

TP caméra CCD

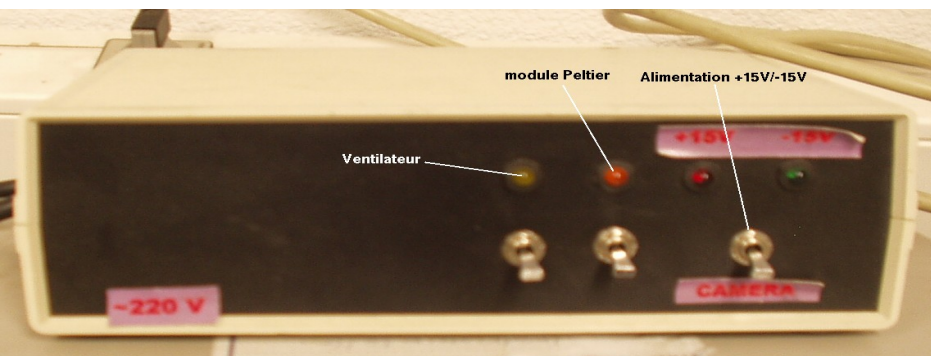
Programmation

Ce TP doit permettre à l'étudiant(e) de se familiariser avec la notion de **chronogramme**. Le chronogramme est le séquençement d'ordres logiques (signaux d'horloges) dans le temps pour actionner les phases de transfert d'un capteur CCD et de traitement de la chaîne électronique (cf. le cours). Selon le chronogramme employé, on peut avoir accès à différents modes de lecture de l'image et donc différentes "formes" de l'image numérique. Ce TP permettra également de se rendre compte du besoin de synchronisation entre l'ordinateur (via le programme de contrôle tp.c) et la caméra pour assurer un bon fonctionnement de cette dernière (cf. *datasheet* du Kodak AF400 sur ma page web).

Allumage de la caméra :

- 1) Allumer l'interrupteur général à l'arrière du boîtier d'alimentation.
- 2) Mise sous tension : interrupteur +15/-15 V. Ne pas allumer le refroidissement par l'élément thermo-électrique Peltier (interrupteur du milieu).
- 3) Mise en marche du ventilateur.

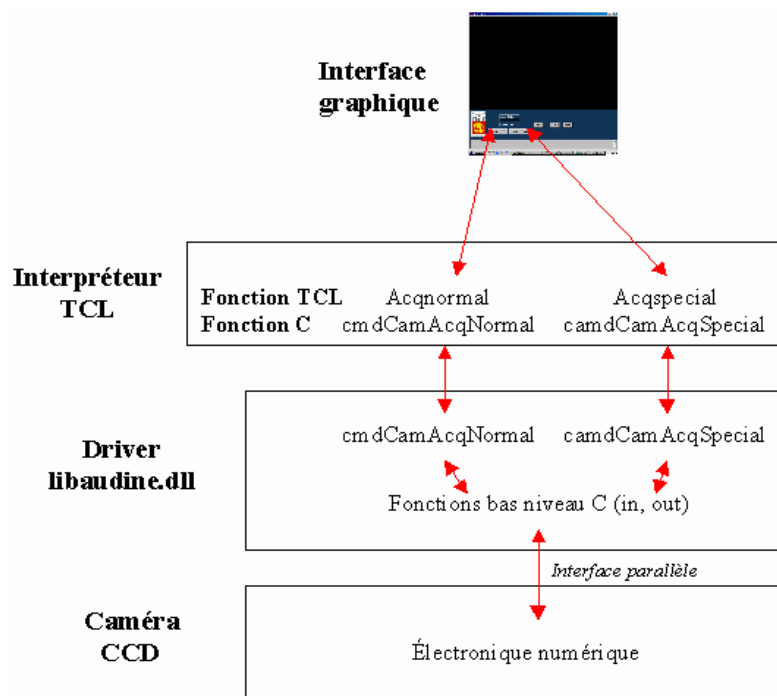
Pour éteindre la caméra, il suffit de reprendre les instructions ci-dessus en sens inverse.



Boîtier d'alimentation de la caméra Audine équipée du capteur CCD Kodak AF400.

1. Le programme d'acquisition

Le programme d'acquisition s'appelle audela.exe. Il se présente sous la forme d'un module d'interprétation de commandes : le **Tcl**. Ce module crée un interpréteur et le programme consiste en des fonctions appelées par cet interpréteur. Afin d'avoir un rendu visuel sur l'écran, l'interface graphique est réalisée avec la librairie **Tk**. Afin d'effectuer l'acquisition des images, la librairie **libaudine.dll** permet de lire les données sur le port parallèle où est connectée la **caméra Audine** (cf. le schéma sur la page suivante).



L'interpréteur de commandes Tcl a un fonctionnement très comparable à la ligne de commande de Matlab. Toute commande est passée à l'interpréteur sous la forme d'une chaîne de caractères (char* en C). Le premier "mot" de la chaîne est le nom de la commande. Ce qui nous intéresse est la commande **cam1** pour piloter la caméra CCD.

Le deuxième argument de la commande **cam1** précise le type d'action à effectuer et dirige l'interprétation vers une fonction C de la librairie libaudine.dll. Par exemple, la commande Tcl **cam1 acqnormal** est associée à la fonction C **cmdCamAcqNormal** qui se trouve dans le fichier **tp.c**. Il s'agit de la fonction d'acquisition qui sera appelée par appui sur le bouton **Acquisition normale** de l'interface.

De même, la commande Tcl **cam1 acqspecial** est associée à la fonction C **cmdCamAcqSpecial** qui se trouve dans le même fichier **tp.c**. Cette fonction est vide pour le moment et le but de ce TP est d'en faire une fonction d'acquisition du mode demi-trame. Le fichier tp.c est compilé dans la librairie libaudine.dll.

2. Analyse du listing tp.c

Cette partie du TP est à préparer à la maison avant le TP. Vous rendrez à l'enseignant(e) un compte-rendu préliminaire incluant des éléments de réponse sur les questions ci-dessous en début de séance.

- 1) Prendre le temps de lire et de comprendre chaque ligne (commentaires) de la fonction **cmdCamAcqNormal** du fichier tp.c.

Expliquer sur votre compte-rendu quelles sont les principales étapes réalisées par cette fonction (utiliser les commentaires pour vous aider).

A quoi correspondent ces différentes étapes physiquement pour la caméra ?

Dans ce fichier, on utilise une structure (camprop) pour assembler toutes les constantes de la caméra. On notera en particulier l'emploi d'éléments suivants de cette structure :

- **nb_photox** : nombre de cellules CCD actives sur l'axe X (=768)
- **nb_photoy** : nombre de cellules CCD actives sur l'axe Y (=512)
- **nb_deadbeginphotox** : nombre de cellules CCD masquées au début du registre horizontal (=14)
- **nb_deadendphotox** : nombre de cellules CCD masquées à la fin du registre horizontal (=14)
- **nb_deadbeginphotoy** : nombre de lignes masquées au début de la matrice (=4)
- **binx** : valeur du binning X
- **biny** : valeur du binning Y
- **exptime** : temps de pause (en secondes)
- **bufno** : numéro du buffer Tcl qui va accueillir l'image

- 2) En vous aidant du schéma du capteur CCD Kodak AF400 (Figure 1), associer les variables ci-dessous aux différentes parties du capteur.
- 3) De quel type de capteur CCD s'agit-il ? Combien y a-t-il d'électrodes par photosite ?
- 4) Décrire la manière dont le transfert de charges est effectué pour ce capteur. Combien d'horloges sont nécessaires pour assurer le transfert des charges ?

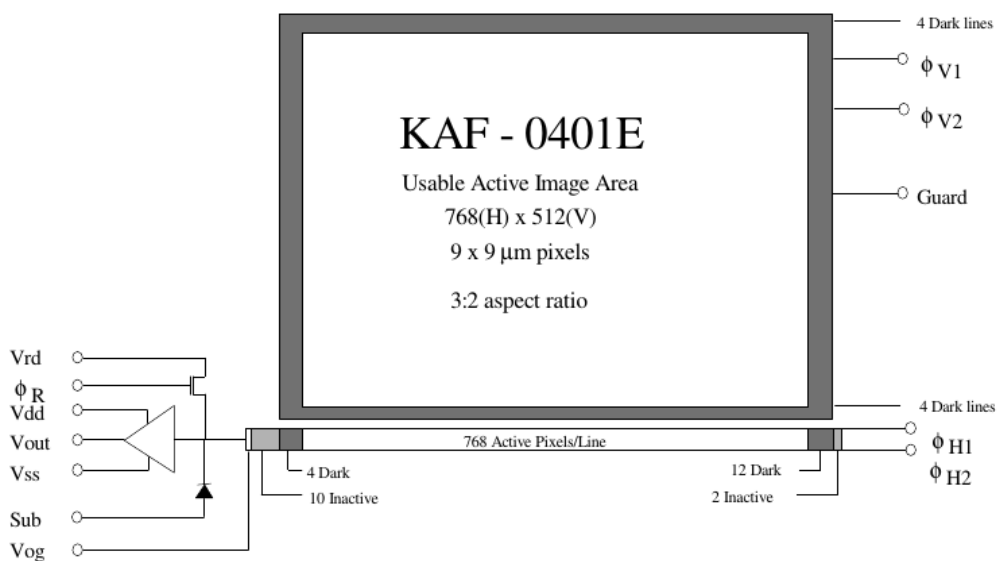


Figure 1 - Schéma du capteur CCD (cf. *datasheet* du CCD)

- 5) Analyser en détails les deux fonctions **fast_vidage(cam)** et **read_win(cam,p)**.

Expliquer sur votre compte-rendu les différentes actions réalisées par ces deux fonctions.

Ne pas oublier d'expliquer le rôle des sous-fonctions incluses dans ces deux fonctions notamment **zi_zh**, **read_pel_fast** & **read_pel_fast2**.

Pour chaque fonction, identifier les horloges actives et ce qu'elles permettent de contrôler sur le capteur et la chaîne électronique.

Remarque 1: La fonction **libcam_out()** est une fonction dont l'exécution permet de commander l'état des horloges du capteur et de la chaîne électronique. Son exécution constitue une base de temps Δt_0 pour décrire l'évolution des différents signaux d'horloges au cours du temps.

Remarque 2: Bien lire les commentaires au début du programme tp.c qui identifie la signification de l'octet (8 bits) utilisé par la fonction **libcam_out()**. Voir ci-dessous. Le **bit 1** est le bit de poids le plus faible (le plus à droite) alors que le **bit 8** est celui de poids le plus fort (le plus à gauche).

- * --- rappel des bits de données du port parallèle pour Audine
- * ordre : 87654321
- * bit 1 : horloge V1
- * bit 2 : horloge V2
- * bit 3 : horloge H1
- * bit 4 : horloge R (reset)
- * bit 5 : horloge CL (clamp)
- * bit 6 : horloge Start Convert (CAN)
- * bit 7 : horloge Select Byte (CAN)
- * bit 8 : horloge Select Nibble
- *
- * N.B. Si le bit est à 0 alors la tension est au niveau haut.
- * Si le bit est à 1 alors la tension est au niveau bas.

- 6) Dessiner les chronogrammes des différentes sous-fonctions de la fonction **fast_vidage** en respectant les durées temporelles de chaque palier.

Pourquoi est-ce que certaines commandes **libcam_out()** sont répétées plusieurs fois à votre avis ? (cf. Tableau – Section 3.4 dans la *datasheet* du Kodak AF400)

- 7) En déduire les chronogrammes créés par la fonction **fast_vidage**.
- 8) Construire les chronogrammes de la fonction **read_win**. En déduire comment le signal de sortie du CCD évolue au cours du temps. Vous pourrez vous aider du cours pour ce faire.
- 9) A quoi correspond le binning ? A quoi peut-il servir ?

Glossaires de quelques fonctions du langage C :

calloc : allocation dynamique de la mémoire et mise à zéro de toutes les valeurs.

sprintf : même usage que **fprintf** sauf que le résultat est copié dans une chaîne de caractères (char*) au lieu d'un fichier.

atoi : transforme une chaîne de caractères en un nombre entier de type int.

3. Utilisation du mode debug

Faire fonctionner le logiciel en compilant la librairie libaudine.dll en mode debug sous Visual C++. S'assurer d'être en mode debug. Demander l'aide de l'enseignant(e) si besoin.

1) Pour compiler la librairie libaudine : Menu <**Build**> et choisir <**Build audine.dll**>

Vérifier que la compilation s'est bien passée (0 errors) dans la zone de statut en bas de l'interface du compilateur.

2) Pour exécuter le programme : Menu <**Build**>, choisir <**Debug**> puis <**Go**>. Une boîte s'ouvre avec le message "C:\audela\binwin\audela.exe does not contain debugging information. Do you want to continue?". Répondre en cliquant sur <**Oui**>.

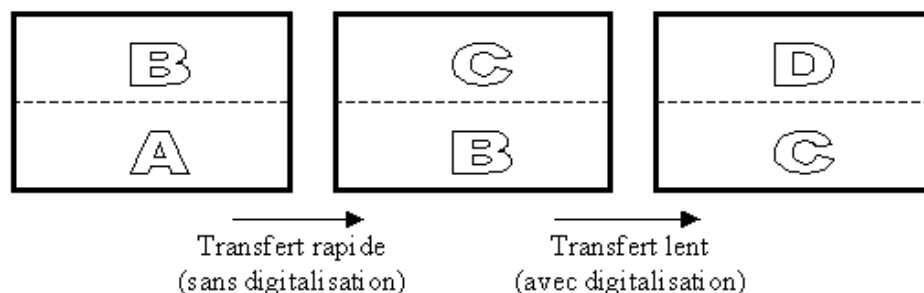
4. Création de la fonction spéciale

4.1 Effet de smearing

- 1) Faire une image du rond blanc sur fond noir sur un temps de pose de 1 s en binning 1x1. La caméra étant sensible à la lumière, la prise d'image se fera dans le noir avec la porte légèrement entrouverte.
- 2) Refaire la même image sur 10 s. Commenter la différence.
- 3) En déduire l'origine du smearing.

4.2 Fonction de lecture demi-trame

Nous désirons créer la fonction de lecture demi-trame. Ce mode d'acquisition consiste à vider sans digitalisation les N/2 premières lignes (partie A), puis à lire normalement les N/2 dernières lignes (partie B). De cette façon, nous pouvons réduire considérablement l'effet de smearing d'un objet brillant sur fond noir.



Principe de lecture du mode demi trame. Seule la partie B sera digitalisée.

Son transfert rapide vers la zone occupée précédemment par A permet d'éviter le smearing. Ce mode est bien adapté si la zone A n'est pas éclairée !

- 1) On commencera donc par écrire la fonction **cmdCamAcqSpecial**. Pour ce faire, recopier le contenu de la fonction **cmdCamAcqNormal** dans **cmdAcqSpecial**.
- 2) Modifier la fonction **cmdCamAcqSpecial** pour l'adapter aux besoins. De plus, on aura besoin d'une fonction assez proche de **read_win(cam,p)** mais un peu différente. On créera donc la fonction **read_win2(cam,p)** en recopiant la fonction **read_win** et en la modifiant.
- 3) Compiler le programme modifié. Expliquer sur votre compte-rendu ce que vous avez modifié en le justifiant.
- 4) Faire l'image du rond blanc sur fond noir sur 1 s en binning 1x1 (attention de bien positionner le rond blanc !) et vérifier ensuite que l'image ne montre pas de smearing.
- 5) Est-ce que cela fonctionne également lorsque le binning est changé ? Faire un essai en binning 2 x 2.
- 6) Quelle autre technique pourrait être utilisée pour supprimer l'effet de smearing ? Justifier votre réponse.
- 7) Sauvegarder les images prises dans un répertoire à vos noms sur le Bureau ainsi que le programme modifié à la fin de la séance de TP.