Carte caméra pour le robot
(Reconnaissance de balles)
- Rapport final -

Auteurs :

BENABEN Yannick (hardware / RGB to HSL) (benabeny@enseirb.fr)
DULUCQ Frédéric (I2C / VHDL) (dulucq@enseirb.fr)
LOCHON Frédéric (RGB to HSL / Asm PIC/…) (lochon@roulaise.net)

Intervenants :

LEGALL Jérôme (composants et matériels) (legall@enseirb.fr)
M. RODES (oscillateur quartz et découplage, don d'un Xilinx CoolRunner au club) (rodes@enseirb.fr)
SCHAEFER Ernest (conseils routage Orcad) (schaefer@enseirb.fr)
VILLIEN Christophe (conversion rgb2hsl) (villien@enseirb.fr)

Rev. 1.02 (27/05/2002)

Ce document a pour but d'expliquer le fonctionnement de la carte caméra fin qu'elle puisse être réutilisée lors des années prochaines.
Les informations contenues dans ce document peuvent être utiles pour la conception des autres parties du robot ou tout simplement par curiosité !

Ce document étant réalisé post-coupe 2002, une liste très utile de tout les bugs rencontrés a été rédigée.(***Notes post-coupe)***

Ces informations doivent rester à l'intérieur du club…

0. TABLE DES MATIERES

1. Principe de base de localisation des balles

Le principe de reconnaissance des balles est le suivant : on considère l'image vue par la caméra représentée ci-dessous :

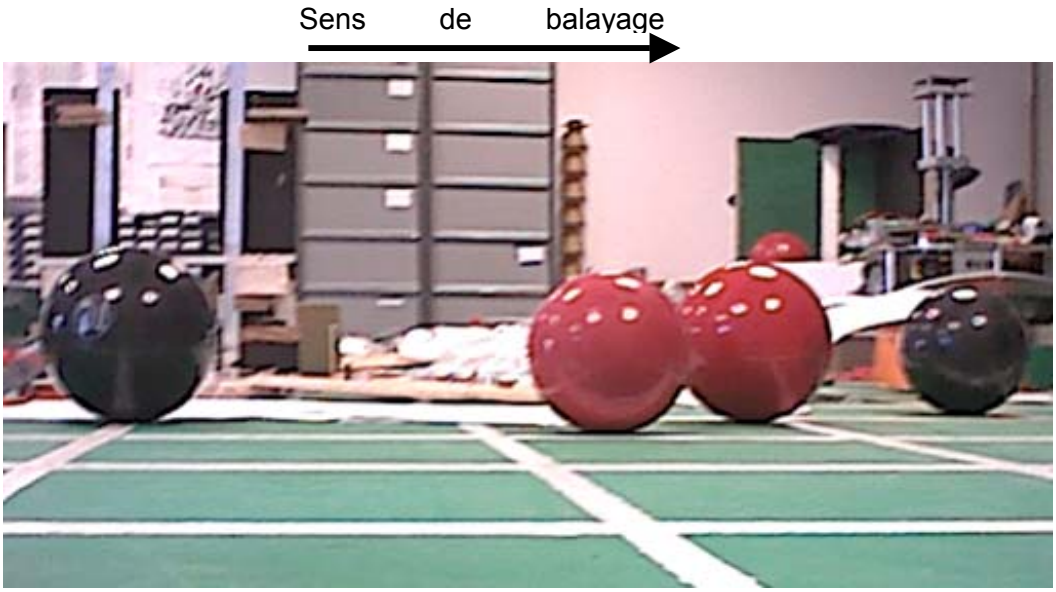


Figure 1 - image originale vue par le robot

Le balayage d'une image est faite de gauche à droite, du haut vers le bas, et ceci une ligne sur deux (trames paires et trames impaires).

Sur l'image ci-dessus, on ignore les lignes supérieures (public, décors) et quelques lignes inférieures (intérieur du robot/terrain) :

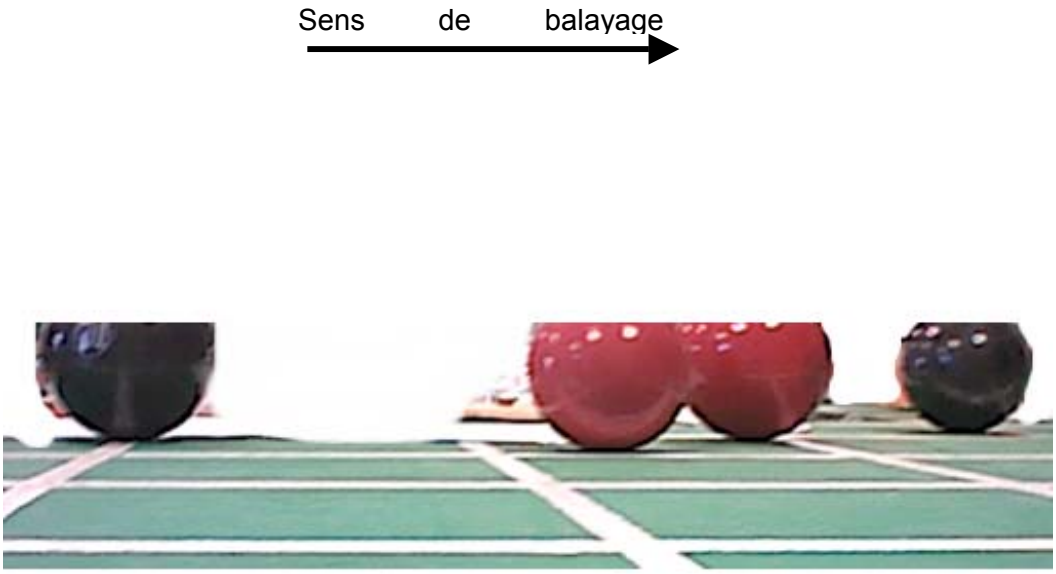


Figure 2 - image originale après exclusion des premières et dernières lignes.

Afin de simplifier le découpage de l'image et ainsi son analyse, on penche la caméra de **90°*** vers la droite et on la place à **ras du sol*** dans le plan des balles ce qui donne le résultat ci-dessous :

Figure 3 - rotation de la caméra de 90° pour découper "naturellement" l'image.

(exclusion des premiers et derniers pixels dans ce cas)

* note : les détails donnés en gras sont à prendre en compte par l'équipe qui s'occupe de la méca…

L'image va alors être découpée ligne par ligne vidéo et on va capturer le début et la fin de chaque balle (en n° de ligne) suivant le nombre de pixels détectés par ligne (exclusion des parasites) :

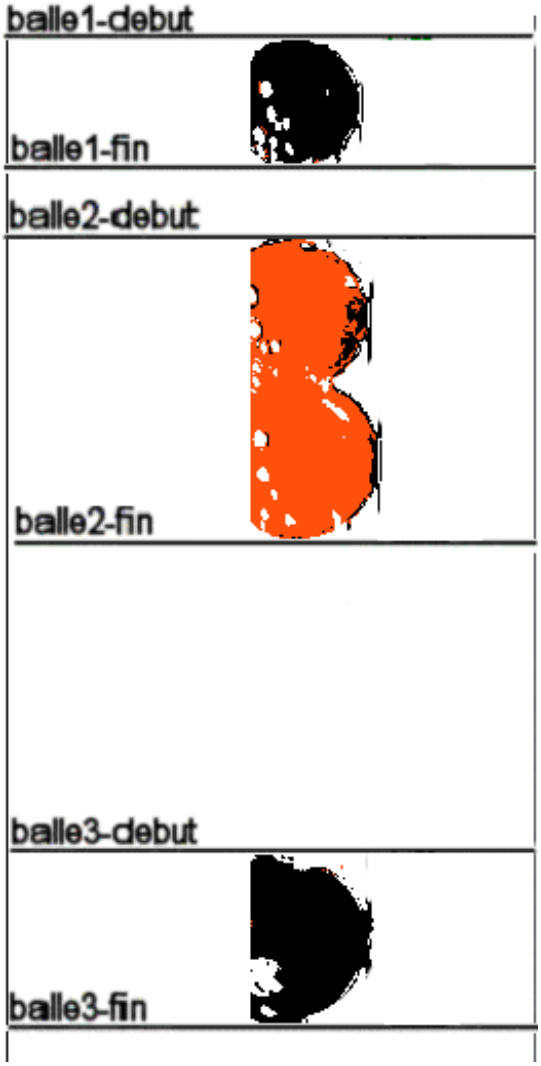
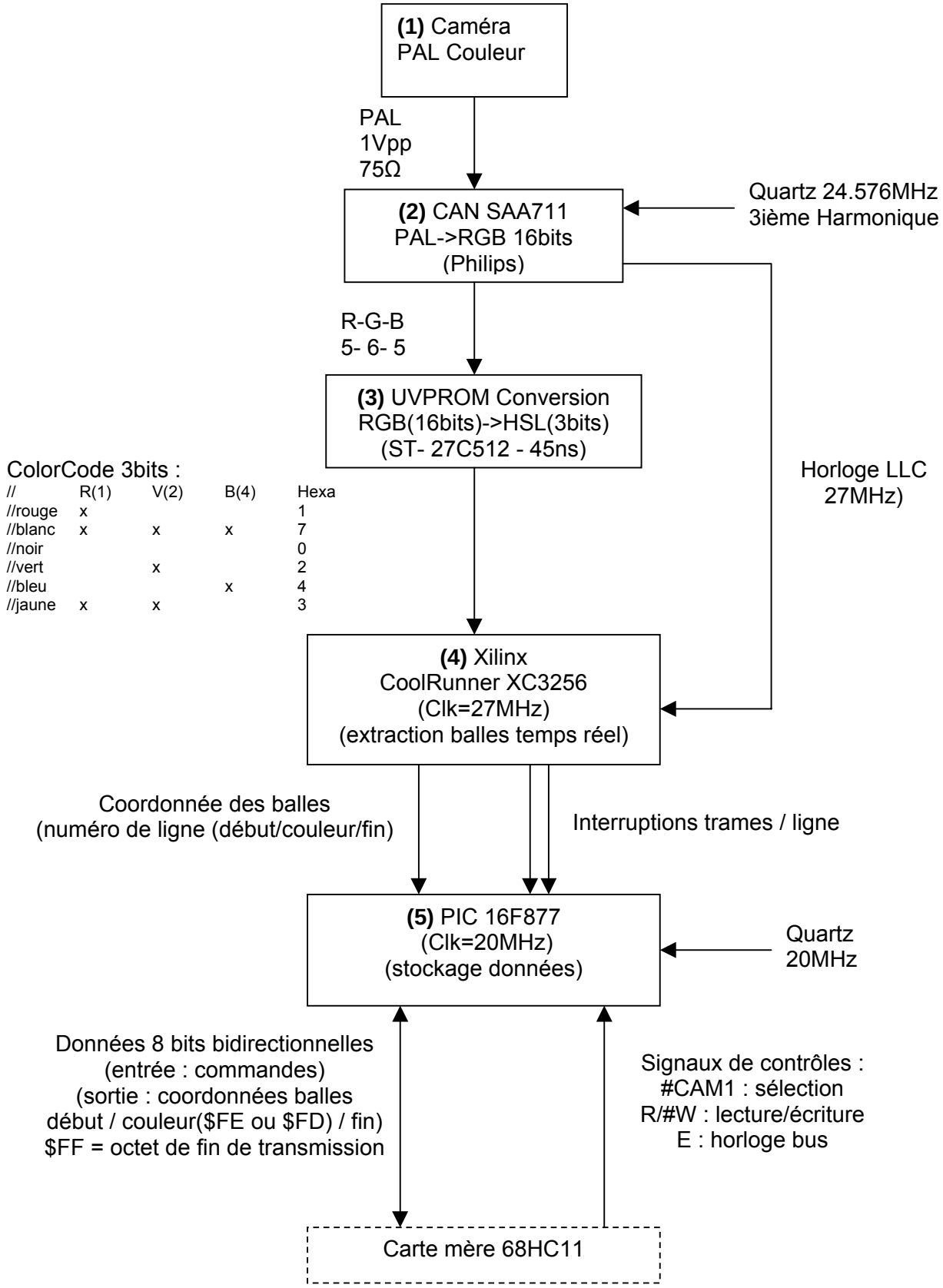


Figure 4 - capture des débuts et fins des balles de couleurs désirées.

Ce travail est effectué par un Xilinx CoolRunner (@27Mhz). La conversion de couleurs RGB 16bits en Teinte, Luminance, Saturation (HSL 3bits) est expliquée ci-dessous. Le Xilinx n'a alors à identifier que 2 codes de couleurs 3bits (rouge et noir, les autres couleurs seront codées par la couleur blanche).

2. Synoptique global

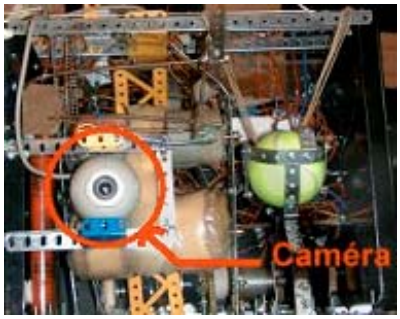


3. Partie (1) : Caméra

3.1 modèles utilisés sur le net

Le modèle de caméra n'est pas encore finalisé mais doit répondre au critères suivants :

- Caméra couleur PAL 50/60Hz (PAL imposé par le CAN)
- Dimension de 48mm x 48 mm x 48mm
- Faible consommation, bonne sensibilité (éviter le bruit du CCD)



Note post-coupe : *une prospection dans les stands et des discutions avec les autres équipes révèlent que les caméras utilisées sont : webcam USB + carte mère d'ordinateur AT PC/portable, webcam + carte mère PC104, caméra à sortie numérique 8/16bits + carte mère à base de ARM-7 (RISC@33MHz), des caméra usines à gaz (GCVP) (liaison série, inscrust temps réel des balles trouvées…), une webcam port // (4sec de transmission d'image), très peu de caméra PAL à cause du problème de numérisation.*

3.2 Modèle retenu

3.2.1 Le modèle retenu est le suivant (réf. du site Conrad) :

Modules caméras C-MOS formats miniatures 3
Code article 150026-62
La pièce : 83,69 (548,97 FF)

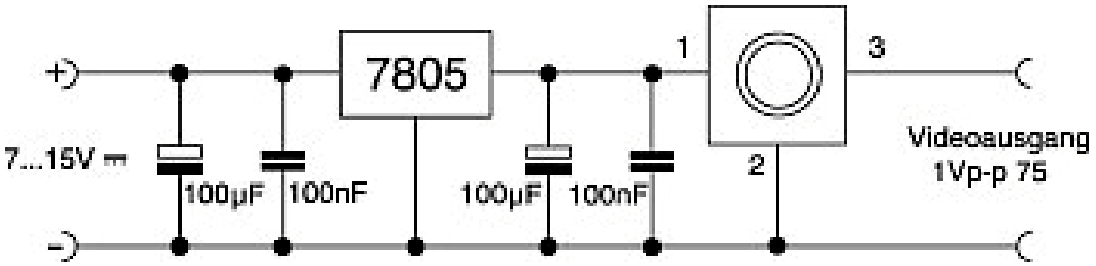
3.2.2 Caractéristiques techniques

Caméra couleur PAL complète haute résolution (628 V x 582 H = 365 496 pixels).
Dimensions réduites : 22 x 22 x 28 mm.
Faible consommation et très bonne qualité d'image.
Caractéristiques techniques : tension d'alimentation 5 VDC. Consommation 10 mA.
Lentille avec filtre IR : angle d'ouverture FOV 43° x 33°. Distance focale f = 6 mm. Diaphragme F = 1,6 mm.
Balance N/B automatique.

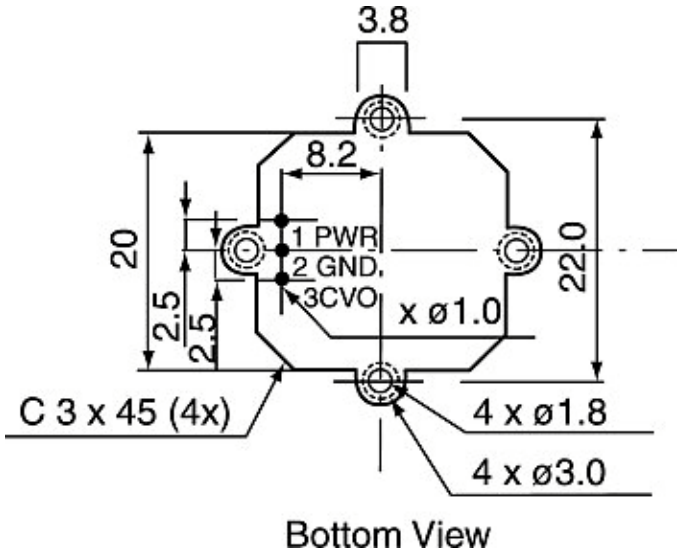
Données techniques :

Caméra	1/3" CMOS
Scanner	2:1 entrelacé
Obturateur	1/60 à 1/1,5000 sec
Sortie vidéo	1 Vcc vidéo composite (75 Ohms)
Résolution	PAL : 628 (H) x 582 (V) NTSC : 510 (H) x 492 (V)
Image	5,78 x 4,19 mm
Rapport S/B	> 48 dB (AGC on)
Distorsion	<0,03 % Vcc
Courant d'obscurité	< 0,2 nA/cm²
Gamme dynamique	> 72 dB
Alimentation	5 VDC +/- 0,5 V
Consommation	10 mA (sans charge)
Lentille	F = 6,0 mm F = 1,6 FOV 43° x 33°

3.2.3 Note d'application



3.2.4 Dimensions physiques et connexion



Adresses :
http://www.produktinfo.conrad.de/datenblaetter/175000-199999/190317-da-01-en-Farbkamera_Modul_3.pdf
http://www.produktinfo.conrad.de/datenblaetter/175000-199999/190317-an-02-fr-minicamera.pdf
http://www.produktinfo.conrad.de/datenblaetter/175000-199999/190317-an-01-fr-minicam,ra.pdf

Note post-coupe : *la caméra marche sans problème entre 4.3V et 5V, la teinte des couleurs change selon l'éclairage, surtout avec l'éclairage dans les stands (rouge=rose). Il ne faut surtout pas la souder (ou alors faire très attention), les pins sont très fragiles…*

4. Partie (2) : Conversion analogique-numérique

La partie conversion analogique numérique est la parti la plus problématique lors de la mise au point. En effet, la puce utilisée est un Video Input Processeur SAA711 de Philips et elle est assez difficile à faire marcher au début concernant la configuration des registres via I2C. Une fois le fonctionnement correct obtenu, la puce se révèle fiable et performante.

Le choix de cette puce a été simplement réalisé par le fait que le processeur vidéo a été généreusement donné par une entreprise. La parti hardware est la note d'application.

Les problème rencontrés sont l'oscillation du quartz qui doit être extrêmement précise pour assurer le fonctionnement de l'étage chrominance (PLL). Un autre problème a été la configuration des registres.

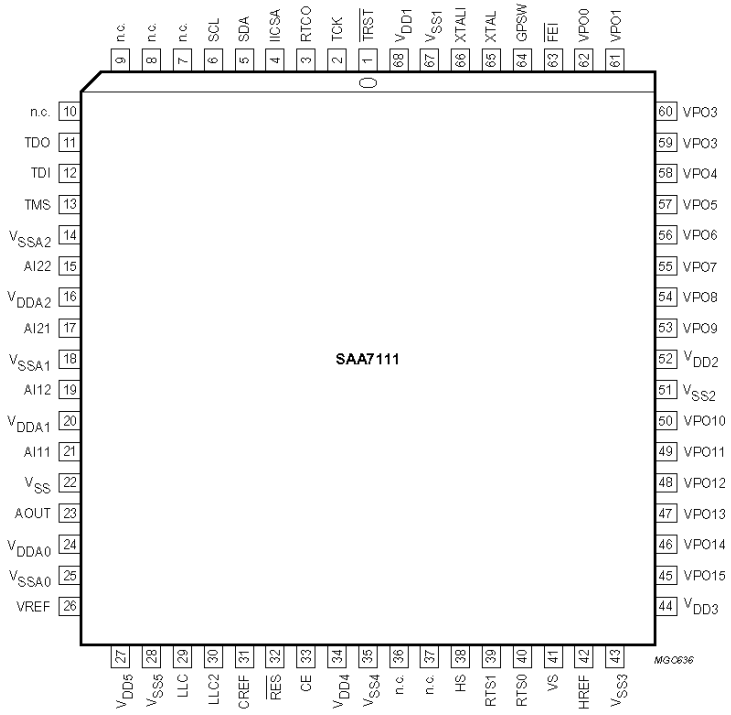
Le quartz utilisé est un quartz de 24.576MHz 3ieme harmonique. Cette valeur est très difficile à trouver (le quartz a été donné également par une entreprise). Le fonctionnement sur son 3ième harmonique est réalisé par le circuit LC attaché au quartz. Il faut prendre garde a choisir une self cms qui supporte le 24.576MHz ainsi que des capacités céramique NP0 (coefficient en température nul). Des caractéristiques du quartz sont données en annexe. Elles ont été mesurées grâce à l'impédancemètre de l'enseirb. Nous pouvons remercier particulièrement M. Rodes pour nous avoir aider à corriger un quartz 24.576MHz fondamental que nous avons utiliser lors des premiers essais.

4.1 Schéma utilisé pour le SAA711

Le schéma pour le SAA711 utilisé est donné en annexe. Il provient essentiellement de la note d'application de Philips.

Les capacités de découplage sont de **10nF** et **100nF** pour TOUT les circuits intégrés vue que les signaux qui transitent sur la carte sont de 27MHZ, 24.576MHz, 13.5Mhz. Les capacités de découplage polarisés sont des tantales et une capacité de 2.2uF doit être placée pour 5 boîtiers (d'après Bob Pease – ingénieur américain gourou électronicien).

Le brochage du SAA711 monté sur la carte finale est le suivant (PLCC68) :



Note : L'oscillation du quartz doit être immédiate si le quartz utilisé est bien un 3ième harmonique. Dans le cas d'un quartz fondamental, une correction doit être faite car le quartz va osciller en charge et ne sera donc pas à la bonne fréquence. **Mr. Rodes** (professeur en filière électronique) nous a donc indiqué comment corrigé ce problème lors des tests avec notre quartz fondamental en insérant en série une self L+ calculée selon la formule suivante :

$$Fprat=24.57945MHz(CchargePrat=56pF)$$

$$\Delta F=Fprat-Fvoulu$$

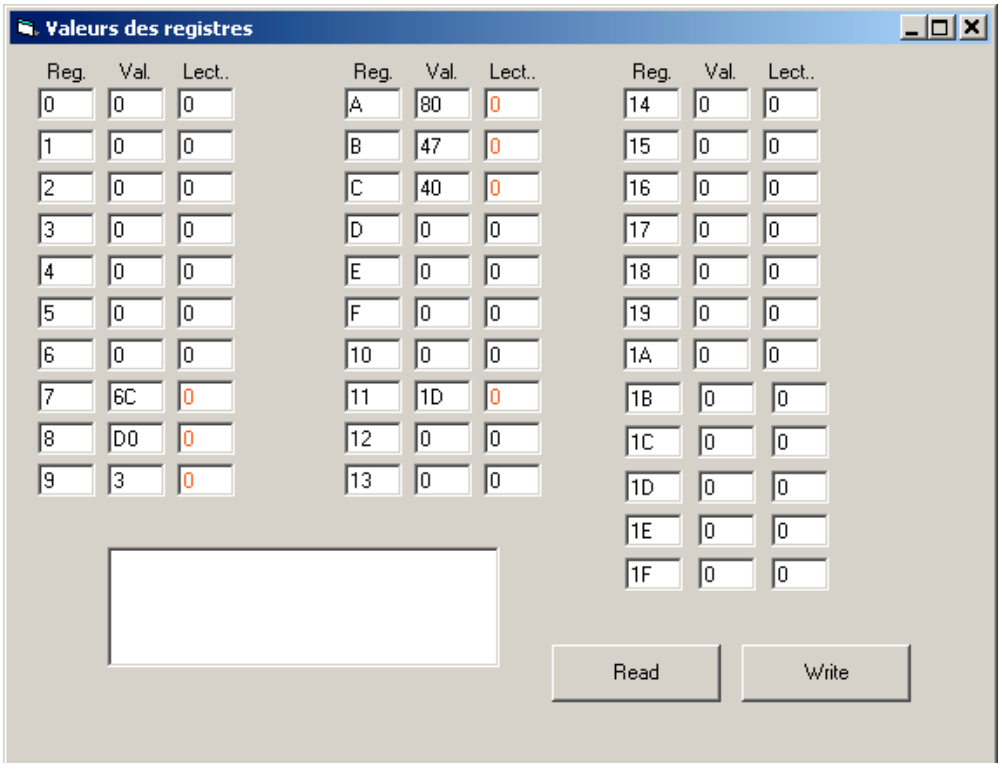
$$F=Fvoulu-\Delta=\frac{1}{2\pi\sqrt{L.C)}}$$

avec L=3.7mH+L+ et C=11.33pF
d'où L+=1.0279uH
(voir détails sur feuilles de diagramme du quartz)

4.2 Configuration des registres : bus I2C

La partie la plus longue a été la configuration des registres par le bus I2C. On a utilisé un montage venant de Supelec Rennes (remerciement à J. Weiss de Supelec Rennes) générant de l'I2C sur port parallèle. Nous avons modifié le soft pour notre puce SAA711 est pour Windows NT (2000). Les sources Visual Basic sont disponibles sur le CD-ROM « carte caméra ».

La page de configuration est la suivante :



Le SAA7111 est configuré pour fonctionner à l'adresse 0x48, caméra branchée sur AI11. **Les valeurs a mettre dans le registre sont données dans la colonne de droite en hexadécimal**.

SLAVE ADDRESS		READ		WRITE		IICSA				CONFIG
		49H and 4BH		48H and 4AH		0 and 1				
REGISTER FUNCTION	SUB-ADDR.	D7	D6	D5	D4	D3	D2	D1	D0	
Chip version	00	ID07	ID06	ID05	ID04	ID03	ID02	ID01	ID00	-
Reserved	01	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	-
Analog input control 1	02	FUSE1	FUSE0	GUDL2	GUDL1	GUDL0	MODE2	MODE1	MODE0	0x00
Analog input control 2	03	(1)	HLNRS	VBSL	WPOFF	HOLDG	GAFIX	GAI28	GAI18	0x00
Analog input control 3	04	GAI17	GAI16	GAI15	GAI14	GAI13	GAI12	GAI11	GAI10	0x00
Analog input control 4	05	GAI27	GAI26	GAI25	GAI24	GAI23	GAI22	GAI21	GAI20	0x00
Horizontal sync start	06	HSB7	HSB6	HSB5	HSB4	HSB3	HSB2	HSB1	HSB0	0x6C
Horizontal sync stop	07	HSS7	HSS6	HSS5	HSS4	HSS3	HSS2	HSS1	HSS0	0x6C
Sync control	08	AUFD	FSEL	EXFIL	(1)	VTRC	HPLL	VNOI1	VNOI0	0x40
Luminance control	09	BYPS	PREF	BPSS1	BPSS0	VBLB	UPTCV	APER1	APER0	0x03
Luminance brightness	0A	BRIG7	BRIG6	BRIG5	BRIG4	BRIG3	BRIG2	BRIG1	BRIG0	0x80
Luminance contrast	0B	CONT7	CONT6	CONT5	CONT4	CONT3	CONT2	CONT1	CONT0	0x47
Chroma saturation	0C	SATN7	SATN6	SATN5	SATN4	SATN3	SATN2	SATN1	SATN0	0x40
Chroma Hue control	0D	HUEC7	HUEC6	HUEC5	HUEC4	HUEC3	HUEC2	HUEC1	HUEC0	0x00
Chroma control	0E	CDTO	CM99	CSTD1	CSTD0	DCCF	FCTC	CHBW1	CHBW0	0x00
Reserved	0F	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	-
Format/delay control	10	OFTS1	OFTS0	HDEL1	HDEL0	VRLN	YDEL2	YDEL1	YDEL0	0x00
Output control 1	11	GPSW	(1)	FECO	COMPO	OEYC	OEHV	VIPB	COLO	0x1C
Output control 2	12	RTSE1	RTSE0	(1)	CBR	RGB888	DIT	AOSL1	AOSL0	0x00
Reserved	13-19	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	-
Text slicer status	1A	(1)	(1)	(1)	(1)	F2VAL	F2RDY	F1VAL	F1RDY	-
Decoded bytes of the text slicer	1B	P1	BYTE16	BYTE15	BYTE14	BYTE13	BYTE12	BYTE11	BYTE10	-
	1C	P2	BYTE26	BYTE25	BYTE24	BYTE23	BYTE22	BYTE21	BYTE20	-
Reserved	1D-1E	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	-
Status byte	1F	STTC	HLCK	FIDT	GLIMT	GLIMB	WIPA	SLTCA	CODE	-

Sur la carte finale, c'est le microcontrôleur PIC qui gère l'I2C. Il permet donc l'initialisation du SAA via le bus I2C.

Note : la carte finale a montré un défaut lors du reset de la carte en cours de fonctionnement. Une erreur sur le bus I2C est provoquée par le PIC lors du Reset, une double initialisation est alors effectuée.

Note post-coupe : Le SAA711 étant obsolète, il faudra changer de modèle (SAA7114 Philips ou BTxxx de Conexant)

5. Partie (3) : conversion RGB vers HSL

5.1 traitement logiciel préalable

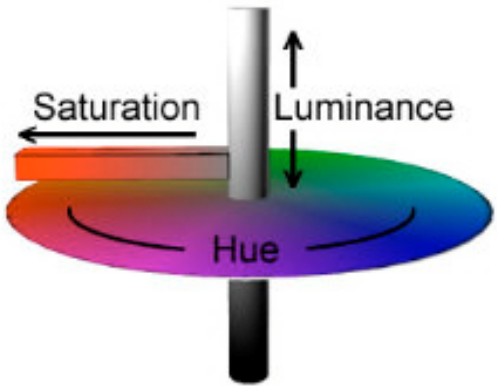
Le but de la conversion de l'espace des couleurs Rouge Vert Bleu vers Teinte (hue), Saturation, Luminosité est de permettre un seuillage des couleurs sur seulement un seul paramètre (Hue ici). Le seuillage ne doit pas prendre en compte la luminosité et doit rester relativement large sur la saturation des couleurs (très dépendante de l'éclairage ambiant).

Finalement, un point essentiel, l'espace de couleur HSL est l'un des seuls espaces a être le plus naturel et le plus proche des phénomènes physiques (espace utilisé dans les logiciels de graphisme)

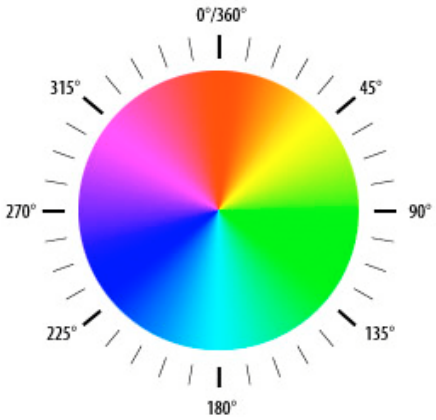
La conversion Rouge, Vert, Bleu vers Hue (teinte), Saturation, Lightness (luminosité) est faite pas une UVPROM ayant un temps d'accès de 45ns (13.5MHz de débit binaire correspond à 75ns environ). Le logiciel de réglage des seuils et de « simulation » a été écrit en Visual C (MS Visual Studio 6) . Les sources sont disponibles sur le CDROM. (Visual Studio est installé à l'ENSEIRB sur des PCs sous Windows NT).

5.1.1 Modèle de couleur en Hue/Saturation/Luminance

Espace HSL

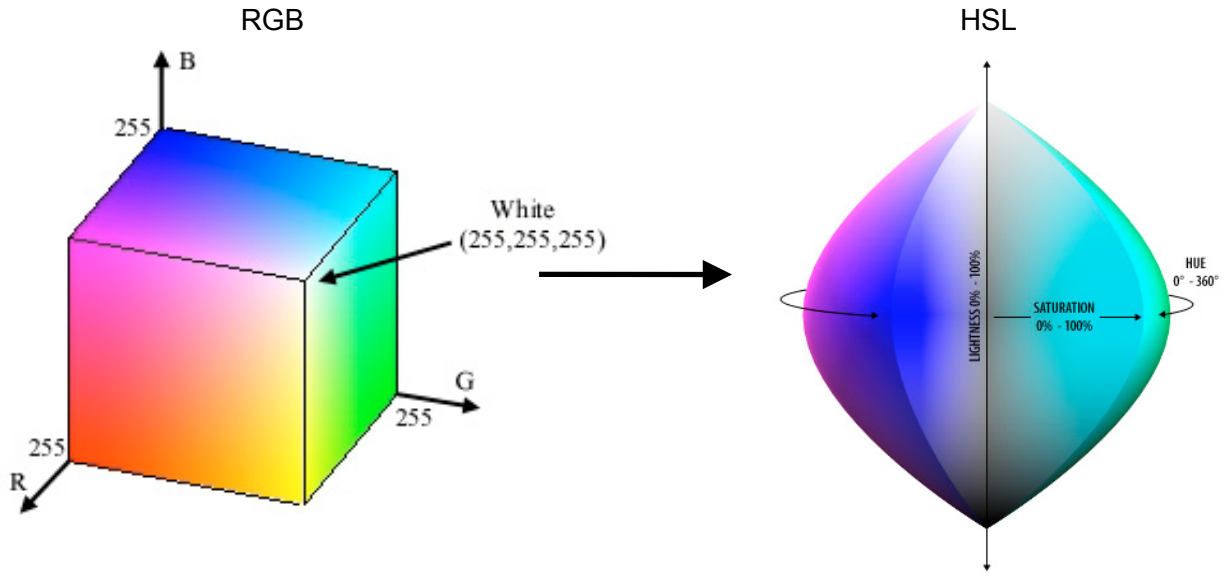


Teinte (Hue)=[0..360°]



5.1.1.1 Transformation effectuée par l'UVPROM :

Au niveau du principe de la représentation des couleurs, on passe de la représentation RVG peut commode pour isoler une couleur à la représentation HSL qui fait varier 1 seul paramètre pour la teinte. Le seuillage de la couleur rouge doit alors prendre en compte que la couleur rouge est la transition 0°/360°. De plus, l'éclairage modifiant la teinte de la balle vue par la caméra, il faut prévoir une saturation forte (rouge prononcé) ou faible (rose) du rouge.



notes :

L'algorithme employé vient de l'adresse : http://blas.cis.mcmaster.ca/~monger/hsl-rgb.html
(voir annexe pour le détail de l'algorithme)

voir aussi :

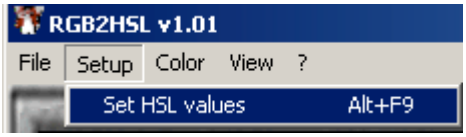
http://www.adobe.com/support/techguides/color/colormodels/hsb.html
http://www.adobe.com/support/techguides/color/colormodels/rgbcmy.html
http://www.nebulus.org/tutorials/2d/photoshop/color/

5.2 Configuration des seuils des couleurs

La conversion Rouge-Vert-Bleu vers Hue(teinte), S(saturation) et L(luminosité) est faite par une UVPROM (pas une Flash car les temps d'accès sont trop grands). Une image du terrain doit alors être préalablement numérisée (**avec la caméra du robot**) pour définir les couleurs.

Le fichier de programmation de l'UVprom est réalisé depuis l'ordinateur grâce à un logiciel qui permet de choisir les couleurs et les seuils optimaux pour trouver les balles rouges et noires tout en excluant le vert, le blanc du terrain et les autres couleurs du robot adverse.

La capture ci-dessous montre la fenêtre de configuration des seuils.



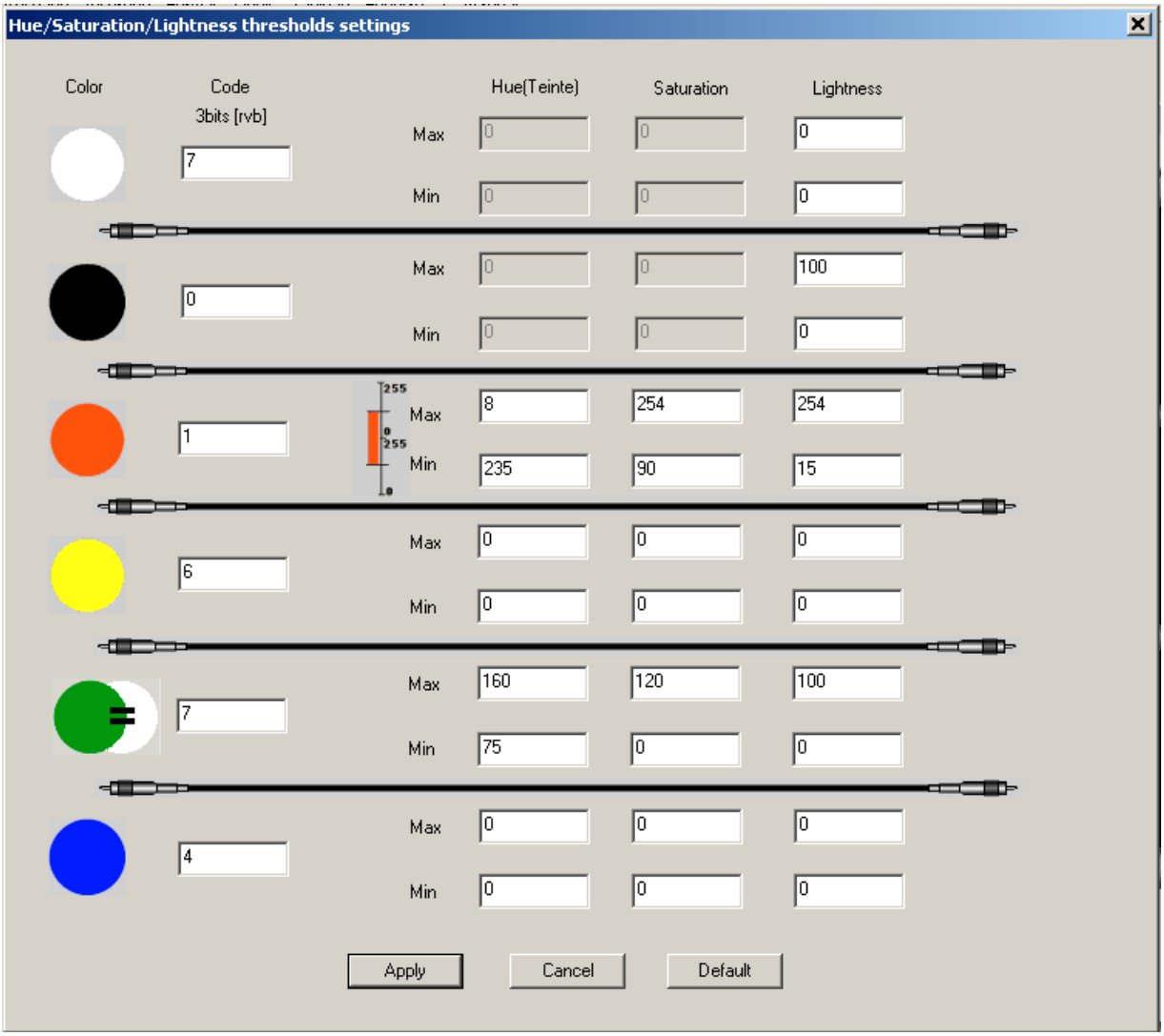


Figure 5 - configuration des seuils

La capture ci-dessus est la configuration utilisée lors des essais au club. La configuration utilisée au stand et sur les terrains officiels est plus tolérante sur le rouge.

Note : Le vert est réglé afin de détecter le vert foncé. Le code attribué est alors celui du blanc. Cette astuce permet de mieux détecter les balles noires en excluant ainsi les bords foncés du terrain. Les priorités des couleurs sont Rouge, vert foncé (traduit en blanc) et noir. Les autres couleurs sont inutilisées et sont traduites en blanc.

5.3 Visualisation des composantes HSL et résultat du seuillage

Une fois les seuils choisis correctement (le plus indépendamment possible de la luminosité), le logiciel affiche le résultat du seuillage dans le cadre inférieur gauche ainsi que les différentes valeurs des composantes Hue, Sat et Light .

Remarque : un bug situé au niveau de la saturation n'a pas été résolus de manière « propre », il s'agit d'une « saturation » de cette composante entraînant un passage de valeurs proches de 255 à 0. Afin de limiter ceci, le coefficient multiplicateur de la saturation a été réduit.

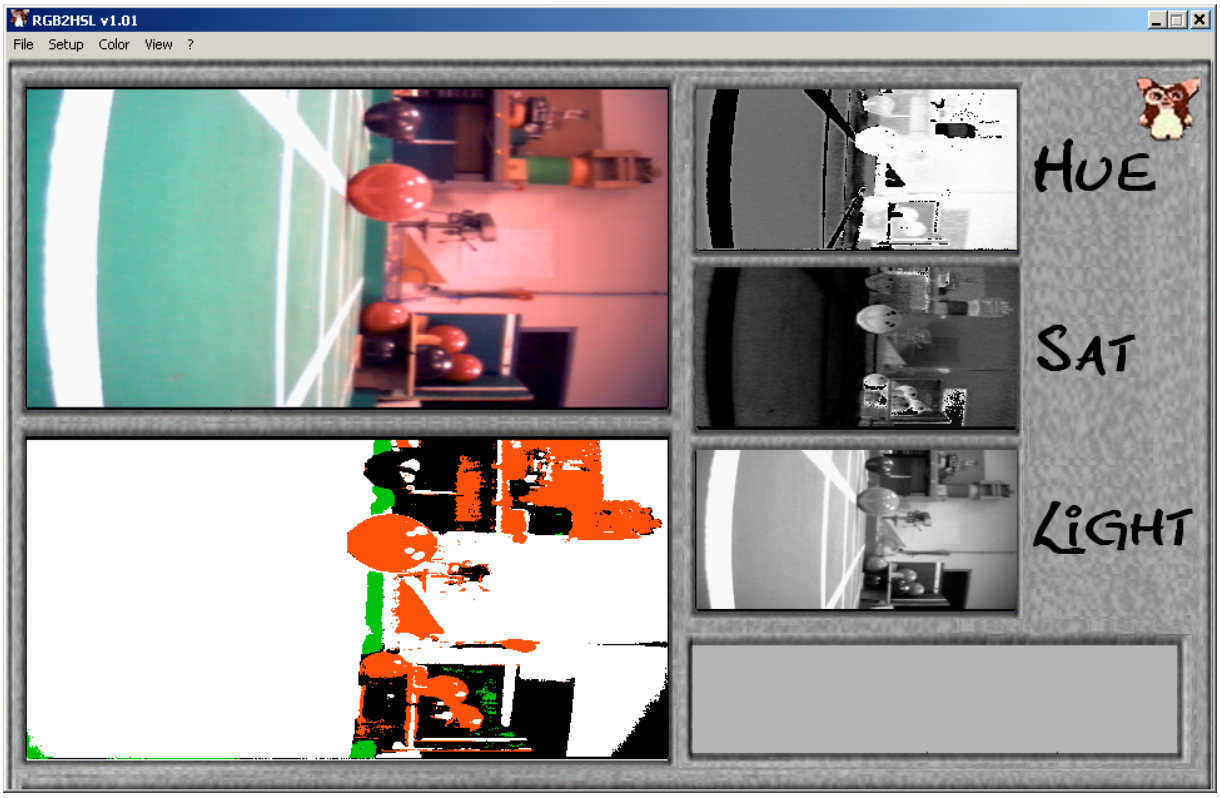
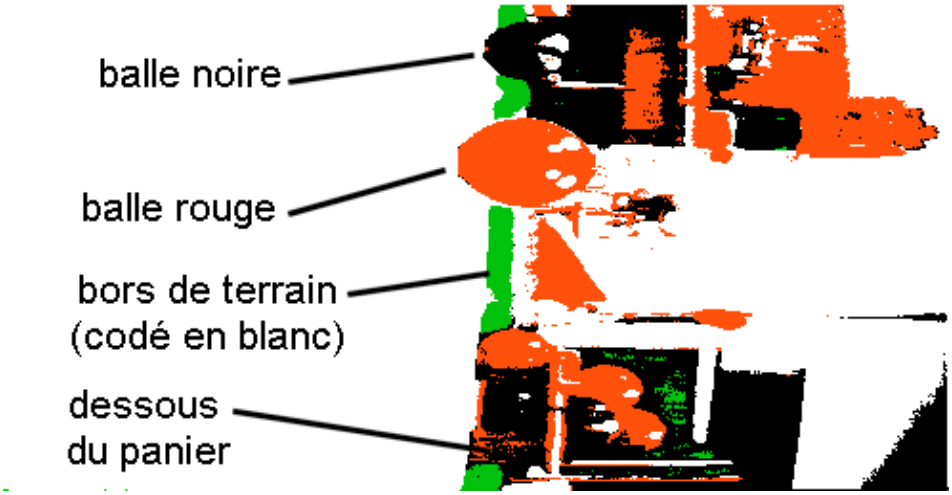


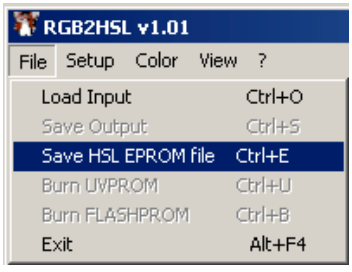
Figure 6 - Résultat du seuillage

Si on analyse le seuillage trouvé on peut voir que seul le bas de l'image (parti de gauche ici) est utilisable, l'analyse de l'image sera alors faite à l'intérieur d'un cadre par le Xilinx :



Après cadrage au prés dans le Xilinx, on a un résultat qui s'approche de l'image suivante en considérant toujours le vert codé en blanc dans l'UVPROM :

Une fois les seuils finaux choisis, on génère le fichier binaire qui va être programmé dans l'UVPROM :



Le codage utilisé est le suivant : ColorCode 3bits :

//	R(1)	V(2)	B(4)	Hexa
//rouge	x			1
//blanc	x	x	x	7
//noir				0
//vert	x	x	x	7
//bleu			x	4
//jaune	x	x		3

Un fichier binaire (.bin) est généré, sa taille est de 64Ko (octets écrits simplement les uns après les autres), il sera programmé dans une **UV**PROM… d'au moins 512Kbits dont le temps d'accès sera très inférieur à 74ns (74ns=1/13.5Mhz, 13.5MHz étant le fréquence du flux binaire RGB en sortie du convertisseur analogique numérique).

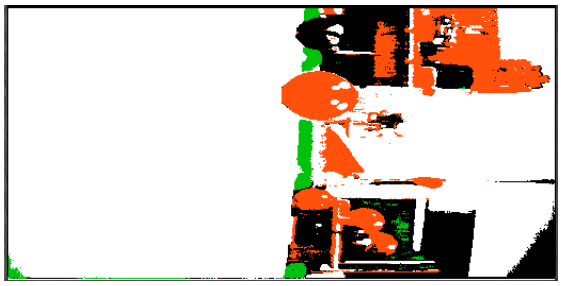
Le modèle retenu pour le prototype est une UVPROM ST-Microelectronics 27C512-45 (45ns de temps d'accès). Le programmateur est ppdos de l'école (TPs microcontrôleurs - numériques) (utiliser « option/fichiers binaire/ouvrir »).

La numérisation et la conversion RGB->HSL sont faites en **temps-réel** (soit 25img/s), la partie traitement par le Xilinx peut introduire un léger retard (de l'ordre de la centaine de µS ?). Les résultats sont alors traités et stockés dans un microcontrôleur PIC.

5.4 Quelques exemples réels :

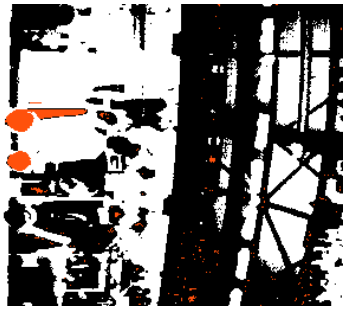
5.4.1 Captures au club robot

Images RGB 24bits | **Images 3 bits après seuillage en HSL**

5.4.2 Exemples de captures prises à la coupe







6. Partie (5) : Xilinx CoolRunner

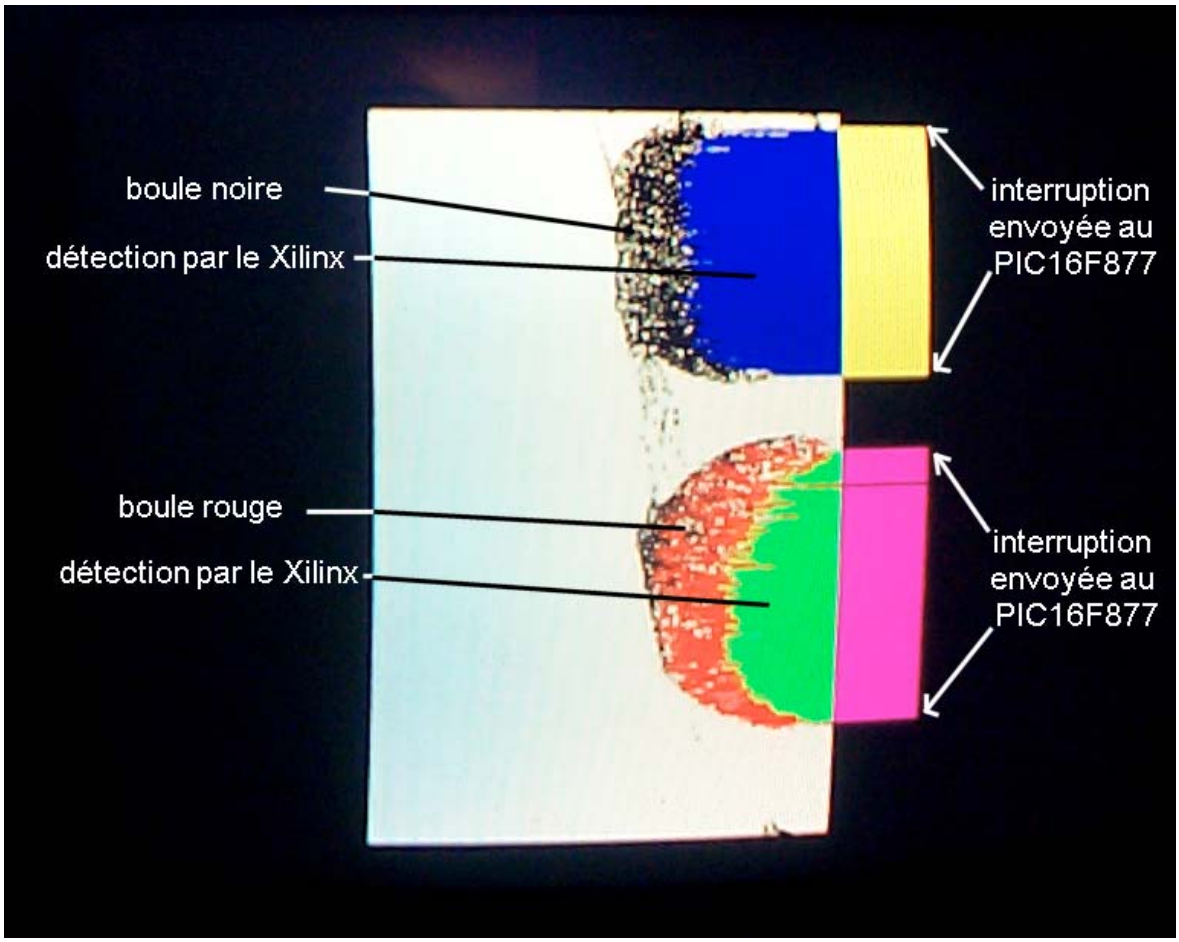
6.1 Principe

Le Xilinx a été programmé en VHDL grâce au Xilinx WebPack disponible gratuit à l'adresse http://www.xilinx.com/support/software.htm (exige une petite registration gratuite).

Il reçoit les 3 bits de code de couleurs et va compter pour chaque ligne le nombre de pixels de couleur rouge ou noir qu'il trouve. Le seuil de détection est réglable, le Xilinx CoolRunner place un cadre noir hors du quel il n'analyse pas l'image. A l'intérieur de ce cadre, le Xilinx va recopier les codes d'entrée vers la sortie. Si le comptage des couleurs est satisfaisant, il passe le reste de la ligne en une couleur arbitraire choisi par le concepteur. Une bande de couleur est également placée sur la droite de l'image afin de visualiser les moments où le Xilinx envoi des interruptions au PIC.

Les interruptions sont au nombre de 2 : une interruption trame qui indique au PIC q'une nouvelle image est en cours d'analyse, l'autre interruption n'apparaît que lorsqu'on a un passage d'une ligne neutre a une ligne contenant une couleur détectée (début balle) et lorsqu'on a un passage d'une balle à une ligne neutre (fin de balle). Le Xilinx envoi alors une Interruption, le PIC capture le numéro de ligne exprimé sur 8bits, ainsi qu'un bit correspondant à l a couleur détectée. A la deuxième interruption, le Xilinx envoi la ligne correspondant à la fin de la balle.

L'image ci-dessous illustre très bien ces explications. L'énorme avantage du Xilinx est qu'on peut avoir un debug en temps réel du code VHDL en visualisant simplement la TV.



(photo d'un écran télé)

6.2 Process VHDL

Le code VHDL est écris dans le fichier main.vhd. Le code comprend différents process synchronisés sur l'horlohe tck0 à 27MHz générée par le SAA7111. Le fluc binaire à 13.5MHz étant synchronisé par rapport à LLC = 27MHz, le Xilinx peut alors lire correctement les différents bits de couleur pour chaque pixel.

Le code décrit un composant ball_finder composé de plusieurs process :

- Un process est dédié au comptage horizontal (pixel) et un autre vertical (ligne) ce qui en renversant la caméra se traduit plutôt en comptage horizontal pour les lignes et vertical pour les pixels.
- Un process permet compter et générer les différentes couleurs sur la télé. Finalement, un process génère les interruptions et l'envoi des données vers le PIC.
- Un process supplémentaire permet de faire clignoter une led avec une période de 1 seconde si l'horloge en entrée est de 27MHz (mini fonction debug).

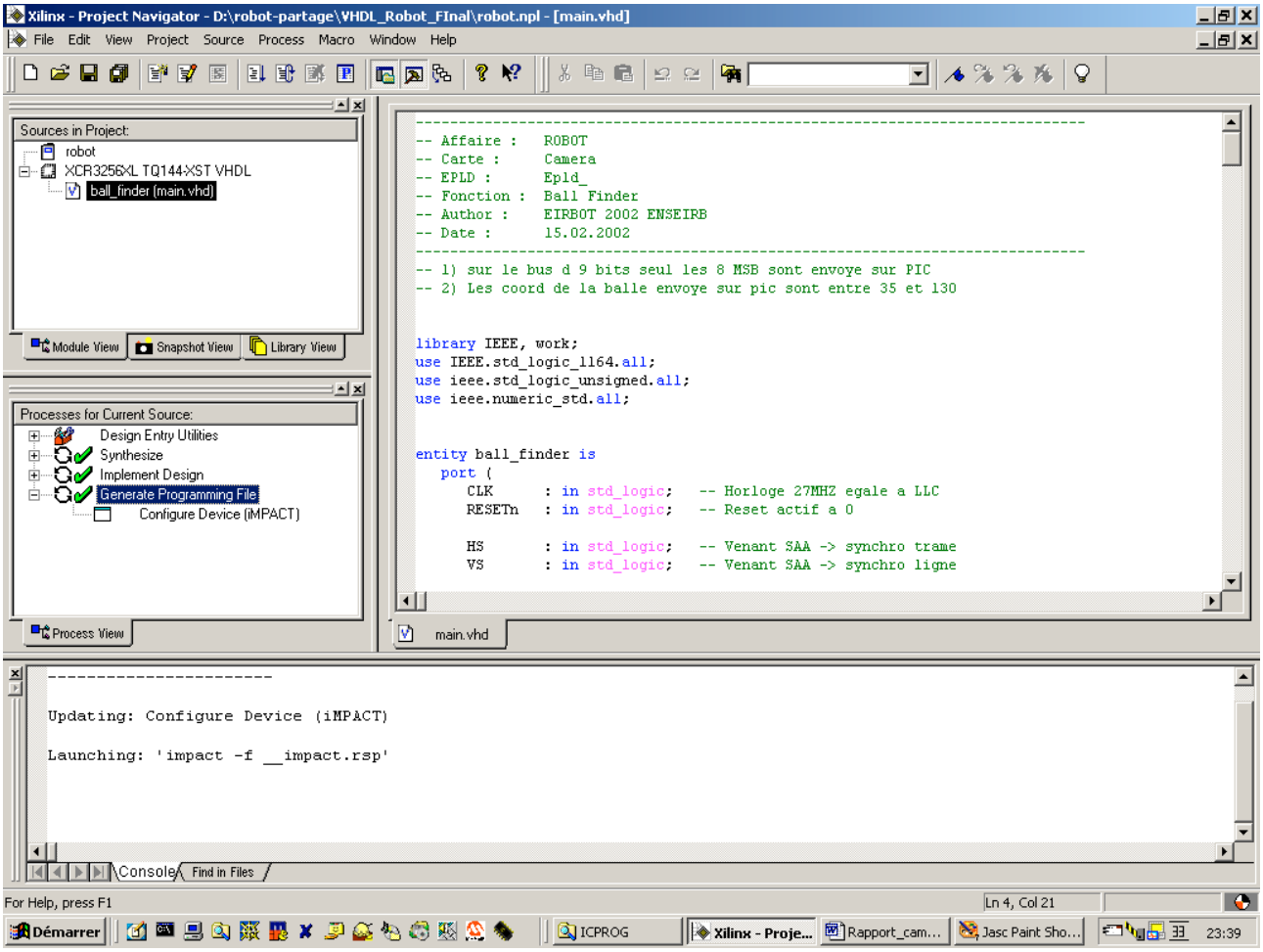
Le code source est donné en annexe ou sur le CDROM associé à ce document dans \VHDL_Robot_Final\main.vhd

Le brochage est donné dans le fichier ball_finder.ucf et configurable depuis le WebPack (design -> User constrainst -> Assign pins).

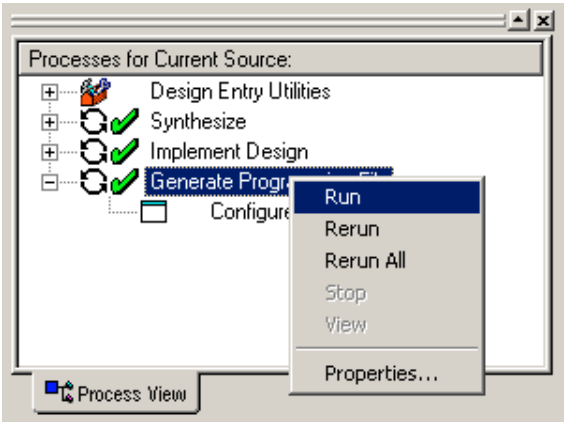
6.2.1 Développement rapide du VHDL

Ce paragraphe explique très brièvement les étapes pour écrire et synthétiser le VHDL avec le Xilinx CPLD Webpack 4.2.

6.2.1.1 Fichier/ouvrir/robot.npl

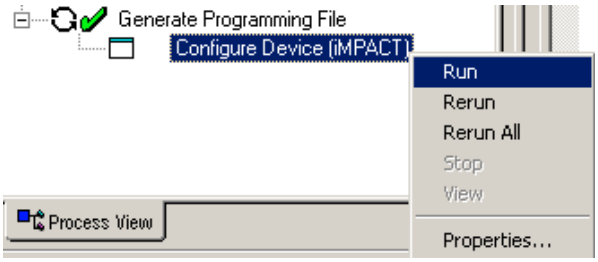


6.2.1.2 Synthétiser/run(rerun all)

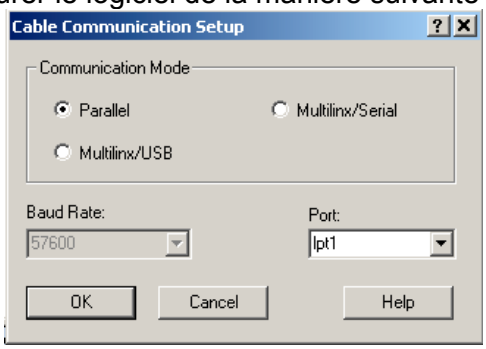


6.2.1.3 Téléchargement dans le Xilinx : brancher le câble JTAG sur le port parallèle

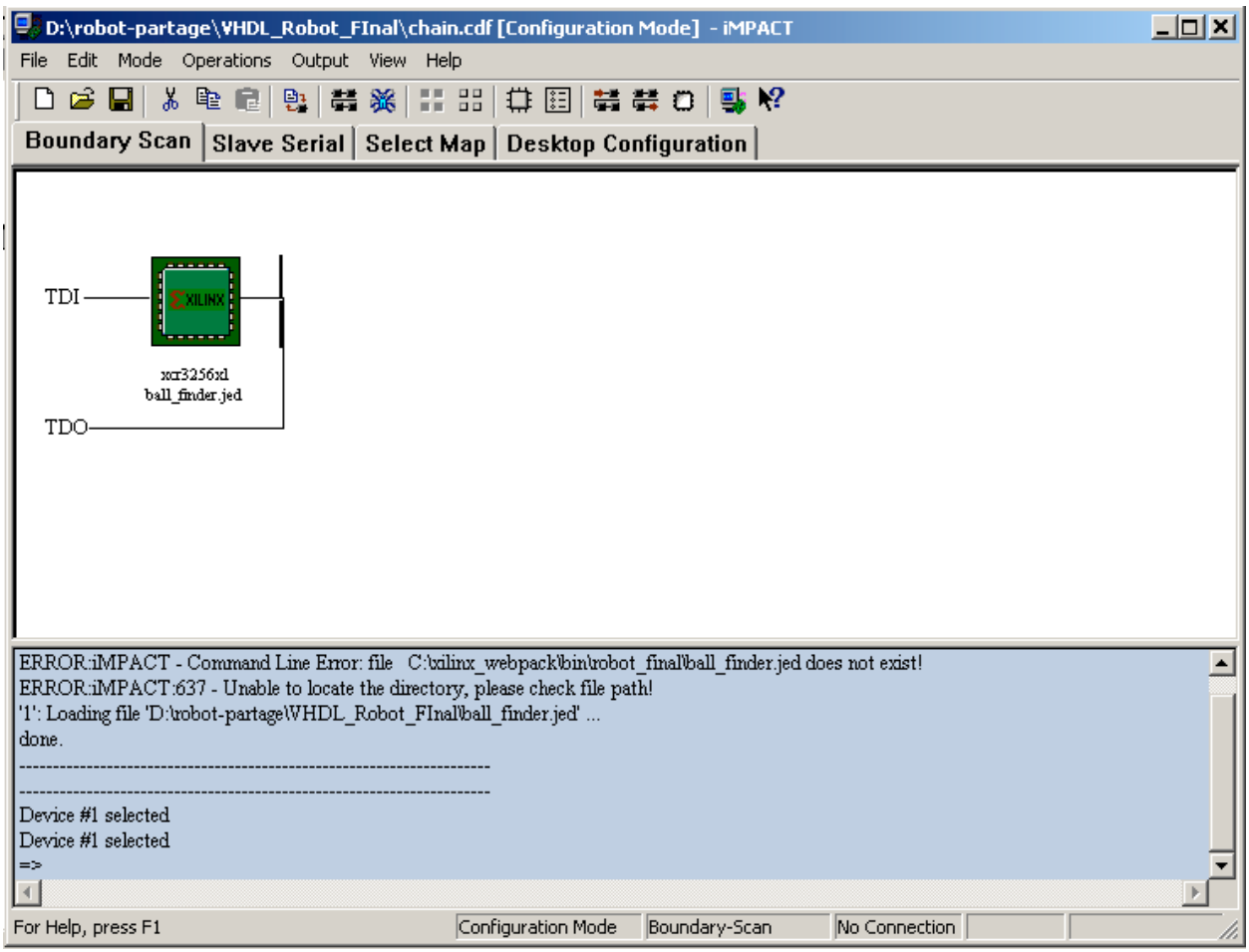
Cliquer run sur 'Configure Device'



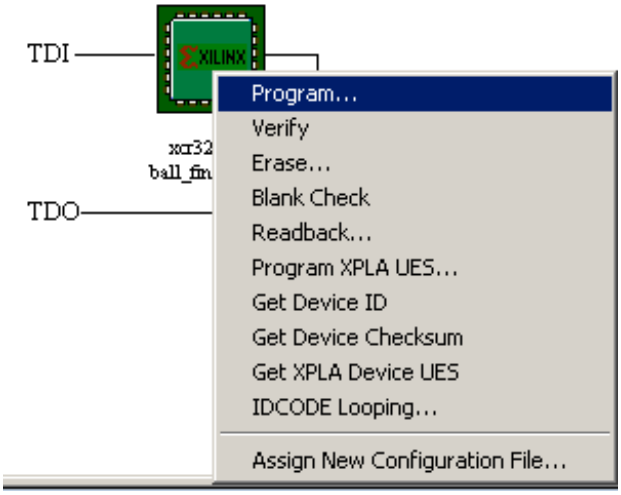
Choisir le bon fichier .cdf et configurer le logiciel de la manière suivante :



On doit arriver à la fenêtre suivante :



Ensuite faire :



That's all folks !

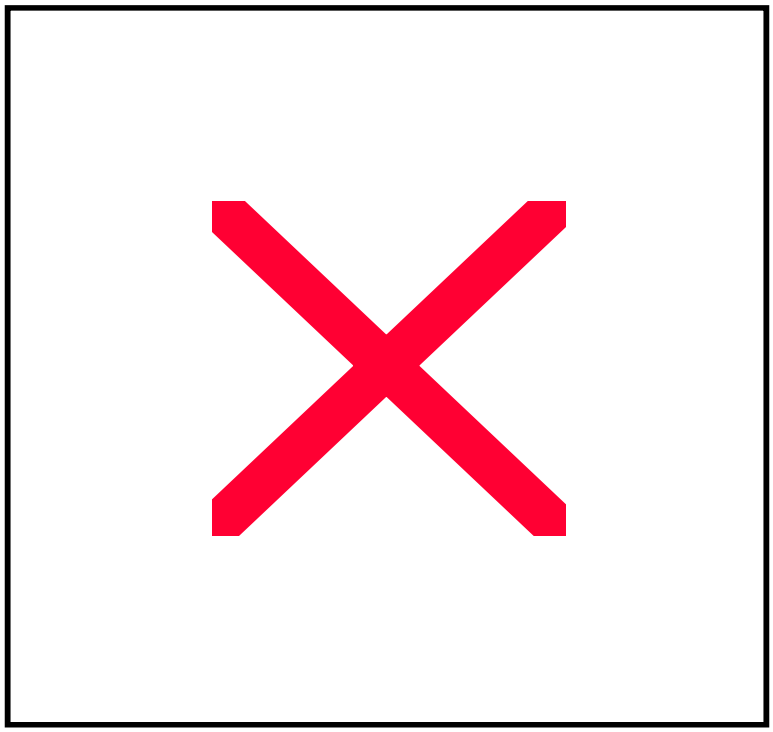
Note : *évidement, vous rencontrerez quelques difficultés mais cela ne serait pas marrant sinon…*

7. Interfaçage avec la carte mère

7.1 Microcontrôleur à bord

7.1.1 Principe

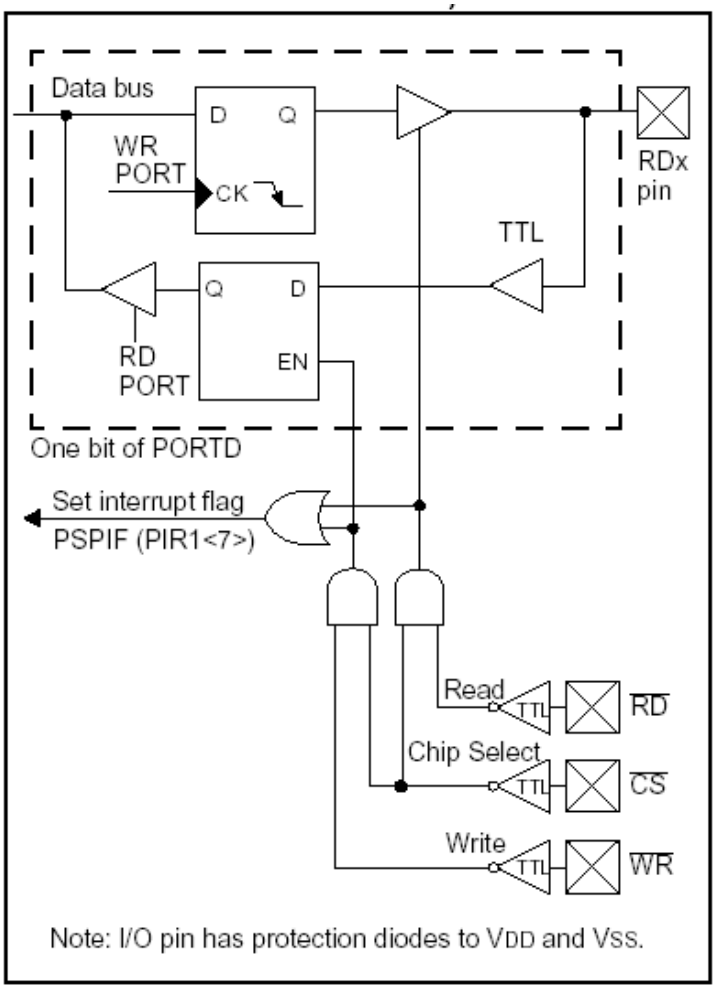
Les informations venant du Xilinx CoolRunner sont stockées dans des pages de mémoires d'un microcontrôleur PIC de la famille PIC16F877. Le brochage du Xilinx utilisé est donné ci-dessous. Les sources sont disponibles sur le CDROM.



Le programme permet en temps normal de stocker les coordonnées des balles dans sa mémoire RAM. Pour cela il utilise 2 pages (banks) mémoire et il alterne les banques pour chaque image reçu du Xilinx (IT trame synchronise le PIC). Lorsque le 68hc11 veut faire une lecture d'une page mémoire, il envoi une commande de lecture au PIC. Le PIC stoppe son rapatriement des balles venant du Xilinx et charge le registre d'émission de la liaison parallèle esclave (il inhibe aussi les interruptions venant du Xilinx entre autres…). Le fonctionnement de l'interface parallèle esclave est donné ci dessous.

7.1.2 Interface

Les exigences côté carte mère seront simplement résolues en stockant alternativement les images dans 2 pages séparées. Lors de la lecture des balles par la carte mère (68hc11F1 cadencé à 8-16MHz), le PIC interrompt sa mémorisation en cours et va envoyer les données de l'autre page (donc de l'image précédente) au 68hc11f1 en utilisant l'interface « Parallel Slave Port » :



Le brochage utilisé est le suivant :

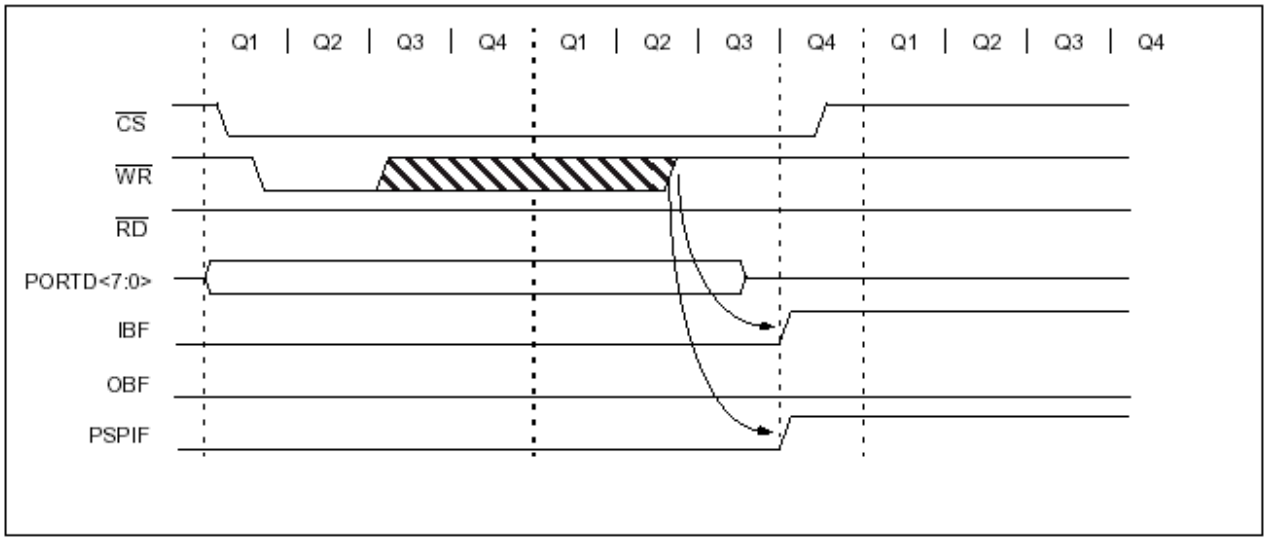
- **Rdx pins** (0à7) reliées aux registres bidirectionnels **8bits 74hc245** de la carte caméra et de la carte mère.
- **/RD** reliée au **ChipSelect1** fabriqué par la carte mère (synchronisé sur E)
- **/WR** reliée au **R/W** (à une inversion prêt).
- **/CS** relié à des portes logiques afin de créer le Chip Select Correct (voir schéma).

Note : *le registre 74hc245 et les portes logiques 74hc02 ont été rajoutées afin de supporter quelques problèmes de surchauffe si on coupe l'alimentation de la carte caméra en la laissant branchée à la carte mère (alimentation de la carte caméra par les fils de bus et de chip select). La première version prototype de la carte utilisée le Xilinx et le PIC*

directement, bilan : des I/Os du Xilinx grillées (pin 99…) et une pin D1 d'un PIC claquée également…

Le chronogramme est donné ci-dessous (cycles d'horloge du PIC : Q1-Q4 = 200ns) :

FIGURE 3-14: PARALLEL SLAVE PORT WRITE WAVEFORMS



7.2 Contraintes logicielles (pour le PIC) :

- le PIC doit initialiser le processeur vidéo par le bus I2C lors de la mise sous tension.
- le 68HC11F1 doit pouvoir lire un octet en moins de 1.6µs (cas typique d'un lecture en mémoire avec LDAA 0,X où X=adresse de la case mémoire mappée sur ChipSelect1), le PIC doit alors gérer une interruption et l'envoi d'une donnée en moins de 8 instructions environ (1cycle PIC@20MHz = 200ns).
- Le PIC doit interpréter des commandes venant de la carte mère (par utilisation du port esclave en lecture ou par le chipselect2). Les commandes seront du type « initialiser la carte « , « changer de camera », « sélection d'un paramètre quelconque… »

7.3 Commandes et fonctions du PIC

7.3.1 Les commandes coté carte mère

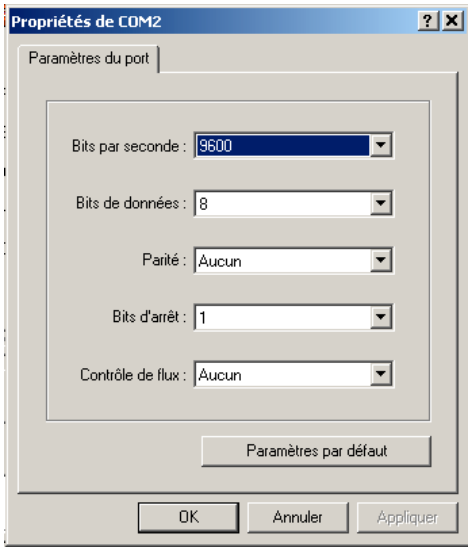
- INS_START = 0x30 = demande le lecture d'une page mémoire (dernière page mémoire remplie)
- INS_FIN = 0x40 = signale la fin de la lecture (**NE PAS L'OUBLIER**)
- INS_CAM1 = 0x10 = commande vers le saa7111 pour numériser la caméra 1 (ai11).
- INS_CAM2 = 0X20 = commande vers le saa7111 pour numériser la caméra 2 (ai21).

Un programme simple en C est donné en annexe pour illustrer cette partie.

Note post-coupe : penser à mettre une faible tempo (**qques ms**) entre chaque octet lu et **SURTOUT** attendre au minimum **40ms** entre chaque requête de lecture afin que le PIC rafraîchisse au moins une image dans sa mémoire RAM.

7.3.2 Les commandes côté liaison série (débug)

Le PIC gère également des fonctions de débugs par la liaison série. La configuration du port série de l'ordinateur est la suivante dans un hyperterminal :



A la mise sous tension ou au Reset, le PIC envoi la chaîne « VisionBoard is alive ! ». Les commandes réservées au port série seulement sont :

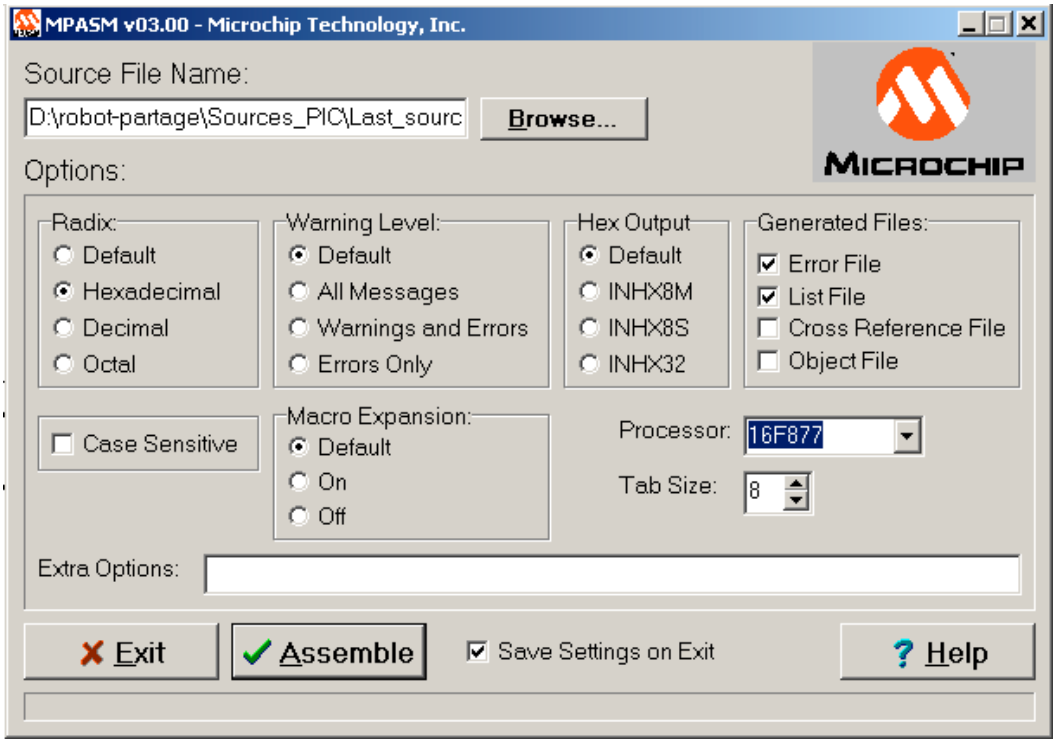
- caractère « **$** » : lecture de la banque mémoire.
- caractère « ***** » : lecture du Xilinx.
- caractère "**!**" : fonction debug…
- caractère "**1**" : sélection camera 1 (AI11).
- caractère « **2** » : sélection caméra 2 (AI21).
- caractère « **0** » : force une initialisation (par I2C) du convertisseur SAA7111.

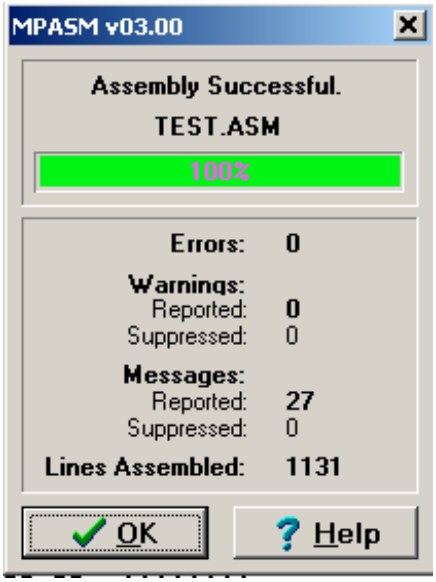
Note : ATTENTION : *ne pas utiliser la liaison série debug **ET** la lecture par la carte mère en même temps…*

7.4 Programmation du PIC

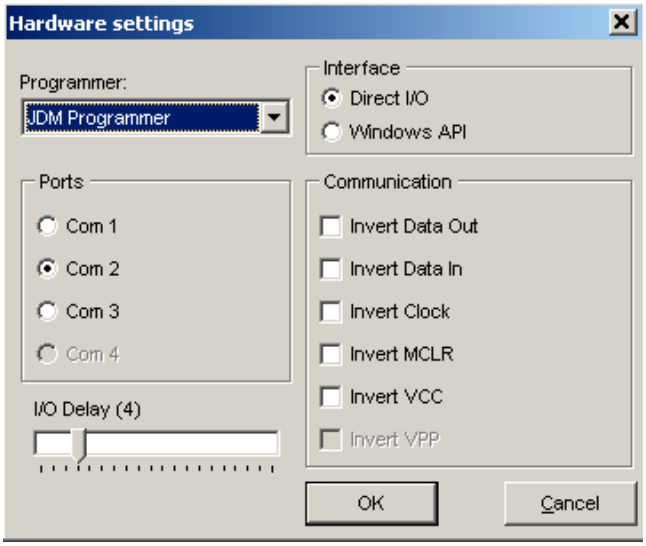
7.4.1 Compilation

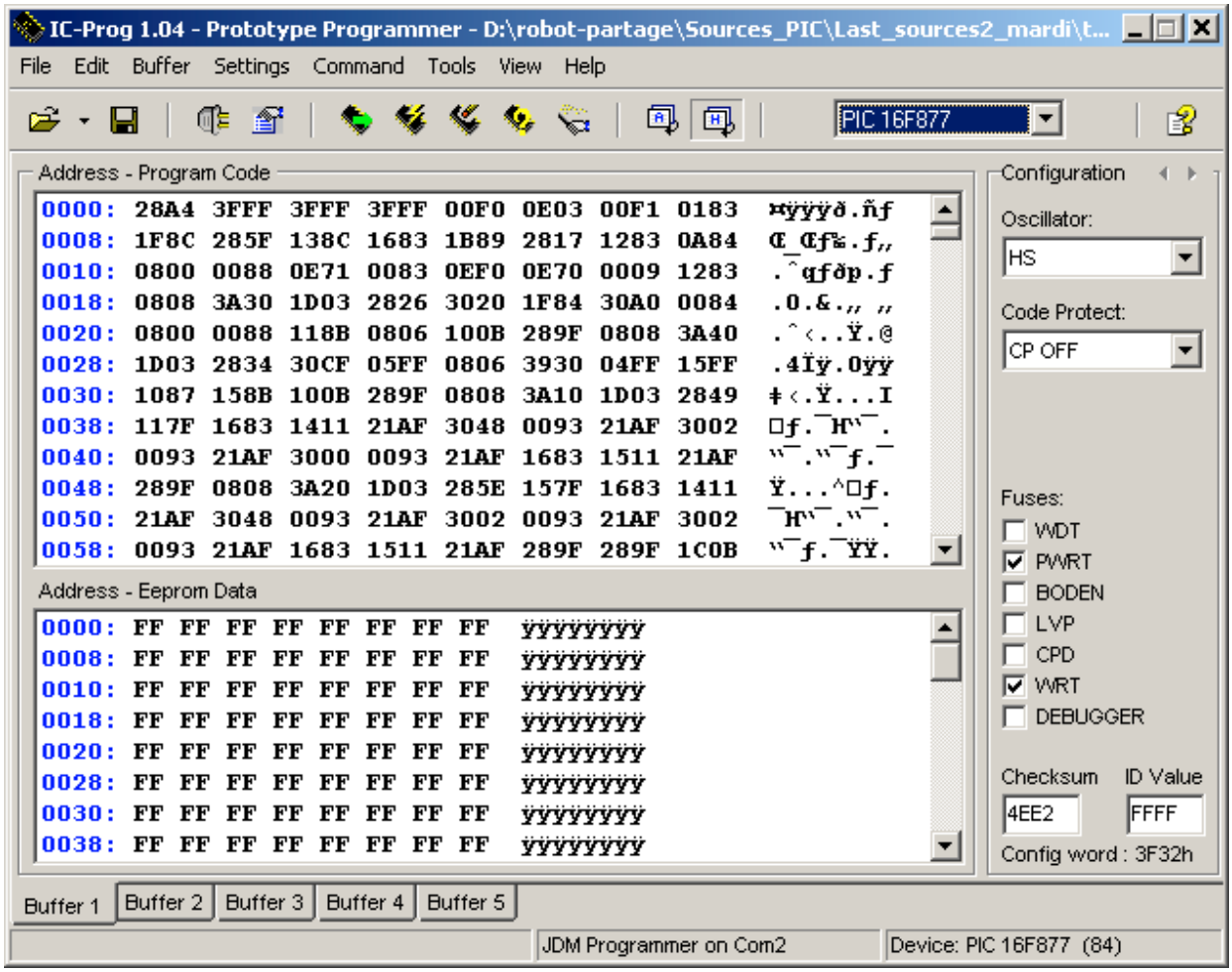
La compilation a été réalisée avec MPLAB téléchargeable gratuitement à l'adresse suivante : http://www.microchip.com/1000/pline/tools/picmicro/devenv/mplabi/mplab5x/index.htm





Le téléchargement du code a été réalisé grâce à un programmateur type JDM et avec ICPROG (http://www.ic-prog.com/) (voir la config globale à respecter…)





Note : ATTENTION : *si on programme **in-situ** le PIC16F877 par la carte debug présentée ci-dessous, il faut basculer le switch de la carte caméra de **Vdd** vers **Prog** afin de déconnecter les alimentations et les masses du PIC du reste du montage (différentes de celle du programmateur).*

8. Carte de Debug & débuggages rapides

8.1 Leds embarquées sur la carte caméra

La carte débug permet comme son nom l'indique de vérifier des signaux. La carte caméra possède quelques leds de contrôle qui sont les suivantes :

- **led +5v** : indique la présence de +5v sur le régulateur concerné
- **led Vs** : indique par un faible éclairage la présence de tops de synchro, ce voyant ne reflète que le fonctionnement correct du SAA711 (initialisation OK)
- **led 27MHz** : elle doit changer d 'état tout les secondes, elle indique en gros que l'horloge à l'entrée du Xilinx (donc en sortie du SAA711) est de 27MHz.
- **led other** indique lorsqu'elle s'allume qu 'un accès par la carte mère est effectué, **si pas de clignotement = pas d'accès** (***Note post-coupe :*** c'est donc la faute des infos qui programment n'importe comment…pas des élecs… inutile donc d'appeler sur le portable 5 minutes avant le début du match…)
- **led run** : un clignotement indique un fonctionnement normal du PIC sur caméra 1, un clignotement rapide indique qu'on utilise la caméra 2. S'il n'y a pas de clignotement, c'est que le PIC est saturé de requêtes de la part de la carte mère (cf. remarque précédente ***Note post-coupe)***

8.2 Carte de débug externe

La carte de débuggage externe se relie par un câble DB25 à la carte caméra. Cette carte permet :

- de visualiser les signaux Rouge, Vert et Bleu en temps réel qui sortent du Xilinx (voir photo de l'écran TV plus haut) sur une TV munie d'une péritel,
- la carte permet de brancher le port série d'un PC pour la liaison série de débuggage du PIC,
- de brancher un port série pour la programmation in-situ (voir plus haut, penser à commuter le switch pour la programmation et à le remettre sur Vdd pour booter le PIC correctement),
- De programmer le coolrunner in-situ grâce au programmateur JTAG intégré et ceci grâce au port parallèle du PC,
- De visualiser les signaux sur un moniteur VGA (fonction **non opérationnelle**, le signal vidéo semble incompatible avec les moniteurs récents…) -> à débugguer

La carte est alimentée de manière séparée en 9V min (tension nécessaire afin de faire commuter la TV sur sa péritel en RVB).

Des leds de visualisations ont été rajoutées, elles donnent quelques renseignements tels que la présence de 5v sur la carte caméra, la présence du signal LLC2 à 13.5MHz utile pour le convertisseur numérique-analogique, des lignes de programmation du PIC (clock et data).

Note post-coupe : seules les fonctions de visualisation sur la télé et le port série de débug ont pu être testées… les autres ne sont pas garantie faute de temps.

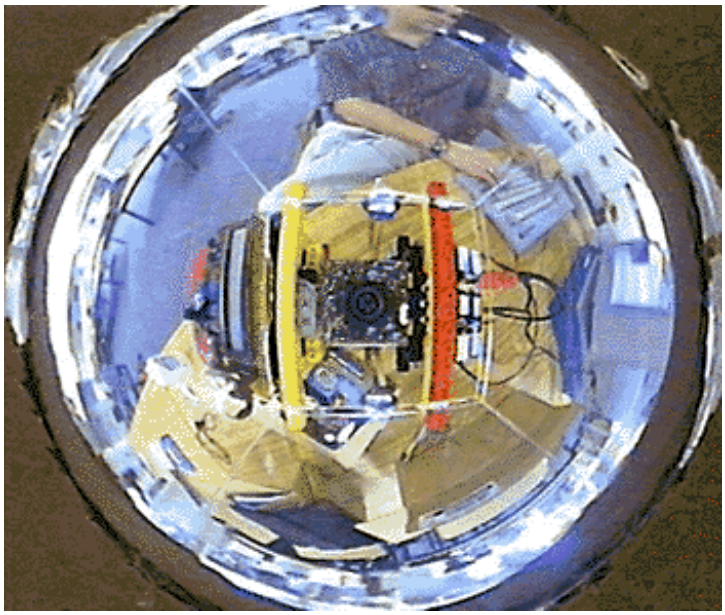
9. Idées d'utilisation supplémentaire de la caméra

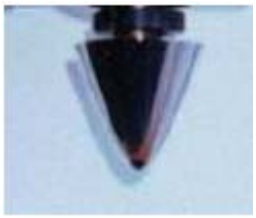
9.1 Balises colorées

Une équipe étrangère a utilisé les couleurs jaune, rouge et bleu (ou vert) pour coder des balises. En fonction de l'ordre des couleurs, on peut déterminer le code de la balise. Le schéma ci-dessous montre les balises (simples cartons coloriés) utilisées par l'équipe en question :

9.2 Détection à l'aide d'une miroir hyperbolique (cône) :

Utilisable pour la détection immédiate (40ms) des balises sur 360°…





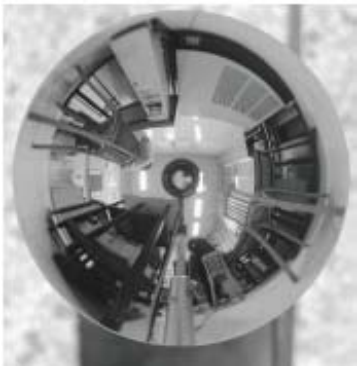
Central Panoramic Camera. Standard perspective camera with hyperbolic mirror I. Virtual camera.

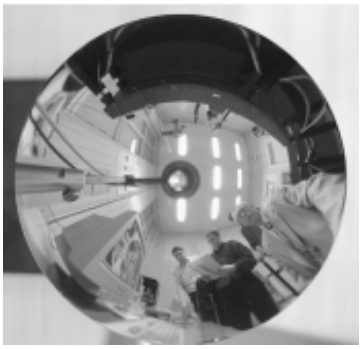
Central Panoramic Camera. Standard perspective camera with hyperbolic mirror II.

Input image. Standard perspective camera observes spherical mirror

Reprojection onto cylindrical surface.

Exemples pris sur http://cmp.felk.cvut.cz/~svoboda/Demos/Omnivis/Hyp2Img/





voir aussi :

http://www.cis.upenn.edu/~kostas/omni.html
http://cmp.felk.cvut.cz/demos/Omnivis/index.html
http://cmp.felk.cvut.cz/demos/OmnidirectionalVision.html
http://cvs.anu.edu.au/bioroboticvision/panoramic/
http://cmp.felk.cvut.cz/~sivic/ibl-exp/index.html
http://www-2.cs.cmu.edu/~iwan/localization.htm
http://cmp.felk.cvut.cz/demos/Omnivis/SphereChr/sphere.html
http://www.mpipf-muenchen.mpg.de/CR/PEOPLE/MORA/projects/samurai.html

10. Annexes

10.1 Sources Visual C++ (M$ Visual Studio 6)

Voir sur le CDROM dans Sources-RGB2HSL\digit.dsw (workspace)

10.2 Algorithme de conversion RGB vers HSL

L'algorithme employé vient de l'adresse : http://blas.cis.mcmaster.ca/~monger/hsl-rgb.html

The conversion algorithms for these color spaces are originally from the book Fundamentals of Interactive Computer Graphics by Foley and van Dam (c 1982,
Addison-Wesley). Chapter 17 describes color spaces and shows their relationships via easy-to-follow diagrams. NB this is a wonderful book, but if you are going to
get a copy, I suggest you look for the latest edition.

10.2.1 RGB -> HSL

1.Convert the RBG values to the range 0-1
Example: from the video colors page, colorbar red has R=83%, B=7%, G=7%, or in this scale, R=.83, B=.07, G=.07

2.Find min and max values of R, B, G
In the example, maxcolor = .83, mincolor=.07

3.L = (maxcolor + mincolor)/2
For the example, L = (.83+.07)/2 = .45

4.If the max and min colors are the same (ie the color is some kind of grey), S is defined to be 0, and H is undefined but in programs usually written as 0

5.Otherwise, test L.
If L < 0.5, S=(maxcolor-mincolor)/(maxcolor+mincolor)
If L >=0.5, S=(maxcolor-mincolor)/(2.0-maxcolor-mincolor)
For the example, L=0.45 so S=(.83-.07)/(.83+.07) = .84

6.If R=maxcolor, H = (G-B)/(maxcolor-mincolor)
If G=maxcolor, H = 2.0 + (B-R)/(maxcolor-mincolor)
If B=maxcolor, H = 4.0 + (R-G)/(maxcolor-mincolor)
For the example, R=maxcolor so H = (.07-.07)/(.83-.07) = 0

7.To use the scaling shown in the video color page, convert L and S back to percentages, and H into an angle in degrees (ie scale it from 0-360). From the
computation in step 6, H will range from 0-6. RGB space is a cube, and HSL space is a double hexacone, where L is the principal diagonal of the RGB cube.
Thus corners of the RGB cube; red, yellow, green, cyan, blue, and magenta, become the vertices of the HSL hexagon. Then the value 0-6 for H tells you
which section of the hexgon you are in. H is most commonly given as in degrees, so to convert
H = H*60.0
If H is negative, add 360 to complete the conversion.

10.2.2 HSL -> RGB

1.If S=0, define R, G, and B all to L

2.Otherwise, test L.
If L < 0.5, temp2=L*(1.0+S)
If L >= 0.5, temp2=L+S - L*S
In the colorbar example for colorbar green, H=120, L=52, S=79, so converting to the range 0-1, L=.52, so temp2=(.52+.79) - (.52*.79) = .899

3.temp1 = 2.0*L - temp2
In the example, temp1 = 2.0*.52 - .899 = .141

4.Convert H to the range 0-1
In the example, H=120/360 = .33

5.For each of R, G, B, compute another temporary value, temp3, as follows:
for R, temp3=H+1.0/3.0
for G, temp3=H
for B, temp3=H-1.0/3.0
if temp3 < 0, temp3 = temp3 + 1.0
if temp3 > 1, temp3 = temp3 - 1.0
In the example, Rtemp3=.33+.33 = .66, Gtemp3=.33, Btemp3=.33-.33=0

6.For each of R, G, B, do the following test:
If 6.0*temp3 < 1, color=temp1+(temp2-temp1)*6.0*temp3
Else if 2.0*temp3 < 1, color=temp2
Else if 3.0*temp3 < 2, color=temp1+(temp2-temp1)*((2.0/3.0)-temp3)*6.0
Else color=temp1
In the example,
3.0*Rtemp3 < 2 so R=.141+(.899-.141)*((2.0/3.0-.66)*6.0=.141
2.0*Gtemp3 < 1 so G=.899
6.0*Btemp3 < 1 so B=.141+(.899-.141)*6.0*0=.141

7.Scale back to the range 0-100 to use the scaling shown in the video color page
For the example, R=14, G=90, B=14

10.3 Sources vhdl

Voir sur le CDROM dans \VHDL_Robot_Final\main.vhd (code VHDL)

10.4 Brochage Xilinx

Voir sur le CDROM dans \VHDL_Robot_Final\ ball_finder.ucf (universal configurator file)

10.5 Source assembleur PIC 16F877

Voir sur le CDROM dans \ Last_sources2_mardi\test.asm

10.6 Exemple simple en C (cosmic pour hc11) de lecture de la caméra

10.6.1 Camera.h

```
#ifndef _CAMERA_
#define _CAMERA_

#define TOLERANCE 15            //ecart maximum pour considere angle bon
#define ORIGINE_CAMERA 83       //centre visu
#define MIN_CAMERA 35 // gauche visu
#define MAX_CAMERA 130 // droite visu
#define VITESSE_VIRAGE 500      //vitesse de virage
#define COEF_VITESSE_VIRAGE VITESSE_VIRAGE / ((MAX_CAMERA - MIN_CAMERA) / 2)
#define tempo_camera 1000       //tempo camera

extern void initCamera();
extern void camera();


extern char balle_detecte; //code de retour si detection d'une balle

extern char line_min_enc;
extern char line_max_enc;
extern char couleur_enc;
extern char milieu;
extern char couleur;
extern char taille;
extern char compteur;
extern char donnee;



#endif
```

10.6.2 camera.c

```
#include "global.h"
#include "camera.h"

#define tempo_camera 1000
// programme de gestion de la camera

char line_min_enc;
char line_max_enc;
char couleur_enc;
char milieu;
char couleur;
char taille;
char compteur;
char donnee;

void initCamera(){}

void camera(){
//initialisation
        wait_millisec(40);
        CAM1=0x30;
        wait(50);
        balle_detecte=0;
        compteur=0;
        donnee=CAM1;
        taille=5;

//boucle d extraction de la plus grosse boule
        while( (donnee != 0xFF)&&(balle_detecte<80) ){
                balle_detecte++;
                switch(compteur){
                        case 0 :{
                                line_min_enc = donnee;
                                compteur = 1;
                                break;
                        }
                        case 1 :{
                                couleur_enc = donnee;
                                compteur = 2;
                                break;
                        }
                        case 2 :{
                                line_max_enc = donnee;
                                compteur = 0;
                                if((line_max_enc-line_min_enc) > taille){
                                        taille = line_max_enc - line_min_enc;
                                        milieu = line_min_enc + taille/2;
                                        couleur = couleur_enc;
                                }       break;

                        }
                }
                wait(50);
                donnee=CAM1;
        }
        wait(50);
        CAM1=0x40;
}
```

// attendre 40ms au moins avant de refaire donnee=CAM1
// Les valeurs des waits sont là à titre indicatives, elles sont modifiables dans la limite du
// fonctionnement.

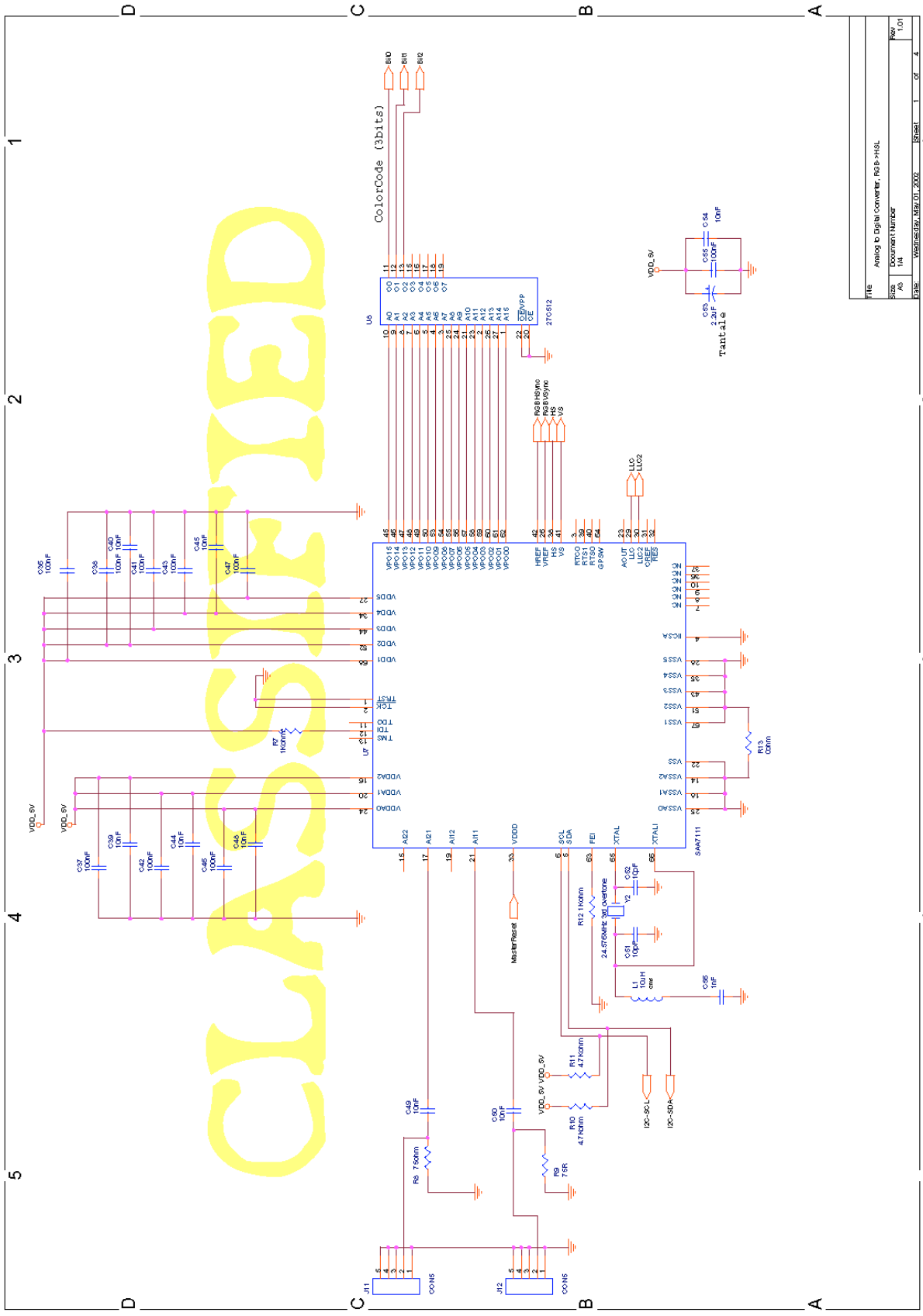
11. Bugs Liste

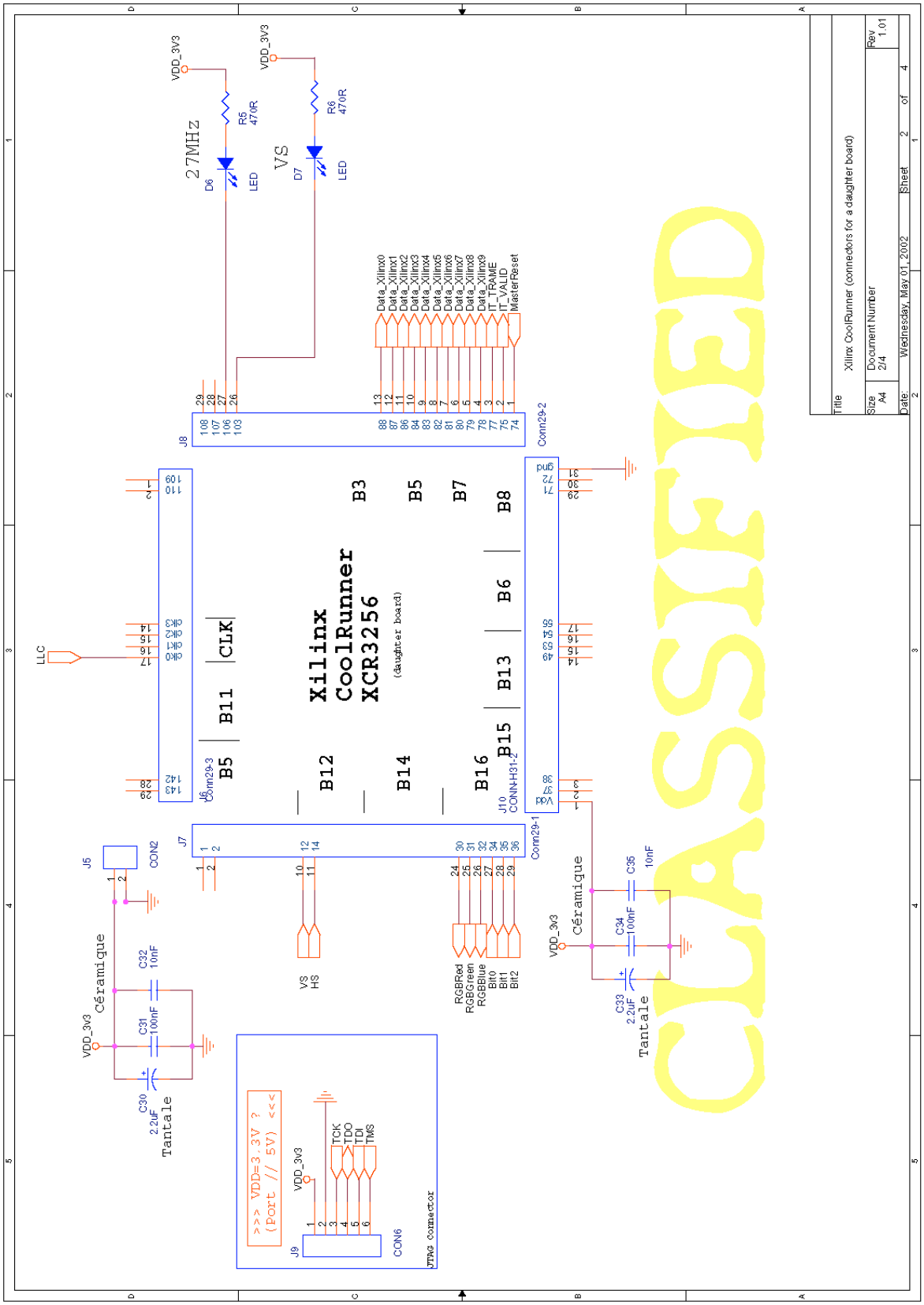
Liste des bugs rencontrés sur la **carte caméra finale** à corriger les années suivantes :

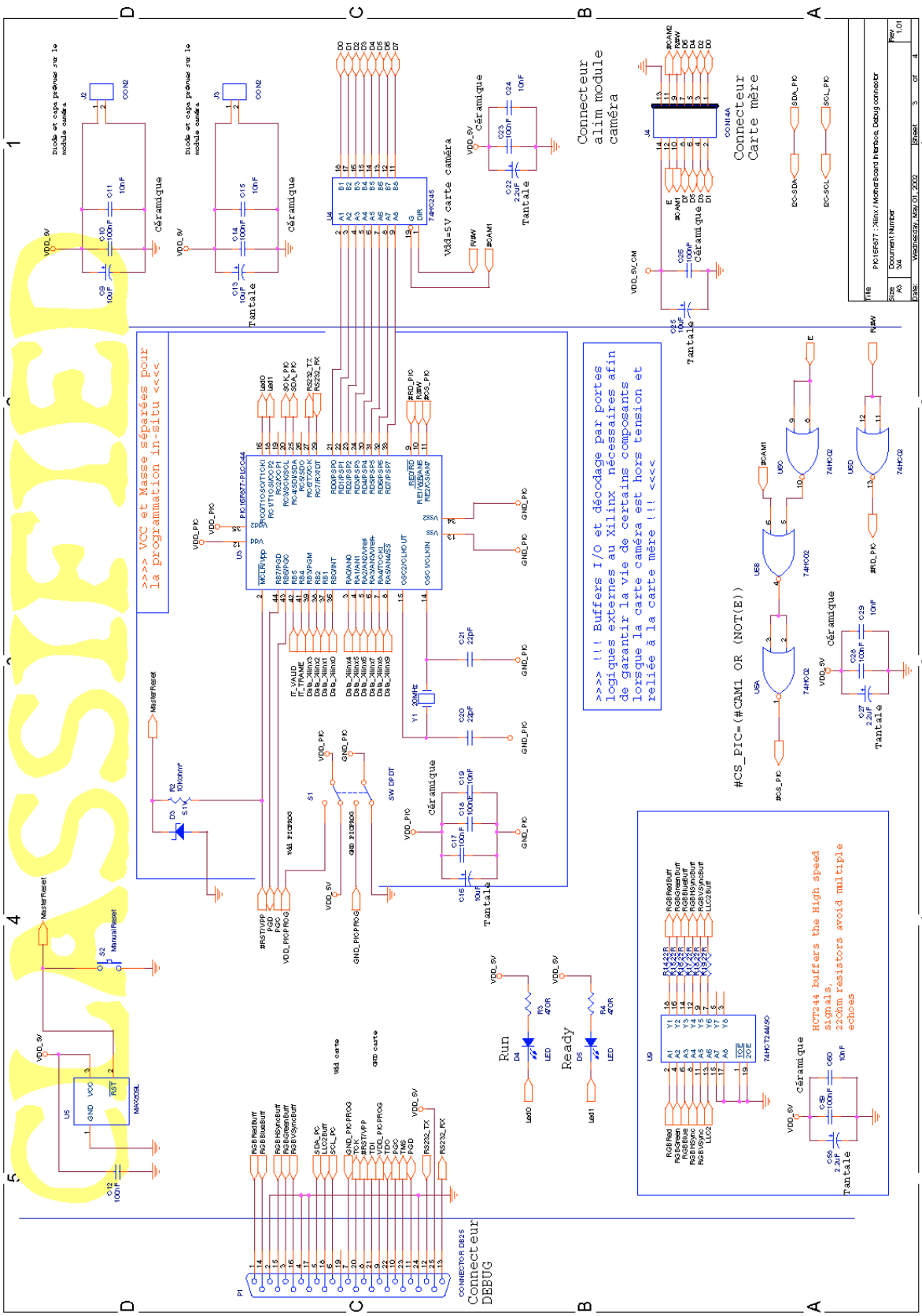
- oubli de résistances de rappel sur PGC et PGD (interruptions intempestives : mettre 2 résistances 100Kohm de pull-down sur PGC et PGD).
- Une inversion (d'origine inconnue) à faire pour les bit2 et bit0 en entrée du Xilinx (inversion faite en VHDL dans le fichier .ucf)
- Une double initialisation de l'I2C à faire pour les resets à chaud (du probablement aux pins du PIC qui envoient un pulse à l'état bas sur les lignes I2C). (1ère initialisation mal interprétée par le SAA7111, 2ième initialisation correctement interprétée).
- Orcad / Capture : erreurs de labels / flags : I2C_SCL et I2C_SCK
- Orcad / Layout : géométrie du support du Xilinx à revoir : horloge tck0 non câblée.
- Orcad / Layout : tourner les géométries des connecteurs RCA de 180°.
- Orcad / Layout : sérigraphies fausses pour les « + » et les « -« à coté des connecteurs d'alimentations caméras et alim générale.
- + quelques erreurs surprises à chercher…

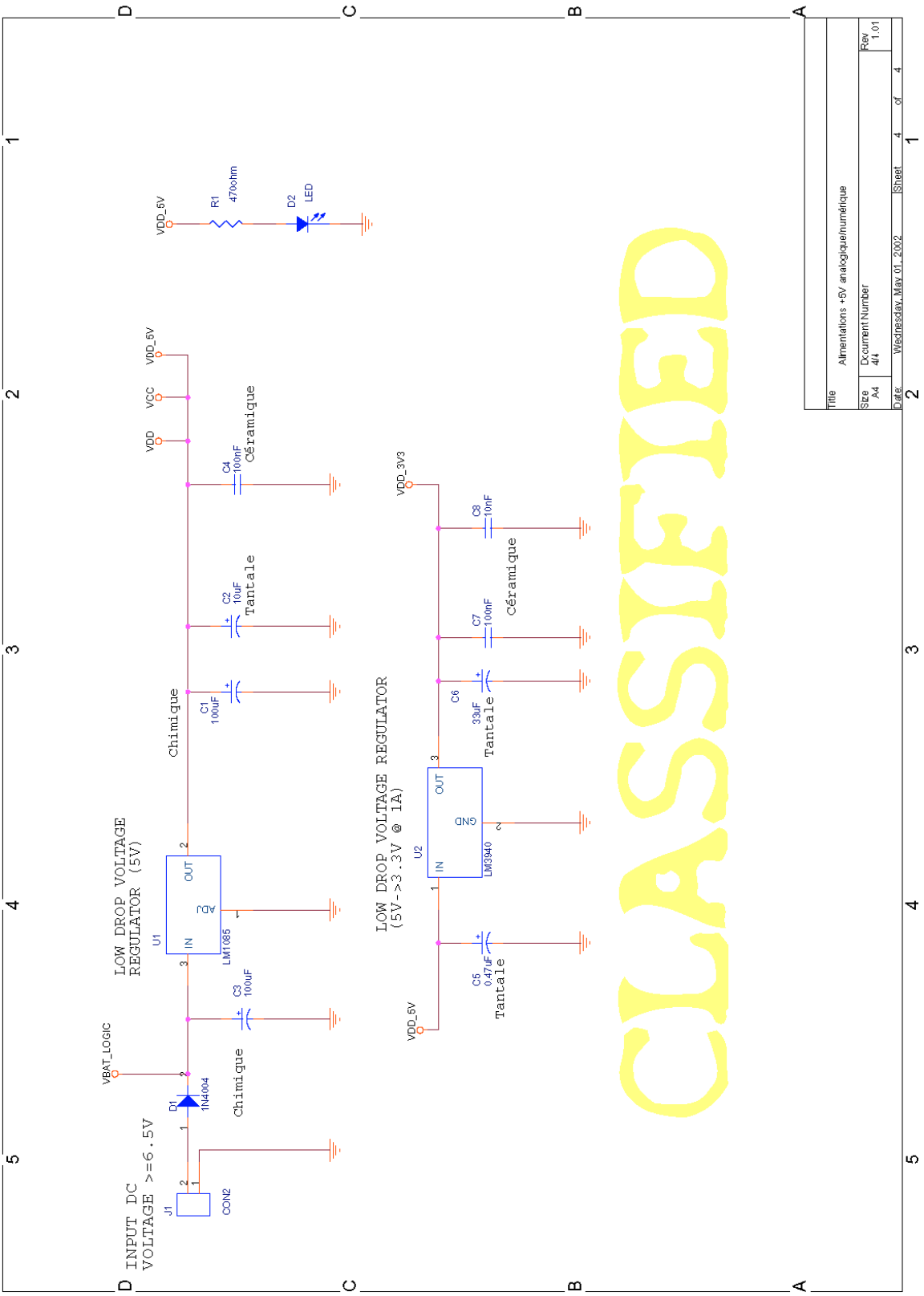
12. Carte caméra montée dans le robot

12.1 Schémas


ColorCode (3bits)
Tantale
MasterReset
I2C-SCL
I2C-SDA
CONN5
CLASSIFIED


Xilinx
CoolRunner
XCR3256
(daughter board)
B3
B5
B7
B8
B6
B13
B15
B16
B14
B12
B11
B5
CLK
J8
Conn29-2
J10
CONN-H31-2
Conn29-1
J7
J6
Conn29-3
J5
CON2
J9
CON6
JTAG connector
VDD_3V3
27MHz
VS
D6
D7
LED
R5
470R
R6
470R
LLC
VS
HS
RGBRed
RGBGreen
RGBBlue
Bit0
Bit1
Bit2
Céramique
Tantale
C30
2.2uF
C31
100nF
C32
10nF
C33
2.2uF
C34
100nF
C35
10nF
TCK
TDO
TDI
TMS
>>> VDD=3.3V ?
(Port // 5V) <<<
Data_Xilinx0
Data_Xilinx1
Data_Xilinx2
Data_Xilinx3
Data_Xilinx4
Data_Xilinx5
Data_Xilinx6
Data_Xilinx7
Data_Xilinx8
Data_Xilinx9
IT_TRAME
IT_VALID
MasterReset
Title
Xilinx CoolRunner (connectors for a daughter board)
Size
A4
Document Number
2/4
Rev
1.01
Date:
Wednesday, May 01, 2002
Sheet
2
of
4
CLASSIFIED


Connecteur alim module caméra
Connecteur Carte mère
Connecteur DEBUG
>>>> VCC et Masse séparées pour la programmation in-situ <<<<
>>>> !!! Buffers I/O et décodage par portes logiques externes au Xilinx nécessaires afin de garantir la vie de certains composants lorsque la carte caméra est hors tension et reliée à la carte mère !!! <<<<
#CS_PIC=(#CAM1 OR (NOT(E))
Vdd=5V carte caméra
Céramique
Tantale
Run
Ready
74HC244 buffers the High speed signals, 22Ohm resistors avoid multiple echoes
CLASSIFIED

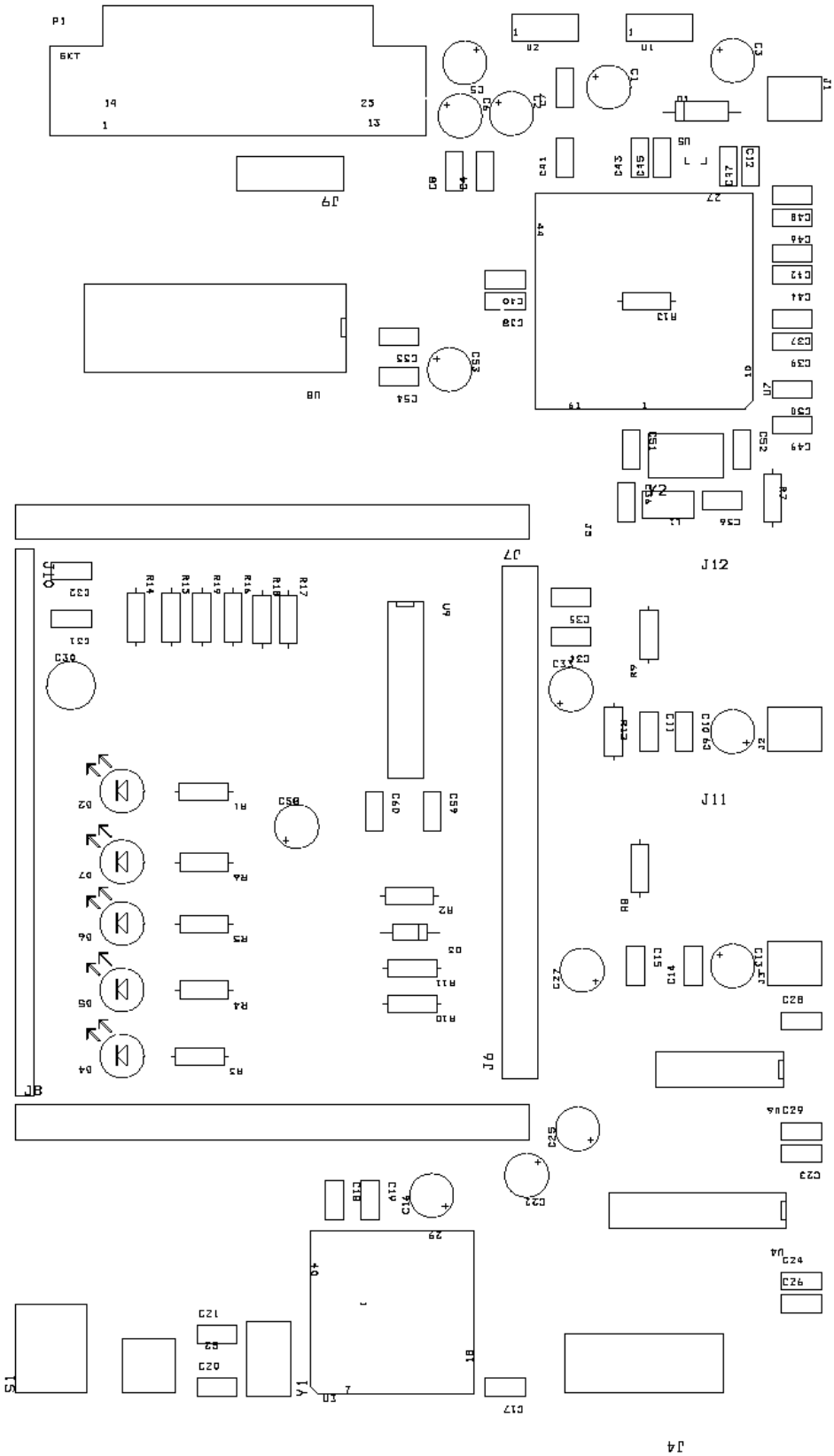

CLASSIFIED
INPUT DC
VOLTAGE >=6.5V
J1
CON2
D1
1N4004
VBAT_LOGIC
LOW DROP VOLTAGE
REGULATOR (5V)
U1
LM1086
IN
OUT
ADJ
C3
100uF
Chimique
C1
100uF
Chimique
C2
10uF
Tantale
C4
100nF
Céramique
VDD
VCC
VDD_5V
R1
470ohm
D2
LED
VDD_5V
LOW DROP VOLTAGE REGULATOR
(5V->3.3V @ 1A)
U2
LM3940
IN
OUT
GND
C5
0.47uF
Tantale
C6
33uF
Tantale
C7
100nF
Céramique
C8
10nF
VDD_3V3
Title
Alimentations +5V analogique/numérique
Size
A4
Document Number
44
Rev
1.01
Date:
Wednesday, May 01, 2002
Sheet
4
of
4

12.2 Nomenclature

Analog to Digital Converter, RGB->HSL Revised: Saturday, April 27, 2002
1/4 Revision: 1.01
Bill Of Materials April 27,2002 13:28:00Page1

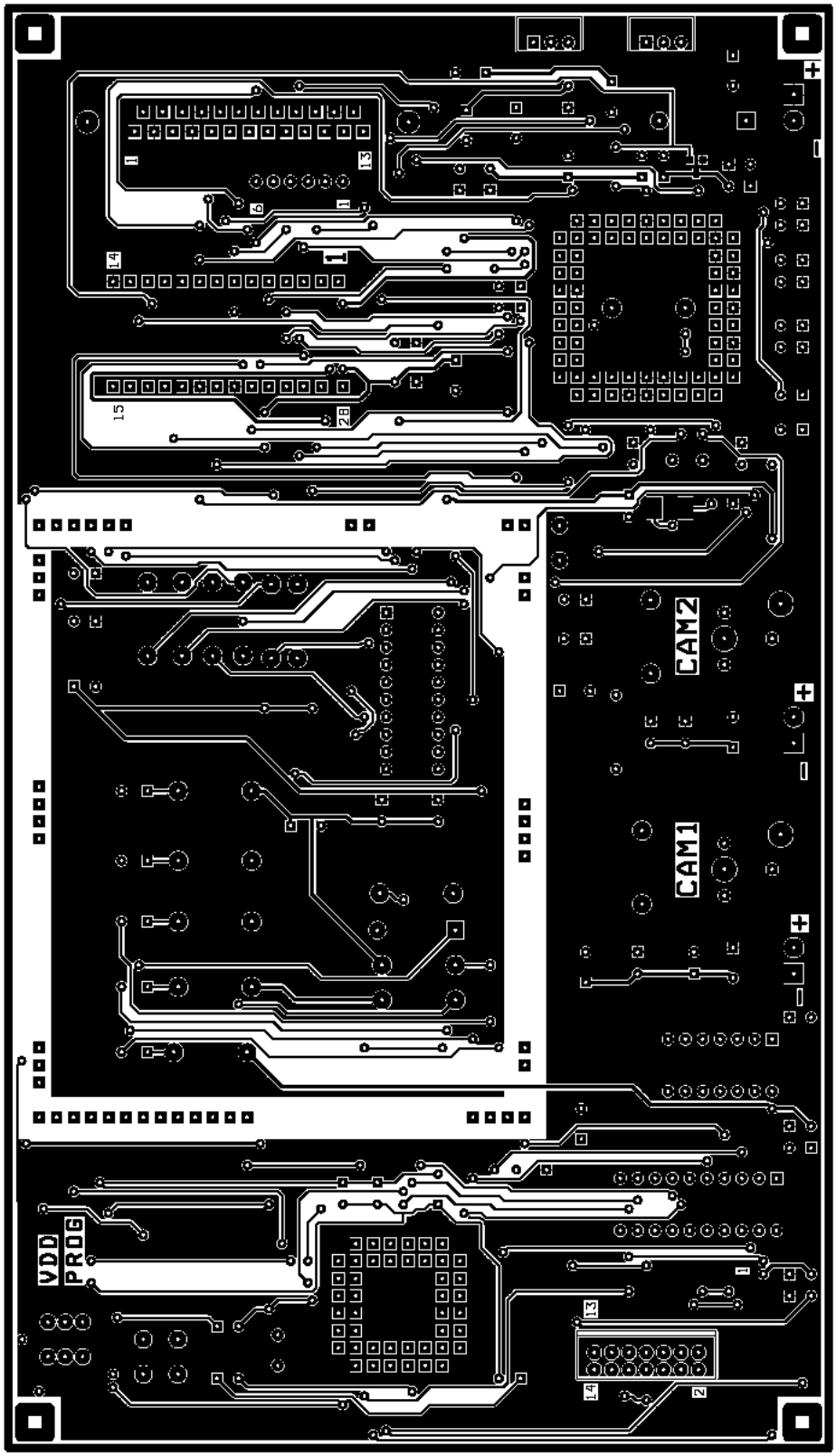
Item	Quantity	Reference	Part
1	2	C1,C3	100uF
2	5	C2,C9,C13,C16,C25	10uF
3	22	C4,C7,C10,C12,C14,C17, C18,C23,C26,C28,C31,C34, C36,C37,C38,C41,C42,C43, C46,C47,C55,C59	100nF
4	1	C5	0.47uF
5	1	C6	33uF
6	17	C8,C11,C15,C19,C24,C29, C32,C35,C39,C40,C44,C45, C48,C49,C50,C54,C60	10nF
7	2	C21,C20	22pF
8	6	C22,C27,C30,C33,C53,C58	2.2uF
9	2	C51,C52	10pF
10	1	C56	1nF
11	1	D1	1N4004
12	5	D2,D4,D5,D6,D7	LED
13	1	D3	5.1V
14	4	J1,J2,J3,J5	CON2
15	1	J4	CON14A
16	1	J6	Conn29-3
17	1	J7	Conn29-1
18	1	J8	Conn29-2
19	1	J9	CON6
20	1	J10	CONN-H31-2
21	2	J11,J12	CON5
22	1	L1	10uH
23	1	P1	CONNECTOR DB25
24	1	R1	470ohm
25	1	R2	10Kohm*
26	4	R3,R4,R5,R6	470R
27	1	R7	1Kohm*
28	1	R8	75ohm
29	1	R9	75R
30	2	R11,R10	4.7Kohm
31	1	R12	1Kohm
32	1	R13	0ohm
33	6	R14,R15,R16,R17,R18,R19	22R
34	1	S1	SW DPDT
35	1	S2	ManualReset
36	1	U1	LM1085
37	1	U2	LM3940
38	1	U3	PIC16F877-PLCC44
39	1	U4	74HC245
40	1	U5	MAX809L
41	1	U6	74HC02
42	1	U7	SAA7111
43	1	U8	27C512
44	1	U9	74HCT244/SO
45	1	Y1	20MHz
46	1	Y2	24.576MHz 3rd overtone

12.3 Implantation



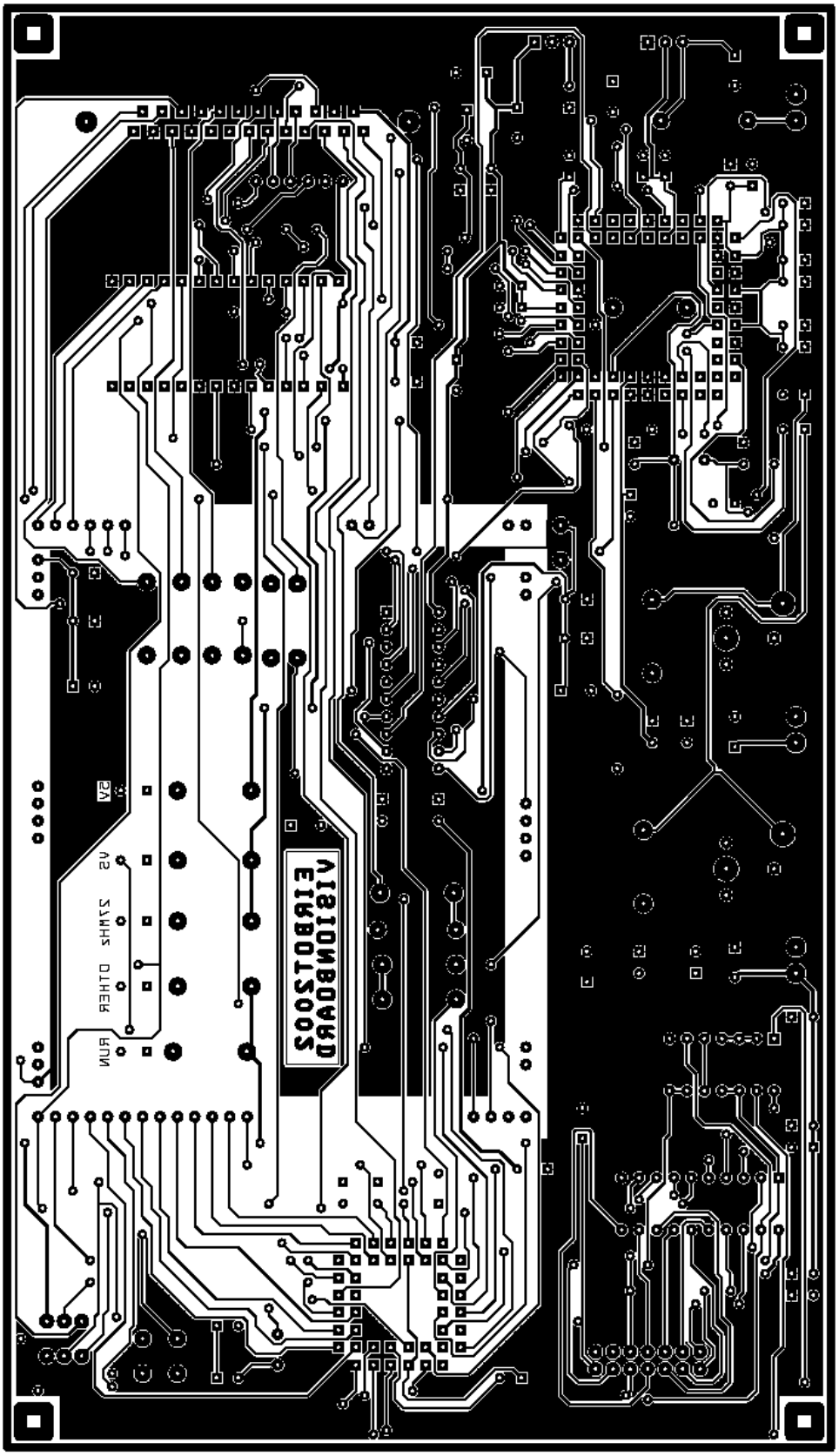
12.4 Typons

12.4.1 Top layer (!!échelle!!)



Les typons à l'échelle sont disponible au format PDF dans : Schematique-typons-Orcad\VisionBoard\typonsPDFs

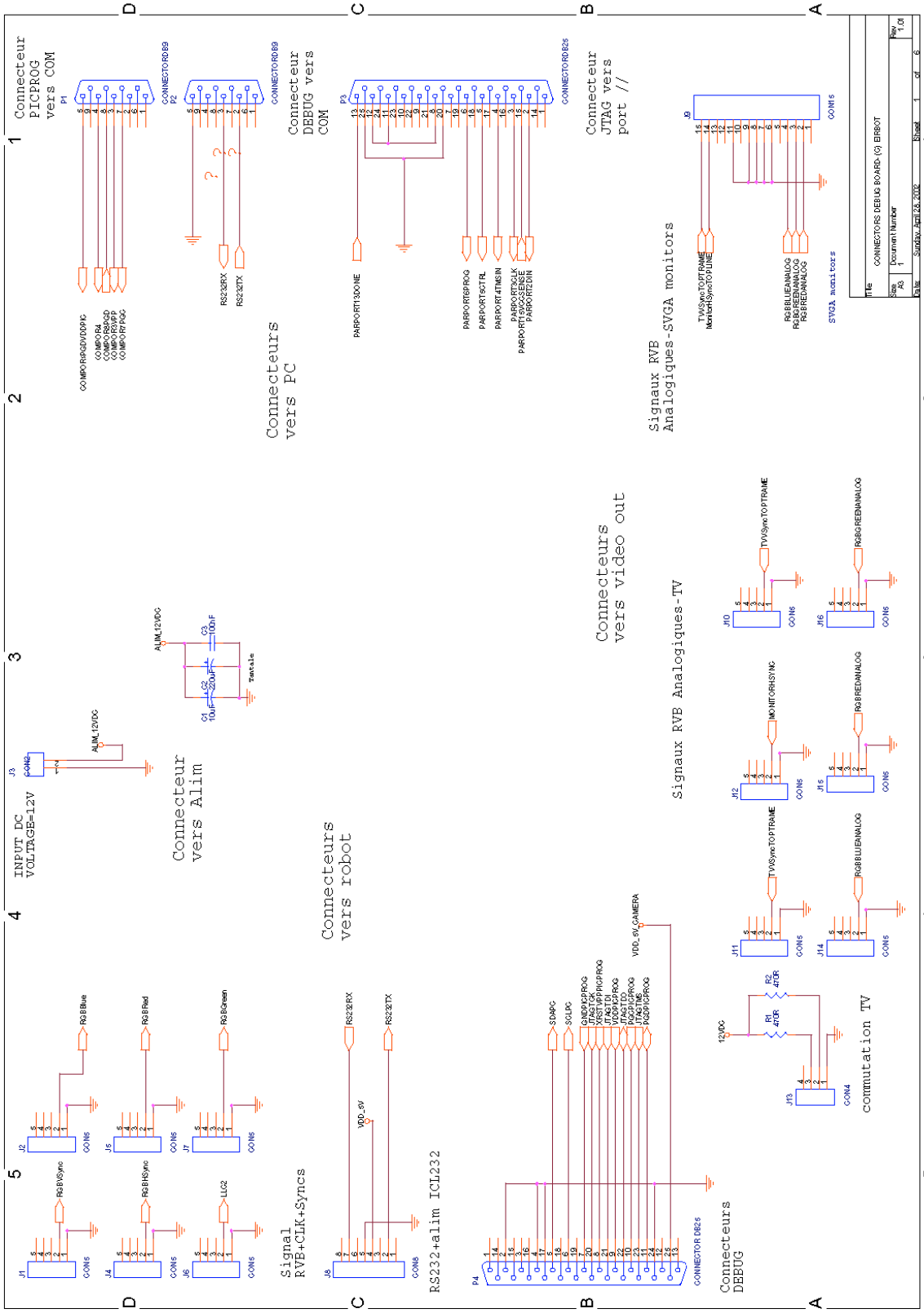
12.4.2 Bottom layer (!!échelle!!)

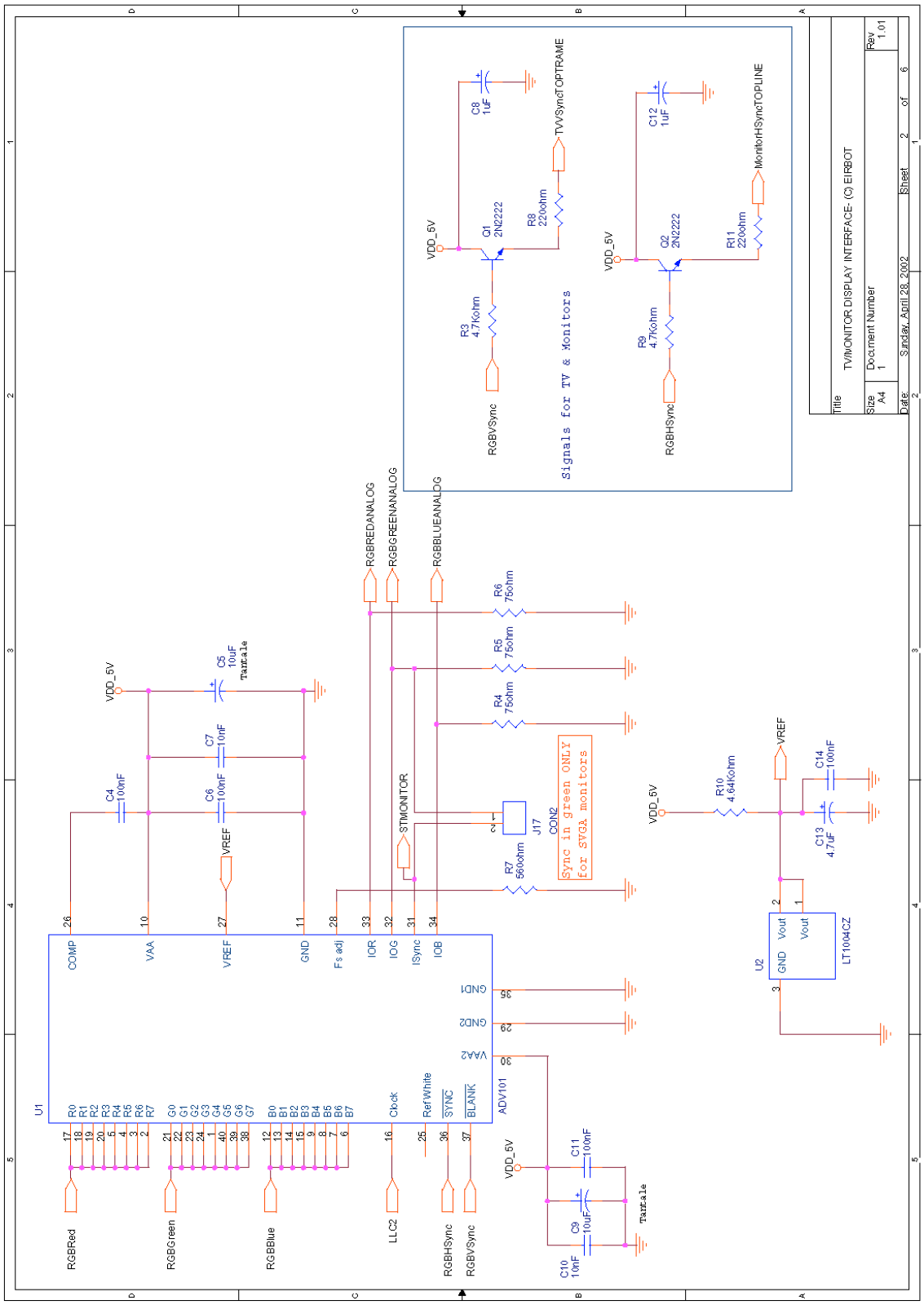


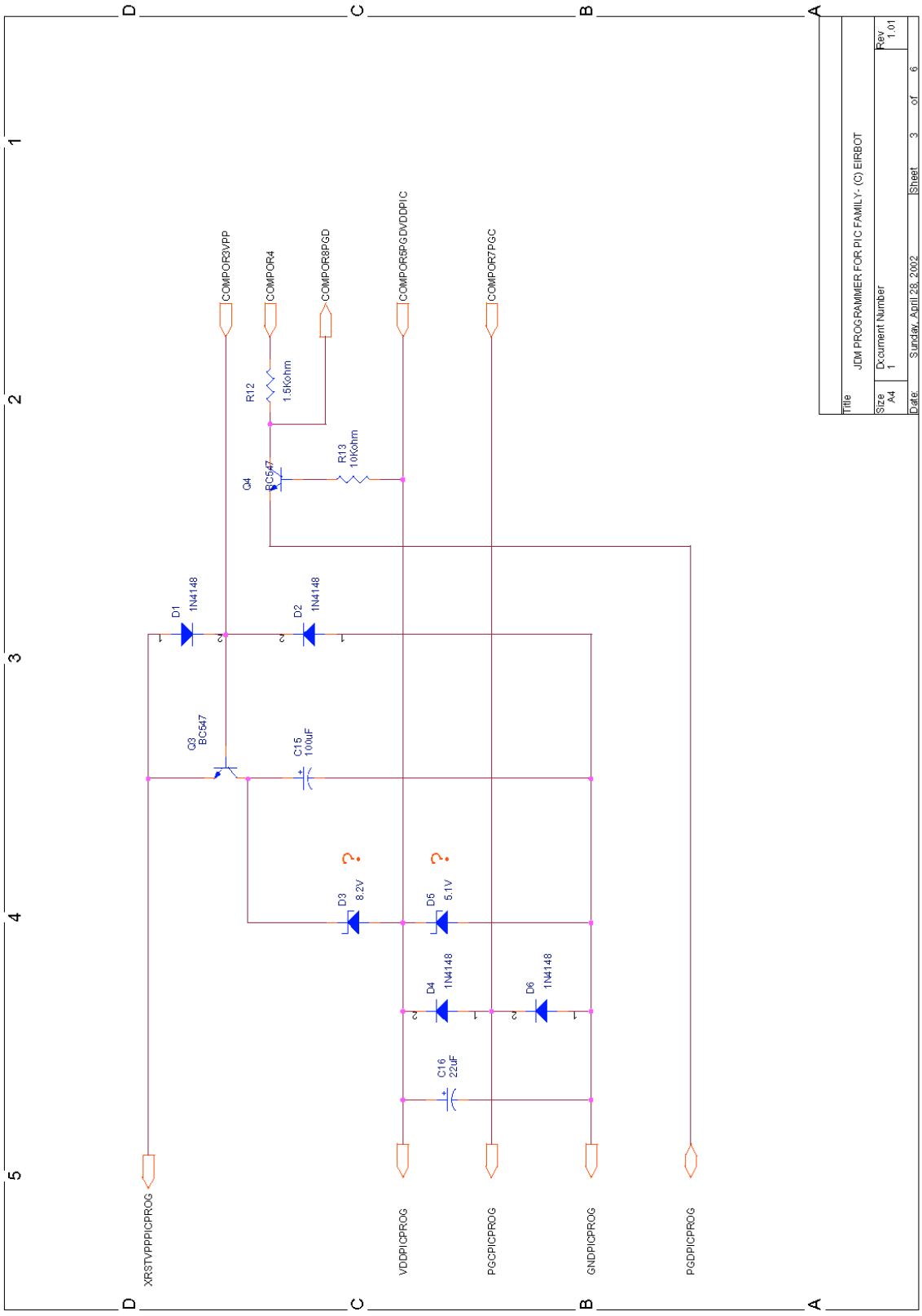
Les typons à l'échelle sont disponible au format PDF dans : Schematique-typons-Orcad\VisionBoard\typonsPDFs

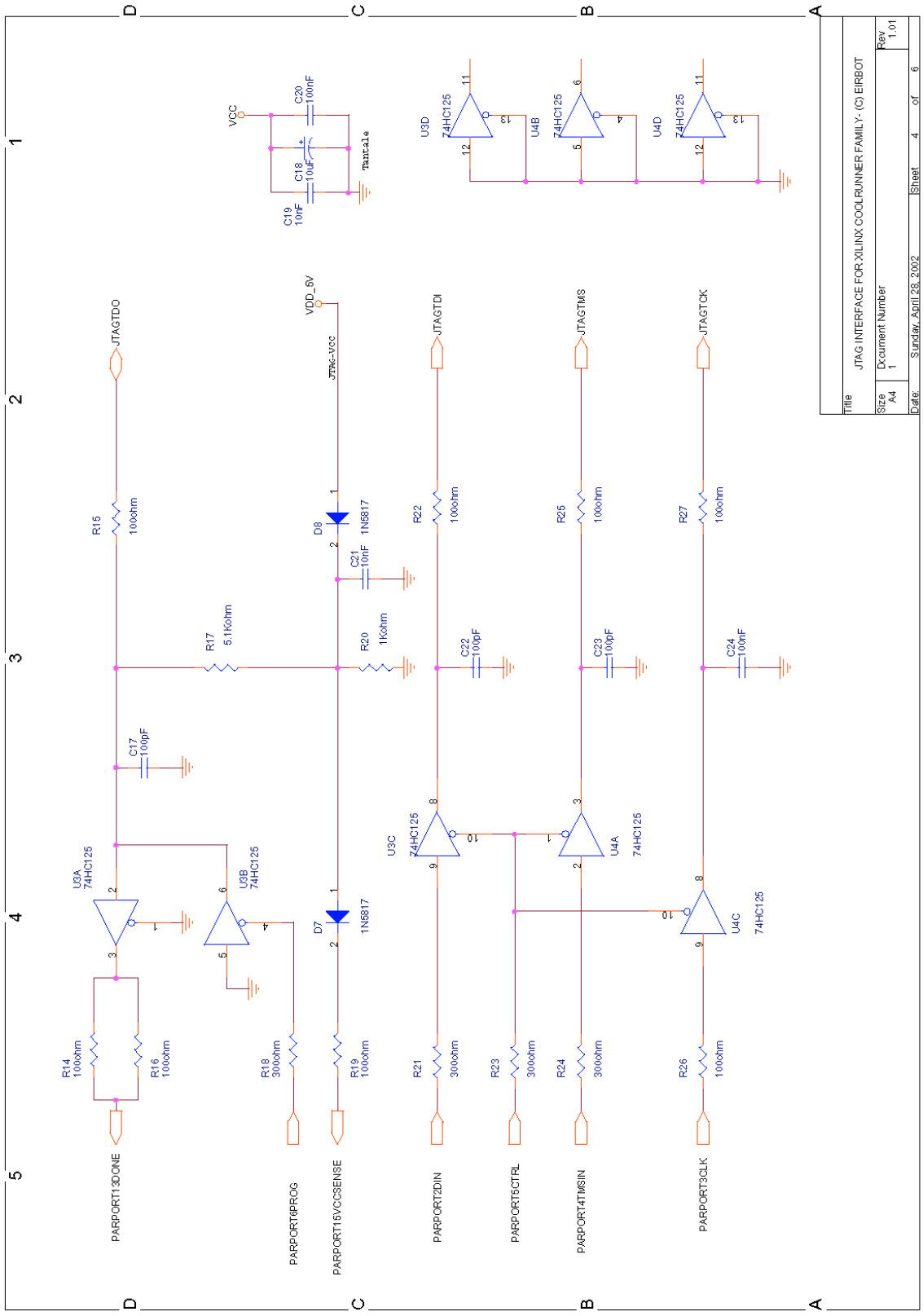
13. Carte débug

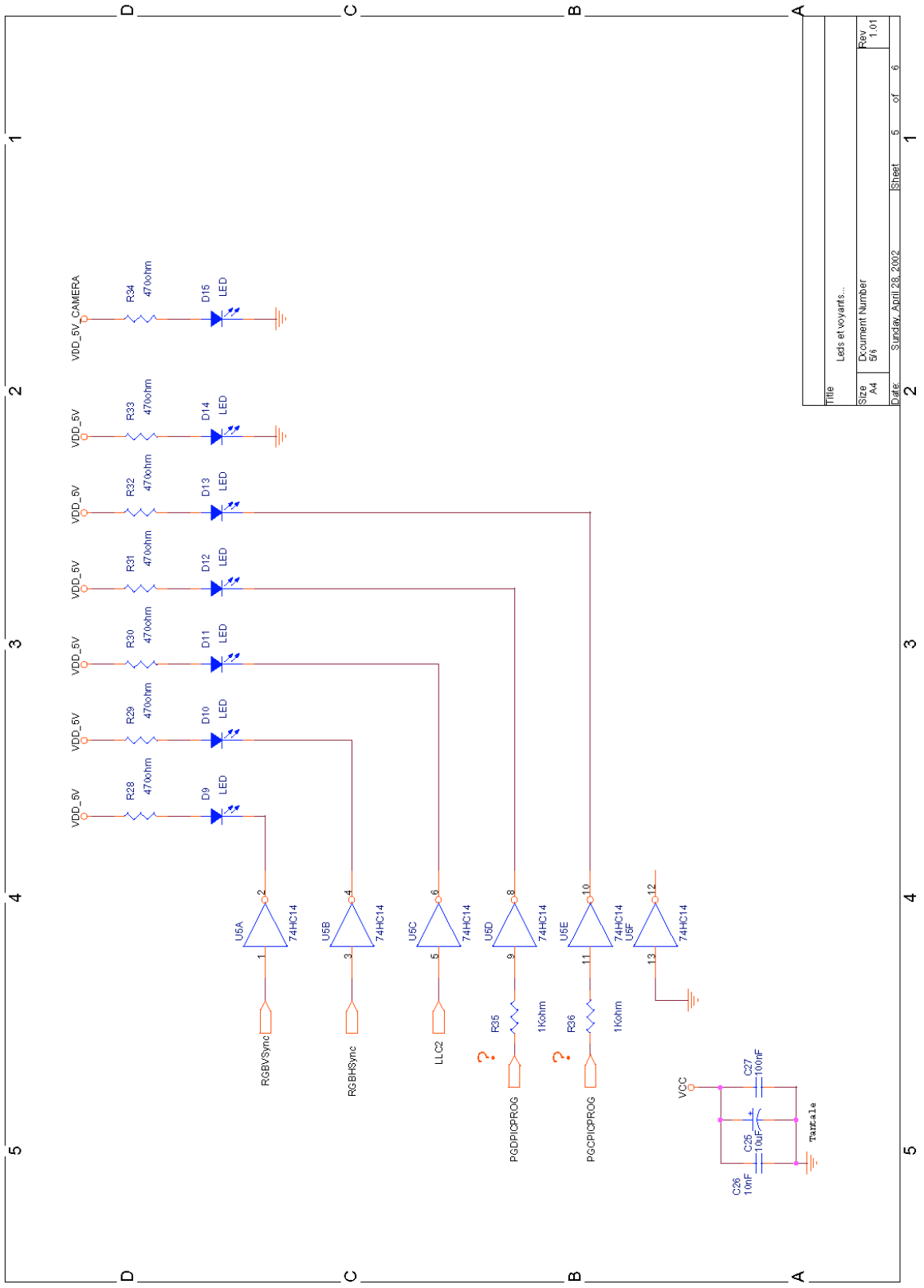
13.1 Schémas

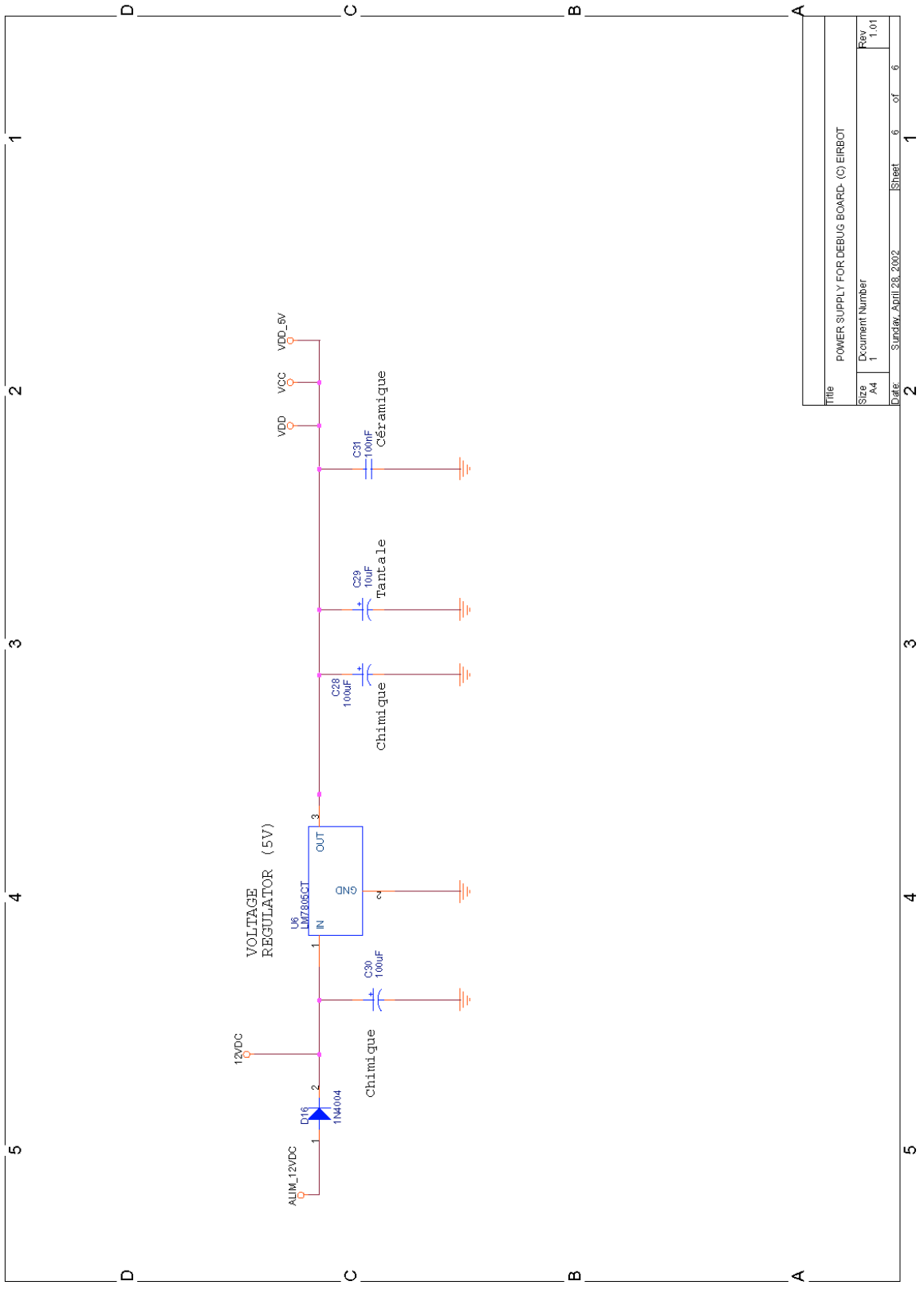

Connecteur PICPROG vers COM
Connecteur DEBUG vers COM
Connecteur JTAG vers port //
Connecteurs vers PC
Signaux RVB Analogiques-SVGA monitors
Connecteurs vers video out
Signaux RVB Analogiques-TV
INPUT DC VOLTAGE=12V
Connecteur vers Alim
Connecteurs vers robot
Signal RVB+CLK+Syncs
RS232+alim ICL232
commutation TV
Connecteurs DEBUG
CONNECTORS DEBUG BOARD (X) EIRBOT


Signals for TV & Monitors
TVVSyncTOPTRAME
MonitorHSyncTOPLINE
RGBVSync
RGBHSync
RGBREDANALOG
RGBGREENANALOG
RGBBLUEANALOG
STMONITOR
Sync in green ONLY for SVGA monitors
ADV101
LT1004CZ
RGBRed
RGBGreen
RGBBlue
LLC2
RGBHSync
RGBVSync
TVMONITOR DISPLAY INTERFACE- (C) EIRBOT
Sunday, April 28, 2002

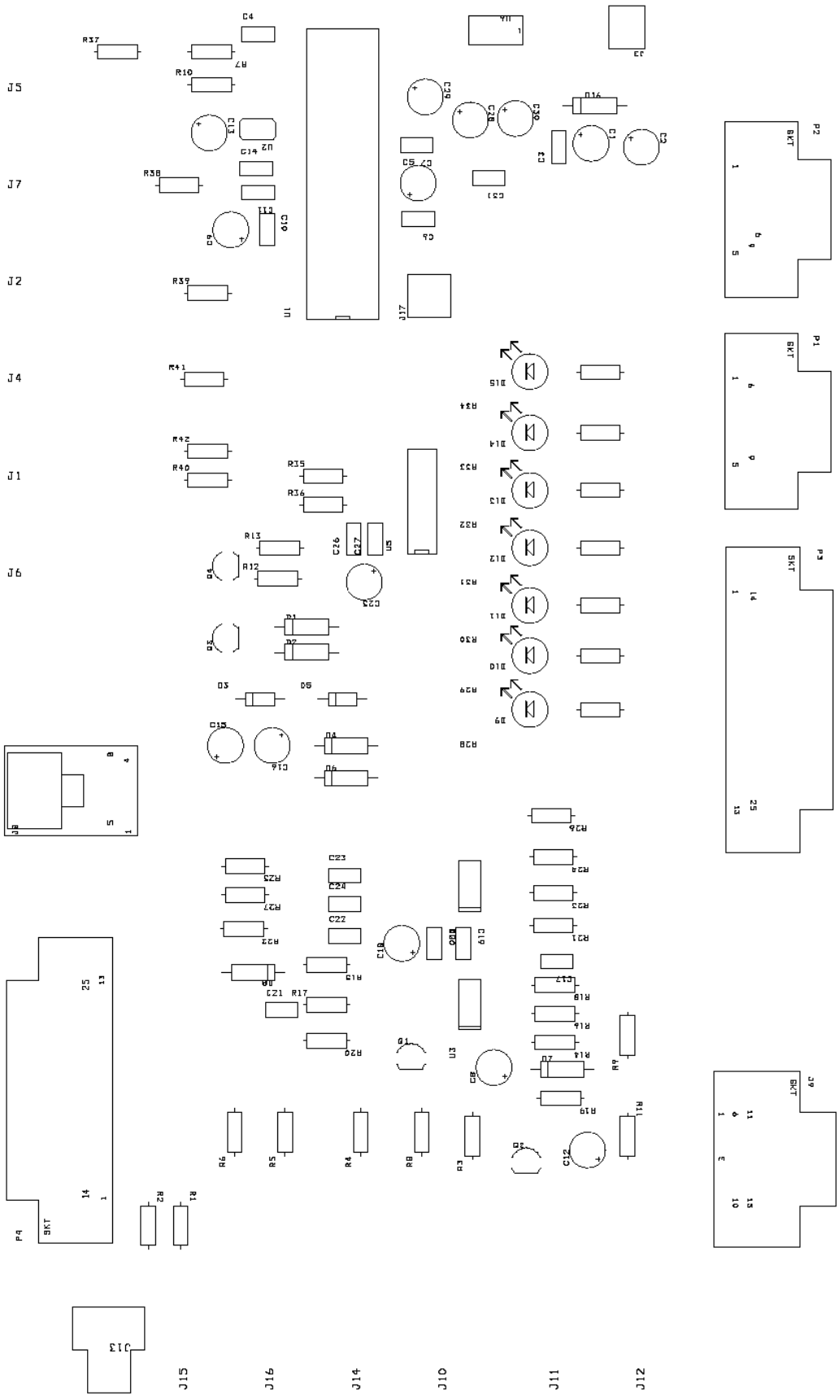

COMPOR3VPP
COMPOR4
COMPOR8PGD
COMPOR8PGDVDDPIC
COMPOR7PGC
R12
1.5Kohm
R13
10Kohm
Q4
BC547
D1
1N4148
D2
1N4148
Q8
BC547
C15
100uF
D3
8.2V
D5
5.1V
D4
1N4148
D6
1N4148
C16
22uF
XRSTVPPPICPROG
VDDPICPROG
PGCPICPROG
GNDPICPROG
PGDPICPROG
Title
JDM PROGRAMMER FOR PIC FAMILY - (C) EIRBOT
Size
A4
Document Number
1
Rev
1.01
Date:
Sunday, April 28, 2002
Sheet
3
of
6


JTAG INTERFACE FOR XILINX COOLRUNNER FAMILY - (C) EIRBOT
Title
Size A4
Document Number 1
Rev 1.01
Date: Sunday, April 28, 2002
Sheet 4 of 6
VCC
VDD_5V
JTAG-VCC
C20 100nF
C18 10uF
C19 10nF
Tantale
U3D 74HC125
U4B 74HC125
U4D 74HC125
JTAGTDO
JTAGTDI
JTAGTMS
JTAGTCK
R15 100ohm
R22 100ohm
R25 100ohm
R27 100ohm
D8 1N5817
C21 10nF
R17 5.1Kohm
R20 1Kohm
C22 100pF
C23 100pF
C24 100nF
C17 100pF
U3A 74HC125
U3B 74HC125
U3C 74HC125
U4A 74HC125
U4C 74HC125
D7 1N5817
R14 1000hm
R16 100ohm
R18 300ohm
R19 100ohm
R21 300ohm
R23 300ohm
R24 300ohm
R26 100ohm
PARPORT13DONE
PARPORT6PROG
PARPORT15VCCSENSE
PARPORT2DIN
PARPORT5CTRL
PARPORT4TMSIN
PARPORT3CLK


Title
Leds et voyants...
Size
A4
Document Number
5/6
Rev
1.01
Date:
Sunday, April 28, 2002
Sheet
5
of
6
VDD_5V_CAMERA
R34
470ohm
D15
LED
VDD_5V
R33
470ohm
D14
LED
VDD_5V
R32
470ohm
D13
LED
VDD_5V
R31
470ohm
D12
LED
VDD_5V
R30
470ohm
D11
LED
VDD_5V
R29
470ohm
D10
LED
VDD_5V
R28
470ohm
D9
LED
U5A
74HC14
U5B
74HC14
U5C
74HC14
U5D
74HC14
U5E
74HC14
U5F
74HC14
RGBVSync
RGBHSync
LLC2
R35
1Kohm
PGDPICPROG
R36
1Kohm
PGCPICPROG
VCC
C27
100nF
C25
10uF
C26
10nF
Tantale


POWER SUPPLY FOR DEBUG BOARD- (C) EIRBOT
Rev 1.01
Sheet 6 of 6
Sunday, April 28, 2002
Document Number 1
Size A4
Title
VDD_5V
VCC
VDD
C31 100nF Céramique
C29 10uF Tantale
C28 100uF Chimique
C30 100uF Chimique
VOLTAGE REGULATOR (5V)
U6 LM7805CT
OUT
GND
IN
12VDC
D16 1N4004
ALIM_12VDC

13.2 Implantation



13.2.1 Nomenclature

CONNECTORS DEBUG BOARD- (C) EIRBOT Revised: Sunday, April 28, 2002
1 Revision: 1.01
Bill Of Materials May 5,2002 17:18:04 Page1

Item	Quantity	Reference	Part
1	6	C1,C5,C9,C18,C25,C29	10uF
2	1	C2	220uF
3	9	C3,C4,C6,C11,C14,C20,C24, C27,C31	100nF
4	5	C7,C10,C19,C21,C26	10nF
5	2	C8,C12	1uF
6	1	C13	4.7uF
7	3	C15,C28,C30	100uF
8	1	C16	22uF
9	3	C17,C22,C23	100pF
10	4	D1,D2,D4,D6	1N4148
11	1	D3	8.2V
12	1	D5	5.1V
13	2	D8,D7	1N5817
14	7	D9,D10,D11,D12,D13,D14, D15	LED
15	1	D16	1N4004
16	12	J1,J2,J4,J5,J6,J7,J10, J11,J12,J14,J15,J16	CON5
17	2	J17,J3	CON2
18	1	J8	CON8
19	1	J9	CON15
20	1	J13	CON4
21	2	P2,P1	CONNECTORDB9
22	1	P3	CONNECTORDB25
23	1	P4	CONNECTOR DB25
24	2	Q2,Q1	2N2222
25	2	Q4,Q3	BC547
26	2	R2,R1	470R
27	2	R9,R3	4.7Kohm
28	3	R4,R5,R6	75ohm
29	1	R7	560ohm
30	2	R8,R11	220ohm
31	1	R10	4.64Kohm
32	1	R12	1.5Kohm
33	1	R13	10Kohm
34	8	R14,R15,R16,R19,R22,R25, R26,R27	100ohm
35	1	R17	5.1Kohm
36	4	R18,R21,R23,R24	300ohm
37	3	R20,R35,R36	1Kohm
38	7	R28,R29,R30,R31,R32,R33, R34	470ohm
39	6	R37,R38,R39,R40,R41,R42	22R
40	1	U1	ADV101
41	1	U2	LT1004CZ
42	2	U4,U3	74HC125
43	1	U5	74HC14
44	1	U6	LM7805CT

14. Adresses mails / contact /support

 Pour d'éventuelles questions après Juin 2002, contacter : 

- Yannick BENABEN : yannick.benaben@wanadoo.fr (http://www.ybnet.fr.st)
  (Tel : 06.82.69.41.11)
- Frederic DULUCQ : jedy@ifrance.com (Tel : 06.73.38.79.29)
- Frédéric LOCHON : lochon@roulaise.net (http://www.crazyfred.org)
  (Tel : 06.14.03.90.04)

Nous dédicaçons ce document à tous les futurs sabotageurs© du club robot…

La coupe robotique e=m6 dans 10 ans…

(pour atteindre les terrains, veuillez suivre la flèche rouge, merci)
(prière de ne pas abîmer la moquette avec les roues merci)