modulaires (seulement deux fils à connecter). Ce type de connexion présente aussi un avantage économique (réduction du prix des boîtiers et de la connectique associée). De plus, au niveau des études, il est facile de réutiliser des modules ou des sous-programmes déjà fabriqués.

En revanche, les priorités sont assez difficiles à gérer. Ainsi l'arrêt des maîtres gênants est seulement possible dans le cas où ils ont une adresse plus basse que celle du maître prioritaire. Parfois, cette difficulté entraîne l'utilisation d'une ligne d'interruption externe nécessitant un troisième fil, ainsi qu'une gestion matérielle et logicielle appropriée.

Evolution

La révision de la norme en 1992 autorise de nouveaux modes de fonctionnement :

- Vitesse de transfert portée de 100 à 400 kbits/seconde, voire 3,4Mbits/s en mode HV,
- Adressage des circuits étendu de 7 à 10 bits.