ELEKTOR

ELEKTOR

ÉLECTRONIQUE et MICRO-INFORMATIQUE

Nº 299
MAI 2003
mensuel
Belgique 6,05 €
Suisse 10,40 FS
Canada 7.70 SCan.
DOM surface 6,45 €
DOM avion 8,10 €
ISSN 0181-7450

www.elektor.fr

Interface USB → RS-232

Multiprise pilotée par RS-232

Echo & Réverb avec kits DSP

Noctilum

Picomire

CADAU

Mono-carte pour le XA

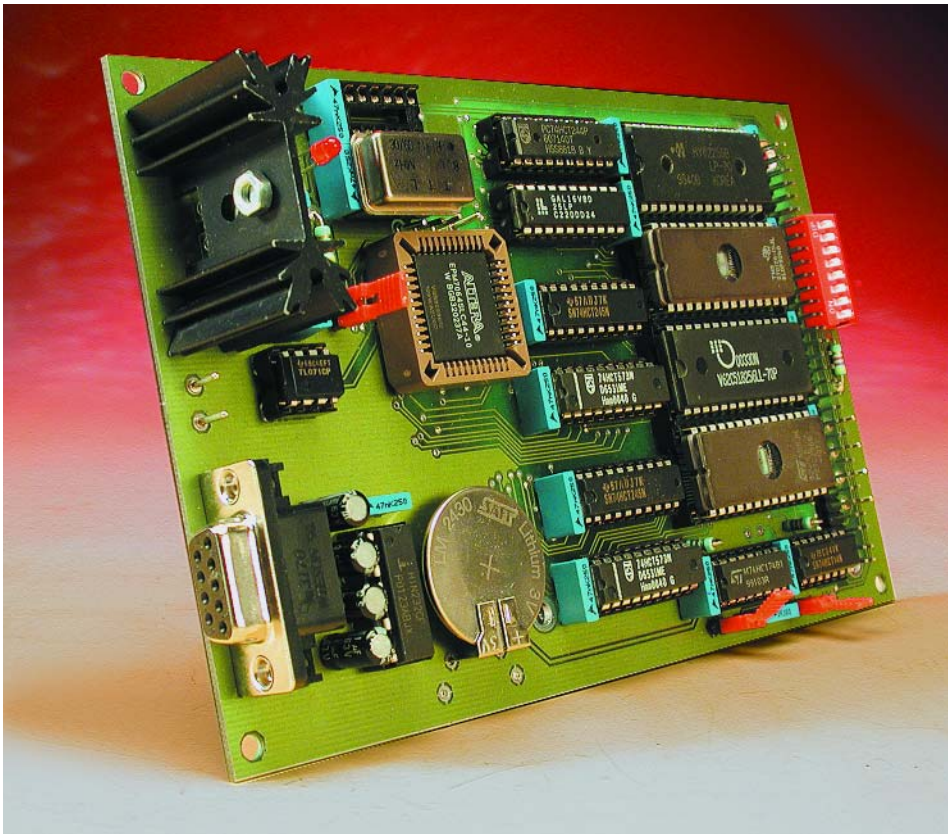
M 01531 - 299 - F: 5,45 €

CaDAU XA

Pour microcontrôleurs 16 bits XA (avec interface PC/104 !)

Benoît Bouchez

Les microcontrôleurs de la série XA ont été mis sur le marché par Philips Semiconductors à partir de 1995. Présentés à l'origine comme des versions 16 bits des célèbres 80C32, ces microcontrôleurs sont en fait bien plus qu'une simple évolution, car leur puissance de calcul est beaucoup plus importante que celle de leurs cousins à 8 bits.



Les XA : une grande famille

À l'origine, la famille des microcontrôleurs XA (pour *eXtended Architecture*) comptait... un seul membre, le XA-G3 (1996), qui disposait de 32 Koctets de ROM interne. Suite à diverses demandes, des versions avec moins de mémoire (et donc moins chères) furent mises sur le marché en 1997, les XA-G1 (8 Koctets) et XA-G2 (16 Koctets). Le G dans les différentes références indique la version Générique (et plus prosaïquement, les versions avec un brochage similaire aux 80C32).

L'année 1997 vit également la mise sur le marché du XA-S3 (le « Super-Chip » selon Philips...), équipé de nombreux périphériques intégrés (voir le **tableau 1**). On entendit également parler à cette époque du XA-SCC, qui se distinguait de ses frères par ses bus Adresses/Données non multiplexés, ainsi que du XA-D3 avec interface DeviceNet (DeviceNet est un bus industriel basé sur CAN). Le XA-SCC ne fut jamais mis en production (à notre connaissance). Quant au XA-D3, il fut remplacé à sa sortie par le XA-C3, dont le « C » indique la présence d'une interface CAN 2.0B, ce qui permettait d'ouvrir le marché aux autres couches applicatives de ce bus (CANOpen par exemple).

Les premiers processeurs de la famille XA portaient une référence P51XAxxxx, alors que les nouvelles versions sont repérées par PXAxxxx (sans le 51). Ce changement de référence est moins anodin qu'il n'y paraît : le noyau XA a en effet été partiellement modifié en vue d'amé-

NdlR : L'acronyme CaDAU permet de réduire très sensiblement la longueur d'un titre tel que : **Ca**rte de **D**éveloppement et d'**A**pplication **U**niverselle.

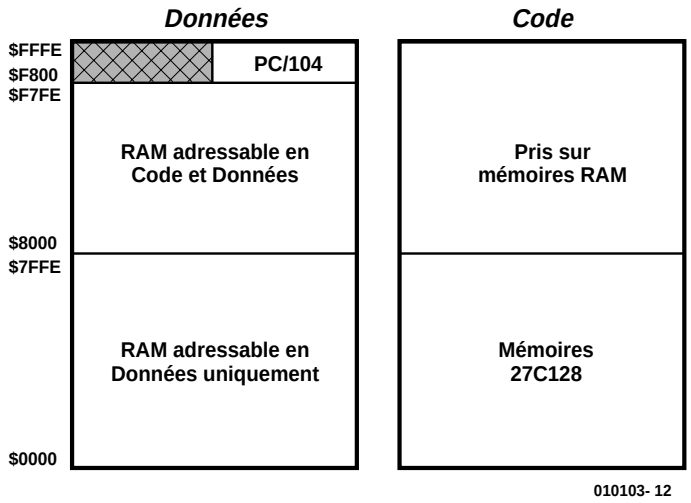


Figure 1 Cartographie mémoire de la carte de développement XA.

liorer ses performances entre temps, mais le code machine reconnu par ces deux versions est le même.
Très récemment, la famille XA s'est encore agrandie, avec l'arrivée des versions à Flash intégrée, appelées XA de type 4, avec des références du type XA-xx49 (les XA de type 3 utilisent une EPROM UV ou sont ROM-less - sans ROM interne).

Un jeu d'instructions puissant

Les XA ont été longtemps présentés comme des remplaçants des 80C32. En fait, les XA sont incapables d'exécuter le code binaire de leurs cousins 8 bits. Philips a en effet préféré revoir complètement le jeu d'instructions, et offrir une translation 80C3x vers XA au niveau source (il faut recompiler les programmes). Le résultat est que le noyau XA fonctionne beaucoup plus vite que celui de son aîné 8 bits à partir du même programme source (donc sans optimisation).
Justement, puisque nous parlons d'optimiser le code, il faut savoir que les XA ont une architecture orientée « registres » : au lieu de n'avoir qu'un seul accumulateur (comme les 80C3x), impliquant d'innombrables opérations de sauvegarde et de chargement, les XA disposent de 16 registres de 16 bits, numérotés R0 à R15, strictement équivalents au niveau du jeu d'instructions. L'ALU peut donc utiliser n'importe lequel de ces registres pour ses calculs. Moyennant une réécriture du code, on économise donc de nombreux cycles machines par rapport à un programme prévu pour des puces 8 bits.
Encore plus fort, les XA peuvent utiliser n'importe lequel de ces registres comme pointeur de données (la mémoire étant partitionnée en pages de 64 Koctets), alors que les 80C3x ne disposent que du registre DPTR (ce registre étant seul, les échanges de données en RAM peuvent devenir un véritable casse-tête à programmer !)
Nous pourrions nous étendre plus encore sur les possibilités de ces processeurs, mais cela demanderait l'ensemble des pages de la revue (et encore !). Aussi, nous ne saurions trop vous conseiller de télécharger les deux documents suivants sur le site de Philips : la fiche de caractéristiques du XA-G3 (ou du XA-C3), ainsi que le *XA User's guide*, qui est **LE** document de référence si vous voulez comprendre comment le noyau XA fonctionne.
Pour les chanceux qui peuvent accéder aux data-books « papier » de Philips, il faut commander le livre « *Data Handbook IC25* », dont le titre est « *16-bit 80C51XA Microcontrollers* ».
Il est important, pour bien comprendre comment fonctionne cette carte de développement, d'en connaître la cartographie de la mémoire, c'est ce qu'illustre le croquis de la **figure 1**.
Les données sont exprimées sur 16 bits, il n'y a donc que des adresses paires. Il est possible de stocker du code exécutable en RAM à par-

Tableau I.

Résumé des caractéristiques de quelques processeurs XA, comparées à celle d'un 80C32.

	80C32	XA-G3	XA-C3	XA-S3
Noyau	8 bits	16 bits	16 bits	16 bits
Architecture	Mono-accumulateur	Orientée registres	Orientée registres	Orientée registres
Pointeurs de code/données	1 pointeur 16 bits 2 pointeurs 8 bits	16 registres de 16 bits (+ registre de page)	16 registres de 16 bits (+ registre de page)	16 registres de 16 bits (+ registre de page)
Bus	Données 8 bits Adresses 16 bits	Données 8/16 bits Adresses 20 bits	Données 16 bits Adresses 20 bits	Données 16 bits Adresses 24 bits
ROM code interne	4 à 32 Koctets	32 Koctets	32 Koctets	32 Koctets
Mémoire code externe	Max 64 Koctets	Max 1 Moctet	Max 1 Moctet	Max 16 Moctets
Mémoire données interne	256 octets	512 octets	1 Koctet	1 Koctet
Mémoire données externe	Max 64 Koctets	Max 1 Moctet	Max 1 Moctet	Max 16 Moctets
Fréquence horloge	Jusque 32 MHz	Jusque 30 MHz	Jusque 30 MHz	Jusque 30 MHz
Cycles d'horloge par instruction	12 à 36	3 à 24	3 à 24	3 à 24
Périphériques intégrés	1 UART 3 timers	2 UART 3 timers	1 UART 1 port SPI 1 port CAN 2.0B 3 timers	2 UART 1 port I^2C 3 timers ADC 8 canaux 5 I/O PCA
Chien de garde	Non	Oui	Oui	Oui

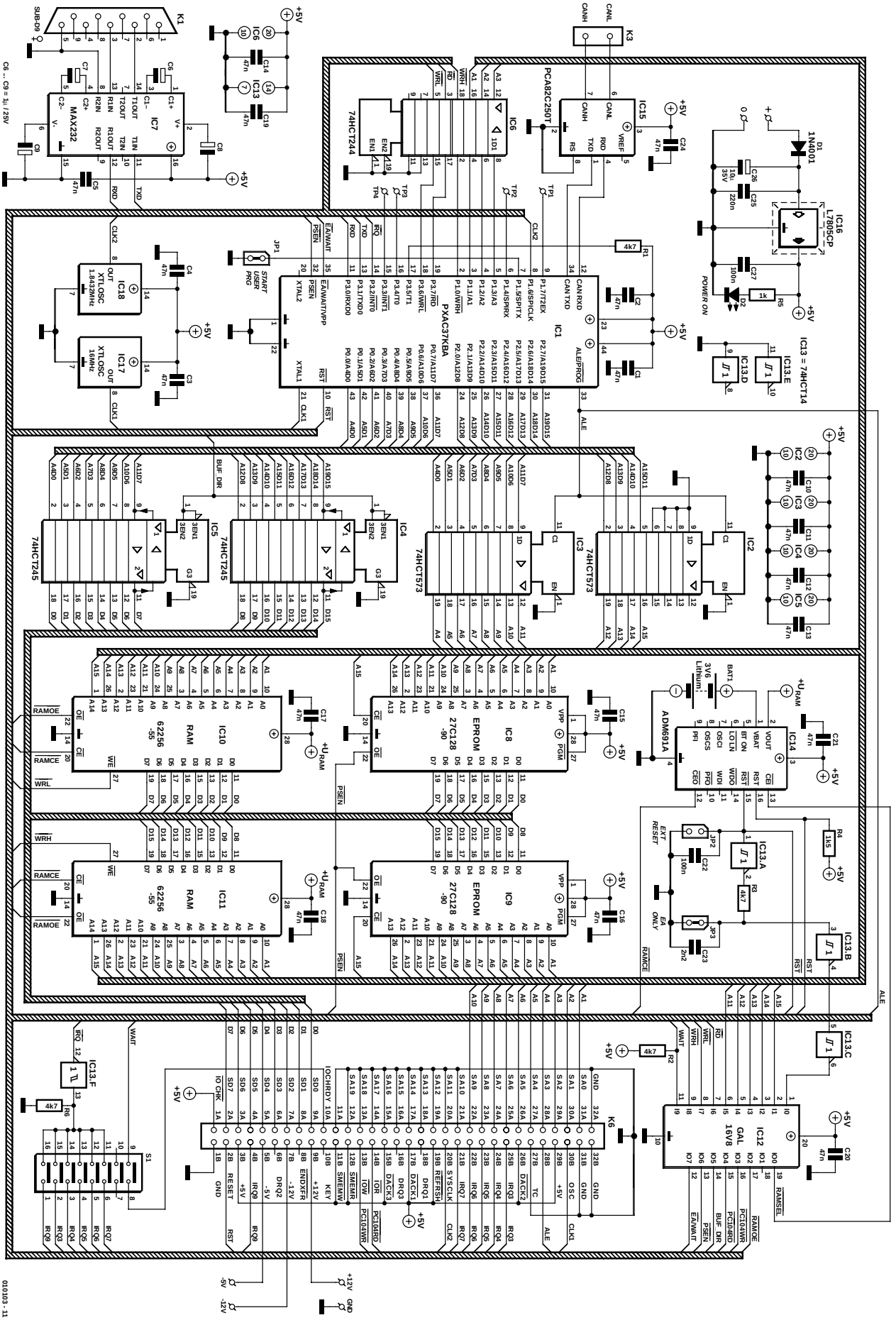


Figure 2. Sur le schéma de cette carte on retrouve la structure classique : microcontrôleur, mémoires et le reste des périphériques.

tir de l'adresse $8000, car cette zone est accessible à la fois en adressage Code et en adressage Données.

Le schéma

Un coup d'oeil sur le schéma (**figure 2**) montre que le qualificatif « d'outil professionnel » s'applique facilement à cette carte. Bien que le schéma soit dense, il reste malgré tout assez simple à comprendre.

Le coeur de la carte est bien évidemment un microcontrôleur XA (IC1), monté en configuration 16 bits de donnée/20 bits d'adresse. Le schéma est prévu pour accepter les XA-G3 et les XA-C3, sans avoir à modifier quoi que ce soit.

Le microcontrôleur est cadencé par un oscillateur DIP (IC17), relié à l'entrée XTAL1. Il est possible d'utiliser n'importe quelle fréquence, tant que celle-ci reste dans les limites acceptées par le micro.

Afin de rendre le débit de communication sur le port série intégré indépendant de la fréquence d'horloge principale, le moniteur configure le Timer 2 comme source de référence pour le port série, d'où la présence d'un oscillateur spécifique (IC18). Si vous n'utilisez pas le moniteur (programme personnel par exemple) ou si vous n'avez simplement pas besoin du port série, vous pourrez vous passer de ce deuxième oscillateur et récupérer ainsi la ligne P1.6.

Tant que nous sommes à parler du port série, passons par IC7, qui est un classique MAX232, chargé de convertir les niveaux TTL en V24 et réciproquement. Ce composant ayant déjà été décrit à de nombreuses reprises dans la revue, nous n'en dirons pas plus sur lui.

Passons maintenant au bus d'adresses, configuré ici en 20 bits (valeur par défaut au démarrage des XA). Vous remarquerez en examinant le schéma qu'il n'existe pas de ligne d'adresse A0, celle-ci étant remplacée par la ligne $\overline{WRL}$ en configuration 16 bits de données. Les adresses sont présentées sur les ports P0 et P2, multiplexées avec les données.

On retrouve donc ici deux démultiplexeurs d'adresse du type 74HCT573, IC2 et IC3, commandés par la ligne ALE du XA. Ces circuits sont également chargés de tamponner les lignes d'adresse, d'une part en raison du nombre important de circuits les utilisant (sans oublier l'interface PC/104), mais également pour protéger le micro en cas de problème sur le bus (court-circuit lors d'une expérimentation par exemple). On notera que les lignes A1 à A3 ne sont pas multiplexées sur les XA, ce qui permet de faire de l'adressage « Burst », diminuant le nombre de cycles d'horloge nécessaires. Comme pour les autres lignes d'adresse, un tampon (IC6) est chargé de protéger le microcontrôleur.

Le bus de données 16 bits est tamponné par IC4 et IC5, pour les mêmes raisons que pour le bus d'adresses. Les signaux de commande ($\overline{RD}$, $\overline{WRH}$, $\overline{WRL}$) sont quant à eux tamponnés par la seconde moitié d'IC6.

Les habitués des 80C3x auront remarqué le brochage similaire (hormis le bus 16 bits) du boîtier PLCC44 des XA-G3 et XA-C3 à celui de leur cousin 8 bits. Il faut cependant savoir qu'il existe certaines différences, subtiles certes, mais **très importantes**.

Première particularité : la configuration dynamique des bus. Les XA sont capables de fonctionner avec un bus de données 8 bits ou 16 bits, et un bus d'adresses sur 12, 16, 20 ou 24 bits. Bien évidemment, l'environnement du micro doit être adapté, car la disposition des lignes de bus est différente dans les deux cas. Notre carte étant configurée pour le mode 16 bits de données, le XA doit démarrer dans ce mode, mais comment le lui faire savoir ?

C'est le port P3.5 (alias T1/BUSW) qui est chargé de ce travail. Lors du Reset, le XA échantillonne cette broche. Si elle est à « 0 », le micro passe en mode 8 bits. Dans le cas contraire (« 1 » logique en entrée), le XA est configuré en mode 16 bits. La broche P3.5 est donc tirée au +5V par R1 pour cette raison. Une fois le micro démarré, la broche T1 peut à nouveau être utilisée comme E/S, mais durant la phase de Reset, il est **CAPITAL** qu'elle soit au niveau logique « 1 ». Notons que la fonction BUSW existe aussi pour le XA-C3, mais que ce micro ne fonctionne qu'en 16 bits !

Deuxième particularité : la broche $\overline{EA}$/WAIT. Comme sur les 80C3x, la broche $\overline{EA}$ indique au microcontrôleur que le code exécutable réside entièrement dans des mémoires externe (EA signifiant *External Adressing*). L'état de la broche $\overline{EA}$/WAIT est lu par le XA lors de la phase de Reset. Si un « 0 » logique y est appliqué, le processeur ira chercher le code programme uniquement dans une mémoire extérieure (les EPROMS IC8 et IC9 dans notre cas), même si le circuit installé est du type XA-G37 ou XA-C37 (avec ROM interne). Bien entendu, pour les XA-C30 et XA-G30, cette broche DOIT être mise à « 0 », puisque ces versions n'ont pas de mémoire programme interne.

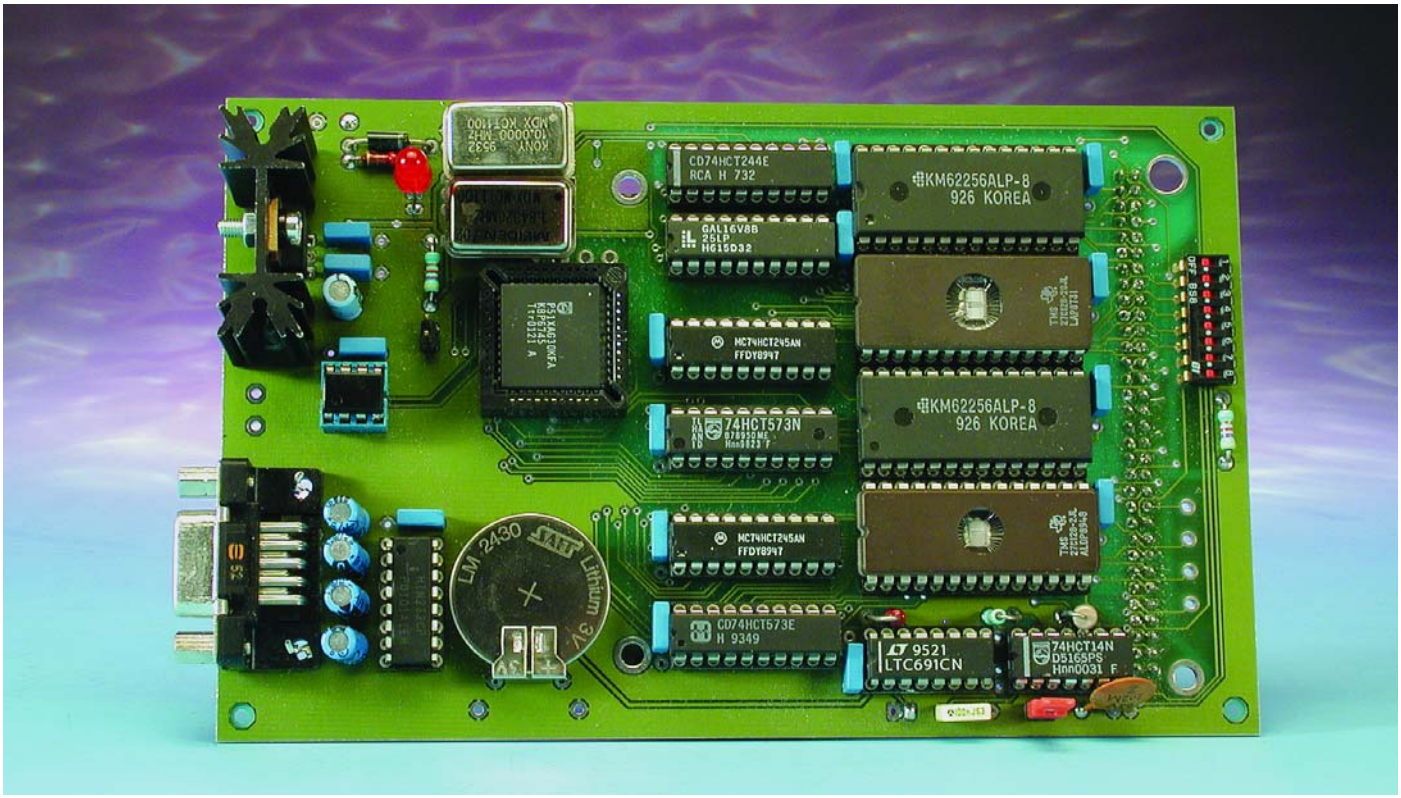
En revanche, lorsque cette broche est mise à « 1 » (toujours au Reset), le XA va commencer l'exécution du programme stocké dans sa ROM interne. Il ne fera appel aux mémoires externes que si l'adresse appelée dans le programme sort des limites supportées par le circuit (voir la fonction *Bus Disable* à ce sujet dans les documents Philips).

Le problème avec les XA est que la fonction $\overline{EA}$ partage la même broche que la fonction WAIT. Inconnue sur les 80C3x, cette fonction permet de figer le microcontrôleur lors d'un accès à un périphérique lent. Les 80C3x n'ayant jamais été des foudres de guerre au niveau de la vitesse des cycles de bus, le problème des périphériques lents était quasi inconnu sur ces processeurs.

À la différence de leurs cousins 8 bits, les XA disposent d'un pipeline susceptible d'aller chercher les instructions (« opcode fetch ») sur un seul cycle d'horloge. À pleine vitesse, un cycle de bus sur un XA cadencé à 30 MHz dure à peine 33 ns (66 ns si un cycle ALE est nécessaire). Or, très peu de périphériques supportent des temps d'accès aussi courts. Pour éviter tout problème avec des périphériques lents, il est possible de régler la vitesse des accès externes par les registres BTRH et BTRL (attention : sur le XA-C3, ces registres doivent être initialisés à une valeur standard. Utiliser en lieu et place les registres MIFBTRH et MIFBTRL), dans un facteur de compris entre 1 à 5.

Il peut toutefois être dommage de réduire la vitesse globale d'un système parce qu'un seul de ses périphériques est trop lent. C'est là que la fonction WAIT entre en scène. Lorsqu'un accès aux bus externes est exécuté, le XA surveille la broche $\overline{EA}$/WAIT. Si la broche reste à « 0 », le cycle de bus s'exécute de façon normale, à la vitesse programmée dans les registres BTRH et BTRL.

En revanche, si la broche WAIT est mise à « 1 » lors d'un accès extérieur, le XA va alors figer son cycle de bus tant que la broche reste à ce niveau. On peut donc allonger indéfiniment le cycle de bus, pour l'adapter aux périphériques les plus lents, et profiter des accès



à pleine vitesse avec des périphériques rapides.

Ce système est d'une efficacité redoutable, mais il est affecté d'un défaut. Prenons le cas d'un XA-C37 ou XA-G37 (à EPROM interne donc). Pour exécuter le programme interne, il faut que la broche $\overline{EA}$/WAIT soit mise à « 1 » au Reset. Tant que le XA accédera à sa mémoire interne, tout se passera bien (la broche WAIT n'a pas d'influence sur les accès internes). Mais au premier accès externe, le XA va contrôler cette broche : comme elle est à « 1 », le micro va se mettre en extension de cycle indéfiniment (tant que $\overline{EA}$/WAIT est à « 1 » en fait). Le système est alors bloqué.

Notre carte pouvant recevoir des XA avec EPROM intégrée, il a été nécessaire d'inclure un petit circuit spécifique, construit autour d'IC13 (portes A, B et C), destiné à maintenir la broche EA/WAIT à « 1 » durant quelques millisecondes après un Reset, le temps que le XA échantillonne la broche. Passé ce délai, la broche est ramenée à « 0 », pour éviter toute perturbation de la fonction WAIT. Soulignons au passage que si la fonction WAIT n'est pas nécessaire sur un système, il est possible de l'inhiber de façon logicielle (bit WAITD).

Si le processeur installé est un modèle ROM-less (ou si on veut forcer un XA-C37 ou un XA-G37 en adressage externe), il faut installer le cavalier JP3 pour forcer la ligne $\overline{EA}$ à « 0 » au Reset. La fonction WAIT n'est cependant pas perdue pour autant : elle est gérée en interne à la GAL IC12. Dans notre cas, cette fonction est reliée à la ligne IOCHRDY du bus PC/104, destinée justement au contrôle des périphériques lents. Signalons au passage qu'il est impératif que les cartes PC/104 installées sur le système maintiennent la ligne IOCHRDY à « 1 » durant les phases de Reset, faute de quoi le signal $\overline{EA}$/WAIT sera perturbé (un tel comportement serait anormal et non conforme aux spécifications PC/104, mais la situation s'est déjà présentée sur un de nos systèmes, autant donc prendre les devants). Si vous avez un doute sur le comportement d'une carte PC/104 à ce niveau, il est possible d'isoler le signal IOCHRDY par le cavalier S1-8.

Puisque nous parlons des cavaliers, intéressons-nous à JP1, connecté à P1.5. Il s'agit ici non pas d'une particularité des XA, mais tout simplement d'une astuce maison pour étendre les possibilités d'utilisation de la carte de développement. Lorsque le moniteur (celui que nous avons écrit) XA-G3 démarre, il teste cette ligne. Si elle est à l'état haut (rappel par une résistance de forçage au niveau haut (*pull-up*) interne au XA), le moniteur prend la main et se met en mode dialogue, afin de pouvoir traiter des commandes externes.

Si JP1 (START USER PRG) est installé (P1.5 à la masse), le moniteur va se désactiver et sauter immédiatement au vecteur de Reset utilisateur, que nous avons placé à l'adresse $8000 (c'est-à-dire en RAM), pour commencer à exécuter le programme utilisateur en mode autonome.

Bien entendu, si la carte est équipée avec des EPROMs qui ne contiennent pas notre moniteur mais une application dédiée, le cavalier JP1 n'a aucune utilité et le port P1.5 est alors utilisable normalement.

JP2 n'est pas un cavalier, contrairement à ce que pourrait laisser croire le schéma, mais un point de connexion pour un poussoir externe de Reset (EXT RESET, le LTC691 n'étant pas muni de cette fonctionnalité). Signalons en passant que la fiche de caractéristique du LTC691 préconise la mise en série d'une résistance de 100 Ω avec le bouton poussoir, non pas pour limiter le courant de court-circuit, mais pour éviter les oscillations. Bien que cette résistance n'apparaisse pas sur le schéma, nous vous incitons à l'installer si vous décidez de monter un poussoir externe de Reset.

Attention : l'installation d'un poussoir de Reset extérieur sur JP2 n'a d'effet que sur le XA, pas sur la carte

Tableau 2.

Principales caractéristiques de CaDAU XA.

– Compatible XA-G3x et XA-C3x
– Compatible versions EPROM (XA-G37 / XA-C37) et ROMLess (XA-G30)
– Bus de données 16 bits
– Moniteur intégré en EPROM
– 32 Kmots 16 bits pour programme en EPROM
– 32 Kmots 16 bits de données
– 32 Kmots 16 bits données/programme pour déverminage du code
– Interface PC/104 en mode I/O, permettant de recevoir des cartes d'interface au standard PC/104 (E/S logiques et analogiques, réseau, etc.)
– Code exécutable en RAM
– Adressage en page zéro (16 bits) pour une vitesse maximale du noyau XA
– Possibilité d'utilisation en carte de développement et en carte d'application
– Contenu des RAM sauvegardé par pile

PC/104 connectée au système. Pour obtenir une réinitialisation complète (XA+PC/104), il est nécessaire de provoquer une coupure d'alimentation de la carte. C'est une bizarrerie du LTC691, nous n'y pouvons rien.

Passons maintenant à l'interface PC/104. Le XA n'étant pas un processeur x86, il était impossible d'implémenter toutes les fonctionnalités du standard PC/104. On ne dispose ici que des accès en mode I/O (entrées/sorties), sur les 2 048 premiers octets. Les spécialistes du PC objecteront que seuls les 1 024 premiers octets sont utilisés sur les PC, mais il faut savoir que l'on trouve des cartes PC/104 adressables sur une gamme plus étendue, d'où notre choix.

Comme les XA ne disposent pas d'un mode d'adressage spécifique des Entrées/Sorties (I/O), l'accès au bus d'extension se fait à travers une fenêtre située entre $F800 et $FFFF dans l'espace Données du processeur. L'accès au bus PC/104 est contrôlé par les signaux $\overline{\text{PC104WR}}$ et $\overline{\text{PC104RD}}$, généré par la GAL IC12. Attention : seul l'octet de poids faible est transféré à l'interface PC/104 !

La carte ne supporte pas les accès DMA (faute de ressources adaptées sur les XA), mais permet en revanche le traitement des interruptions IRQ2 (nommée IRQ9 dans certains documents de référence) à IRQ7 du bus PC/104. Bien que le XA dispose de deux entrées d'interruptions externes ($\overline{\text{INT0}}$ et $\overline{\text{INT1}}$), une seule d'entre elle a été affectée au PC/104, le choix du canal se faisant par S1. Noter qu'avec cette approche, il ne peut y avoir qu'une seule carte PC/104 installée avec support des interruptions (le nombre de cartes sans interruptions n'étant limité que par la capacité des tampons de bus). La polarité des signaux d'interruption sur PC/104 étant l'inverse de celle utilisée par les XA, la porte IC13.F se charge de faire rentrer les choses dans l'ordre.

Les accès en mode Mémoire au bus PC/104 ne sont pas possibles, ce qui n'est qu'un inconvénient mineur, la plupart des cartes d'extension PC/104 n'utilisant que les accès I/O. Néanmoins, avant de choisir et d'installer une carte PC/104 sur la carte de développement, il vous faudra vérifier ce point, afin d'éviter tout problème ultérieur.

Pour terminer sur l'interface PC/104, soulignons que notre carte ne peut fournir que la tension +5 V sur le connecteur. Certaines cartes PC/104 peuvent nécessiter des tensions de +12 V, –5 V et –12 V. Ces tensions n'étant pas utilisées, ni générées sur la carte de développement, nous avons prévu des points d'entrée (K4 et K5), qui devront être reliés à des sources extérieures si besoin est.

Revenons sur IC14, qui est chargé principalement de générer l'impulsion de Reset lors de la mise sous tension de la carte. Pour ce faire, il utilise un comparateur interne, qui détecte le passage de la tension d'alimentation en dessous de 4,75 V. Lorsque cette situation se présente, les signaux Reset et $\overline{\text{Reset}}$ sont générés automatiquement, jusqu'à ce que la tension repasse au dessus de 4,75 V. En pratique, l'impulsion de Reset est maintenue 35 ms après stabilisation de la tension.

IC14 contient également un circuit très utile de commutation de source d'alimentation, qui permet de maintenir sous tension les mémoires RAM, via la pile de 3,6 V, BAT1. La commande de ce circuit est entièrement automatique, dès détection d'une tension d'alimentation incorrecte.

Parallèlement à cette fonction de commutation de source, IC14 se charge également d'inhiber les signaux de validation des RAM, lors des phases de Reset, afin d'éviter toute corruption de leur contenu.

Grâce à ce circuit, il est possible de mémoriser un programme ou des données en RAM, et de les conserver hors tension. Le moniteur contrôlant la ligne P1.5 au démarrage, le système peut donc être rendu totalement autonome, sans nécessiter la programmation spécifique des EPROMS IC8 et IC9, ce qui s'avère d'une très grande utilité lors des phases de mise au point de programmes. Bien entendu, dans le cas où· la carte est utilisée en tant que carte d'application (avec un programme spécifique en EPROM), ce système est utilisable pour permettre, par exemple, de maintenir des données spécifiques à une application en RAM.

Les circuits de la famille LTC691 sont munis d'autres fonctionnalités, comme un chien de garde intégré et des entrées/sorties Power-Fail, mais elles ne sont pas utilisées sur cette carte. Signalons pour terminer que ce circuit existe sous plusieurs dénominations telles que ADM691, LTC691, MAX691, etc. Le circuit IC15 est un adaptateur TTL/CAN, qui ne se justifie que si la carte est équipée d'un XA-C3 (le XA-G3 n'a pas d'interface CAN).

Dernière partie du schéma et non des moins importantes : l'alimentation. La carte se contentant d'une seule tension d'alimentation, un simple régulateur de la famille 7805 fait le plus gros du travail. La diode D1 est là pour protéger le circuit contre les inversions de polarité, et non pas pour permettre l'alimentation du circuit par une source alternative. Insistons lourdement sur le fait que le circuit est équipé de nombreux condensateurs de découplage (un par circuit intégré), et que leur présence est absolument indispensable. Bien que le schéma spécifie des valeurs de 47 nF, on peut utiliser sans crainte des 100 nF, parfois plus faciles à trouver.

Nous voici arrivés à la fin de cette description du schéma. Dans l'article du mois prochain nous passerons à la partie la plus intéressante de ce montage, sa réalisation et son utilisation. Pour vous mettre l'eau à la bouche nous vous proposons, dans le **tableau 2**, un résumé succinct des caractéristiques les plus importantes de CaDéAU XA.

(010103-I)

Effets d'écho et de réverb avec kits d'évaluation DSP

Théorie et pratique avec les cartes-filles audio et les processeurs TMS320C5402 et TMS320C6711 de Texas Instruments

Richard Sikora

De nos jours, la technique de l'audio (numérique) n'hésite pas à utiliser les effets d'écho et de réverbération à toutes les sauces. Le présent article a pour but d'en décrire l'implémentation à l'aide de cartes DSP standard de source TI et d'aborder la programmation des effets en C.

La carte DSP avec le processeur de signal TMS320C6711 de Texas Instruments.

Note : Le code de programme en C a été écrit en 2 versions, l'une pour un processeur travaillant en virgule fixe l'autre pour un processeur travaillant en virgule flottante. Ces codes (en allemand) sont à votre disposition pour téléchargement depuis notre site Internet.
On utilise souvent les effets d'écho et de réverbération pour obtenir, dans un studio d'enregistrement ou dans une petite salle, les mêmes effets que ceux produits par une salle de concert ou un auditorium. Nous allons nous intéresser, dans les paragraphes qui suivent, aux mécanismes servant à produire l'écho et la réverbération dans le cadre du traitement numérique de signaux (DSP = *Digital Signal Processing*).

Le logiciel écrit en C implémente les effets décrits à condition d'utiliser les kits d'évaluation (stater kit) DSP TMS320C5402 DSK (processeur à virgule fixe) et TMS320C6711 DSK (processeur à virgule flottante) en association avec une carte-fille audio. Au niveau de cette carte compatible avec les cartes DSP il s'agit d'un Codec Audio stéréo à base de PCM3003 de Burr-Brown.

L'effet d'écho

La **figure 1** montre le principe (simplifié) utilisé pour produire un écho.

Le rectangle représente les murs d'une salle de concert ou d'un auditorium. Le son arrive à l'auditeur par 2 voies différentes. Tout d'abord lui arrive le son en provenance directe de l'instrument de musique ou des enceintes. Simultanément, le son rebondit sur les murs et arrive, très peu de temps après, aux oreilles de l'auditeur; on parle de réflexion. Le niveau (puissance) du signal réfléchit est légèrement inférieur à celui du son direct, différence due à l'atténuation induite par frottement sur le mur.

Comme la vitesse du son est (relativement !) faible, de l'ordre de 343 m/s dans l'air à 20 °C, l'auditeur entendra tout d'abord le son venant directement de la source sonore et très peu de temps après (quelque 100 ms plus tard par exemple), le même son une seconde fois. Une petite pièce est caractérisée normalement par un petit écho de quelque 50 ms seulement. À l'inverse, si l'on se trouve dans une large vallée encaissée, le son peut avoir besoin de quelques secondes avant de revenir aux oreilles de l'auditeur, sous la forme d'un écho, après réflexion sur la paroi rocheuse d'en face.

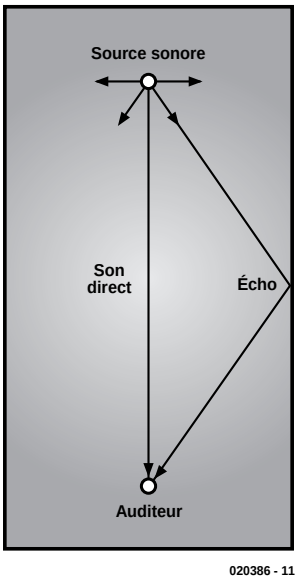


Figure 1. Principe de création d'un écho.

la valeur de mesure instantanée générée à cet instant par le CAN (**C**onvertisseur **A**nalogique/**N**umérique). La valeur du signal de sortie calculée servant à la génération du signal audio est transmis, par le biais du CNA (**C**onvertisseur **N**umérique/**A**nalogique), à l'amplificateur voire directement aux enceintes (actives). Le symbole de somme Σ implique une addition.
Mathématiquement, le concept illustré en figure 2 peut s'exprimer à l'aide de l'équation suivante :

Signal de sortie = signal d'entrée + signal d'entrée retardé

Il faudra, si l'on veut créer un écho réaliste, que le signal retardé ait subi une atténuation de quelque 15% de manière à simuler l'atténuation introduite par la réflexion du son.

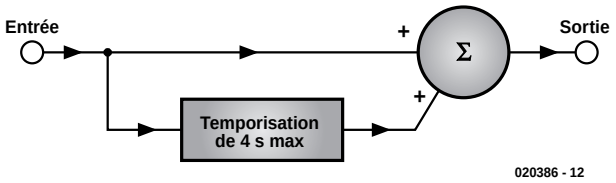


Figure 2. On reproduit un écho par addition au signal d'entrée d'un signal retardé.

Exemple 1.

```
int buffer[8] = {0,0,0,0,0,0,0,0};
```

Implémentation logicielle d'un écho

Il faut, si l'on veut implémenter un écho logiciellement, ajouter au signal d'entrée une version décalée du dit signal. C'est ce processus que schématise le croquis de la **figure 2**.
Le signal d'entrée prend la forme de

Temporisation induite par tampon

Si, dans la réalité, un tel tampon de retardement se devait de comporter plusieurs milliers d'éléments, nous allons ici, pour ne pas trop compliquer les choses, utiliser un tampon simple ne comportant que 8 éléments. **L'exemple 1** montre comment implémenter un tampon en C. Le tampon est implémenté comme une matrice (*array*) de 8 mots; il a reçu la dénomination `buffer`.
Les éléments du tampon sont numérotés de 0 à 7. Le premier élément du tampon a été baptisé `buffer[0]`, le second `buffer[1]` et le dernier `buffer[7]`. La liste `{0,0,0,0,0,0,0,0}` indique que chacun des éléments s'est vu attribuer un « 0 » comme valeur de départ.
L'une des techniques permettant d'utiliser le tampon en module d'introduction de retard consiste à mémoriser la valeur de mesure la plus récente (échantillon, *sample*) à l'une des extrémités du tampon, la valeur la plus ancienne étant elle stockée dans l'emplacement situé à l'autre extrémité. Nous pourrions ainsi, par exemple, utiliser `buffer[0]` pour l'échantillon le plus récent et `buffer[7]` pour l'échantillon le plus ancien. Cette disposition est connue sous la dénomination de *straight buffer* (tampon en ligne).
À chaque fois que le CAN effectue une mesure, on a placement d'une nouvelle valeur dans le tampon `buffer[0]`. Nous éjectons la valeur de mesure la plus ancienne du `buffer[7]` et décalons toutes les valeurs d'une position. Ainsi, la valeur de `buffer[6]` aura été déplacée vers `buffer[7]`.
La raison physique du retard ainsi introduit tient au fait qu'il faut un certain temps pour permettre à une valeur de mesure de passer de l'entrée du tampon à sa sortie. De ce fait, le signal arrive à la sortie en ayant subi un certain retard. Les valeurs de mesure les plus récentes se décalent progressivement vers la sortie. Ce processus est représenté schématiquement en **figure 3**. Nous supposons que le tampon comprend déjà les valeurs 1, 2, 3, 4, 5, 6, 7 et 8.
Cette disposition ne convient malheureusement pas pour une implémentation logicielle d'effets d'écho et de réverbération. Si le tampon est de bonne taille et qu'il comporte, par exemple, plusieurs milliers de valeurs, le programme passe la majeure partie de son temps à transférer les données d'un endroit à l'autre. Il ne reste tout simplement pas suffisamment de temps pour le traitement des signaux audio. Une autre approche, plus adaptée, est le *circular buffer* connu sous nos latitudes comme le tampon en anneau.

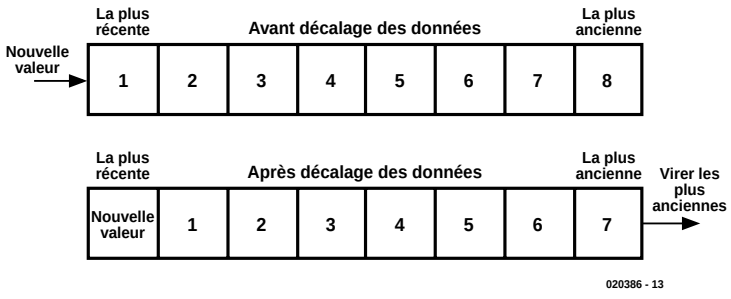


Figure 3. Stockage des valeurs dans un tampon en ligne.

Tampon en anneau

Lorsque, dans le cas de la figure 3, on ajoute une nouvelle valeur, toutes les valeurs plus anciennes subissent un décalage vers l'avant, la plus ancienne étant éjectée. Le principe d'un tampon en anneau est différent.
Les valeurs présentes dans le tampon ne

Exemple 2. Tampon en anneau en C

```
int buffer[8];              /* Delay buffer */
int temp ;                  /* Temporary storage for oldest */
unsigned int x = 0;         /* Retain value between calls */

temp = buffer[x];           /* Read oldest value from buffer*/
buffer[x] = input;          /* Over-write with newest */

if ( x < 7 )                /* Test for end of buffer */
   {
     x++;                   /* Increment pointer */
   }
else
   {
    x = 0;                  /* Go back to beginning */
   }
```

subissent pas de décalage. Au lieu de cela, l'élément le plus ancien est purement et simplement écrasé par le plus récent. Ceci signifie que la position de l'élément le plus ancien du tampon ne cesse de se déplacer au fur et à mesure de la mémorisation d'un nouvel élément. Le suivi de la position de l'élément le plus ancien se fait à l'aide d'un pointeur.

La **figure 4** illustre le principe de fonctionnement d'un tampon en anneau. Nous avons à nouveau supposé que le tampon contenait déjà les valeurs 1, 2, 3, 4, 5, 6, 7 et 8. L'adjonction d'une nouvelle valeur au tampon se traduit par un écrasement de la valeur la plus ancienne et le décalage du pointeur d'une position. À l'inverse de ce qui se passe dans le cas d'un tampon en ligne toutes les autres valeurs restent à leur place (et partant ne subissent pas de décalage).

On voit en **figure 5** ce qui se passe lorsque l'on arrive en fin du tampon. À nouveau on a écrasement de la valeur la plus ancienne par la valeur la plus récente. Dans ce cas-là le pointeur est repositionné au début du tampon, ce qui fait de lui un tampon pseudo-circulaire.

Tampon en anneau en C

En cas d'utilisation d'un tampon en anneau on n'a pas de déplacement des données. Le seul changement se situe au niveau de la position du pointeur.

L'exemple 2 donne l'implémentation et l'utilisation, en code C, d'un tampon en anneau. Dans l'exemple le pointeur a été implémenté en index pour le champ `buffer[8]` et s'est vu attribué le nom x.

En vue d'incrémenter la valeur de x nous commençons par voir si la valeur x est inférieure à la valeur maximale de 7. Si cela est le cas, on pourra incrémenter la valeur x. SI au contraire, la valeur x se trouve déjà à sa valeur maximale de 7, il faudra donner à x la nouvelle valeur de zéro.

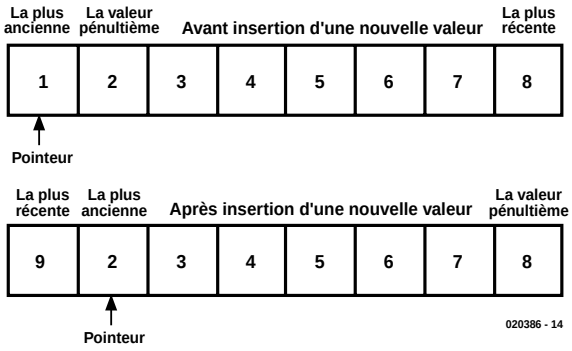


Figure 4. Stockage des valeurs dans un tampon en anneau.

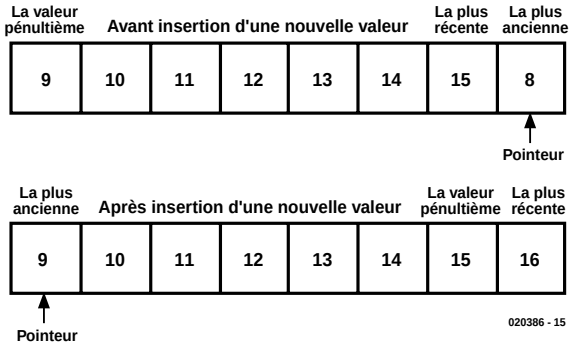


Figure 5. Lorsque l'on arrive à la fin d'un tampon en anneau, le pointeur retourne au début du tampon.

Fonction d'écho en C

Pour l'implémentation d'un écho en code C il faut commencer par désigner un tampon de temporisation. La taille du tampon dépend de 2 facteurs : le taux d'échantillonnage du CAN et de la durée de retard. Un exemple : la carte-fille audio échantillonne le signal d'entrée 24 000 fois par seconde. Il nous faudra partant, si nous voulons un retard (*delay*) de 4 secondes, disposer d'une taille de tampon de 24 000 x 4 = 96 000.

L'exemple 3 donne le code C requis pour l'implémentation de la fonction `echo()`. La valeur de mesure fournie par le CAN est le premier paramètre à être pris en compte, la valeur de sortie est la somme de la valeur d'entrée et de l'écho ajouté. Nous commençons par la définition du tampon. Nous utilisons pour cela la fonction `#define BUFFER_SIZE` de manière à pouvoir modifier facilement la taille du tampon de retard.

Le pointeur du tampon de retard a été baptisé x. Comme cette variable est toujours positive, elle a été déclarée en *unsigned* (non signé). Le mot-clé `static` signifie que la valeur est conservée lors de retour de la fonction. En l'absence de ce mot-clé, x se trouverait toujours à zéro.

La valeur la plus ancienne est graduée vu qu'elle subit une multiplication par l'équivalent de 0,8. Des processeurs à virgule fixe tels que le TMS320C5000 ne peuvent travailler qu'avec des entiers. Dans ce cas

Exemple 3. Implémentation d'une fonction d'écho en langage C

```
#define ECHO_CONSTANT 26214       /* Percentage of delayed signal */
#define BUFFER_SIZE 96000         /* Size of delay buffer */
int buffer[BUFFER_SIZE];          /* Declare delay buffer */

int echo(int input)               /* Echo function */
{
 static unsigned int x = 0;       /* Pointer to delay buffer */
 int temp;
 long output;

 if ( x < BUFFER_SIZE-1)          /* Test for end of buffer */
   {
     x++;                         /* Increment pointer */
   }
 else
   {
     x = 0;                       /* Back to beginning of buffer */
   }

 output = (long) temp *           /* Use 80% of delayed signal */
                ECHO_CONSTANT;
 output >>= 15;                   /* Remove fractional part */
 output += input;                 /* Add input to delayed input */
 return ( (int) output);          /* Return input + delayed input */
}
```

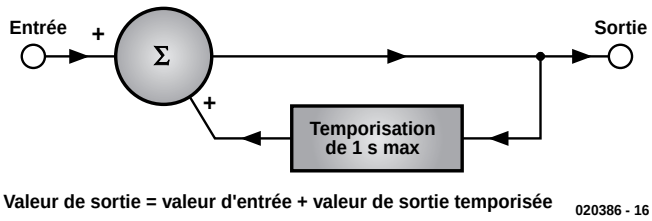


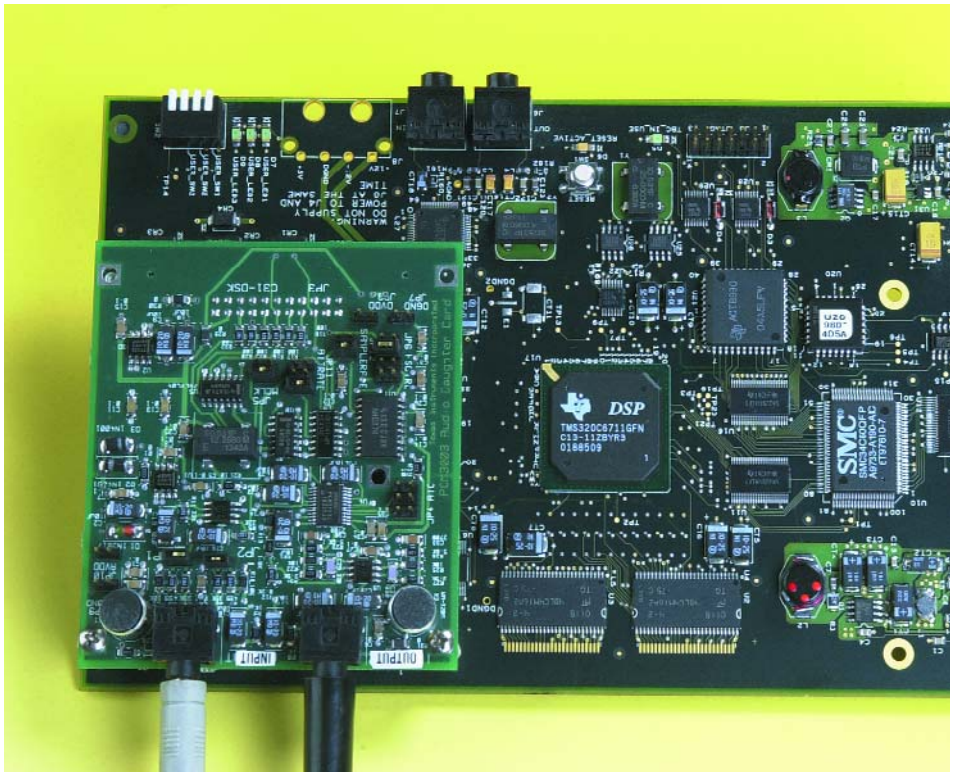
Figure 6. Pour la réverbération, c'est le signal de sortie temporisé est réinjecté à l'entrée.

Comme le signal de sortie output peut dépasser la valeur maximale de 32767 ou tomber en deçà de la valeur minimale de –32768, nous allons limiter la plage des valeurs en divisant la valeur par 2. Nous avons utilisé pour cela un décalage vers la droite et non pas une division de manière à être assurés que le compilateur procédera bien à une opération de décalage et qu'il ne fera pas appel à une fonction de librairie, plus lente.

nous pouvons remplacer le nombre 0,8 par un entier qui sera lui-même divisé par un autre entier, opération dont le résultat sera 0,8 :

0,8 = 26 214 / 32 768

Ainsi, pour multiplier un nombre entier par 0,8, nous commençons par multiplier ce nombre par 26 214 avant de diviser le résultat par 32 768. Il est important, en vue de simplifier l'opération, que le diviseur soit une puissance de 2 (2, 4, 8, 16, 32, 64, etc.). Dans la pratique, une division par 32 768 (2^{15}) est extrêmement simple; il suffit de décaler le produit de 15 positions vers la droite.



La carte-fille audio implantée sur la platine DSP à TMS320C6711 intègre un Codec (CAN/CNA) et comporte, outre une entrée ligne stéréo, une paire de capsules micro à électret.

Exemple 4. Implémentation d'une fonction de réverbération en C

```
#define ECHO_CONSTANT 26214      /* Percentage of delayed signal */
#define BUFFER_SIZE 2400         /* Size of delay buffer */
int buffer[BUFFER_SIZE];         /* Declare delay buffer */

int reverberation(int input)     /* Reverberation function */
{
 static unsigned int x = 0;      /* Pointer to delay buffer */
 int temp;
 long output;

 temp = buffer[x];               /* Make copy of oldest value */

 output = (long) temp *          /* Use 80% of delayed signal */
                 ECHO_CONSTANT;
 output >>= 15;                  /* Remove fractional part */
 output += temp);                /* Add input to delayed input */

 if ( output > 32767)            /* Test for greater than maximum */
   {
     output = 32767;             /* Limit positive output */
   }
 else if ( output < -32767)      /* Test for less than minimum */
   {
     output = -32767;            /* Limit negative output */
   }

 buffer[x] = (int) output;       /* Over-write oldest with newest */

 if ( x < BUFFER SIZE-1)         /* Test for end of buffer */

   {
     x++;                        /* Increment pointer */
   }
 else
   {
     x = 0;                      /* Back to beginning of buffer */
   }

 return ( (int) output);         /* Return input + delayed input */
}
```

Principe de l'effet de réverbération

Le mécanisme d'un effet de réverbération diffère quelque peu de celui d'un écho. Prenons l'exemple de quelqu'un qui, sur une scène, parle dans un micro. Le son subit dans ce cas-là une réflexion sur le mur situé derrière l'orateur avant de revenir vers la scène. Le microphone capte ce son réfléchi en même temps que le son direct de l'orateur. Le peut dans ces conditions suivre le même chemin et si le gain de cette boucle est trop important l'amplitude ne cesse d'augmenter ce qui se traduit très rapidement par une réaction acoustique : le fameux effet de Larsen qui se manifeste sous la forme d'un hurlement strident dû à l'entrée en oscillation du système en raison de cette réaction.

Dans le principe de l'effet de réverbération représenté schématiquement en **figure 6** ce n'est plus à la valeur d'entrée que l'on fait subir un certain retard introduit par un tampon, mais à la valeur de sortie. Cette valeur de sortie est, après cette temporisation, additionnée au signal d'entrée. Vu que de ce fait le signal de sortie est réinjecté plusieurs fois à l'entrée la durée de retard doit être sensiblement plus courte que dans le cas d'un effet d'écho. Partant, l'effet de réverbération sera la configuration préférentielle lorsque la taille de la mémoire de données (RAM) que l'on pourra consacrer au tampon se trouve être limitée. En cas d'utilisation de cette approche dans laquelle il s'agit, ne l'oublions pas, d'une réinjection (même si c'est celle du signal retardé), il faut travailler avec soin sachant que sinon le système risque de devenir, comme dans le cas de la réaction acoustique, instable.

Fonction de réverbération en C

L'exemple 4 donne le code C permettant d'implémenter une fonction de réverbération. La différence se trouve ici dans le fait, nous le disions plus haut, que c'est le signal de sortie et non le signal d'entrée qui subit une temporisation. Le tampon sera en outre moins long. Pour éviter toute instabilité la plage des valeurs du signal de sortie va, pour la réverbération, de -32767 à +32767.

Téléchargements

Nous proposons, pour le processeur TMS320C5402 DSK avec carte-fille audio, un exemple de code écrit en C au téléchargement depuis notre site (www.elektor.fr). Cela est également vrai dans le cas du TMS320C6711 DSK avec carte-fille audio. Le fichier comportant les 2 exemples de programme sont à trouver sous la rubrique TELECHARGEMENTS du numéro dans lequel cet article est publié; sa dénomination est **EPS020386-11**.

En guise de conclusion

Nous avons vu, dans le présent article, comment implémenter de l'écho et de la réverbération dans le cas des processeurs TMS320C5402 DSK et TMS320C6711 DSK, et ceci en langage C. La réverbération requiert moins de mémoire mais présente l'inconvénient potentiel d'une instabilité système. Les configurations illustrées par les figures 2 (écho) et 6 (réverbération) sont très importantes dans le cas d'un traitement numérique de signal. Les deux sont utilisées pour l'implémentation de filtres numériques. Si l'on diminue la durée de retard de la figure 2 nous obtenons un filtre FIR (*Finite Impulse Response*). Avec le concept de la figure 6, le choix d'une durée de retard plus courte donne un filtre IIR (*Infinite Impulse Response*).

(020386)

Littérature

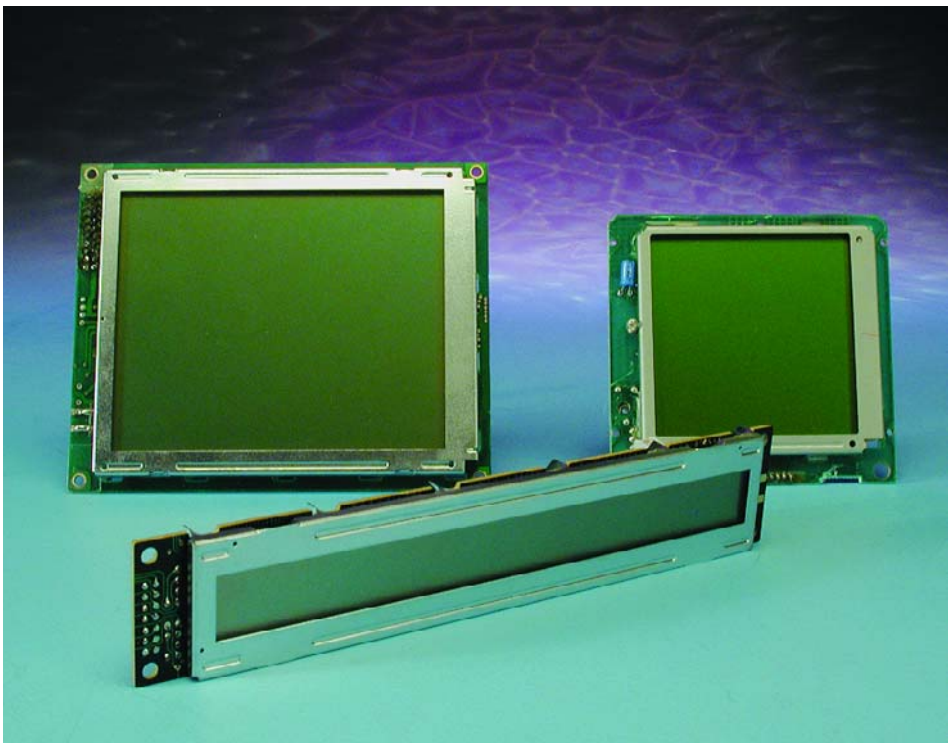
A Digital Signal Processing Primer with Applications to Digital Audio and ComputerMusic.
Ken Steinglitz,
ISBN 0-8053-1684-1

Pilotage de LCD

1ère partie : un LCD : ça marche comment ?

Wim Huiskamp

Sur le marché des occasions, on trouve régulièrement des modules matriciels à cristaux liquides (LCD). Il s'agit parfois d'afficheurs flambant neufs, inutilisés, et d'autres fois ce sont des pièces de récupération. Ils s'accompagnent le plus souvent du pilote, mais alors c'est la mémoire ou l'interface qui fait défaut. Elektor va publier un circuit de commande pour ces afficheurs, basé sur le microcontrôleur 80C51. Mais d'abord, examinons le fonctionnement et les fondements de l'afficheur à cristaux liquides.



On peut trouver à bon compte des afficheurs LCD, mais c'est souvent leur pilotage qui donne des soucis. C'est complètement différent de ceux que nous proposons régulièrement, à titre de projet de construction, dans les pages d'Elektor, et qui sont généralement équipés de mémoire et d'un microcontrôleur HD44780.

Bien sûr, on peut trouver les puces de commande des afficheurs matriciels. Un exemple en est le SED1335 de Seiko. Ce genre de puce fournit les signaux de contrôle nécessaires à l'afficheur et gère la mémoire. Mais pour l'amateur, de nouveaux soucis se préparent, parce que ces circuits intégrés sont d'habitude sous boîtier CMS à un prix rédhibitoire et on les obtient difficilement en petite quantité.
Aussi allons-nous, dans un prochain article, vous proposer une commande d'affichage matriciel à construire soi-même, capable de piloter des écrans jusqu'à 200 x 200 pixels, basée sur le 80C51. Mais pour débuter, nous allons examiner comment fonctionne l'afficheur à cristaux liquides proprement dit.

Liquid Cristal Displays

La première différence marquante entre un afficheur LCD et un écran normal, c'est qu'un LCD n'émet aucune lumière. Il travaille souvent avec l'appoint d'un éclairage d'arrière plan à LED ou par source fluo-

rescente. Les pixels se contentent de laisser ou non passer la lumière pour former l'image. Mais d'autres ne disposent pas de source de lumière, ils travaillent alors sur la lumière ambiante qu'ils doivent réfléchir.
Les cristaux liquides sont enfermés entre deux feuilles de verre séparées d'à peine quelques microns. Ils changent une de leurs propriétés optiques, la polarisation de la lumière qui les traverse, sous l'influence d'un champ électrique issu d'électrodes transparentes très fines, déposées sur les lames de verre.
Sur les afficheurs qui fonctionnent par réflexion, on a ajouté un filtre de polarisation et derrière, un miroir. La lumière ambiante atteint le polariseur, traverse le cristal liquide, subit une réflexion, retraverse le cristal, puis de nouveau le polariseur avant d'atteindre notre œil. La lumière doit donc passer deux fois par le filtre polarisant. Si la lumière n'a pas changé de sens de polarisation, nous la voyons revenir naturellement. Mais si les cristaux liquides ont modifié l'orientation de la polarisation, la lumière ne peut plus retraverser le filtre. Le pixel concerné apparaît alors en noir.
Les afficheurs éclairés par l'arrière possèdent deux filtres polariseurs, de part et d'autre des cristaux liquides, l'effet est donc le même.
Il est possible de donner aux électrodes la forme que l'on veut, lors de la vaporisation, celle de signes fixes par exemple, comme un chiffre à sept segments ou une échelle du genre thermométrique pour représenter l'état de charge d'un accumulateur.

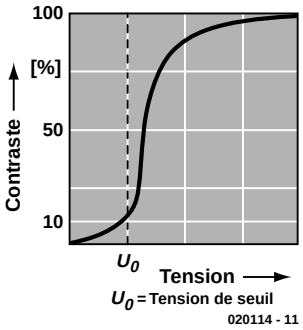


Figure 1. Courbe de contraste du LCD.

Commande

On attaque généralement les afficheurs à petit nombre d'électrodes en mode statique, c'est-à-dire que chacune d'elles dispose de sa broche propre. Mais quand elles sont nombreuses, la méthode deviendrait vite encombrante. On passe alors au multiplexage. Plus de commande individuelle des pixels, ils sont adressés à tour de rôle, par ligne et par colonne. La commande en multiplex se doit d'être rapide, parce qu'il ne faut que quelques millisecondes aux cristaux liquides pour revenir au repos après la disparition du champ électrique. Il faut donc rafraîchir continuellement l'image.
Sur un tube écran normal, on peut aisément conférer à chaque pixel une intensité différente. Il y a une relation plus ou moins linéaire entre le nombre d'électrons qui vont bombarder le pixel et la luminosité qu'il fournit. Rien de tout cela sur un LCD. Comme on le voit à la **figure 1**, c'est pratiquement du tout ou rien. S'il est effectivement possible d'influencer légèrement l'intensité lumineuse restituée par le pixel, en changeant la tension entre les électrodes, pour le rendu de la gamme des nuances de gris sur chaque pixel, il faut utiliser d'autres moyens. Souvent, on y parvient par des commutations à haut rythme entre les deux états. Le rapport cyclique influence alors la teinte de gris.

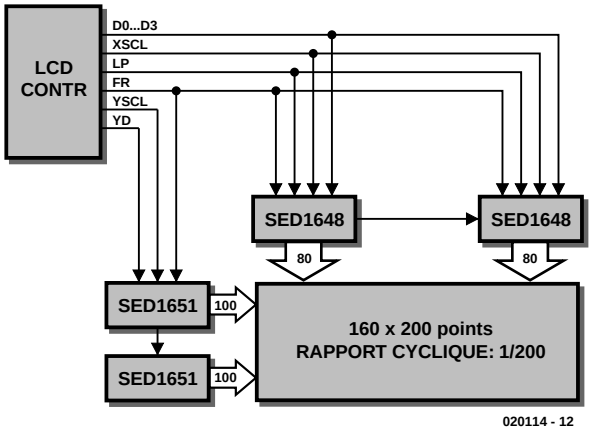


Figure 2. Schéma fonctionnel d'une commande de LCD en multiplex.

Lignes et colonnes

Les afficheurs à matrice opèrent en multiplex. La **figure 2** nous en fournit un exemple. Ce sont des circuits d'attaque spécialisés pour segments, les SED1648 qui commandent les colonnes et des circuits d'attaque de lignes communes du type SED1651 qui s'occupent des rangées, sous la supervision du contrôleur. Les informations à reproduire, le microcontrôleur les prend en mémoire RAM. La totalité de l'image est ainsi passée en revue à une fréquence de récurrence de 60 à 70 Hz.

Attaque de segments

Les circuits d'attaque (*driver*) de segments pour LCD disposent d'un registre à décalage embarqué. Le contrôleur leur expédie les informations sous forme sérielle, scandées par la ligne XSCL. Grâce au transfert sériel des données, il ne faut plus que quatre ligne au lieu de 80. Les registres à décalage ont donc aussi une largeur de quatre bits. Après 20 impulsions sur la ligne XSCL, l'information relative à une ligne se retrouve dans le registre à décalage. Une impulsion sur la ligne LP (*Latch Pulse*, verrouillage) et les données passent du registre à décalage dans la mémoire tampon qui commande les sorties. Un processus que vous retrouvez à la **figure 3** sous forme de diagramme temporel. Il est de coutume de traiter les pixels de gauche à droite, comme représenté à la **figure 4**.

Attaque des lignes communes

Pour les lignes communes, la structure des circuits d'attaque ressemble à celle des segments, sauf que le registre à décalage ne fait qu'un seul bit de large. Il n'y a qu'une seule ligne active à la fois. Le contrôleur s'arrange toujours pour placer un « 1 » dans le registre au début de chaque trame, c'est le marqueur de première ligne. Seule la première ligne comporte ce « 1 », pour toutes les autres, ce sera un « 0 » jusqu'au début de la trame suivante. Un circuit d'attaque des lignes communes n'a pas besoin de tampon de sortie. Le « 1 » qui circule en boucle dans le registre à

décalage constitue la commande dont nous avons besoin pour les rangées.
La **figure 5** reproduit l'évolution dans le temps de la commande des lignes communes. Les sorties des circuits d'attaque s'activent tour à tour au moment où les commandes de segments contiennent la donnée correspondante à la position de la rangée. Lorsqu'une trame complète a été traitée, le « 1 » sort du registre à décalage et le microcontrôleur s'empresse d'en glisser un nouveau au début du registre. La fréquence d'horloge des lignes communes est la même que celle du signal LP d'attaque des segments. C'est pourquoi les deux signaux sont souvent combinés.
Comme les données des pixels passent en multiplex, le rapport cyclique de la commande dépend du nombre de lignes. Pour un afficheur de 100 lignes, chacune d'elles n'est attaquée que pendant un centième du temps. C'est fort peu, aussi faut-il souvent les grouper, si l'afficheur compte plus de cent lignes. On peut ainsi commander plusieurs lignes en même temps, une par groupe.
Les afficheurs TFT (luminescents à transistors en couche mince) fonctionnent autrement. Chaque pixel possède son transistor pour mémoriser la dernière valeur. C'est pourquoi les écrans TFT peuvent totaliser beaucoup plus de lignes et offrir une qualité d'image supérieure à celle des matrices dont nous parlons ici.

Brûlure

Sur les moniteurs habituels, on connaît bien le phénomène de « brûlure » d'écran, un genre d'empreinte d'une image restée fixe trop longtemps. Avec le LCD, on remarque un effet comparable, mais qui se manifeste bien plus vite que sur un tube cathodique. L'écran LCD a horreur de la tension continue. Envoyer à un pixel la même tension pendant quelques secondes peut déjà provoquer des dommages irréparables, du fait que les cristaux liquides vont conserver la polarisation.
La bonne méthode de commande d'un afficheur à cristaux liquides consiste à alterner la polarité. C'est aussi une tâche du microcontrôleur et des circuits d'attaque. Le plus souvent, on inverse à chaque trame la polarité du champ électrique à l'aide du signal FR. Les cristaux liquides changent alors leur polarisation de gauche à droite et inversement, d'une trame à l'autre, et la composante continue moyenne sur chaque pixel reste nulle. Pour y parvenir, on utilise différentes tensions continues, symbolisées par V0 à V5. On les appelle les tensions de polarisation (*bias voltage*) comme sur la **figure 6**. Pour la plupart des LCD, il faut que

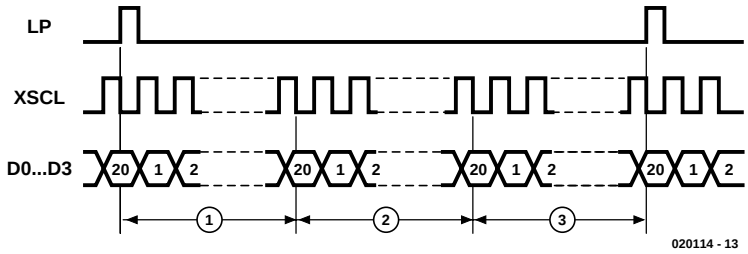


Figure 3. Diagramme temporel en illustration de la commande de segments.

Set 1 D3...D0	Set 2 D3...D0	Set 3 D3...D0	.....	.....	.....	Set 59 D3...D0	Set 60 D3...D0

020114 - 14

Figure 4. Le microcontrôleur traite les pixels de gauche à droite.

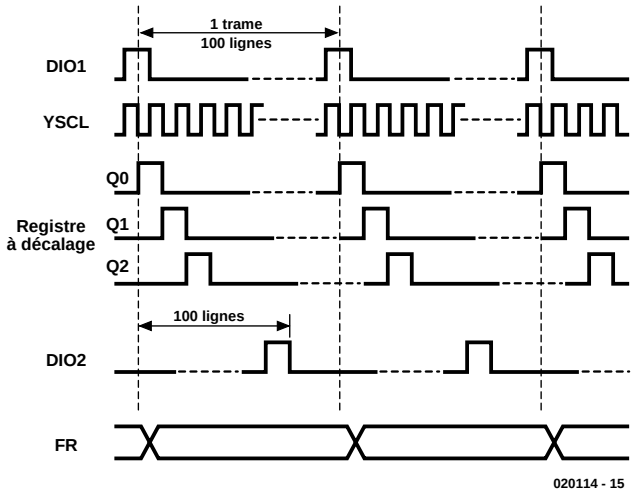


Figure 5. Diagramme temporel pris sur un circuit d'attaque de ligne commune.

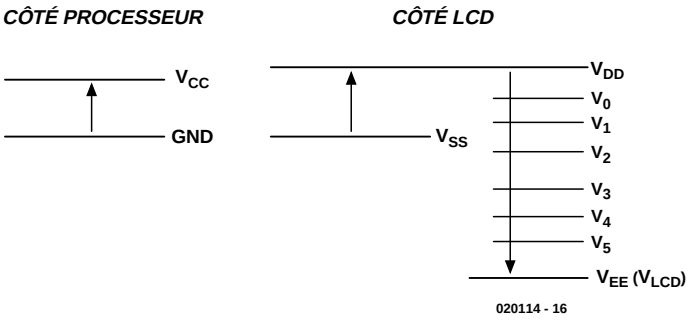


Figure 6. Les tensions de polarisation du LCD : Vdd >= V0>V1 >V2>V3>V4>V5 >= V_{ee}.

Tableau 1. Tension de sortie pour la commande des segments

$\overline{\text{Dsp\_off}}$	Registre à décalage	FR	Tension de sortie
L	–	–	V0
H	L (pixel éteint)	L	V2
H	L (pixel éteint)	H	V3
H	H (pixel allumé)	L	V0
H	H (pixel allumé)	H	V5

Tableau 2. Tension de sortie pour la commande des lignes communes.

$\overline{\text{Dsp\_off}}$	Registre à décalage	FR	Tension de sortie
L	–	–	V0
H	L (rangée inactive)	L	V1
H	L (rangée inactive)	H	V4
H	H (rangée active)	L	V5
H	H (rangée active)	H	V0

Vcc = Vdd = V0 = 5 V.

Le **tableau 1** reprend les valeurs de tension pour la commande des segments, le **tableau 2** pour les électrodes communes. Lorsque le signal de commande « Display off » est actif, les deux circuits d'attaque fournissent la tension V0. Tous les pixels sont éteints et il n'y a aucune tension entre les électrodes.

Exemple

À la **figure 7** vous trouverez le diagramme d'impulsions des tensions de sortie des deux circuits d'attaque dans le cas d'une image simple. Il s'agit ici d'une matrice de 34 communs et au moins 5 colonnes de segments. Le diagramme temporel ne reproduit que deux rangées et deux colonnes.

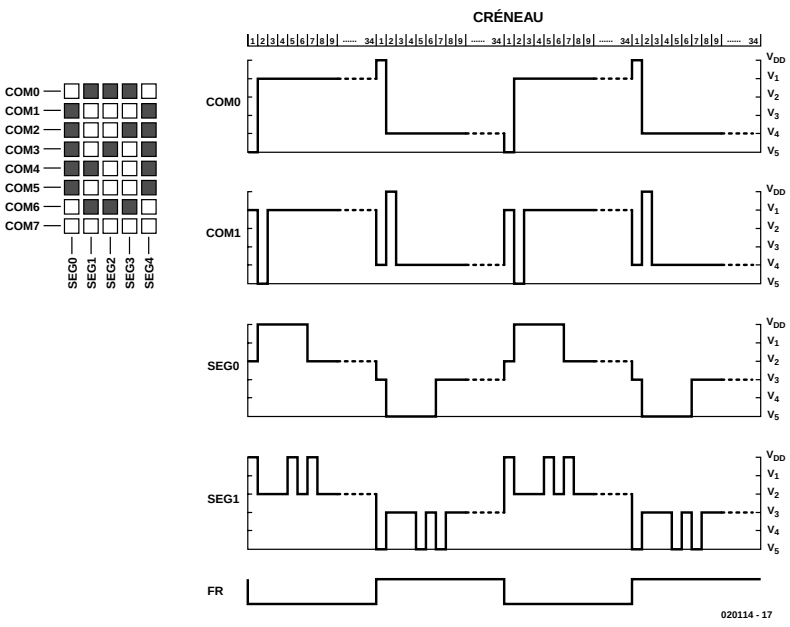


Figure 7. Tensions de sortie des circuits d'attaque de segments et de lignes communes.

Les rangées sont adressées à tour de rôle, il y en a donc toujours une reliée à la tension V5 (*active common*), pendant que les autres sont à V1 (*inactive common*). On applique aux colonnes ou bien V0 (*active segment*, donc pixel noir), ou bien V2 (*inactive segment*, donc pixel blanc). On reconnaît bien, dans le diagramme d'impulsions, le motif de l'image, formé de pixels lumineux et sombres, pour les deux colonnes reproduites.
Même s'ils sont inactifs, on commande les pixels. La tension est alors réduite et reste sous la valeur de seuil. Sous l'action du signal FR, chaque pixel se voit appliquer une différence de potentiel alternativement positive et négative entre ses deux électrodes. On le remarque également dans le diagramme. Les tensions lors des trames paires forment précisément le signal inversé de celui des trames impaires. La différence de tension entre électrodes est donc toujours de même grandeur, mais inversée, avec pour résultat qu'aucune composante continue ne peut apparaître.

Diviseur de tension

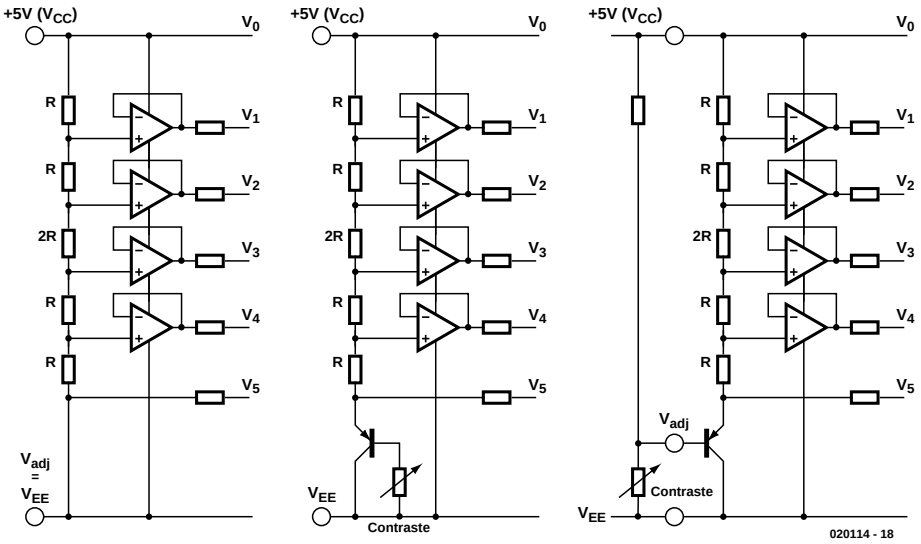
La **figure 8** vous dévoile la manière de produire les tensions nécessaires à la commande se l'afficheur. Ce sont des résistances qui forment diviseur de tension entre Vcc et Vee. Mais pour supporter la charge dynamique que constitue l'affichage, les tensions V0 à V5 ont été tamponnées par amplificateurs opérationnels et condensateurs électrolytiques. Dans le schéma de la figure 8, au milieu, nous pouvons apercevoir une résistance variable : elle sert à régler le contraste. À droite, dans la même figure, une autre version fait appel à une tension de commande externe par l'entrée Vadj. Autant que possible, on conditionne les tensions auxiliaires à proximité immédiate de l'afficheur, question de s'affranchir de l'influence de signaux parasites qui seraient visibles dans l'image. Mais il ne faut pas perdre de vue qu'on ne peut pas envoyer Vee quand Vdd n'est pas encore appliquée, sous peine d'endommager l'afficheur.

Bibliographie :

Les afficheurs à cristaux liquides (LCD), Elektor n°24, juin 1980, page 42 et suivantes

Liens Internet :

– www.seiko-instruments.de/
– www.optrex.co.jp/us/product/catalog/index.html
– www.eio.com/public/lcd



Brochages

Il n'y a pas de norme concernant les dénominations des broches d'un LCD. Chaque fabricant y va de sa petite idée et même dans la gamme proposée par le même constructeur, on trouve des différences. Le tableau ci-dessous devrait vous permettre d'associer appellations et fonctions des différents raccordements.

(020114-I)

Figure 8. Voici, par trois exemples, comment produire les tensions nécessaires à la commande de l'affichage.

***D0-D3** (Data0 à Data3), signaux de données à afficher sur LCD simple. S'appellent aussi parfois : XD0-XD3.
Plusieurs variantes possibles. Certains afficheurs fonctionnent avec un seul bit de donnée, appelé alors D ou XD. Il y a aussi les LCD à double écran, sur lesquels UD0-UD3 concernent la moitié supérieure et LD0-LD3, la moitié inférieure.

XSCL (*X Shift CLock*). S'appelle aussi : CL2, CP2, CP, Dot Clock, DCLK
L'horloge du registre à décalage. On lit D0-D3 sur le flanc descendant de XSCL. La fréquence d'horloge dépend de la récurrence des trames et du nombre de pixels par ligne. Il est courant de rencontrer une fréquence de 5 MHz.

LP (*Latch Pulse*). On l'appelle aussi : CP1 (clock pulse), CL1, HS, HSYNC, LOAD
Signal de mémorisation d'affichage, comparable au HSync du signal vidéo. Au flanc descendant de LP, on transfère les données des pixels du registre à décalage vers les tampons de sortie. Cet instant coïncide avec le moment où le circuit d'attaque de ligne commune doit envoyer la rangée suivante de l'afficheur. C'est pourquoi ce signal est souvent branché à l'entrée YSCL du registre à décalage du commun.

FLM (*First Line Marker*). On l'appelle aussi : DI, DIN, YD (Y data), S (Scan start-up), VSYNC
Entrée de données du registre à décalage du commun. Une impulsion sur FLM marque le début d'une nouvelle trame. À comparer à VSync du signal vidéo. Sa fréquence est proche de 60 Hz.

FR (*FRame signal*). On l'appelle aussi : M, DF, WF
C'est le signal qui assure l'alternance entre positif et négatif de la tension de commande de l'afficheur. Sa fréquence est d'habitude la moitié de la récurrence d'image, une trentaine de hertz, donc. À son sujet, on parle aussi de « AC drive waveform » ou d'expressions similaires, mais cela n'a rien à voir avec l'alimentation alternative pour illuminer l'arrière plan !

$\overline{\text{DISP\_OFF}}$ (*DISplay OFF*). On l'appelle aussi : $\overline{\text{DISP}}$, $\overline{\text{D\_OFF}}$, ENA, $\overline{\text{INH}}$
Signal de mise en/hors service de l'afficheur (H = marche, L = arrêt). C'est lui qui débranche tous les circuits d'attaque, pour économiser l'énergie, composer un nouvel écran en coulisses ou protéger l'afficheur quand le processeur n'a pas encore démarré.

Vdd, Vss (*Logic supply voltage*). On les nomme aussi : Vcc, Gnd
Tension d'alimentation pour la logique (5 V). Le courant consommé est faible, normalement moins de 100 mA, à moins qu'il ne serve aussi au convertisseur CCFL.

Vee (*LCD supply voltage*). On l'appelle aussi : Vlcd, Vssh
Tension d'alimentation des circuits d'attaque du LCD. Selon le type d'afficheur, on utilise des tensions entre –5 V et –25 V. Dans les feuillets de caractéristiques, on renseigne souvent la tension par rapport à Vdd, donc pas en référence à la masse !
Vee ne peut jamais être présente avant Vdd.

Vadj (*Contrast Adjust*). On l'appelle aussi : V_0, Vcon
Réglage de contraste du LCD. Cette tension influence souvent l'angle de vision optimale. Elle est parfois réglable. Il ne faut pas confondre « V_0 » avec « V0 » ! V0 est la tension d'alimentation la plus positive de l'afficheur.

A, C (*Anode*, *Cathode*)
Alimentation de l'éclairage d'arrière plan à LED. Parfois plusieurs LED sont en série, d'autres fois pas. Il y a donc lieu de vérifier !

AC On l'appelle aussi : CCFL, HV, HOT & GND
Alimentation de l'éclairage par l'arrière plan. On utilise une CCFL (*Cold Cathode Fluorescent Lamp*), une lampe fluorescente à cathode froide qui requiert 200 V environ et consomme une centaine de mA (hé oui !!!). Consultez soigneusement les fiches techniques avant de brancher une tension aussi élevée ! Pour des raisons de sécurité, les broches de la CCFL se placent toujours à l'écart des autres.

FG (*Frame Ground*)
Cette broche est reliée à toutes les parties métalliques de l'afficheur, on la branche à la masse du circuit.

Pratique des réseaux neuraux (4)

4ème (et dernière) partie : applications et grands réseaux neuraux

Chris MacLeod et Grant Maxwell

Dans la dernière partie de cette série, nous allons examiner quelques-unes des autres applications des réseaux neuraux, et jeter un coup d'œil sur certaines des questions plus complexes que doivent résoudre les chercheurs en réseaux neuraux.

Au cours de cette série d'articles, nous nous sommes concentrés sur la reconnaissance d'images. La raison en est qu'il s'agit non seulement d'une des plus importantes applications des réseaux neuraux, mais aussi de celle qui est la plus facile à illustrer puisque nous pouvons « voir » les images.
Toutefois, comme cela est expliqué dans la première partie, le réseau neural peut être considéré comme un système logique universel, capable (sous réserve qu'il dispose de trois couches) d'apprendre n'importe quelle table de vérité, *cf.* **figure 1**.
Si la sortie des neurones est une fonction sigmoïdale, le réseau agit alors selon une « logique floue », et peut produire des sorties analogiques -ce qui peut se révéler utile pour traiter de problèmes qui dans la vie courante ne sont pas « noir ou blanc ».

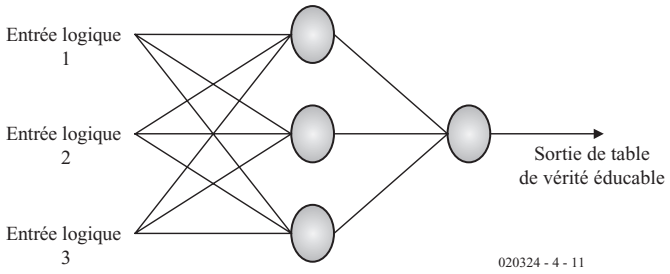


Figure 1. Logique universelle éducable.

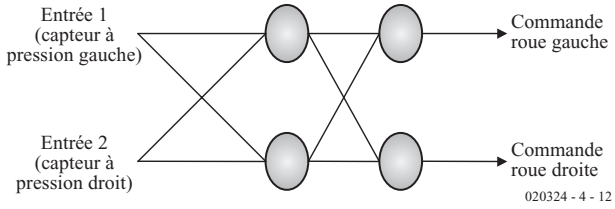


Figure 2. Contrôleur de robot.

Applications en robotique

Pour illustrer l'utilité d'une table de vérité, considérons un contrôleur de robot comme celui présenté en **figure 2**.
Le réseau peut apprendre à faire obéir le robot, si on l'entraîne à réagir aux entrées du capteur en fournissant des sorties qui commandent les moteurs. C'est un exemple simple de ce qu'on appelle un système nerveux artificiel, et un robot ressemblant à un animal, commandé par un système de pilotage éducable, est quelquefois nommé un « animat ».

On peut entraîner ce robot avec l'algorithme de rétro-propagation. Mais on peut aussi lui apprendre à survivre par lui-même en utilisant un algorithme d'apprentissage **d'exécution renforcée**. Les méthodes de renforcement d'exécution opèrent en inculquant au robot des comportements bénéfiques induisant des récompenses (par exemple en évitant des obstacles) et par une augmentation des poids qui encouragent le comportement récompensé. Ceci peut être réalisé automatiquement dans le robot lui-même (il faut bien sûr évaluer ce qui est un bon et ce qui est un mauvais comportement et le programmer, ce qui aboutit à introduire un **système de valeurs** dans le robot).

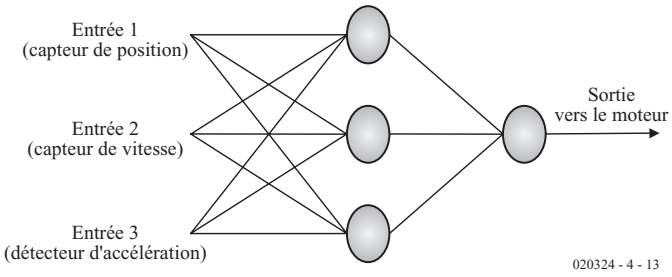


Figure 3. Pilotage du moteur.

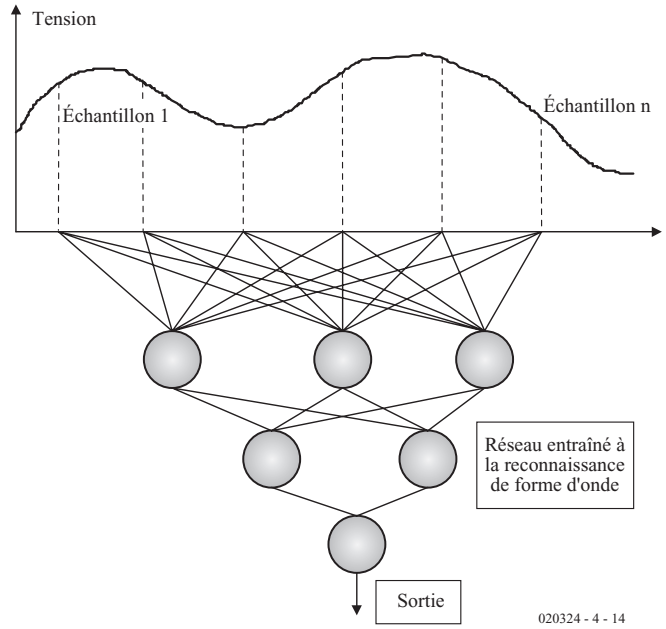


Figure 4. Un forme d'onde simple présentée à un réseau neural.

Une légère variante consiste à utiliser un algorithme génétique pour engendrer des types de réseaux à partir d'une population manifestant de bons comportements -on appelle cet algorithme un algorithme évolutionniste pour l'apprentissage de l'exécution renforcée (EARL = *Evolutionary Algorithm for Re-enforcement Learning*).

Applications de pilotage

On peut étendre les aspects de pilotage mentionnés ci-dessus à d'autres systèmes. Par exemple, supposons que l'on veuille obtenir un système de pilotage d'un moteur à courant continu, *cf.* **figure 3**.
On pourrait reconnaître les entrées comme des identifiants de paquets types (PID = *Packet IDentifiers*) utilisés dans les contrôleurs habituels. Un tel **neuro-contrôleur** pourrait être entraîné avec la rétro-propagation, à partir d'un contrôleur habituel, en utilisant les sorties de celui-ci comme cibles. Il pourrait aussi, et de manière plus intéressante, apprendre par l'expérience, en utilisant un algorithme d'apprentissage d'exécution renforcée ou un algorithme génétique pour déterminer ses poids.

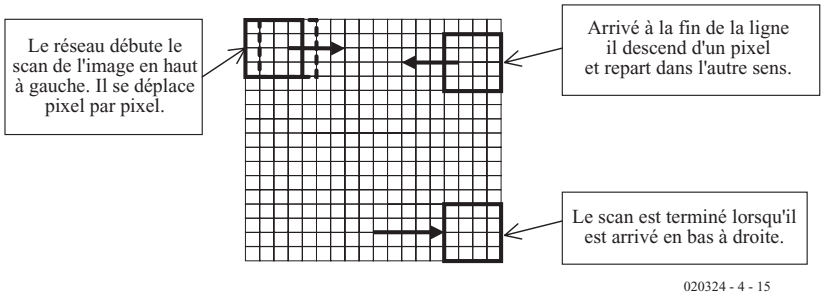


Figure 5. Scan d'image.

Interface homme-machine et détection intelligente

Dans le système de pilotage ci-dessus, le réseau réagit aux informations des capteurs et produit une sortie adéquate. Les réseaux neuraux sont efficaces pour démêler des entrées imbriquées comme celles-ci -vous n'avez pas besoin de comprendre la base théorique du système, juste de disposer d'exemples pour éduquer le réseau. Le réseau neural artificiel se chargera lui-même, pendant la phase d'apprentissage, de donner un sens aux interrelations et transformations entre les entrées et les sorties.
Une application future de ce type de système « à détection intelligente » est une interface homme-machine. Si l'on doit, par exemple, tester et réaliser des membres artificiels et les relier à une partie du système nerveux, il faut tester et comprendre les connexions des nombreux nerfs (et peut-être d'autres entrées et parasites biométriques), qui se manifestent toutes avec des fréquences différentes. Certaines des entrées peuvent n'avoir aucun rapport et d'autres être reliées entre elles de manière subtile. Traiter les informations de ce type de système complexe et non-linéaire relève manifestement du domaine de compétence des réseaux neuraux.

La reconnaissance des formes d'onde

Dans l'interface homme-machine évoquée ci-dessus, il fallait traiter les formes d'onde produites par les nerfs. En effet, un réseau neural peut reconnaître des formes dans le temps (comme celles-ci) et dans l'espace (comme des images), en les échantillonnant et en les présentant en entrée. La **figure 4** en montre le principe. Ceci conduit, bien sûr, au problème, déjà mentionné dans la deuxième partie, de l'image, ou dans ce cas de la forme d'onde, qui doit posséder une dimension précise et se trouver placée au centre de la grille de lecture.

Pré-traitement

Donner la bonne dimension à une image ou une donnée et la centrer de façon à ce qu'elle

soit lisible par un réseau s'appelle le pré-traitement. Il y a bien des manières de le réaliser. Dans le cas d'une forme d'onde échantillonnée, le réseau observe une « fenêtre » de temps. La forme d'onde, un échantillon à la fois, qui traverse celle-ci va finir par se positionner au centre de la fenêtre et être reconnue par le réseau.

On peut dire exactement la même chose d'un réseau de reconnaissance d'images. Il peut lui aussi faire un scan (balayage) de l'image, comme le montre la **figure 5**. La forme à reconnaître va finir, de la même façon durant ce processus, par se placer au centre de la grille de lecture. Dans la réalité ceci n'est pas très loin de ce que font nos yeux lorsque nous étudions une scène, en balayant (inconsciemment) l'image pour chercher des formes reconnaissables. Un point important de l'éducation d'un réseau pour ce genre de tâche consiste à l'entraîner à reconnaître autant les « parasites » ou les données sans rapport que la forme recherchée, sinon il signalera de fausses reconnaissances.
Autant pour le centrage de l'image ; qu'en est-il de son bon dimensionnement ? Eh bien, il peut aussi donner lieu à un pré-traitement.

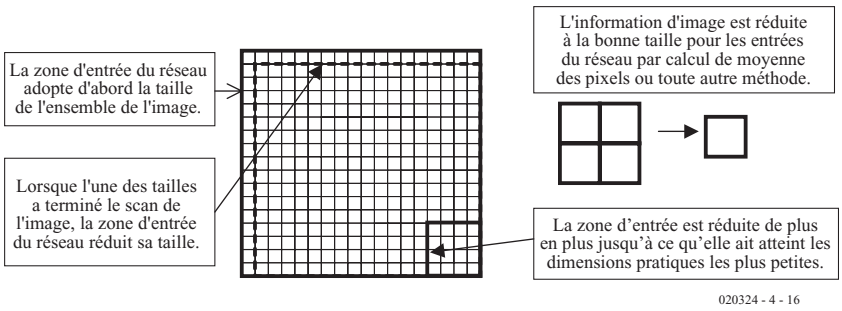


Figure 6. Dimensionnement d'image.

Réseaux modulaires

Les réseaux neuraux du cerveau reconnaissent les objets à travers certaines caractéristiques. Prenez la lettre L, par exemple. Elle a deux caractéristiques, une ligne verticale et une ligne horizontale. Peu importe son emplacement sur la grille de lecture, peu importe sa dimension, elle possède toujours ces deux caractéristiques.
Si nous avons un réseau reconnaissant les verticales et un autre les horizontales, alors un troisième peut les combiner pour former un tout -*cf.* **figure 7**.
C'est un exemple simple de réseau modulaire. Plutôt qu'un grand réseau disposant de multiples connexions et essayant de tout apprendre, on dissocie les tâches en sous-tâches plus réduites exécutées par un réseau plus petit. La recherche a démontré que c'est ainsi qu'est organisé le cerveau - non comme un

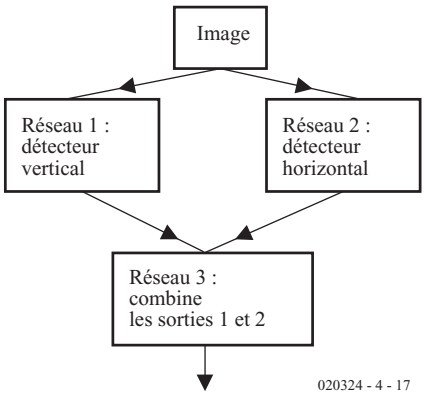


Figure 7. Réseaux modulaires.

Si nous commençons par une grande fenêtre et lui faisons balayer l'image, nous pouvons ensuite diminuer progressivement sa taille de façon à ce que, à un moment ou à un autre, la forme à reconnaître se dimensionne correctement sur la grille de lecture, *cf.* **figure 6**.
Les entrées du réseau neural étant de taille fixe, il faut manipuler la dimension originale des images de différentes tailles jusqu'à ce qu'elle corresponde à celle de la grille de lecture. On y arrive facilement en calculant une moyenne des pixels adjacents jusqu'à obtenir une image de dimension adaptée au réseau.

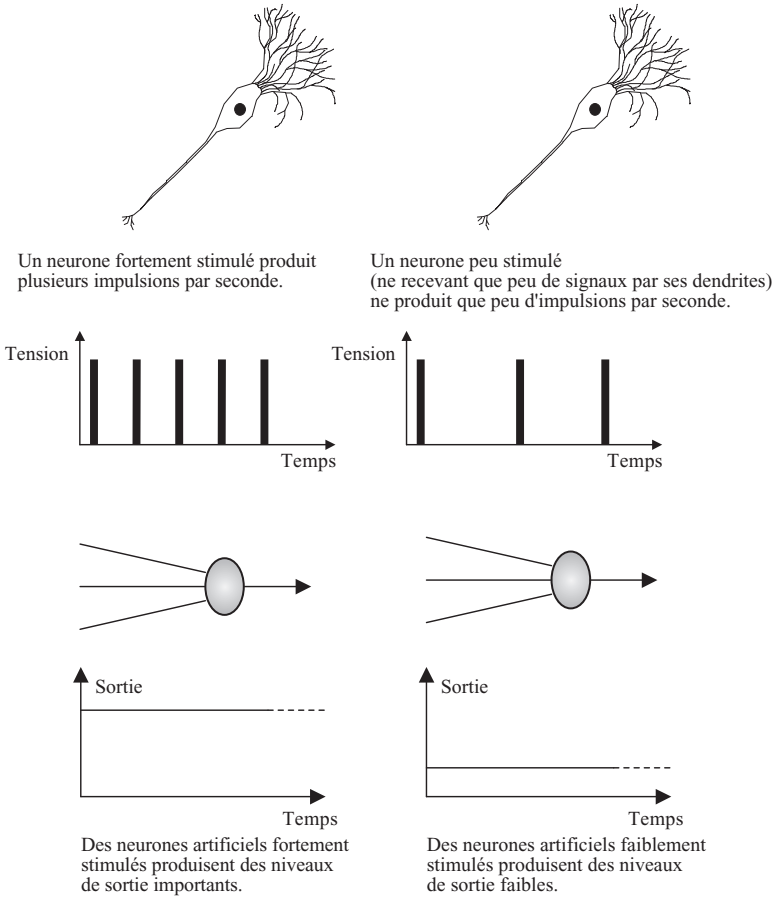


Figure 8. L'activité dans un neurone biologique et un neurone artificiel.

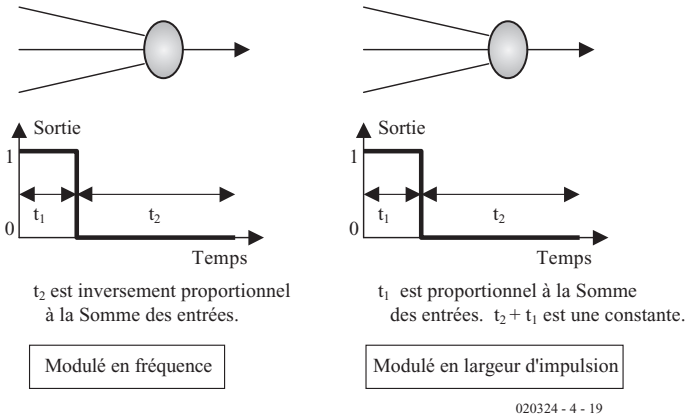


Figure 9. Neurones stimulés (on pourrait presque dire piqués au vif).

énorme réseau, mais en une multitude de petits modules exécutant leurs tâches et intégrant leurs résultats dans un ensemble cohérent.
La recherche se poursuit actuellement pour voir s'il est possible d'utiliser les algorithmes évolutionnistes pour développer des groupes de tels modules et former un système intégré.

Fonction neurale

Le lecteur attentif aura repéré que la description du neurone artificiel présentée en première partie n'a guère de ressemblance avec le neurone biologique que nous venons de décrire. En particulier, le réseau neural biologique module son activité en fréquence, comme le montre la **figure 8**.
Certains scientifiques y attachent une certaine importance, et pensent que ce que nous appelons « pensée » et « conscience » sont en fait des séquences d'impulsions traversant le système nerveux en interagissant les unes avec les autres. D'une manière plus pragmatique, il est plus facile de travailler avec des systèmes de pilotage modulant ainsi l'information (ou la modulant par largeur d'impulsion).
Des modèles artificiels ont été définis pour fonctionner de cette manière et sont souvent appelés des neurones « stimulés ». La **figure 9** montre deux modèles simples, l'un d'un neurone « stimulé » (modulé en fréquence) et l'autre d'un neurone modulé en largeur d'impulsion. Les poids et autres paramètres de ceux-ci sont facilement déterminés en utilisant un algorithme génétique (ou si un peu de mathématiques vous amuse, en modifiant la rétro-propagation).

Connexions et technologie

Une dernière zone d'intérêt réside dans la taille du réseau neural biologique comparée à celle du réseau artificiel. Vous vous souvenez sans doute que le cerveau est constitué de 100 milliards de neurones (autant que d'étoiles dans notre galaxie). Ceci en fait la structure la plus complexe de l'univers connu.
De plus, chacun de ces neurones peut être connecté avec 1 000 autres (certains ont 100 000 connexions). Ce qui signifie qu'il existe peut-être 100 000 milliards de connexions dans le cerveau.
La technologie moderne ne peut pas rivaliser avec la densité des neurones et de leurs connexions. Le neurone est actuellement de la même taille qu'un transistor, mais organisé en structure dense à trois dimensions. En revanche, nos circuits électroniques sont essentiellement plats -à deux dimensions.
On peut penser que la densité des neurones et de leurs chemins de câblage sont des problèmes insurmontables pour créer des réseaux neuraux biologiquement réalistes. Mais on doit se rappeler que les signaux ne traversent les neurones réels qu'à la vitesse de quelques deux cents mètres à la seconde, alors qu'ils peuvent circuler dans leurs équivalents artificiels à 60 % de la vitesse de la lumière. Le réseau neural artificiel gagne ainsi en vitesse ce qu'il perd en connectivité.
Il existe une possibilité de résoudre le problème de connectivité, même avec la technologie actuelle, si l'on attribue à chaque neurone une adresse équivalente à une base d'un réseau de communication, et en l'utilisant pour la transmission de signaux depuis et vers les neurones à la place d'un câblage spécifique pour chaque signal.
Réaliser des réseaux neuraux physiques n'est cependant pas simple, parce qu'il n'y a pas de technologie de circuits imprimés disponible immédiatement pour les amateurs (les grands réseaux neuraux discontinus nécessitent de nombreux composants) -en particulier ceux disposant de câblages reconfigurables, nécessaires pour les réseaux neuraux artificiels évolutionnistes. Quelques expérimentateurs ont essayé les circuits FPGA (*Field Programmable Gate Arrays* = circuits de portes programmables), mais il ne sont pas idéalement adaptés à la tâche parce que chaque neurone occupe un grand espace « immobilier » de puces. C'est d'avoir modélisé le réseau neural sur le cerveau, sans prévoir sa mise en œuvre électronique - ce qui la rend inefficace, -qui est à l'origine de ce genre de problème. On a aussi besoin de développer un neurone (dans le sens d'une unité de traitement généraliste et éducable) qui soit réalisable électroniquement et efficace.

En conclusion

Nous espérons que vous avez apprécié ce voyage au sein des réseaux neuraux, suffisamment pour en tenter l'expérimentation vous-même. Vous trouvez ci-dessous quelques références de base, qui sont un bon point de départ pour des explorations ultérieures.

(020324-4)

Remerciements
Les auteurs tiennent à exprimer leur gratitude à Ann B. Reddipogu, Niccolo Capanni et Sethuraman Muthuraman pour leur aide dans la mise au point de ces articles.

Lectures
Quelques livres réputés sur les réseaux neuraux pour ceux qui veulent approfondir le sujet :

K. Gurney, An Introduction to Neural Networks, UCL Press, 1997.

I. Pratt, Artificial Intelligence, MacMillan, 1994.

P. D. Wasserman, Neural Computing : theory and practice, Van Nostrand Reinhold, 1989.

Pilote de port sériel pour Windows

Programmer avec Serial.dll

Paul Goossens

Nous avons, dans le numéro de novembre 2002, publié un article décrivant comment programmer correctement le port sériel sous Windows. Il apparaît que la méthode décrite semble trop complexe pour nombre d'entre nos lecteurs. C'est la raison pour laquelle nous avons écrit une .dll qui facilite très sensiblement la programmation du port sériel. Cet article a pour prétention de vous présenter cette .dll.

En dépit de sa disparition prochaine sur les nouveaux modèles d'ordinateurs, le port sériel reste fort utilisé par nombre d'amateurs d'électronique pour relier l'ordinateur avec le monde extérieur, rien de plus logique d'ailleurs vu que cette interface se trouve (encore) sur tout PC standard et met à disposition, outre sa fonction de moyen de communication sérielle, un certain nombre d'entrées et de sorties. Il est vrai que tout le monde ne parle plus aujourd'hui que du bus USB dont les possibilités n'ont pas à rougir de celles de l'interface sérielle mais qui requiert l'adjonction d'un minimum d'électronique et dont la mise en oeuvre dépasse les capacités de la plupart des électroniciens amateurs. Dans ce cadre, nous passerons sous silence les vieilles cartes ISA enfichables et ne parlerons pas des cartes PCI de réalisation personnelle. En tout état de cause, le port sériel reste l'interface désignée lorsqu'il s'agit de réaliser une interconnexion simple entre de l'électronique et un PC.

Programmer

Le temps passant, la programmation du port sériel a bien évolué. Sous DOS il était facile de l'attaquer directement sans que le système d'exploitation (SE) n'ait la moindre objection et ne mette de bâtons dans les roues. Sous Windows les choses se sont quelque peu compliquées. Si les premières versions de Windows permettaient un accès direct au matériel (périphériques) il n'en fallait pas moins, de temps à autre, faire appel à toutes sortes de trucs et astuces pour pouvoir utiliser le port sériel.

Les versions les plus récentes de ce SE interdisent tout simplement à un programme de piloter directement du matériel faisant partie de l'ordinateur. Windows a ses responsabilités et entend garder les rênes en main. Ce n'est que dans ces conditions que Windows peut orchestrer le pilotage du matériel.

La seule (bonne) méthode, sous Windows, de piloter le port sériel est de le faire par le biais d'un API (*Application Programming Interface*) Windows. Dans le cas du port sériel l'API est doté d'un certain nombre de fonctions (nous vous renvoyons à l'article cité en référence [1]). Il n'en reste pas moins que le pilotage du port sériel par le biais de ces fonctions n'est pas une sinécure. Il serait

Tableau I.

Récapitulation de toutes les fonctions et routines de la .dll.

OpenCOM
CloseCOM
IsPortOpen
GetPortNr
COMPortExists
SendCharCOM
ReadCharCOM
SetTxD
ResetTxD
SetRTS
ResetRTS
SetDTR
ResetDTR
GetCTS
DetDCD
GetDSR
GetRI
GetHandle
CheckInputs
BaudRateSet
ParitySet
BitPerByteSet
StopBitsSet

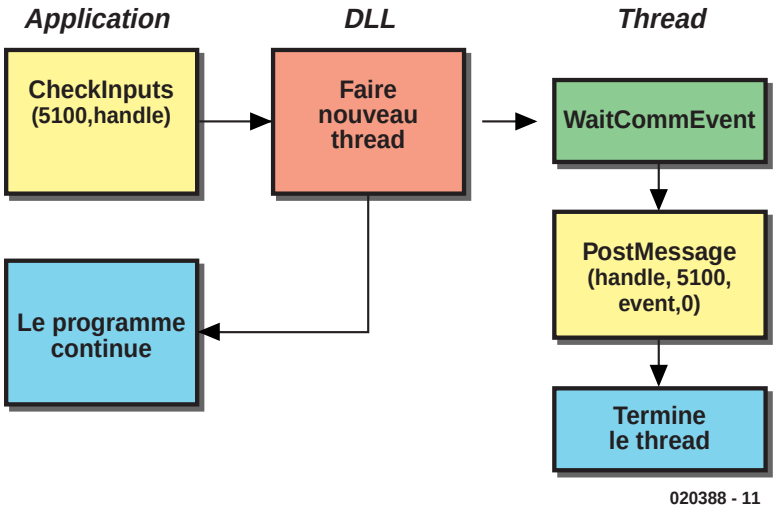


Figure 1. Exécution de la fonction CheckInputs.

plus pratique de faire exécuter par une .dll toutes les routines de base simple pour la commande du port sériel, sans que le programmeur n'ait à se soucier de la communication entre le programme et Windows.

Serial.dll

Nous pensons que nombre de nos lecteurs seront heureux d'apprendre que nous avons écrit une .dll permettant un pilotage facile d'un port sériel. Elle permet, outre l'émission et la réception de caractères, de procéder à une interrogation de l'état des entrées et à une commande différenciée de chacune des sorties.
Fonction additionnelle, la .dll a été dotée d'une option qui permet de surveiller le port sériel et faire émettre un message par Windows en cas de changement à ce niveau. Ceci permet de ne plus avoir à interroger (le fameux « *polling* ») le port sériel. Ce n'est qu'après un changement d'état que le programme en est informé, tout comme Windows signale une action sur un bouton ! Tout le monde ne pourra pas utiliser cette fonction à tous les coups : rassurez-vous, il n'est pas indispensable d'utiliser cette routine, elle est en fait plutôt un extra pour le programmeur expérimenté.

Les fonctions

Le **tableau 1** énumère les fonctions et les routines de notre .dll. Toutes les fonctions concernées ont reçu un nom parlant de sorte que l'on sait au premier coup d'oeil quelle est leur tâche. Nous n'entrerons partant pas dans le détail de chacune d'entre elles. Nous vous renvoyons à l'exemple correspondant que vous pourrez, tout comme la .dll, télécharger depuis notre site Internet (adresse :www.elektor.fr) sous la dénomination **020388-11**.
Il faut, pour pouvoir utiliser un port, commencer par l'ouvrir. C'est à quoi sert la fonction *OpenCOM*. Cette fonction a besoin, comme paramètre, du numéro de port du port sériel à ouvrir. 1 correspond à COM1, 2 à COM2, etc. Si l'ouverture s'est faite correctement on a, en retour, mention du numéro de port; si les choses ont mal tourné, la réponse retour est un 0.
Après utilisation, lors de la fermeture du programme par exemple, il faudra refermer le port. C'est là la fonction de la routine *CloseCOM*. On pourra, entre ces 2 événements, envoyer et recevoir des caractères par les routines *SendCharCOM* et *ReadCharCOM* respectivement.
Les fonctions *Getxxx* (*GETDCD*, *GETDST* etc.) permettent respectivement l'interrogation des entrées DCD, DST, CTS et RI. Le pilotage de la sortie DTR se fait par le biais des fonctions *SetDTR* et *ResetDTR*. Il existe des fonctions comparables pour les 2 autres sorties, TxD et RTS.
La commande directe de la ligne TxD cache un petit piège. La fonction de cette ligne est normalement l'émission de données sérielles. Il est vrai qu'il est possible de commander cette ligne directement mais si nous essayons d'envoyer un caractère après que la fonction *SetTxD* a forcé la ligne au niveau haut, cette opération d'émission ratera et risquera fort bien de provoquer le plantage (crash) du programme. Il suffit de commencer par effectuer une réinitialisation (reset) de la ligne par le biais de la fonction *ResetTxD*, nous pourrons ensuite envoyer autant de caractères que nous le voudrons.
Les fonctions précédentes ne requièrent en fait que peu d'explications. Il en va autrement dans le cas de la fonction *CheckInputs*. Il va nous falloir, avant de pouvoir mettre en oeuvre cette fonction, dire quelques mots au sujet des messages Windows. Que cela ne nous empêche pas d'utiliser la présente .dll. Vous pouvez sauter le paragraphe suivant si vous n'avez pas l'intention d'utiliser la fonction de signalisation par Windows de tout changement d'état au niveau du port sériel.

Messages Windows

Les programmes peuvent faire effectuer une tâche au système d'exploitation par le biais d'une API et dans la plupart des cas Windows fournit en réponse un message sous une forme ou une autre. Il ne s'agit cependant pas là du seul et unique moyen de communication entre Windows et les applications.
Le *messaging system* est un autre moyen de communication important. La quasi-totalité des programmes Windows utiles s'en sert. Le système d'exploitation informe, par le biais de ces fameux Windows messages, une application qu'il y a eu action sur une bouton de souris ou lorsqu'il faut reproduire le programme sous forme minimisée.
Un message se résume à 3 variables seulement que l'application doit interpréter avant de pouvoir ensuite entreprendre l'action correspondante. La première variable donne le type de message. Les 2 variables suivantes sont utilisées en cas de besoin d'information additionnelle. Ainsi, à quoi cela servirait-il que Windows informe une application que l'utilisateur a bougé la souris si le SE ne lui donne pas la nouvelle position de la souris.
Windows connaît un bon nombre de message spécifiés, mais il est également possible d'envoyer un message personnel vers une application. C'est à cet effet que sont réservés les message de type 5000 et au-delà, dits « utilisateurs ». Il s'agit des messages définissables par l'utilisateur (*user-definable*); ils ne sont pas émis par Windows soi-même, mais peuvent être utilisés par les différentes applications pour communiquer entre elles. La plupart des environnements de programmation (tels que Delphi, Visual C++, etc.) traitent

cette communication sans que le programmeur n'ait à intervenir. Cependant, la plupart des environnements de programmation permettent au programmeur de traiter ces messages lui-même lorsque cela est souhaité.
Dans le cas de Delphi on dispose de la possibilité d'utiliser l'événement (*event*) *Application.OnMessage* pour commencer à traiter soi-même les messages de Windows entrants. En Visual C++ on pourra arriver au même résultat par le biais de *Application::OnMessage*. Notre .dll utilise ce mécanisme de communication pour informer l'application le cas échéant d'un changement d'état du port sériel. Nous pouvons utiliser, pour l'émission d'un message, la fonction *PostMessage* de l'API Windows.
L'API Windows dispose en outre d'une fonction qui attend jusqu'à ce qu'il y ait un changement d'état d'un port sériel. cette fonction s'appelle *WaitCommEvent* et convient fort bien à notre objectif, mais elle présente l'inconvénient d'interrompre l'exécution du programme qui ne reprendra que lors d'un changement d'état au niveau du port sériel, ce qui n'est pas le but de la manoeuvre. Il est cependant facile de résoudre ce problème en faisant appel à ce que l'on appelle les *threads* (fils).

Threads

Une application peut exécuter des parties de son programme dans un *thread* distinct. Cela signifie que 2 programmes sont actifs simultanément. Le premier est le programme principal, le second est le *thread*. En cas de blocage de ce *thread*, cela n'empêche pas l'exécution du programme principal.

La **figure 1** illustre l'exécution, par la .dll, de la fonction *CheckInputs*. Le programme appelle la fonction, la .dll créant un *thread* avant que celui-ci ne soit ensuite démarré automatiquement. À partir de ce moment, le programme principal s'exécute normalement indépendamment de l'exécution du *thread*. Le *thread* appelle la fonction Windows *WaitCommEvent*, Windows arrêtant l'exécution du thread jusqu'à ce qu'il y ait un changement d'état au niveau du port sériel.
Comme Windows considère le *thread* comme un programme distinct l'exécution de notre propre programme se fait en toute quiétude. Dans le cas présent, le programme principal s'occupe de la routine *CheckInputs* qui fait partie de la .dll. La .dll termine la routine et le programme principal peut tranquillement vaquer à ses propres occupations.
Dès l'instant de changement d'état du port sériel, Windows lève le blocage du *thread* et signale, à l'aide d'une variable, quel est l'événement ayant pris place sur le port sériel. À son tour, le *thread* transmet cette information à notre programme sous la forme d'un message Windows.
À la réception par le programme d'un message Windows on a appel de la fonction *AppMessage*. Cette fonction s'assure si le message signale un changement d'état du port sériel (dans le cas présent nous y avons attribué la valeur 5100 mais on pourra utiliser toute autre valeur à condition qu'elle ne soit pas utilisée par Windows). Si tel est bien le cas, il est fait appel à une routine qui réagit au nouvel état. Sinon, le message doit être traité par la boucle Message proprement dite.

En conclusion

Nous pouvons fort bien nous imaginer qu'il vous reste, après la lecture de cet article, encore l'une ou l'autre question. Il vous est partant recommandé de jeter un coup d'oeil au code-source des programmes donnés en exemple correspondants. L'examen et l'adaptation d'un programme fonctionnel existant est souvent la méthode d'apprentissage la plus rapide et la plus facile. C'est à dessein que le programme d'exemple a été écrit en Delphi sachant que ce langage est très explicite. Il est facile de convertir le code-source en C++ par exemple.
Dans ce même numéro nous vous proposons un petit testeur de port COM qui vous permettra de vérifier très vite le fonctionnement des programmes que vous aurez écrits.

(020388)

Bibliographie

[1] Ports sériels sous Windows, Elektor n° 293, novembre 2002, pages 30 et suivantes

Programmateur Atmel, Elektor n° 279, septembre 2001, page 38 et suivantes

L'auteur de ce projet a mis à notre disposition des versions réactualisées du programme tournant sous Windows (010005-11) et du progiciel (firmware) du processeur (010005-41). Ces files sont disponibles sous la rubrique Téléchargement de notre site. Il semblerait que, en cas d'utilisation de la version 4 Koctets du microcontrôleur d'Atmel, les derniers octets ne se laissaient pas programmer. Ce problème paraît être dû à des erreurs de chronologie et a été résolu par la nouvelle version du programme.

(010005)

Capacimètre auto-calibre, Elektor n° 298, avril 2003, page 74 et suivantes

Il existe une différence entre le schéma et la liste des composants de cet article au niveau du transistor T1. Bien que tant le BC557B que le BC559B puissent fort bien travailler ici, le composant préférentiel est le BC559B.
Le code du coffret proposé chez Farnel est 736-442 et non pas 736-351.

(020144)

Analyseur logique 20/40 MHz à géométrie variable, Elektor n° 296, février 2003, page 38 et suivantes

La liste des composants de ce montage requiert 2 corrections mineures :

IC11 = 74LS688 (vu qu'il n'existe pas de 74F688 !)
IC13 = 74LS573 ou **74F573**

Notons qu'il existe une source de kit et de composants pour ce projet, encore qu'elle soit située aux Pays-Bas :
C-I Electronics, PO Box 55544 NL 3008-AM Rotterdam Pays-Bas, l'adresse de leur site (aussi en anglais) : www.dil.nl

(020032)

UART pour USB, Elektor n° 283, janvier 2002, page 8 et suivantes

Il existe de nouvelles versions des fichiers USBUART.SYS et USBUART.INF qui permettent l'installation du programme sous Windows XP. Ces nouveaux fichiers sont disponibles dans la rubrique Téléchargement de notre site.

(010207)

Relais à puissance réduite

Des relais modernes qui fonctionnent sur piles

par Klaus-Jürgen Thiesler

Bien que leur principe n'ait pas évolué en un siècle, les relais sont loin d'être en voie d'extinction ! L'une des raisons : le circuit de commande est isolé électriquement du circuit de commutation.

Les interrupteurs télécommandés électriquement ont encore de beaux jours devant eux : ils sont robustes, la tension à commuter ne dépend pas de la tension de commande, et la configuration de commutation s'étend d'un simple contact de travail à des contacts inverseurs multiples non court-circuitants.
Bref, il serait économiquement illogique de les remplacer par des circuits à semiconducteurs bien plus coûteux, complexes et volumineux. Et n'oublions pas que le développement technologique exerce aussi son influence sur ces bons vieux relais, qui rapetissent continuellement mais deviennent toujours plus puissants, et dont le brochage est prévu de plus en plus souvent pour le montage en surface.

Particularités des relais

Veiller au point suivant lors du choix d'un relais : le rebondissement lors de la fermeture des contacts de commutation ne doit pas nuire au bon fonctionnement de la charge. La pointe de courant initial des charges à coefficient de température positif, comme les lampes halogène ou à incandescence, est très élevée ; elle peut atteindre 50 fois la valeur du courant nominal. Dans le cas des moteurs et d'autres charges inductives, les contacts de commutation et les enroulements doivent résister à d'éventuelles charges de pointe ou d'autres conditions exceptionnelles comme le courant de démarrage, les courts-circuits ou une chute subite de tension sous peine d'être détruits par la chaleur, de rester collés ou de s'ouvrir.

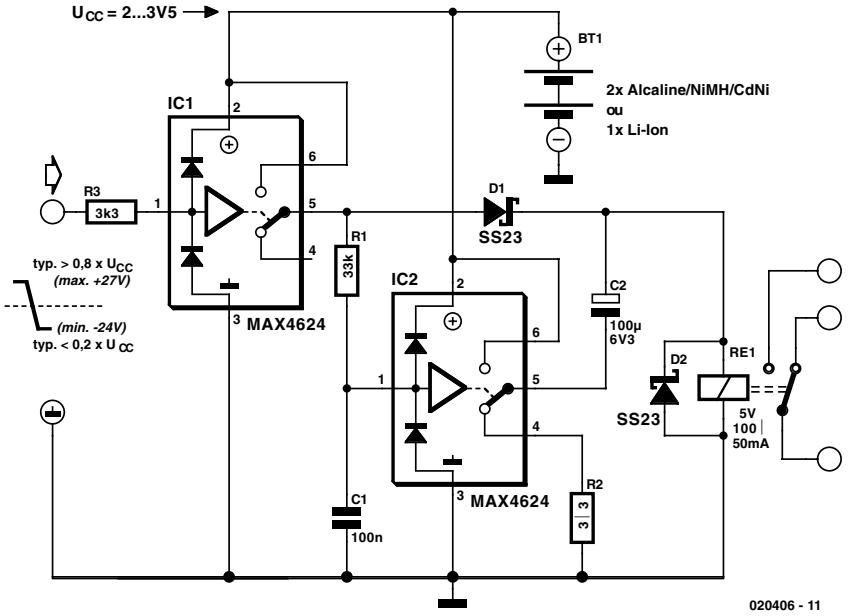


Figure 1. Les 8 composants adaptent le courant du relais aux caractéristiques du processus de commutation.

Fonctionnement sur piles

La bobine d'un relais commuté est en général alimentée continuellement. Cette perte de puissance constitue un désavantage considérable dans les applications sur piles. L'absence de consommation en position de repos constitue par contre un avantage pour ces mêmes applications.
Comment réduire cette perte de puissance dans l'état commuté ? La puissance dissipée par la bobine du relais est égale à U^2/R. Elle diminue fortement, en fait comme le carré de la tension, lorsqu'on abaisse cette dernière. La consommation type d'un relais

utilisé dans le domaine des télécommunications est de 250 mW à 5 V pour une résistance interne d'environ 100 Ω. Le tableau fournit des indications sur la puissance dissipée en fonction de la tension de commande. Les valeurs nominales sont basées sur un fonctionnement garanti à 100 % en mode DC ! Il faut les déterminer empiriquement ou les obtenir auprès du fabricant.
Il est toutefois presque toujours possible d'activer un relais avec une tension plus basse. L'entrefer du flux magnétique se réduit dès que l'armature est attirée, de sorte que la bobine du relais n'a besoin que d'un courant plus faible pour la maintenir. Dans un environnement dépourvu de vibrations, l'armature n'est relâchée que si la tension de la bobine diminue jusqu'à 1,5 V.
Les valeurs réelles subissent une dispersion due à la fabrication par rapport aux valeurs du tableau ; il faut donc tester empiriquement plusieurs relais et tenir compte de leur type. Les piles coûtent cher ; il est donc logique d'adapter la tension de travail d'un relais au moyen d'un commutateur analogique.

Le circuit

Le circuit de réduction des pertes de commutation est représenté dans la **figure 1**. Les commutateurs analogiques sont représentés déclenchés (au repos). Si une tension positive (>0,5 Uss) est appliquée à l'entrée de commande (broche 1), le courant passe par le commutateur analogique IC1 et par D1 avant de parvenir à la bobine du relais S1. La bobine est prémagnétisée au moyen de la charge rapide de C2 par R2. Dans ce cas, la tension de la pile n'est pas suffisante pour attirer l'armature du relais.
C1 se charge en même temps par R1. Grâce à la fonction logarithmique RC, IC2 commute après 3 ms, lorsque la valeur de commutation logique atteint environ 0,5 Uss. À ce moment-là, C2 est presque entièrement chargé. R2 limite le courant de charge maximum à 1 A. Le circuit analogique MAX4624 est en outre protégé en interne contre les surcharges de courant jusqu'à 1,2 A –mais cette fonction ne doit être active que très brièvement sous peine de surcharge thermique.
La commutation du circuit analogique IC2 place le condensateur électrolytique C2 en série avec la tension d'alimentation Uss et la bobine du relais. L'impulsion de courant fournie par C2 élève brusquement la tension de la bobine au double de la tension d'alimentation. Et le relais colle ! C2 se décharge immédiatement jusqu'à la tension de la pile +Uss moins la tension à l'état passant de la diode de Schottky D1. Le relais reste attiré car la tension est encore supérieure à la tension de retombée de 1,5 V.
S'il est impossible d'exclure l'éventualité d'une inversion de polarité des éléments de la batterie, intercaler une diode de Schottky type SS23 dans chaque ligne d'alimentation de la broche 2 de chacun des 2 circuits analogiques IC1 et IC2.
Les composants dépendent de la tension d'alimentation fournie par les éléments de la batterie et des caractéristiques du relais. Il faut qu'ils soient bien adaptés les uns aux autres, mais cela ne devrait pas constituer un problème.
R3 protège l'entrée de commande contre les dommages qui pourraient être causés par les tensions transitoires. Toutes les broches du composant MAX4624 sont dotées en interne de diodes pouvant dériver jusqu'à 10 mA au maximum vers les 2 potentiels d'alimentation.
La plage de tension d'alimentation des circuits intégrés de commutation analogique s'étend de +1,8 à 5,5 V. Le composant MAX4624 présente la particularité de limiter le courant de commutation maximum possible à 1,2 A en cas de MCA (*Maximum Conceivable Accident*, accident maximal prévisible). En tenant compte des contraintes thermiques, il peut commuter continuellement ±200 mA en régime nominal et ±400 mA en mode impulsionnel (durée d'enclenchement de 10 %, 1 kHz). Sa résistance de commutation est de l'ordre de 0,6 Ω.
À l'instant de la commutation, le courant de charge de C2, qui passe lorsque celui-ci est déchargé, est très élevé ; il n'est limité que par la résistance interne des éléments de la batterie et du condensateur électrolytique venant s'ajouter à R_{DSon} du circuit analogique de IC2. R2 doit dissiper la majeure partie de la puissance au cas où le condensateur électrolytique subirait un court-circuit interne. La constante de temps de R1|C1 doit dépasser de plusieurs fois celle de R2|C2 (en l'occurrence d'environ 10 fois). On assure ainsi que le condensateur électrolytique est complètement chargé lorsque IC2 commute.
La capacité de C2 fixe l'énergie impulsionnelle nécessaire pour attirer avec certitude l'armature du relais utilisé. Il va sans dire qu'elle peut varier fortement en fonction du type de relais et du fabricant. La seule solution consiste à effectuer des essais. On peut toutefois se baser sur une règle empirique : 1 000 µF par ampère. La valeur que doit avoir C2 croît avec l'énergie nécessaire pour commuter le relais.
La constante de temps de R1|C1 détermine le délai entre l'impulsion haute à l'entrée de commande et l'activation de l'armature.
Le condensateur électrolytique C2 est le composant le plus volumineux de ce circuit (le relais étant bien entendu hors concours). Les commutateurs analogiques IC1 et IC2 sont livrables en boîtier SOT23 5 broches de taille fortement réduite.

Les conséquences

Le rendement subit une amélioration extraordinaire ! Comme la tension nominale du relais n'est nécessaire que pendant un instant, la contribution de la puissance nominale

Tableau 1. Valeurs nominale et limite d'un relais Telecom déterminées empiriquement

Processus de commutation	Tension [V]	Courant [mA]	Résistance interne [Ω]	Dissipation [mW]	Rendement relatif
Tension de relais nominale	5	50	100	250	100 %
Tension de sortie	3,5	35		125	50 %
Tension de maintien	2,5	25		62	25%
Tension de décollage	1,5	15		25	10%

dissipée (250 mW) devient négligeable. En résumé, la tension nominale de la bobine du relais utilisé dans un circuit peut être sans inconvénient le double de la tension de la pile. La pile ne doit alors fournir que la tension de maintien, d'où réduction de la puissance moyenne dissipée : il faut choisir le type de relais en fonction de la tension de retombée qui doit être inférieure à la tension de la pile partiellement déchargée. Dans le cas de 2 éléments NiMH ou NiCd (2·1,2 V) la puissance dissipée est réduite de 75 % par rapport à la puissance nominale dissipée à 5 V. Dans le cas d'un élément Li-Ion (3 V), la puissance dissipée est réduite de 64 %. Même un élément lithium-chlorure de thionyle (3,6 V) permet encore d'atteindre une réduction de 48 % par rapport à un relais dont la tension nominale de fonctionnement est égale à celle de la pile.

(020406)

Straight, ampli à tubes

2ème partie : platines et réalisation

Bob Stuurman

La construction du « Straight » est (quasiment) un jeu de (grand) enfant. En fait, une version stéréo requiert 2 platines d'amplificateurs, une platine d'alimentation fournissant la haute tension et la tension de grille négative, une paire de transformateur de sortie et un transformateur d'alimentation. Nous avons développé 2 circuits imprimés pour faciliter la réalisation de cet amplificateur mais rien n'interdit d'envisager de travailler « à l'ancienne » à l'aide de plots de soudage.

Le châssis comporte 2 parties : un bac en forme de U en aluminium recouvert d'une plaque d'aluminium. Le bac sert de support aux transformateurs de sortie sur le dessus et au transformateur d'alimentation sur le dessous. Le seul poids des transformateurs dépasse les 8 kilos, ce bac donnant la rigidité suffisante au châssis. La branche arrière du bac se trouve à la même hauteur que l'arrière de la plaque; c'est à cet endroit que se trouvent tous les connecteurs et la commande de volume « principale » (*master*). L'utilisation d'une entrée secteur de type euro à filtre secteur, interrupteur marche/arrêt et porte-fusible intégrés, nous limitons au strict indispensable le câblage à effectuer au niveau du secteur. Il n'est pas nécessaire de prévoir d'ampoule de visualisation de la présence de la tension d'alimentation vu que lorsque l'amplificateur est sous tension les tubes l'indiquent à l'évidence.

Mesures de sécurité

Cet amplificateur véhicule des tensions élevées létales. Les condensa-

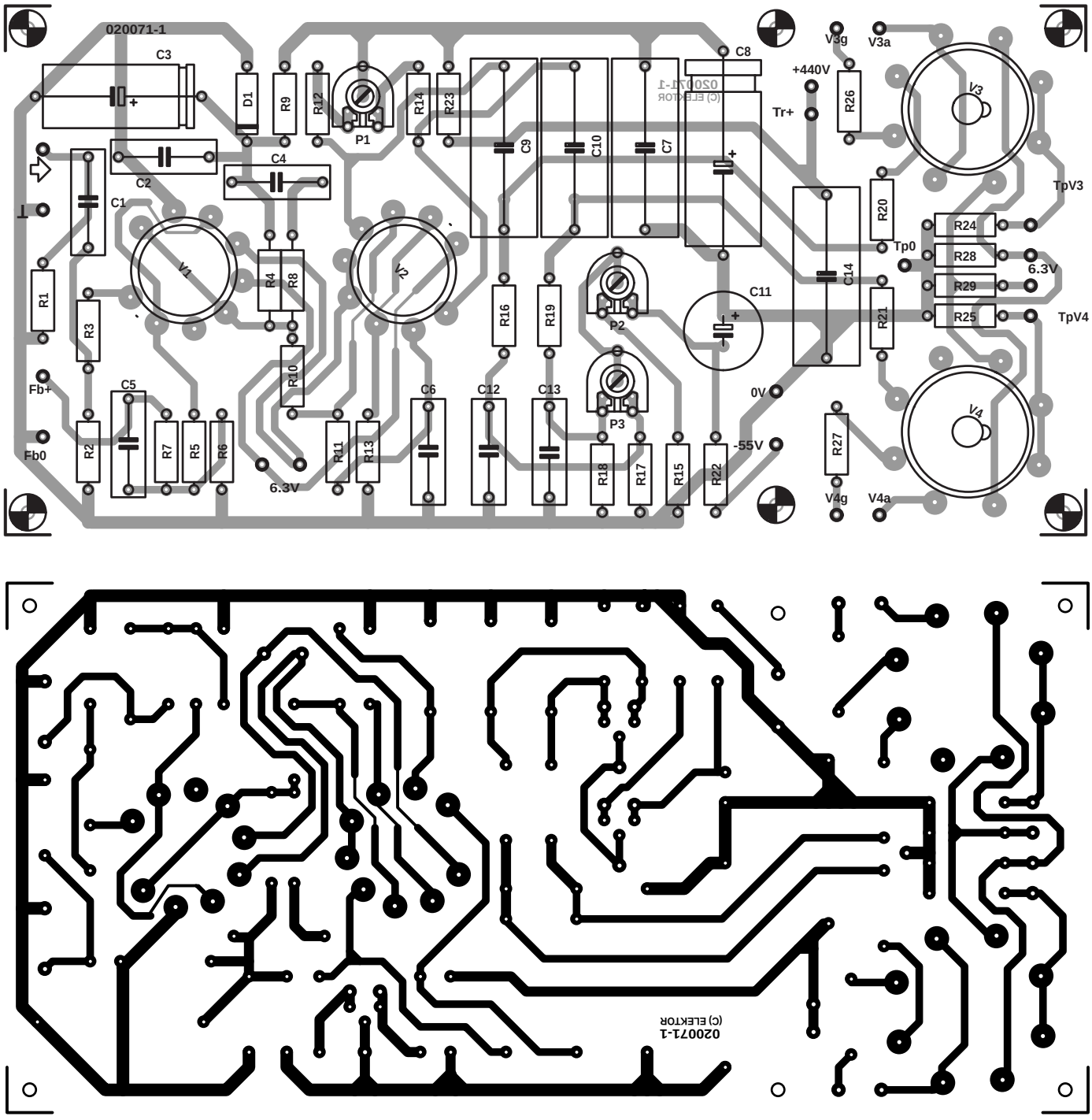


Figure 1. Dessin des pistes et sérigraphie de l'implantation des composants de la platine de l'amplificateur en version mono.

teurs électrochimiques présents sur la platine de l'alimentation ont une capacité importante de sorte qu'il faut, après la coupure de l'alimentation, un certain temps (et un temps certain) avant que la haute tension ait chuté à une valeur ne présentant plus de danger. Il faudra partant, pendant la période des essais, brancher 2 ampoules à incandescence de 220 V/15 W en série sur la haute tension. Dans ces conditions, dès la coupure de la tension du secteur les condensateurs se déchargent en quelques secondes. Elles n'ont pratiquement aucune influence sur le fonctionnement de l'amplificateur.

Réalisation des platines

La **figure 1** vous propose le dessin des pistes et la sérigraphie de l'implantation des composants du circuit imprimé de l'amplificateur. Le seul élément à ne pas y trouver place est le transformateur de sortie. Il s'agit d'une platine simple face qu'il vous sera facile de réaliser vous-même, mais elle est également disponible toute faite auprès des adresses habituelles sous la dénomination **EPS020071-1**. Si vous décidez de réaliser une version stéréo de cet amplificateur il vous en faudra 2 exemplaires.
Pour toutes les connexions nous avons fait

Liste des composants du « Straight » par canal

Résistances :

(Toutes les résistances sont des Beyschlag MBE0414 ou BC Components PR-02, dim. 4 x 12 mm à film métal)
R1,R2,R11 = 1 MΩ
R3 = 4kΩ7
R4,R17,R18 = 47 kΩ
R5 = 390 Ω
R6,R22,R28,R29 = 100 Ω
R7 (HP 8 Ω) = 3kΩ3
R7 (HP 4 Ω) = 2kΩ2
R8 = 27 kΩ
R9 = 100 kΩ
R10,R26,R27,R30 = 1 kΩ
R12,R14 = 150 kΩ
R13 = 82 kΩ
R15 = 15 kΩ
R16,R19 = 390 kΩ
R20,R21 = 2kΩ2
R23 = 10 kΩ
R24,R25 = 10 Ω
P1 = ajustable 50 kΩ
P2 = ajustable 10 kΩ
P3 = ajustable 20 kΩ
(Tous les ajustables sont des Bourns de type 3386P)

Condensateurs :

(Ce sont tous, sauf mention contraire, des condensateurs film Wima MKS4)
C1 = 470 nF/100 V au pas de 15 mm
C2 = 100 nF/400 V au pas de 15 mm
C3 = 10 µF/350 V ou 450 V axial dim. 12 x 25 mm
C4 = 100 pF/630 V polypropylène dim. 5 x11 mm
C5 (HP 8 Ω) = 680 pF/630 V polypropylène dim. 5,5 x 15 mm
C5 (HP 4 Ω) = 1 000 pF/630 V polypropylène dim. 5,5 x 15 mm
C6,C12,C13 = 220 nF/250 V au pas de 15 mm
C7,C14 = 470 nF/630 V au pas de 27,5 mm
C8 = 10 µF/450 V, axial dim. 15 x 30 mm
C9,C10 = 100 nF/630 V au pas de 22,5 mm
C11 = 470 µF/63 V radial dim. 12,5 x 25 mm

Semi-conducteurs :

D1 = diode zener 200 V/1,3 W

Tubes :

V1 = EF86
V2 = ECC83
V3,V4 = EL34 appairés

Divers

2 supports encartables Noval (9 points) céramique
2 supports encartables Octal (8 points) céramique
Tr1 = transformateur équilibré de sortie Lundahl LL1620 P-P

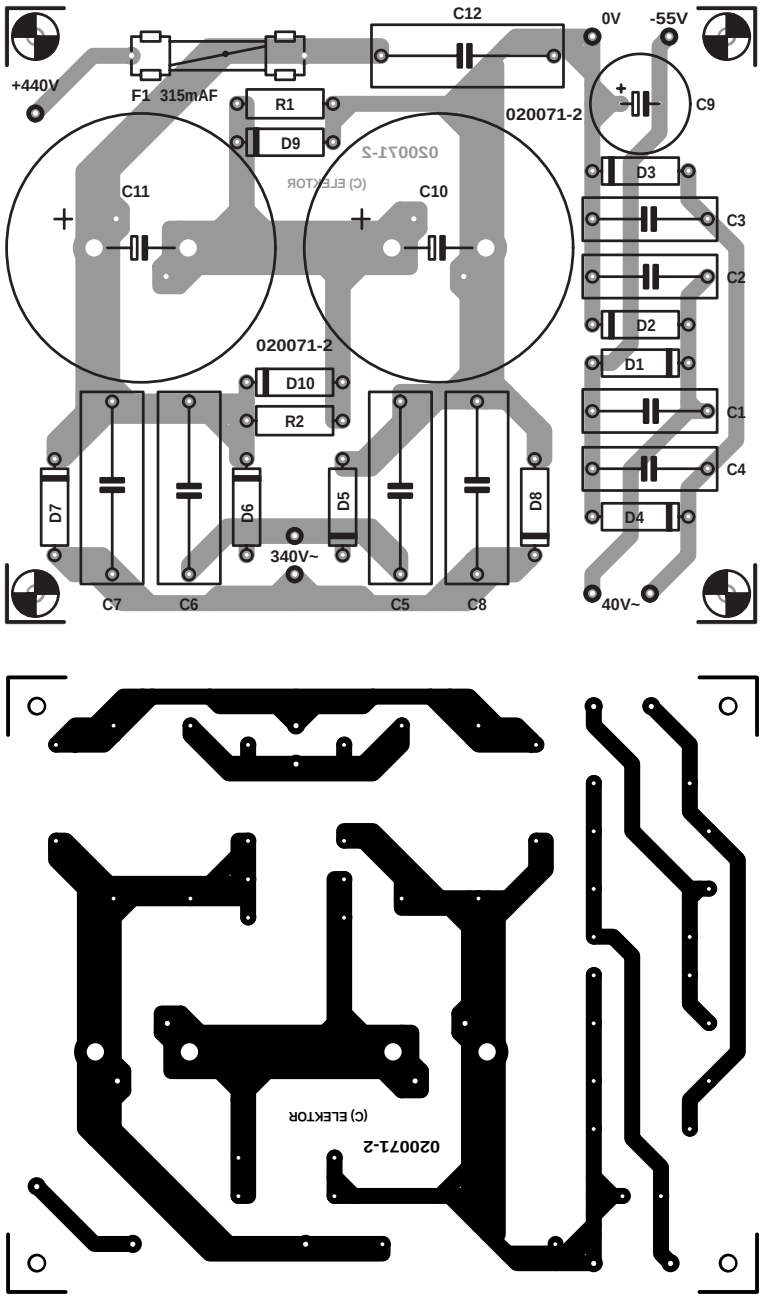


Figure 2. Dessin des pistes et sérigraphie de l'implantation des composants de la platine de l'alimentation.

appel à des picots de 1,3 mm de section et aux connecteurs correspondants. Nous avons utilisé, pour les tubes V1 et V2, des supports noval dont il existe 2 modèles, à savoir l'un en plastique, l'autre en céramique. La platine a été dessinée pour les versions céramiques.
Pour ce qui est de V3 et V4, les EL34, il s'agit de support octal en céramique. Ces supports possèdent des languettes de soudage de 2 mm de large et de 0,5 mm d'épaisseur. Il faudra, pour garantir une fixation solide des supports à la platine, élargir quelque peu les orifices présents dans la platine en utilisant un petit foret comme une sorte de fraise.
6 orifices de fixation permettent de fixer la platine solidement sur la plaque de base, la partie où se trouvent les supports des tubes de puissance étant ainsi encore mieux supportée.
Si tant est que l'on respecte les spécifications de la liste des composants la mise en place des différents éléments est très simple, ils d'adap-

tent tous parfaitement. Les résistances de type PR-02 de BC Components (Ex-Philips) ont une tolérance de 1% et possèdent 4 anneaux de couleur. Il est judicieux, comme il peut arriver que la lecture de la valeur soit délicate, de toujours vérifier sa lecture par une mesure à l'ohmmètre.
Les supports pour les tubes sont soudés côté « pistes ». On pourra, pour pouvoir fixer les contacts de façon optimale lors des opérations de soudage, enficher les tubes dans les supports. Faites bien attention, dans le cas des supports octal, au positionnement correct du repère vu que ces supports se laissent mettre dans pratiquement n'importe quelle position et qu'il devient pratiquement impossible de les enlever une fois soudés en place.
La platine simple face de l'alimentation (**figure 2**) est disponible aux adresses habituelles sous la dénomination **EPS020071-2**. Ici à nouveau nous avons utilisé, pour les connexions, des picots de 1,3 mm et les connecteurs correspondants. La réalisation de la platine de l'alimentation n'appelle pas de remarque particulière il suffit de faire attention à la polarité des diodes et des condensateurs électrochimiques.

La construction

Dans la partie inférieure gauche du plan de câblage de la **figure 3** nous indiquons les dimensions de la plaque du châssis et du bac. Ce dernier est fait d'une plaque d'aluminium de 370 mm de long et de 290 mm de large, que l'on plie en U, chaque plan vertical faisant 80 mm (le fond de l'U faisant partant 130 mm de large).
L'intérêt de ce châssis est que l'on peut travailler indépendamment l'un de l'autre sur le bac et la plaque. Certains des orifices existent et dans le bac et dans la plaque de sorte qu'il faudra, pour les effectuer, fixer (momentanément) ces 2 éléments l'un à l'autre. C'est la raison de la présence, sous les couvercles de protection des transformateurs, d'orifices-repères de fixation pouvant recevoir des boulons de 2 mm.
Il va nous falloir, pour l'étape suivante, des gabarits en papier, en papier transparent de préférence. Les gabarits des platines des amplificateurs et de l'alimentation pourront prendre la forme de photocopies des sérigraphies de l'implantation des composants, les dimensions des platines et les positions des orifices de fixation y étant indiquées. Il faudra effectuer, pour le transformateur de sortie y compris son couvercle de protection, un relevé des dimensions extérieures (qui sont en fait celle du couvercle) et des orifices à percer (pour le transformateur et le couvercle cette fois). Le gabarit du transformateur d'alimentation prend en fait la forme d'un cercle avec point central. Effectuez également des gabarits de l'entrée de tension secteur euro et du réglage de volume ALPS. Ces gabarits sont collés sur la plaque du châssis de manière à ce que les platines des amplificateurs se trouvent à 13 mm des bords avant et latéraux (espace nécessaire aux poutrelles de support du coffret). Les écrous de fixation des couvercles des transformateurs doivent eux aussi rester à l'intérieur du bac en U. Les transformateurs aux noyaux en forme de C se trouvent dans le prolongement l'un de l'autre. Le transformateur de l'alimentation vient se placer au milieu du bac en U.
Il est possible maintenant de centrer les orifices et de les percer. Chaque transformateur de sortie requiert 2 orifices servant au passage des conducteurs. Si on veille à disposer ces orifices à l'intérieur de la partie recouverte plus tard par le couvercle, ils ne seront pas visibles une fois l'amplificateur terminé.
Tout autour des supports des tubes, nous avons percé 6 orifices de 8 mm de diamètre destinés à permettre une circulation d'air, à des fins de refroidissement, les EL34 atteignant une température relativement importante.
Les conducteurs allant aux filaments sont placés dans un guide-câble collé à l'intérieur de l'un des angles du support en U (identifié par un « 1 » en bas à droite du plan de câblage). Il est prévu des orifices de passage sur les platines au niveau des points de connexion aux filaments.
Le câblage vers les points 0 V, –55+V et +440 V prend place dans un guide-câble disposé en position « 2 ».

Liste des composants de l'alimentation

Résistances :
R1,R2 = 47 kΩ (Beyschlag MBE0414 ou BC Components PR-02, dim.. 4 x 12 mm)

Condensateurs :
C1 à C4 = 100 nF/400 V au pas de 15 mm
C5 à C8 = 100 nF/1 000 V au pas de 22,5 mm
C9 = 470 µF/63 V radial au pas de 5 mm dim. 12,5x25 mm
C10,C11 = 470 µF/400 V radial au pas de 10 mm (tel que, par exemple, Roederstein série EYS)
C12 = 100 nF/630 V au pas de 22,5 mm

Semi-conducteurs :
D1 à D4,D9,D10 = 1N4007
D5 à D8 = BYW96E

Divers
porte-fusible encartable + fusible 315 mA/rapide
transformateur d'alimentation sec. 340 V/700 mA, 6,3 V/6,8 A et 40 V/0,1 A (tel que, par exemple, Amplimo 7N607)

Les transformateurs Lundah avec leurs couvercles sont disponibles, entre autres, chez Aquablue à Anvers (www.diyparadiso.com), mais peuvent également être commandés directement chez Lundahl (www.lundahl.se).

Canford Audio PLC (www.canford.co.uk) est également distributeur de Lundhal. Ils ont un bureau en France.
Les supports pour les tubes sont disponibles, entre autres, chez Aquablue, Conrad (www.conrad.nl) et Amplimo (www.amplimo.nl). Les tubes électroniques peuvent également être trouvés à ces adresses. Les résistances PRO2 sont vendues par DOS (De Onderdelen Specialisten www.dos.nl/index.html) et Farnell (www.farnell.com), les condensateurs MKS4 -par DOS, Conrad et Farnell.

Autres composants
un ensemble entrée secteur châssis euro à filtre secteur, porte-fusible et interrupteur marche/arrêt intégrés +fusible 1,5 A retardé
2 résistances NTC 5 Ω/5 W (Amplimo ou Conrad)
1 potentiomètre audio 2 x 100 kΩ log (tel que, par exemple, Alps type RK-27112) avec bouton
2 embases châssis Cinch (isolées)
2 bornes à fourche rouge (isolées)
2 bornes à fourche noir (isolées)
2 couvercles pour les transformateurs de sortie

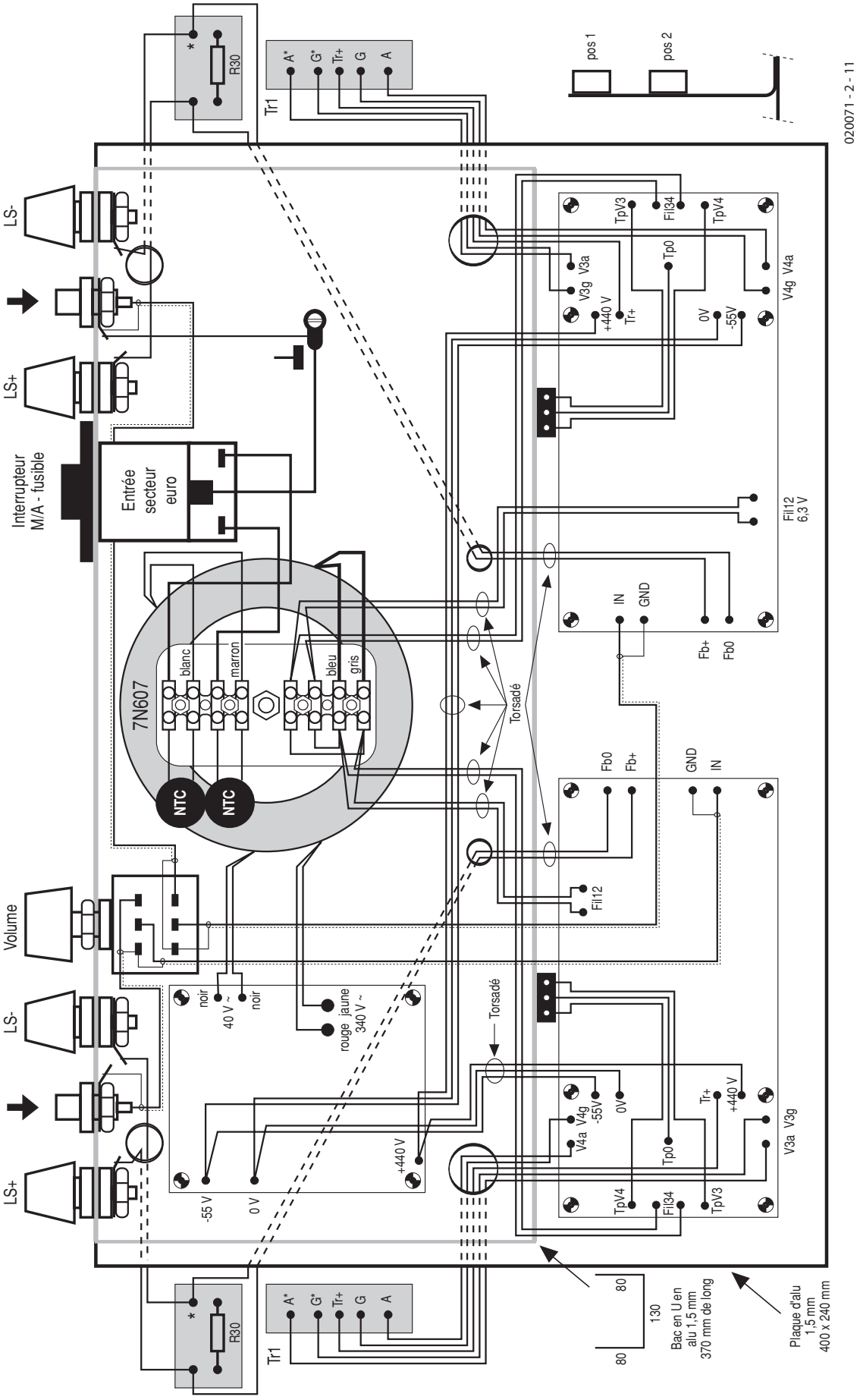
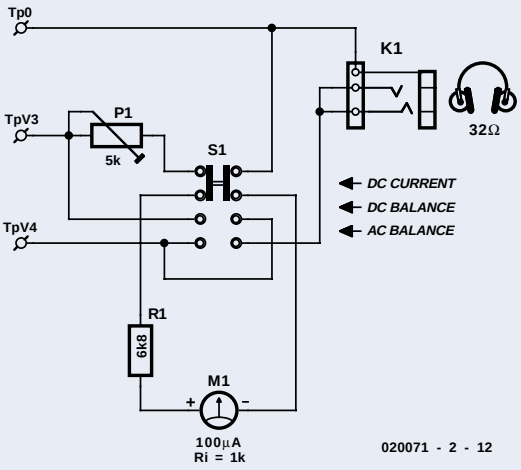


Figure 3. Exemple de câblage d'une version stéréo de l'amplificateur.

Réglages

Il faudra, lors du réglage de l'amplificateur et de mesures, brancher à la sortie haut-parleur (lire enceinte) une charge de 8 ou 4 Ω. On pourra donner à cette charge la forme d'une série de résistances de puissance montées sur un radiateur. Si l'on oublie de charger l'amplificateur on court le risque d'une surtension au niveau du transformateur de sortie avec comme conséquences possibles une destruction de ce dernier.
Les tubes de puissance ne sont pas auto-régulateurs (nous n'utilisons pas ici de résistance de cathode mais une résistance de grille négative), de sorte qu'il est recommandé de les acheter appariés par paire.
Il faudra procéder, et dans l'ordre donné, aux réglages suivants : DC current, DC balance et AC balance (respectivement courant CC, équilibre CC et équilibre CA (CC = continu, CA = alternatif). Le vieillissement des tubes en modifie les caractéristiques; il est partant judicieux, au début, de vérifier le réglage au bout de quelques semaines et plus tard tous les quelques mois.
Le courant au travers des tubes peut présenter de légères variations. Il est pour cette raison relativement délicat de procéder au réglage en utilisant un voltmètre numérique; il est préférable d'utiliser un instrument analogique à affichage à aiguille. Vu que le réglage est une opération répétitive au fur et à mesure des années, il n'est pas mauvais de prendre quelques dispositions facilitant ce processus. Nous allons utiliser un petit dispositif auxiliaire. À cet effet on fixera, à l'aide d'un morceau de scotch double face, à un endroit facilement accessible, et pour chaque étage, une embase à 3 contacts. Le contact central est relié au point de test Tp0, les contacts extérieurs l'étant respectivement aux points TpV3 et TpV4. Notre dispositif auxiliaire sera doté d'une embase femelle à 3 contacts à l'extrémité de son câble de connexion (qui ira s'enficher dans l'embase évoquée quelques lignes plus haut).



Chaque EL34 doit être traversée par un courant (anode + grille de protection) de 50 mA, la dissipation de chaque tube étant de quelque 22 W. On relève, aux bornes de la résistance de cathode de chaque tube, une tension de l'ordre de 0,5 V.
Il va falloir commencer par le réglage de notre dispositif auxiliaire. Appliquez aux points de connexion Tp0 et Tp3 du circuit représenté ci-contre, une tension continue de 0,5 V et ajustez, par action sur P1, l'affichage, sur l'instrument de mesure, à 50 (lire des mA au lieu des µA affichés). S1 doit se trouver en butée vers le chaut, dans la position DC current (courant CC).
On mesurera, dans la position DC balance (équilibre CC), la différence de tension entre les points TpV3 et TpV4. S'il circule, à travers les tubes, des courants identiques, l'instrument affichera « 0 ». Notons que la sensibilité est ici plus importante, vu que seule R1 fait office de résistance de limitation de courant.
Dans la position AC balance (équilibre CA) les points TpV3 et TpV4 sont interconnectés, un casque d'écoute branché sur le jack femelle K1 permettant d'entendre le signal.

Réglages de DC current et de DC balance
Mettez, sur chacune des platines des amplificateurs, les ajustables P1 et P3 et position milieu, P2 étant lui tourné en butée dans le sens anti-horaire, de manière à ce que la tension de grille ait sa valeur la plus négative possible. Mettez l'auxiliaire de réglage en position DC current et connectez-le à l'embase ajoutée à son intention sur l'amplificateur. Mettez l'amplificateur sous tension. Attendez quelques minutes et ajustez, par action sur P2, l'affichage de l'instrument de mesure à 40 mA, avant d'ajuster, après être passé en position DC balance (S1 en position centrale) l'affichage aussi bien que possible à « 0 » par action sur P3. Après une dizaine de minutes de fonctionnement on pourra augmenter à 50 mA le DC current et reprendre le réglage de la valeur de DC balance.

Réglage de AC balance
Le réglage de cet élément se fait, normalement, à l'aide d'un distorsiomètre. Mr Byrith a imaginé une solution pour effectuer ce réglage à l'oreille. Mettez le commutateur en position AC balance et appliquez à l'entrée de l'amplificateur un signal sinusoïdal de 1 kHz/100 mV_{eff}. Écoutez au casque le signal produit et jouez sur P1 jusqu'à ce que le signal de 1 kHz soit devenu le plus faible possible. Si l'on entend aussi le ronflement du secteur ainsi que des harmoniques du sinus et que le signal acoustique surfe sur des « vagues » il n'en reste pas moins possible de trouver une position dans laquelle le signal de 1 kHz est minimum. On trouve alors sur les cathodes le signal de phase opposé mais d'amplitude identique. Une excellente idée.

Réglage du signal carré
Le condensateur C5 pris dans la boucle de contre-réaction sert à la correction de l'évolution de la phase. S'il a une valeur trop faible, les coins du signal carré sont rabotés et, si sa valeur est trop importante, on constate des pics. Il faudra, pour cette mesure, disposer d'un générateur de signal carré et d'un oscilloscope.

Les platines des amplificateurs sont fixés sur la plaque à l'aide d'entretoises de 10 mm. À l'aide de rondelles on ajuste leur position de manière à ce que les supports des tubes de puissance arrivent en butée contre la plaque supérieure. La platine de l'alimentation est elle aussi fixée à l'aide d'entretoises. Les platines des amplificateurs sont séparées par une plaquette de protection en aluminium. Le réglage de volume ALPS sera encapsulé dans un blindage réalisé dans une tôle de faible épaisseur.

Premiers essais

Tant que les platines des amplificateurs n'ont pas encore été mises à leur place définitive il reste possible d'atteindre n'importe quel endroit. Il est judicieux, pour pouvoir en vérifier le bon fonctionnement, d'avoir déjà réa-

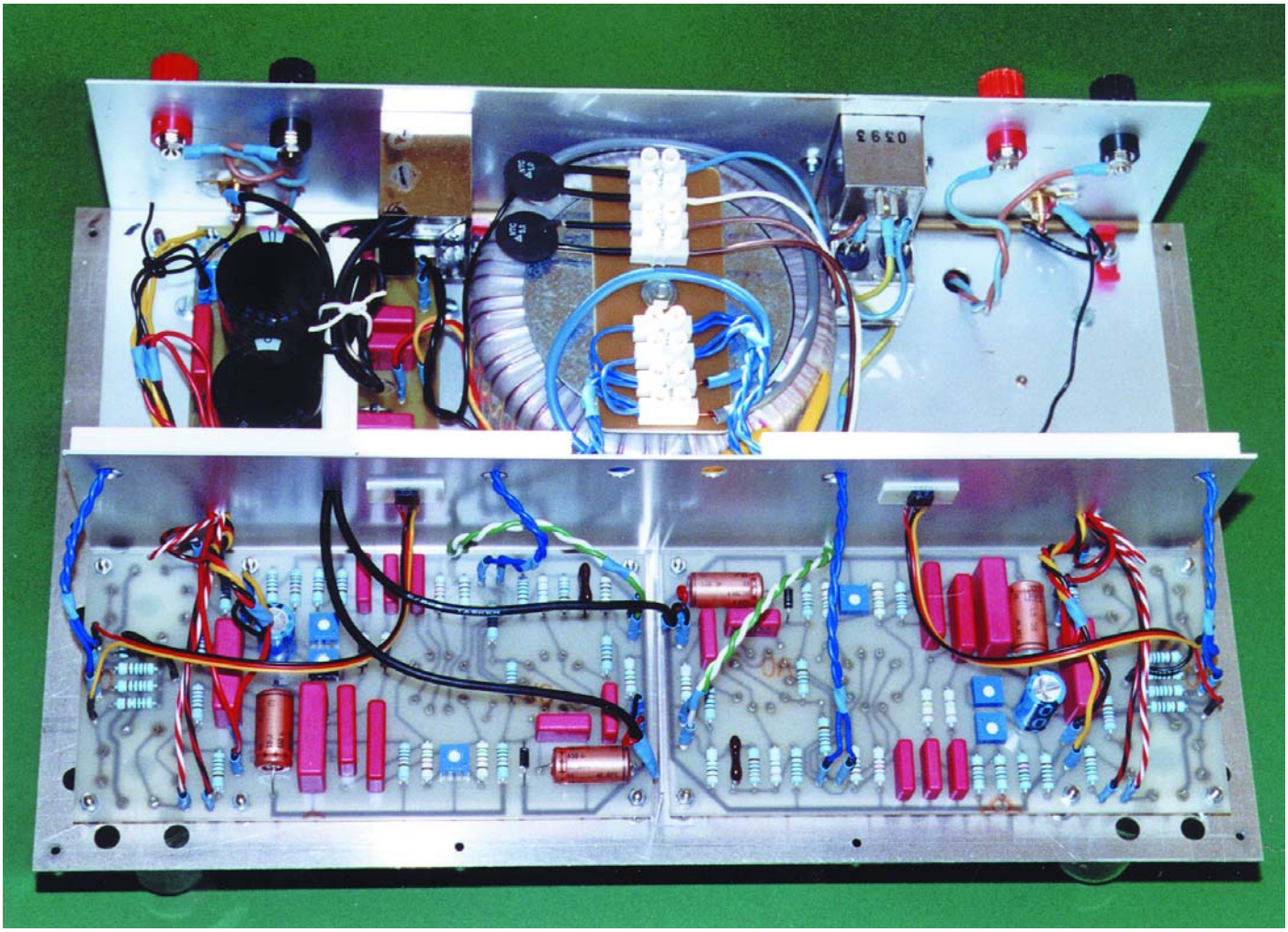


Figure 4. Vue plongeante sur le dessous d'un exemplaire terminé d'un amplificateur « Straight ».

lisé l'alimentation. Pour cela, nous allons monter le transformateur d'alimentation et la platine de l'alimentation dans le bac en U qui recevra également l'entrée secteur de type euro que l'on aura dotée d'un fusible de 1,5 A retardé. Sur une plaquette d'époxy nous avons placé une double rangée de 4 borniers fixés à l'aide de vis de 3 mm dont la tête biseautée vient s'encastrer sur le dessous de la plaquette. Cette dernière est fixée, par le biais d'un écrou additionnel, sur le boulon de fixation du transformateur d'alimentation. Les 4 borniers inférieurs servent à la connexion des filaments. La quasi-totalité du câblage (exception faite des conducteurs plus gros allant vers les bornes de connexion des haut-parleurs) prend la forme de câble souple d'une section de 0,5 mm^2 de différentes couleurs. Un bornier accepte sans le moindre problème jusqu'à 3 de ces conducteurs.

Les 4 borniers supérieurs servent à la connexion des conducteurs secteur du transformateur vers la prise secteur euro d'entrée. Chacun de ces conducteurs est doté d'une résistance NTC prise en série avec lui. Ces composants atténuent le choc de mise sous tension. Ils ne sont pas indispensables au fonctionnement correct de l'amplificateur mais constituent une solution à la fois simple et efficace de temporisation à la mise en fonction.

Une fois que l'on aura effectué le câblage entre l'entrée secteur, le transformateur d'alimentation et la platine de l'alimentation on pourra tester le fonctionnement de cette dernière. Pour cela on connecte les 2 ampoules à incandescence de 220 V/15 W prises en série aux lignes +440 V et 0 V avant de mettre le système sous tension. Si les ampoules s'allument on pourra vérifier, au voltmètre (et en prenant les précautions requises !), les valeurs de la HT et de la tension de grille négative.

Après avoir coupé la tension, on connectera la platine de l'amplificateur (sans la fixer dans le boîtier) à l'alimentation et on y reliera le transformateur de sortie. Attendez avant de brancher la HT et commencez par vérifier que les tubes s'allument. Dans le cas du EF86 il est possible de s'en assurer (encore que cela ne soit pas très évident) en regardant le tube par le haut. Sortez les tubes de puissance (après avoir coupé l'alimentation bien entendu) et connectez la HT. On pourra, après avoir remis l'ensemble sous tension et laissé le temps aux EF86 et ECC83 de chauffer, vérifier les tensions aux bornes des supports de ces tubes. Il se peut que les valeurs de tension relevées présentent de petites tolérances, mais si vous mesurez une valeur très différente il est fort probable qu'il y ait une erreur au niveau de l'une des résistances.

Si tout colle, il sera temps de remettre les tubes en place (après avoir coupé l'alimentation) et l'on pourra enfin procéder à un réglage

provisoire de l'amplificateur. Nous vous renvoyons, pour cette procédure, au paragraphe de l'encadré intitulé « Réglages ». On pourra, dans l'état actuel des choses, monter les platines des étages d'amplification dans le coffret et effectuer le reste du câblage.

Finissage

Un amplificateur tel que celui-ci requiert purement et simplement un joli coffret en bois. Nous l'avons réalisé en pièces de bois aggloméré (multiplex) de 9 mm d'épaisseur de manière à camoufler les orifices percés dans la plaque du châssis et le bas en U. Dans la partie arrière du coffret nous avons percé 2 orifices carrés destinés aux connecteurs et à la commande de volume. Le coffret est recouvert d'une fine couche de bois naturel mais rien n'interdit non plus d'utiliser des morceaux de bois massif pour réaliser le boîtier. Le dessous du bas est doté de supports autocollants.

Même s'il n'est pas doté d'un coffret, l'amplificateur constitue un ensemble solide grâce à son bac en aluminium. Si l'on dispose, sur le dessus des couvercles des transformateurs, des petites lattes que l'on y fixe à l'aide de scotch double face, on pourra, après l'avoir retourné et y avoir implanté les tubes, le poser ainsi sur une table. Il est possible ainsi d'avoir accès partout sur l'amplificateur et cela facilite la mise en place du coffret en bois.

On pourra le cas échéant fermer le dessous du coffret à l'aide d'une plaque d'aluminium (à mettre à la terre) que l'on aura dotée d'orifices d'aération.

Il est recommandé, dans une chaîne audio, de mettre l'amplificateur le dernier sous tension de manière à éliminer tout risque de bruit d'enclenchement (ploc ou autre crac).

(020071-2)

NdlR :

Nous ne pouvons pas ne pas ajouter ci-dessous le courriel qui nous est arrivé en toute dernière minute.

Chère Rédaction,
La lecture de votre premier article m'a poussé à vous envoyer cet E-mail avec une adresse qui me paraît très intéressante.
C'est un petit magasin à Paris qui a énormément de références de tubes anciens en stock.

D.M.
Société RADIO TUBES
40, Boulevard du Temple
75011 PARIS
Tél : 01 47 00 56 45
du lundi au jeudi de 14 h à 18h.
E-Mail : radio.tubes@laposte.net
www.radio-tubes.net/fr.html

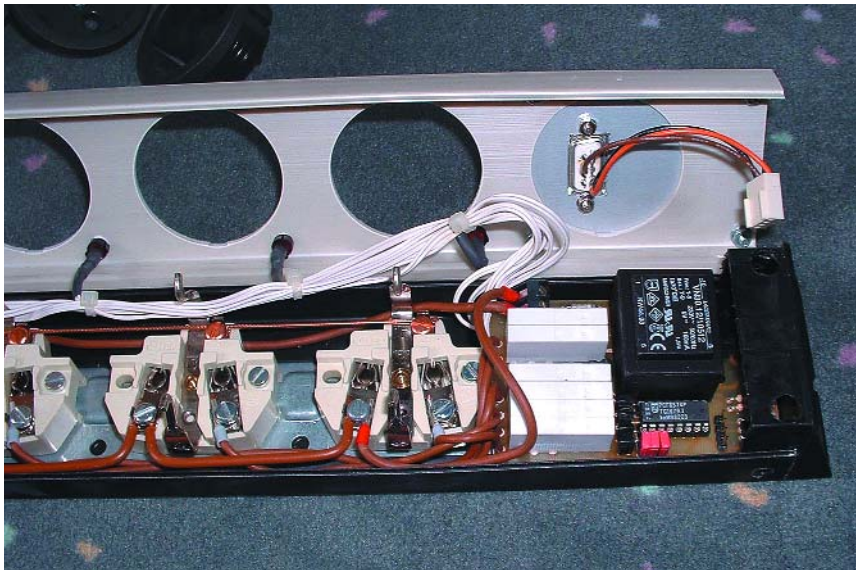
Multiprise pilotée par RS-232

Télécommande via RS-232 et I2C

Werner Frank

La présente électronique, que l'on pourra intégrer dans une multiprise, permet de commander un maximum de 6 prises. Leur pilotage se fait par le biais d'une interface RS-232, d'un composant pour bus I²C et d'un PCF8574P se chargeant du décodage des signaux RS-232.

Il est possible, sans avoir à faire appel à un matériel complexe, de réaliser une multiprise pilotée depuis un PC par son interface RS-232 pour peu que l'on dispose des pilotes (driver) adéquats. Nous vous proposons une version pour Windows, pour DOS et pour Linux. Le clou de cette réalisation : c'est un circuit intégré I²C qui se charge du décodage des signaux RS-232 et s'occupe, individuellement de chacune des prises secteur.



Expanseur d'E/S 8 bits

Le synoptique de la structure interne représenté en **figure 1** montre que le PCF8574 est, en principe, un convertisseur sériel/parallèle. En 3 mots, les données sérielles véhiculées par le bus I²C sont transférées, par le biais d'un registre à décalage, dans un verrou de sortie d'une largeur de 8 bits. Ce processus n'a en fait rien d'extraordinaire et peut également fort bien être implémenté à l'aide de circuits intégrés de logique standards.

L'approche à base de PCF8574 présente cependant certaines spécificités intéressantes et utiles. L'une d'entre elles est la possibilité d'adresser le composant I²C. L'utilisateur peut paramétrer à son gré 3 bits de l'adresse du PCF8574, les 4 bits de poids fort étant en tout état de cause à 0100 dans le cas du PCF8574 et à 0111 lorsqu'il s'agit d'un PCF8574A. Il est possible ainsi, théoriquement, de connecter au bus I²C jusqu'à 16 de ces circuits intégrés et de les piloter indépendamment l'un de l'autre.

Le PCF8574 possède un réseau de remise à zéro à la mise sous tension (*POWER ON RESET*) interne de manière à ce que le verrou de sortie se trouve, après l'application de la tension d'alimentation, dans un état parfaitement défini. Les signaux I²C traversent un filtre d'entrée (*INPUT FILTER*) et arrivent ainsi, débarrassés de parasites, jusqu'à la logique de commande du circuit intégré.

Les lignes de port P0 à P7 ne sont pas uniquement des sorties mais travaillent quasiment en mode bidirectionnel. Il est possible ainsi non seulement d'écrire les niveaux présents sur les lignes P0 à P7 en mode d'écriture mais aussi de les lire en mode lecture (possibilité non utilisée ici). sans que l'on n'ait besoin de signal de commande définissant la direction de circulation des données.

Le PCF8574 possède une sortie d'interruption. Cela permet de doter pour ainsi dire, le circuit, qui, en mode lecture, travaille en fait en esclave (*slave*), d'une sorte de fonction de

maître (*master*). Chaque changement d'état des lignes de port (flancs montants et descendants) produit une interruption que pourra enregistrer un microcontrôleur connecté au système. Sachant cependant que la présente réalisation n'utilise pas la sortie d'interruption nous pouvons nous passer d'entrer dans le détail de son fonctionnement.

Le reste du matériel

Un coup d'oeil au schéma de la **figure 2** permet de constater qu'il ne comporte guère de composant excitant si l'on fait abstraction du PCF8574. Une triplette de cavalier, JP1 à JP3 servent à définir l'adresse de base du circuit intégré. Si l'on est certain de son affaire (en ce qui concerne l'adresse de base s'entend) on pourra supprimer les résistances de forçage au niveau haut (pull up) R1 à R3 et relier directement les broches concernées à la masse (–) ou à la tension d'alimentation (+).
L'interface RS-232 du PC (reliée au circuit par le biais du connecteur K6) émule le bus I²C. Les seules lignes utilisées outre la masse sont DTR (*Data Terminal Ready*) et RTS (*Request To Send*). Les diodes assurent la conversion de niveau du ±12 V (RS-232) au +5/0 V logique dont a besoin le PCF8574 en tant que signal d'horloge et signal de donnée. Cette approche ultra-simple ne permet de circulation des données que dans un sens seulement, du POC vers le PCF8574, mais comme on n'a pas besoin d'autre chose ici...
Les 6 lignes d'E/S utilisées attaquent chacune, au travers d'un transistor de commande, un relais doté en parallèle d'une diode de protection montée en sens inverse. Les relais sont alimentés en quelque 12 V, tension dérivée directement de la tension secondaire redressée brute fournie par Tr1 et disponible en sortie du pont B1. Le PCF8574 est lui alimenté en +5 V régulés.

On pourra, comme l'illustre l'encadré en pointillés, doter chaque sortie de relais (collecteur du transistor de pilotage) d'une LED reliée au +12 V par le biais d'une résistance de 1kO2, de manière à visualiser l'état du relais concerné.

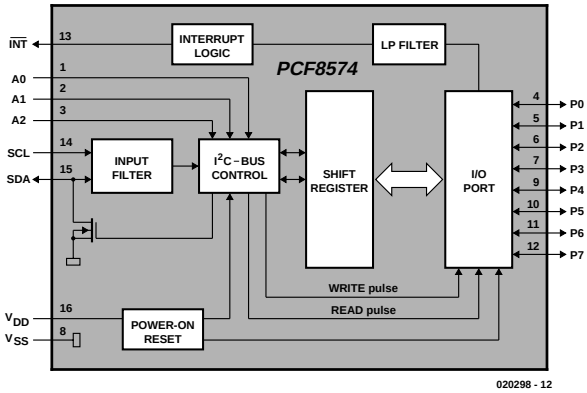


Figure 1. Structure interne du PCF8574.

Construction et mise en boîtier

Le dessin de la platine représenté en **figure 3** a été conçu de manière à pouvoir trouver place dans une multiprise à 7 prises secteur du commerce classique. Il va sans dire, mais il vaut peut-être mieux le préciser, que la multiprise en question doit être démontable. Pas question partant d'utiliser une multiprise à coquilles soudées même si elles sont moins chères. Il faut

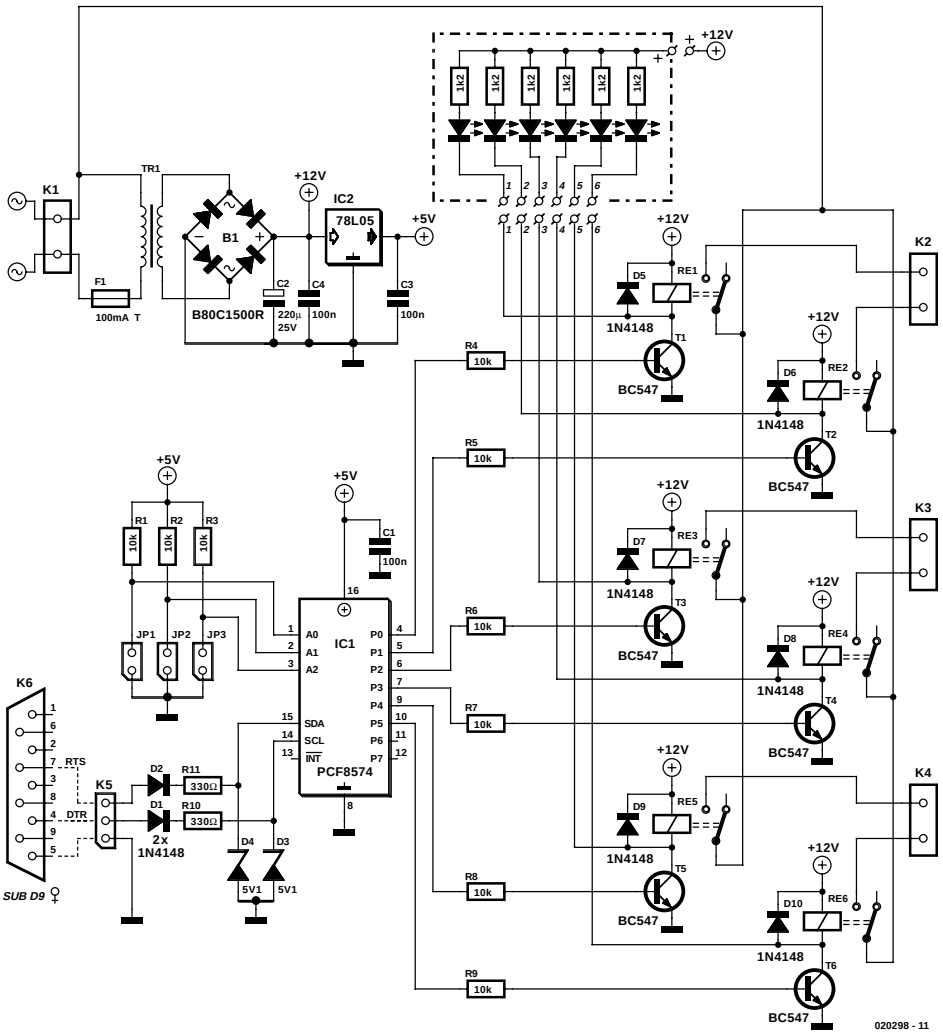


Figure 2. Le circuit intégré I²C entouré d'une interface sérielle et de 6 relais de sortie.

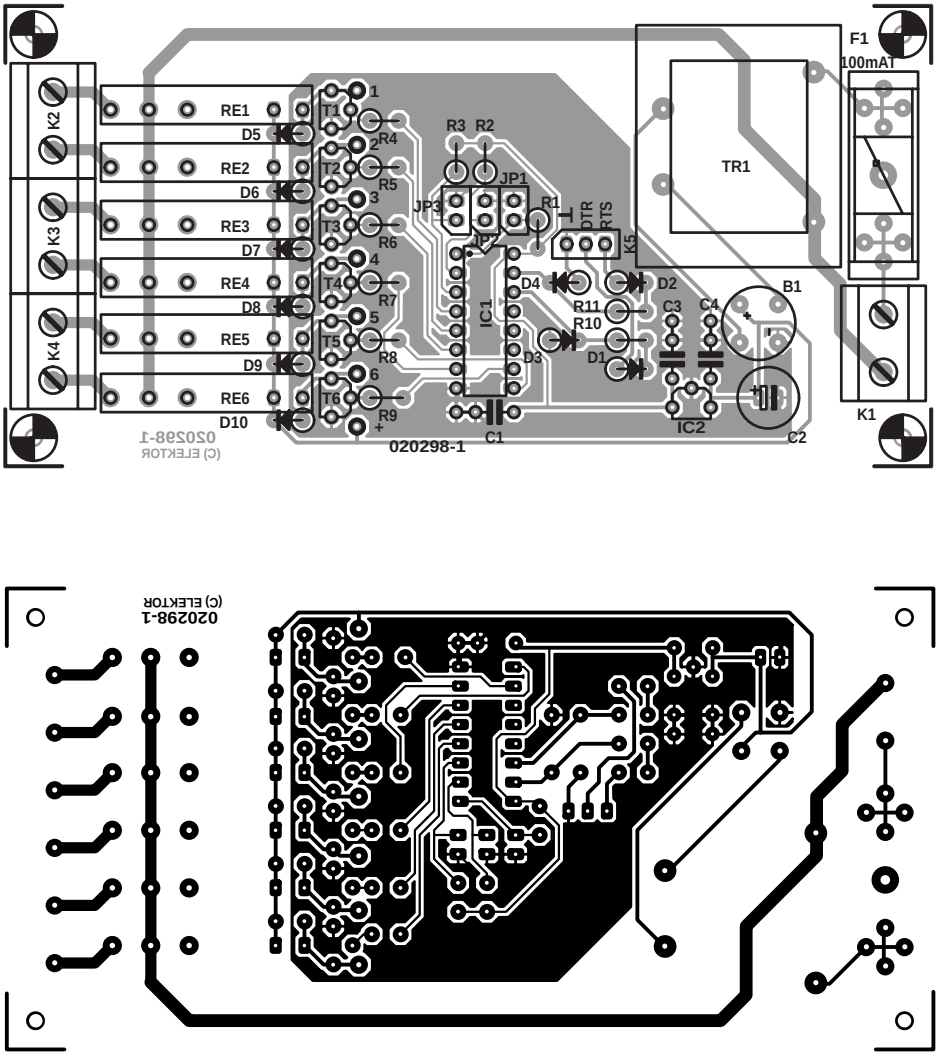


Figure 3. La platine se glisse tout juste dans le boîtier de la multiprise.

Liste des composants

Résistances :
R1 à R9 = 10 kΩ
R10,R11 = 330 Ω

Condensateurs
C1,C3,C4 = 100 nF
C2 = 220 µF/25 V radial

Semi-conducteurs :
D1,D2,D5 à D10 = 1N4148
D3,D4 = diode zener 5V1/500 mW
T1 à T6 = BC547
IC1 = PCF8574(A) (Philips)
IC2 = 78L05

Divers :
JP1 à JP3 = cavalier ou pont de câblage*
K1 à K4 = bornier encartable à 2 contacts au pas de 75, mm (RM7,5)
K5 = embase SIL à 1 rangée de 3 contacts
K6 = embase Sub-D 9 points châssis
TR1 = transfo 9 V/1VA5 tel que, par exemple BV EI 302 2021 (Hahn)
RE1 à RE6 = relais tel que Finder 34.51.7.012.00 (12 V) (Conrad RFA 504459)
F1 = porte-fusible avec capuchon et fusible 100 mA retardé
B1 = B80C1500 (rond)
en option 6 résistances de 1kΩ2 et 6 LEDs

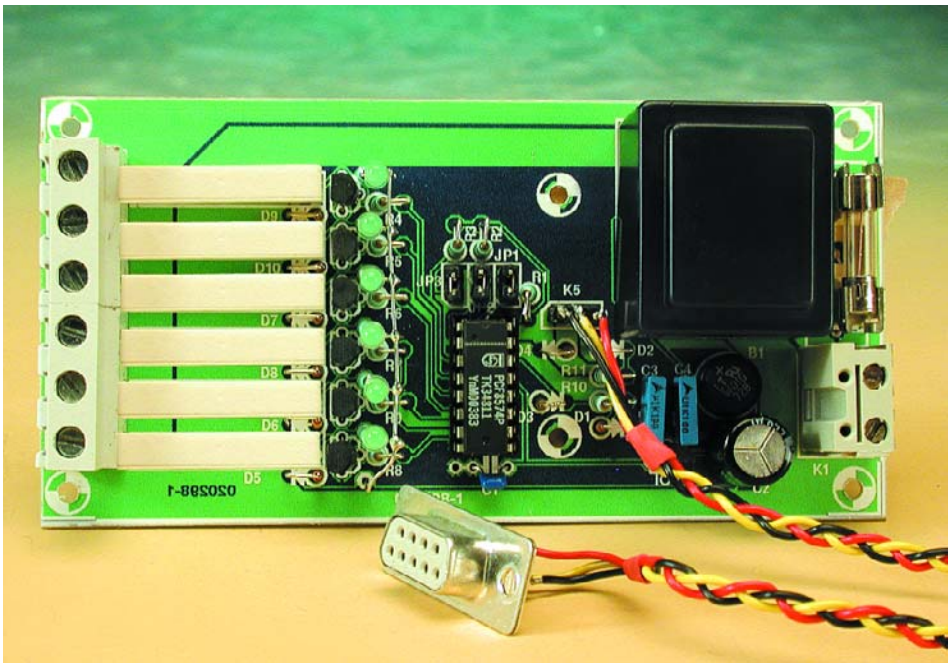


Figure 4. Exemplaire terminé de notre multiprise pilotée par RS-232.

en outre s'assurer que les différentes prises ne sont pas interconnectées par le biais de barres métalliques mais à l'aide de câbles. La première des prises sera démontée pour disposer de la place suffisante pour l'électronique. Les 4 orifices que comporte la platine permettront de la fixer solidement en place dans la multiprise.

La mise en place des composants sur la platine ne devrait pas poser de problème. Les composants passifs et les semi-conducteur s discrets sont tous montés verticalement. On pourra, nous le disions plus haut, remplacer les 3 cavaliers par autant de ponts de câblage mis, selon le cas, à la masse ou à la tension d'alimentation positive. Si l'on opte pour cette solution il ne faudra pas implanter les résistances R1 à R3. Les relais secteur très étroits de la

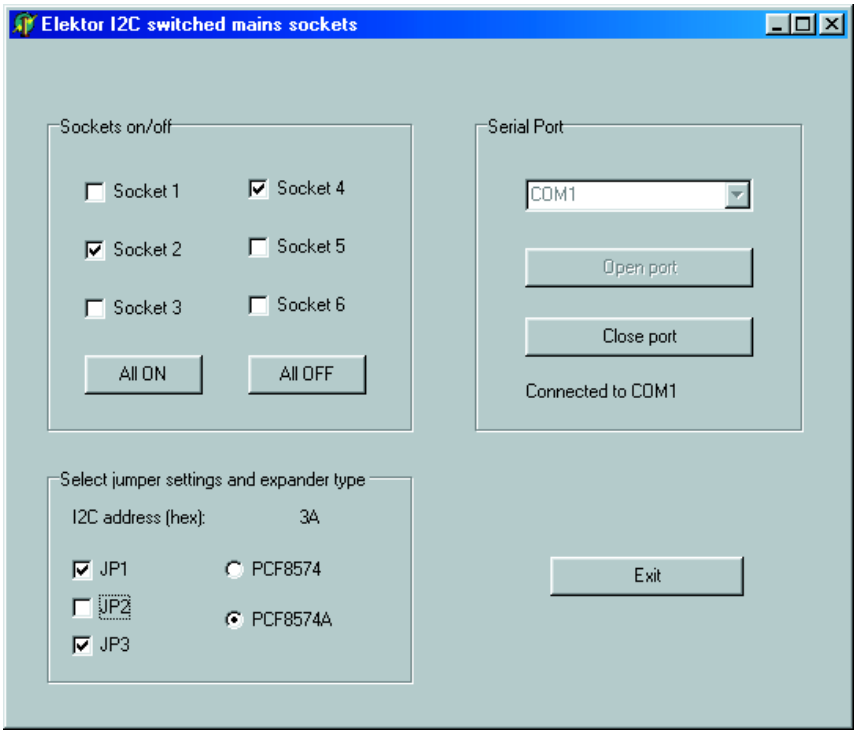


Figure 5. Commande de la multiprise par le biais d'une fenêtre Delphi5 tournant sous Windows.

société suisse Finder permettent une réalisation extrêmement compacte.

Les pilotes

Ce projet est accompagné de codes-source et de fichiers exécutables pour Windows (application Delphi), Linux et DOS (tous deux en C). Vous pouvez télédécharger tous ces programmes gratuitement depuis notre site Internet (www.elektor.fr) dans le mois concerné voire les obtenir sur disquette auprès des adresses habituelles (**EPS020298-11**).

Le coeur du programme est bien entendu la simulation du bus I²C à l'aide de 2 des lignes d'une interface sérielle de PC « normale ». Nous le disions plus haut, la ligne DTR pilote la ligne d'horloge I²C sérielle, SCL (*Serial CLock*), RTS pilotant la ligne de données I²C, SDA (*Serial DAta*). Comme il n'est pas question que le PC reçoive des données en provenance du circuit intégré I²C, il n'est pas prévu, au niveau du logiciel, d'accusé de réception (*aknowledge*) du mot de donnée reçu par le PCF8574.

Lors du lancement du logiciel PC (ou après envoi d'un paquet de données) le bus est au repos et les 2 lignes I²C se trouvent au niveau haut. Le maître lance la transmission par mise d'un flanc descendant sur la ligne DSA, SCL restant au niveau haut (conditions initiales). SCL passe ensuite au niveau bas. Au fur et à mesure des flancs montants du signal SCL on a émission successive des 7 bits d'adresse suivis d'un bit $\overline{R/W}$ de clôture. Le niveau de la ligne SDA ne peut (et ne doit) changer que pendant les passages au niveau bas du signal SCL. Dans le cadre de cette application, le bit $\overline{R/W}$ se trouve en permanence au niveau bas (« 0 » = Write). Normalement, l'esclave a la possibilité de confirmer la réception d'une adresse valide par l'émission d'un accusé de réception. Bien que cela ne soit pas prévu ici, le maître doit générer une impulsion d'horloge additionnelle que l'esclave interprète comme confirmation de l'accusé de réception.

L'esclave est ensuite paré pour la réception des 8 bits de donnée. Chaque bit est en relation directe avec l'une des sorties. On a encore ensuite une impulsion SCL de confirmation du pseudo-accusé de réception, à la suite duquel le maître clôture la transmission par une instruction d'arrêt (*stop*) qui prend la forme d'un flanc montant sur SDA pendant que SCL se trouve au niveau haut.

Pilotage

Le pilotage de l'interface sérielle sous Windows se fait par le biais des fonctions mises à disposition par la .dll Serial.dll que nous décrivons exhaustivement et vous proposons ailleurs dans ce numéro. Cette. dll (téléchargée ou sur disquette **EPS020388-11**) fait également partie de ce projet et devra être recopiée dans le répertoire \WINDOWS\SYSTEM. Le programme proprement dit, SOCKETS, fait appel aux fonctions SetDTR, Reset-DTR, SetRTS et ResetRTS. En outre, COMPortExist détermine quels sont les ports existants; OpenCOM/CloseCOM est elle aussi nécessaire en vue d'activer ou de désactiver le port auquel est connectée la multiprise.

La procédure la plus importante s'appelle SendI2C; elle envoie un octet de donnée à une adresse I²C spécifiée. Un coup d'oeil au fichier source permet de constater que tout ce dont il s'agit est d'assurer un pilotage correct des 2 lignes dans l'ordre convenable. Après chaque changement de niveau apporté aux lignes, on a inséré une petite pause de manière à maintenir la vitesse de transmission sur le bus à l'intérieur des limites (autorisées).

La **figure 5** reproduit la fenêtre du programme d'exemple en Delphi5, programme dont le code-source est disponible. Il ne requiert que peu d'explications. Tout débute par la sélection et l'ouverture du port COM souhaité; on choisit ensuite l'adresse du circuit intégré qui dépend et du type de composant (PCF8574 : adresse de base $20, variante A : adresse de base $38) que de l'état (ouvert ou fermé) des embases à cavalier JP1 à JP3. Une fois ce paramétrage effectué il sera possible d'activer (mettre sous tension) ou de désactiver (mettre hors-tension) chacune des prises. Lorsque l'on veut quitter l'application il suffira de clôturer le port sériel ou tout simplement d'appuyer sur le bouton « Exit ».

(020298)

Téléchargements

Vous trouverez, à l'adresse
www.elektor.fr/dl/dl.htm
les fichiers suivants de ce projet pour un télédéchargement :
– Fichiers de code-source et exécutable pour Windows (application Delphi), Linux et DOS (ces 2 derniers en C)
– Dessin des pistes au format .pdf

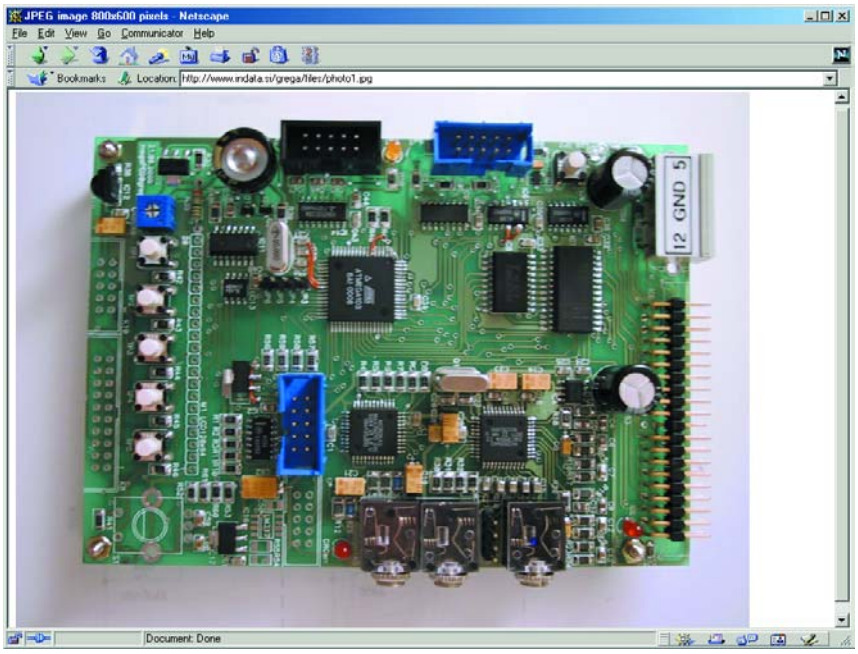
Lecteurs MP3 portables

Plus compacts grâce à de nouveaux circuits

Harry Baggen

Le format MP3 s'est fait, au cours des dernières années, une jolie place au soleil dans le monde de la musique. On trouve aujourd'hui nombre de modèles de lecteurs de fichiers MP3 à acheter, en version de salon ou portable, avec lecteur de CD, disque dur ou mémoire Flash. Mais réaliser son propre lecteur MP3 garde tout son attrait pour l'électronicien amateur.

Le format MP3 a déclenché une vraie révolution dans le monde de la musique. Grâce à son facteur de compression élevé il a permis une distribution rapide et facile de fichiers de musique par le biais d'Internet.
Mais l'utilisation de MP3 dans la voiture ou chez soi présente aussi un certain nombre d'avantages. Il devient ainsi possible de stocker des heures et des heures de musique sur un CD (NdlR : notre record actuel, 368 morceaux, plus de 25 heures de fichiers MP3 Pro sur un CD de 700 MB), les systèmes MP3 à disque dur intégré permettent eux de stocker une collection complète de CD (on parle de plus de 8 000 morceaux sur un DD de 20 Goctets, soit plus de 650 CD standard, une belle collection).
Il s'agit ici du troisième électronique en ligne consacré à ce sujet, mais il n'en est devenu, au fil des ans, que plus populaire. Grâce à l'arrivée de nouvelles puces de décodage MP3 telles que le populaire VS1001 du fabricant finlandais **VLSI Solution** [1], la conception de son propre lecteur de MP3 est devenue bien plus intéressante.
Lorsque l'on passe en revue les concepts de lecteurs MP3 de réalisation personnelle on constate que l'approche la plus prisée fait appel à une mémoire CompactFlash ou à disque dur de 2,5". Cela permet de réaliser un appareil compact qui, dans certains cas, mérite réellement le qualificatif de « portable ». Certains des modèles que nous avons découverts au cours de nos pérégrinations sur le Net sont aussi compacts et « design » que des appareils du commerce auxquels ils pourraient même se mesurer.



Le **Super-Simple pocket size MP3 player** [2] est une réalisation basée sur un PIC16LF877 de Microchip. Le circuit intégré de décodage est le VS1001k de VLSI Solution évoqué quelques lignes plus haut.
La mémoire prend la forme d'une carte CompactFlash. La taille de l'appareil ne dépasse pas celle d'un paquet de cigarettes; il s'agit d'un concept minimaliste sans le moindre luxe, ne comportant partant pas même d'affichage, mais l'électronique est bien en mesure de lire les fichiers MP3 présents sur la carte Flash. Le PIC garde suffisamment de ressource pour une éventuelle implémentation d'une telle fonction.
YAMPP (*Yet Another MP3 Player*) est un site pour réalisations MP3 qui existe depuis quelques années déjà. Le concepteur qui se cache derrière ce site est très productif. Il en est à son 7ème projet. Le **YAMPP-7** [3] est un joli lecteur compact aux dimensions minimum utilisant un contrôleur ATMega161 et un décodeur VS1001. Il intègre en outre une inter-

face USB et un chargeur Li-Ion. L'affichage, de même d'ailleurs que l'accumulateur Li-Ion a été récupéré sur un téléphone GSM. Le concepteur offre les différents composants et platine requis pour la réalisation de son projet voire plusieurs versions montées et testées.

Les « technautes » « cachés » derrière **MP3ar** [4] n'ont pas fait du surplace depuis la dernière fois que nous en avions parlé (septembre 2001). Ils proposent sur leur site un lecteur de MP3 portable à disque dur de 2,5". Le contrôleur utilisé ici est un PIC18C452, le décodeur étant à nouveau un VS1001k. Il est possible, pour de l'ordre de 100 $US, de commander un kit complet comprenant la platine dotée de ses composants, un affichage LCD et un CD-ROM avec logiciel. Il vous reste tout juste à acheter le disque dur.

Le lecteur **megaPEG** [5] de Gregor Horvat est un concept qui n'a pas été réactualisé depuis plus d'un an, mais les images placées sur son site permettent de découvrir une jolie petite platine à laquelle viennent se connecter un disque dur et un affichage graphique. Pour son projet, Horvat utilise un microcontrôleur ATMega103 et un décodeur MAS3507D.

Fallguy [6] est un concept universel que l'on pourra utiliser aussi bien chez soi, dans sa voiture, voire en portatif.

Le matériel prend la forme d'un 68HC912 et d'un STA013. Un affichage LCD de 128 x 64 pixels affiche les titres et informe quant aux organes de commande. L'une des spécificités de ce concept est la possibilité de le doter de 2 disques durs (maître/esclave), ce qui se traduit par une capacité maximale de 256 Goctets !

Le **Beatbox 2002** [7], s'il ne fait pas partie de la catégorie des mini-lecteurs, n'en est pas moins un appareil compact. Il possède les dimensions d'un disque dur de 3,5" et se compose d'une platine, d'un disque dur de 2,5"et d'un affichage graphique de bonnes dimensions à commande tactile. Le microcontrôleur utilisé est un MC68332 de Motorola, épaulé par un décodeur du type VS1001.

Une interface USB-2.0 garantit un transfert rapide des fichiers MP3. Ce lecteur offre de nombreuses possibilités d'autant plus qu'il est capable de lire différents formats audio (MPEGI et II layer 1, 2, 3, plus MPEG2.5).

Le **IMAP2** [8] de David Freudenberger ne mérite pas le qualificatif de portable, mais n'en est pas moins un lecteur autonome pouvant se voir connecter un lecteur de CD ou un disque dur. Son pilotage est confié à un PIC, le traitement MP3 est l'affaire du fameux MAS3507D de Micronas.

Le **MP3 Player** [9] de Codepuppies est un concept similaire (platine avec disque dur ou lecteur de CD + affichage, le tout dans un boîtier compact). La dernière réactualisation de ce site date cependant d'il y a 18 mois, raison pour laquelle il est fortement recommandé, avant de se lancer dans ce projet, de s'assurer de la disponibilité des composants utilisés. Les composants-clé de ce projet sont un PIC et un MAS3507D.

Le site **mp3projects.com** [10], que nous mentionnions voici 2 ans reste la meilleure source pour se faire une idée de tous les projets MP3 actuels à réaliser soi-même proposés sur Internet. Ce panorama, qui occupe plusieurs pages, a été subdivisé en 3 groupes : autonome (*stand-alone*), portable et à base de PC. L'auteur de ce site propose, pour 20 $US, un CD-ROM où sont rassemblés quantité de logiciels pour une réalisation personnelle d'un lecteur de MP3. On y trouve, entre autres, des fiches de caractéristiques, des assembleurs/désassembleurs, des logiciels de dessin de platine, des émulateurs ainsi que des codes-source et des routines utilisables pour différentes fonctions.

(035026)

Adresses Internet

[1] VLSI Solution: www.vlsi.fi/
[2] Super-Simple pocket size MP3 player: www.walrus.com/%7Eraphael/html/mp3.html
[3] YAMPP-7: www.myplace.nu/mp3/index2.htm
[4] MP3ar: www.mp3ar.com/
[5] megaPEG: www.indata.si/grega/megapeg.htm
[6] Fallguy: www.loetronic.com/home-e.htm
[7] Beatbox 2002: www.beatbox2002.de/
[8] IMAP2: www.chez.com/imap3/index2_e.html
[9] MP3 Player: www.codepuppies.com/~ben/sens/pic/mp3/
[10] mp3projects.com: www.mp3projects.com/

Picomire

La mire vidéo à 10 €

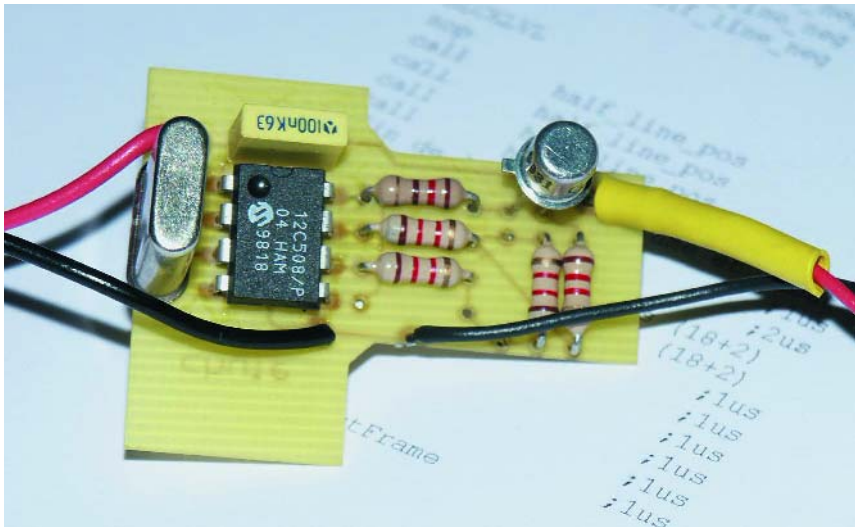
Florent Simonnot

Qui n'a jamais eu besoin de synchroniser un poste de télévision ou un moniteur vidéo, mais ne disposait pas des outils nécessaires ? La mire vidéo à 10 € à PIC tombe alors, c'est le cas de le dire, à pic.

Le présent montage est en fait un générateur de signal vidéo rustique permettant de faire synchroniser un téléviseur ou un moniteur vidéo dès lors que l'appareil en question est doté d'une entrée vidéo composite. Les timings et niveaux nécessaires sont synthétisés à l'aide d'un microcontrôleur du type PIC12C508.

Nombreux sont les magnétoscopes capables d'enregistrer l'audio en stéréophonie. Il est partant intéressant d'exploiter cette possibilité pour enregistrer une émission de radio dans de bonnes conditions et ce, de manière programmée avec le timer. Malheureusement, les magnétoscopes récents ne « voyant » pas de signal vidéo à leur entrée, commutent sur un fond de couleur afin de ne pas enregistrer de parasites et ce faisant coupent du même coup l'audio !

Il ne faut que peu d'électronique pour générer les tops synchro de lignes et de trames capables de tromper le magnétoscope en lui faisant croire qu'il a affaire à une source de signal vidéo en bonne et due forme, et puis, tant que nous y sommes, pourquoi ne pas visualiser à l'écran autre chose qu'un fond noir ? De plus, on disposerait ainsi q'un petit outil simple, fiable et autonome pour la recherche de l'origine d'une panne vidéo.



Le schéma

Un PIC12C508 est-il en mesure de générer un signal vidéo simple ? Pour rappel, comme l'illustre le croquis de la **figure 1**, une ligne de vidéo dure 64 µs; elle débute par un top synchro d'une longueur de quelque 5 µs à la suite duquel on trouve le palier avant, de 5 µs environ lui aussi, puis le signal vidéo proprement dit suivi d'un palier arrière d'une durée de 2 µs environ (les *burst* de synchronisation ne sont bien sûr pas générés dans le présent montage). Si nous faisons un petit calcul nous constatons que la longueur de la ligne « visible » est de :

64 –5 –5 –2 = 52 µs.

Même si l'on ne peut pas dire du PIC12C508 C508 qu'il soit un foudre de guerre quant à sa vitesse, ses performances, s'il est cadencé au maximum de ses possibilités par le biais d'un quartz de 4 MHz, sont telles qu'il exécute une instruction par microseconde, ce qui est parfaitement suffisant pour remplir la tâche que nous allons lui confier. De plus, son architecture RISC (*Reduced Instruction Set Computer*) le rend tout particulièrement adapté à une programmation synchrone.

Les niveaux requis pour le signal sont au nombre de 3 : le niveau d'effacement ou de top, le niveau du noir et le niveau de blanc en respectant un rapport d'amplitude entre les différences topologie noir et noir/blanc de 33 et 66% sous 1 V et avec une impédance caractéristique de 75 Ω. Ceci établi et compte tenu du fait que le PIC12C508 ne possède pas de CNA (Convertisseur Numérique/Analogique), il va falloir trouver une technique pour l'en doter. Heureusement que la pratique est moins compliquée que ne le donnerait à penser la dénomination de CNA. Quelques résistances font parfaitement l'affaire et permettent d'avoir un niveau visible entre le blanc et le noir. En uti-

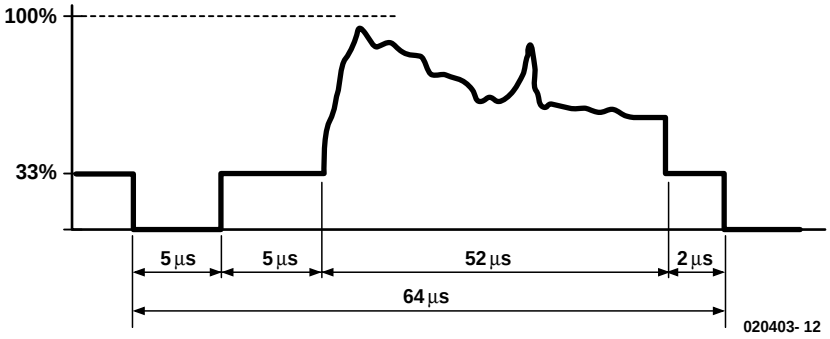


Figure 1. Chronodiagramme d'une ligne vidéo.

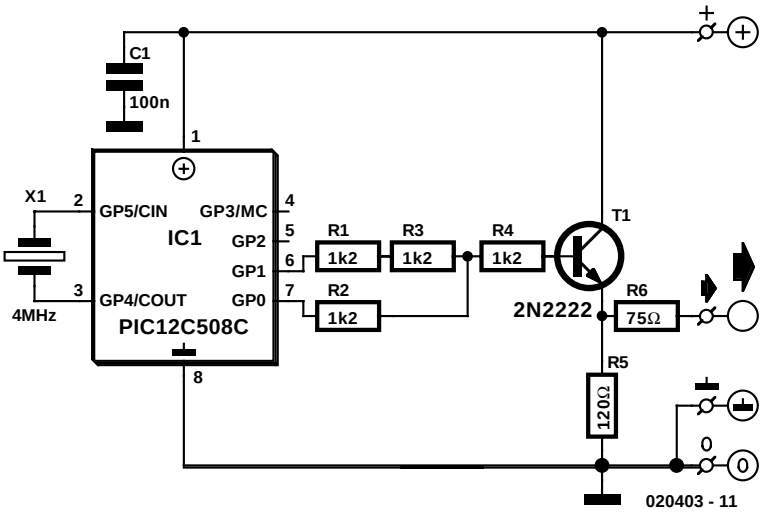


Figure 2. L'électronique de Picomire se résume à très peu de choses.

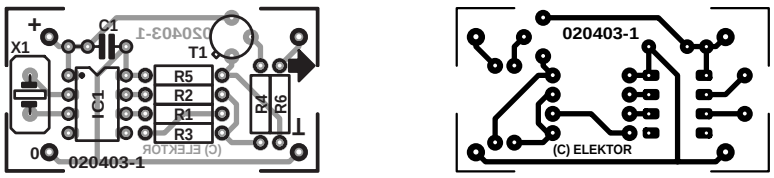


Figure 3. Dessin des pistes et sérigraphie de l'implantation des composants de la platine dessinée à l'intention de Picomire.

Liste des composants

Résistances :
R1 à R4 = 1kΩ2
R5 = 120 Ω
R6 = 75 Ω
R5 = 4kΩ7

Condensateurs :
C1 = 100 nF

Semi-conducteurs :
T1 = 2N2222
IC1 = PIC12C508C (Microchip)

Divers :
X1 = quartz 4 MHz

lisant 2 sorties du PIC12C508 nous aurons une loi de réponse en sortie du réseau telle que, dès lors que l'on choisit des résistances ayant toutes une valeur de 1,2 kΩ, V_{out} = 1/3 VCC (2MSB+LSB) avec le MSB en broche GP0 et le bit de poids faible (LSB) en broche GP1. Si nous utilisons une alimentation par pile 4,5 V, cela nous donne les niveaux suivants : 0 V pour le top, 1,5 V pour le noir, 3 V pour le gris, 4,5 V pour le blanc, valeurs qui correspondent exactement au cahier des charges qui définissait un top égal au 1/3 de l'amplitude totale du signal et un rapport de 66% pour le blanc.

Le reste de l'électronique

Un circuit d'adaptation d'impédance basé autour d'un montage transistor-suiveur (collecteur commun) permet d'abaisser la résistance de sortie du convertisseur tout en fournissant une impédance proche de 75 Ω pour le signal vidéo. Il n'en faut pas plus pour générer un signal vidéo noir et blanc (N&B) de bon aloi. Nous en avons terminé avec l'électronique, il est temps de nous intéresser d'un peu plus près au coeur de ce montage, le PIC.

La platine

Un rapide coup d'œil au dessin de la platine représenté en **figure 2** aura vite fait de vous convaincre qu'il existe des différences sensibles entre le dessin de l'auteur (cf. la photo en tête d'article) et le nôtre. L'absence de la résistance inférieure sur le prototype de l'auteur s'explique par l'implantation de cette dernière directement dans le fil de sortie (caché ici par la gaine thermorétractable).
La platine n'est pas disponible par les canaux habituels. Il vous faudra utiliser le dessin des pistes proposé ici (voire celui que vous aurez télédéchargé depuis notre site (adresse : www.elektor.fr) voire celui que vous aurez dessiné vous-même à l'aide de votre programme de CAO (saisie de schéma + dessin de platine) pour (faire) réaliser votre propre platine.

Le programme du PIC

Comme nous le disions plus haut, la programmation du PIC doit être synchrone car le noyau du PIC12C508 (si toutefois on peut encore, vu la taille de ce dernier, parler de noyau) ne peut être interrompu (INT) et quand bien même il aurait pu l'être, il n'aurait pas été assez rapide pour traiter une interruption : à 4 MHz, le PIC effectue une opération toutes les microsecondes (µs). Quelles performances peut-on en attendre (c'est-à-dire quel est le plus petit motif affichage avec ce microcontrôleur) ? Une ligne visible dure quelque 52 µs, soit le temps correspondant à l'exécution de 52 instructions. Comme il n'est pas possible, en raison de la structure du convertisseur N/A, d'utiliser l'instruction de manipulation de bit (BCF et BSF), il nous faudra utiliser un movlw xx suivi d'un

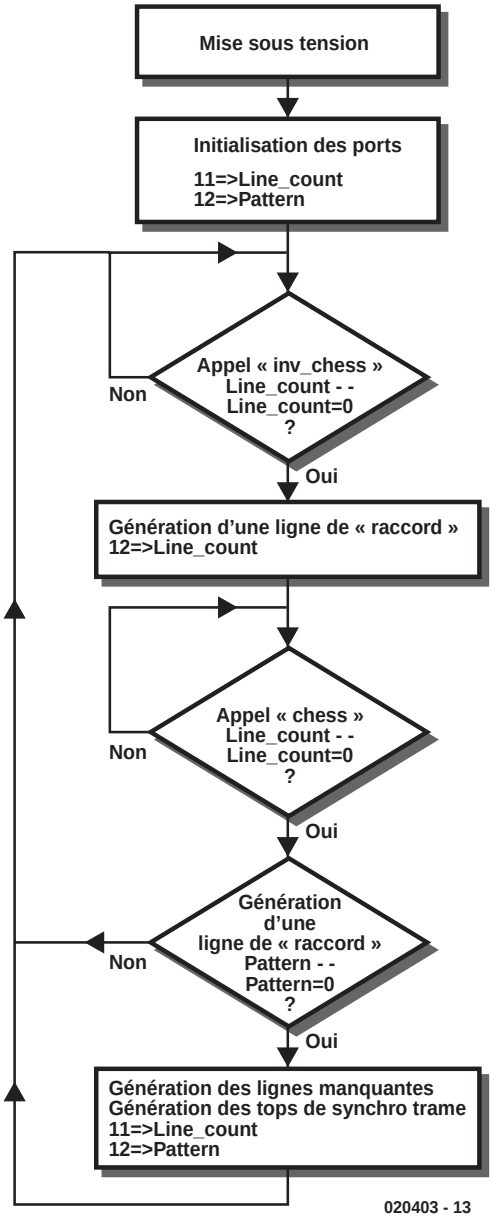


Figure 4. Ordinogramme du programme grillé dans le PIC.

movwf GPIO, soit 2 µs pour passer d'un niveau à l'autre. La taille minimum d'un « pixel » affichable sera donc de 625*(4/3)*2/52 = 32 colonnes-images; comment en sommes-nous arrivés à ce nombre ? Avec un rapport 4/3 on obtient pour 625 lignes 830 colonnes. 830 colonnes en 52 µs cela nous donne 16 colonnes par µs et donc 32 colonnes pour 2 µs. Comme on peut s'en douter, si l'on connaît les normes de TV, la boucle principale du programme dure 20 ms, ce qui correspond à la période d'une trame (ou une demi-image). L'auteur a choisi l'affichage d'un damier de rectangles alternativement blancs et noirs, ceci afin de pouvoir utiliser la Picomire en 2 occasions courantes lors d'un dépannage « en campagne » : la vérification des convergences d'un tube douteux et la vérification des étages d'attaque des canons.

Tableau I. Décomposition d'une ligne vidéo

Temps (µs)	niveau	instruction	commentaires
–1	noir	macro TOPLVL	fin de la ligne précédente
0	top		début du top synchro ligne
1	top	return	il dure 7 µs
2	top		
3	top	decfsz line_count	
4	top	goto $–2	
5	top		
6	top	macro BLCKLVL	fin du top synchro ligne
7	noir		début du palier arrière
8	noir	call inv_chess	il dure 5 µs
9	noir		
10	noir	nop	
11	noir	macro WHITELVL	fin du palier arrière
12	blanc		début de la ligne visible
13	blanc	macro BLCKLVL	elle dure 52 µs
14	noir		
15	.....		
62	blanc	macro BLCKLVL	
63	noir		palier avant
64	noir	macro TOPLVL	il dure 2 µs
65	top		

Le programme proprement dit se décompose, principalement, en plusieurs routines d'attente calibrées afin d'économiser de la mémoire en mémoire de programme (ROM) et 2 routines de synthèse de lignes vidéo, l'une baptisée « chess » et l'autre « inv_chess » qui grâce à l'utilisation de macro-commandes vont permettre l'affichage à l'écran, en alternance, de traits blancs puis noirs, complémentaires l'un de l'autre.

Nous vous proposons, en **figure 4**, l'ordinogramme du programme grillé dans le PIC et disponible sur notre site Web et auprès des adresses habituelles sous la dénomination **EPS020403-11**.

Remarques

Ce montage prend quelques libertés avec le standard vidéo : en effet, les dites normes font une différence entre trame paire et trame impaire dans l'organisation des demi-lignes et demi-lignes inversées qui servent à déclencher le retour du spot du téléviseur; ici c'est toujours la même trame qui est générée, mais aucun problème particulier ne devrait apparaître. Une autre liberté a été prise avec le standard en ce qui concerne la structure de la ligne elle-même : en effet le top ligne dure 7 µs (au lieu de 5 µs), adaptation imposée par la structure du PIC.

Le **tableau 1** illustre la structure d'une ligne vidéo.

La réalisation

La mise en place des composants n'appelle pas de remarque particulière. On pourra bien entendu adapter cette réalisation à ses besoins propres. Une fois que vous aurez terminé la réalisation de Picomire, il vous restera à l'essayer. Après avoir branché une pile de 4,5 V aux lignes disposées sur la gauche du quartz, attention à la polarité, il restera à connecter les 2 conducteurs de sortie, signal et masse, aux contacts correspondants d'une fiche Cinch qui viendra s'enficher dans l'embase Vidéo que comportent la plupart des téléviseurs et magnétoscopes actuels. (Si vous voulez passer par le biais d'une prise Péritel, il faudra utiliser les contacts 20 pour la vidéo et 17 pour la masse). Si, après mise sous tension de Picomire et du téléviseur à tester, vous obtenez l'écran représenté en **figure 5**, c'est que votre montage fonctionne correctement (électroniquement et logiciellement).

Mieux connaître le PIC12C508

Le PIC12C508 de Microchip est, avec ses 8 boches seulement, l'un des plus microcontrôleurs les plus petits que l'on puisse trouver actuellement sur le marché. La famille des PIC12C5XX comprend, outre le 508, les 509 et 518. Le 508 possède une EPROM de 512 mots de 12 bits et une RAM de 25 mots. Le set d'instruction de cette famille se limite à 33 instructions seulement. Le 509 possède une EPROM de 1 024 mots et une RAM de 41 mots. Les CE518 et CE519 sont des versions à EEPROM de données (16 mots de 8 bits). Les composants de cette famille exécutent toutes les instructions qu'ils connaissent en un seul cycle d'horloge (soit 1 µs à une horloge de 4 MHz).

Notons qu'il existe une version plus récente de ce composant, le PIC12C508A, de même qu'il existe d'ailleurs une version 12C509A.

Pour de plus amples informations à leur sujet, on pourra faire un tour sur le site Web de Microchip à l'adresse : www.microchip.com/1000/pline/picmicro/category/digictrl/8kbytes/devices/12c508/

(020403)

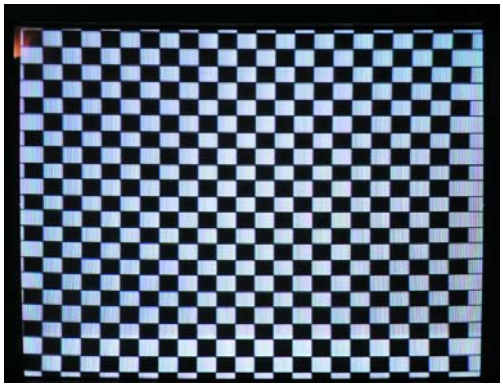


Figure 5. Voici ce que devrait donner l'écran de votre téléviseur après connexion de Picomire.

Interface CompactFlash

Pour système MCS-BASIC52

Gilbert Meers

Le prix des cartes CompactFlash baisse très rapidement. Ce phénomène n'a pas échappé à l'attention de l'auteur de cet article qui a, pour cette raison, conçu une interface PC parallèle permettant, sous MCS-BASIC, de transférer des données ou des programmes vers ce type de mémoire à semi-conducteur dont l'utilité n'est plus à démontrer.

Une part importante du marché des appareils photo numérique utilise une carte CompactFlash comme médium de stockage et que ce type d'appareil connaît une expansion quasi-exponentielle, ces cartes de mémoire extrêmement compactes, d'où leur nom, deviennent de plus en plus populaires. La taille de la mémoire stockée dans une CompactFlash ne cesse de croître, leur prix au mégaoctet ne cessant de devenir plus attrayant et ce à un point tel que ce type de carte intéresse de plus en plus les possesseurs de PC, raison pour laquelle nous nous y sommes intéressés dans le numéro d'avril 2002 d'Elektor (n°286, page 48 et suivantes) et que nous vous avons proposé une carte d'adaptation à leur intention. La première carte de 16 Moctets fournie avec votre appareil photo numérique a depuis lors fait place à sa grande soeur de 256 voire 512 Moctets, de sorte qu'elle ne sert plus à grand chose, alors que sa capacité est celle de plus de 10 disquettes.

Cette publication fut l'une des raisons qui poussa l'auteur de cet article à développer une interface reliée aux bus d'adresses de données et de commande du PC et qui associée à un programme simple, permet de transférer et stocker sur la carte CompactFlash des programmes en BASCI52.

Le programme nécessaire est disponible au téléchargement sur le site d'Elektor (www.elektor.fr); il est doté des fonctions requises pour la signalisation de saisie de paramètres invalides et l'interdiction d'écrasement involontaire de blocs par réécriture. La consommation de courant de l'interface est de quelque 42 mA, la vitesse de transfert est elle de 50 Koctets/s. L'auteur utilise cette interface sur un système tournant à 20 MHz et certifie que le transfert de données se fait de façon très fiable.

Le schéma

La **figure 1** vous propose l'électronique de l'interface. Comme le montre un coup d'oeil au schéma, l'électronique nécessaire se limite à un 82C55, un 74LS04, une paire de LED d'indication et 2 connecteurs.

Le connecteur à 50 contacts K2 sert de « point d'ancrage » de la carte CompactFlash. La broche 9 de ce connecteur est reliée à la masse, ce qui force la carte à se comporter comme un disque dur IDE. En cas d'utilisation de la carte dans ce mode il est impératif de mettre le cavalier JP1 (pris entre la broche 39 et la masse) en place.

Le pilotage de l'interface IDE est confiée à IC1, un 82C55. Par le biais de ses ports bidirectionnels PB0 à PB7 il commande les lignes de donnée D0 à D7, ses ports bidirectionnels PA0 à PA7 prenant à leur compte le pilotage des lignes de donnée D8 à D15. Les ports PC0 à PC7 assurent le pilotage des lignes de commande de l'interface IDE. Les ports bidirectionnels PA0 à PA7 et PB0 à PB7 sont activés en entrée ou en sortie au moment précis où cela est nécessaire. Les lignes du port PC, PC0 à PC7 sont en permanence des sorties.

Les inverseurs intégrés dans IC2 sont requis en raison du fait que lors de la commutation de sens de transfert des données le 82C55 met les dites lignes de port au niveau bas (« 0 »). Ces inverseurs évitent tout risque de situation d'états incontrôlée au niveau des lignes de commande.

Il faudra amener sur le connecteur K1 les signaux système suivants : 5 V et GND (masse); D0 à D8; $\overline{RD}$, $\overline{WR}$, A0, A1 et RESET; $\overline{CS}$.

À noter que le montage ne comporte pas le décodeur d'adresse fournissant le signal CS (*Chip Select*). Le signal $\overline{CS}$ devra se trouver au niveau bas en cas d'adressage des adresses

se trouvant dans le domaine commençant à $0C000_{HEX}$. Le 82C55 requiert 4 adresses :

```
Registre de données PA, adresse de base +0
Registre de données PB, adresse de base +1
Registre de données PC, adresse de base +2
Registre de commande, adresse de base +3
```

Terminons par un passage en revue de quelques détails intéressants du schéma : la LED D1 s'allume en cas d'activité au niveau de la carte CompactFlash. La LED D2 signale la présence de la tension d'alimentation de 5 V. Les condensateurs C1 et C2 servent enfin au découplage de la tension d'alimentation.

Le logiciel

Le logiciel requis par cette interface est écrit en ASM51, son code source étant proposé sous 2 versions :

– PARCFFU.ASM. (pour des instructions PCFF depuis MCS BASIC52);
– PARCFF.ASM. (pour des instructions CALL depuis MCS BASIC52).

Le logiciel intègre un pilote IDE de base qui subdivise la carte CompactFlash en blocs de 8 192 octets. Une carte classique de 64 Moctets possède de ce fait une capacité de 7 824 blocs, ce qui nous donne un total de 64 086 016 octets. Cette valeur est légèrement plus faible que la capacité nominale, ceci étant du au fait que le logiciel n'utilise pas la totalité du dernier cylindre.

Le programme source comporte une procédure d'attribution des adresses du 82C55 :

```
P8255A    EQU    0C008H    ;ADRESSE PORT A
P8255B    EQU    0C009H    ;ADRESSE PORT B
P8255C    EQU    0C00AH    ;ADRESSE PORT C
P8255CO   EQU    0C00BH    ;ADRESSE COMMANDE
```

Il faudra, si le signal $\overline{CS}$ du connecteur K1 sélecte des adresses différentes de celles indiquées plus haut, modifier en conséquence les attributions d'adresse.

Comme le logiciel est destiné à servir d'extension au MCS-BASIC52, le système concerné devra être doté de suffisamment de mémoire de programme additionnelle à partir de l'adresse 2000_{HEX}. La taille du programme est de 1,6 Koctet.

Il faudra, si l'on a déjà doté son système d'instructions MCS-BASIC52 additionnelles, modifier en conséquence le programme source et en effectuer un nouvel assemblage. Il ne faudra pas oublier que le programme source utilise des instructions ACALL, instructions connaissant une limite de 2 Koctets !

Le logiciel est compatible avec les version V1.1 et V1.3 de MCS-BASIC52.

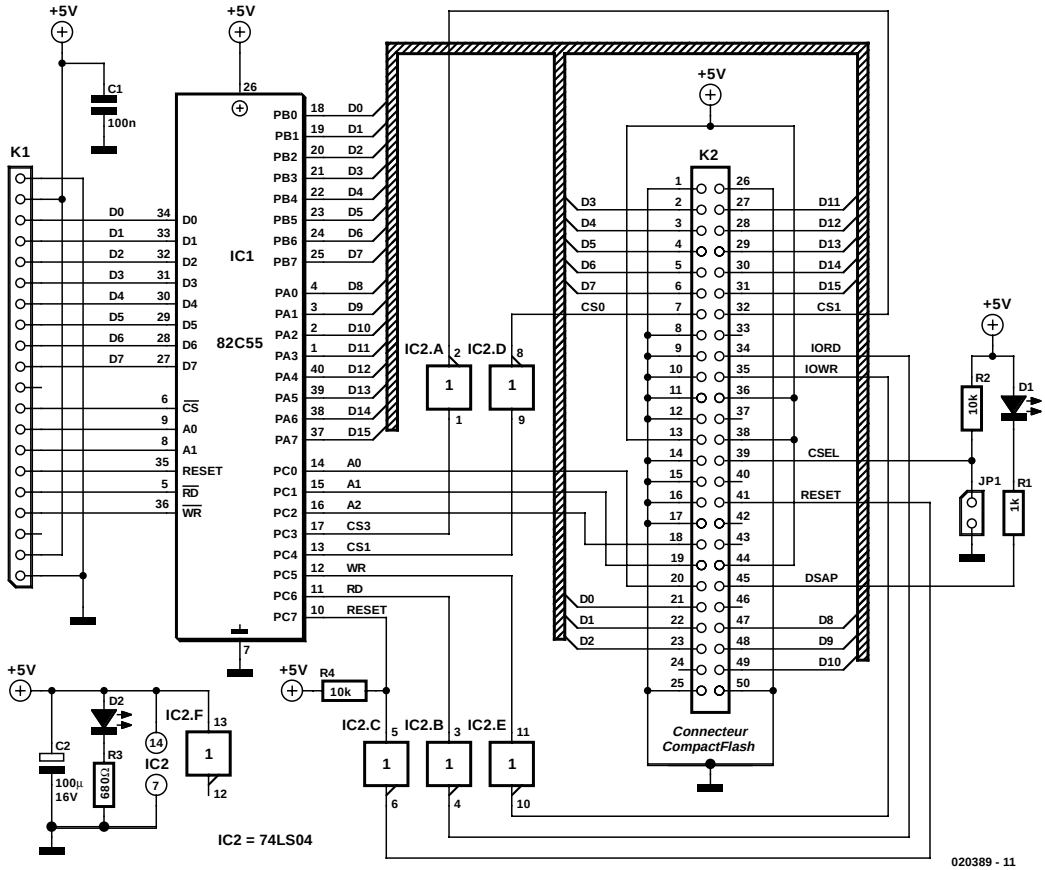


Figure 1. Le matériel requis par l'interface se limite à bien peu de choses.

PARCFFU.ASM

Il s'agit donc là du code-source pour doter le MCS-BASIC de nouvelles instructions (*statement*). Le code résultant de l'assemblage, PARCFFU.HEX, devra être placé dans la mémoire de programme du système, et ceci à compter de l'adresse 2000_{HEX}.
Voici les nouvelles instructions disponibles :

– **PCFF V** – Donne la version de la carte CompactFlash connectée au système.
– **PCFF R** – Remise à zéro (reset) de la carte CompactFlash.
– **PCFF L**, **numéro de bloc**, **adresse** – Écriture de données de la carte CompactFlash vers le système. Le numéro de bloc est celui défini sur la carte (0 à 7 823); il possède toujours une taille de 8 Koctets. « Adresse » correspond à l'adresse de début de la mémoire de données du système (00000_{HEX} à $0FFFF_{HEX}$). Après traitement de chaque secteur, on a émission d'un L à destination du terminal.
– **PCFF S**, **numéro de bloc**, **adresse** – Écriture de données de la mémoire de données du système vers la carte CompactFlash. Mêmes conditions et remarques que pour l'opération précédente, à ceci près que chaque transmission de secteur est suivie par l'émission d'un S.
– **PCFF F**, **numéro de bloc** – Cette instruction se traduit par la mise, dans la totalité du bloc concerné, de la valeur $0FF_{HEX}$. Chaque écriture de secteur est suivie par l'émission d'un caractère F.
– **PCFF D**, **numéro de bloc** – Affiche l'état (*status*) du numéro de bloc concerné : B = donnée, V = vide.

Si la carte n'est pas prête, que le numéro de bloc est trop élevé ou qu'il existe un risque d'écrasement de contenu de bloc, ces situations sont indiquées par l'affichage de messages d'erreur très « parlants ».

Mode d'emploi des instructions PCFF

Les programmes développé sous MCS-BASIC se trouvent en mémoire RAM à partir de l'adresse 0200HEX. L'instruction PCFF S, 10, 0200H écrit le programme dans le bloc 10 de la carte. Par PCFF L, 10, 0200H il est possible de remettre le programme dans la mémoire en RAM.
Lorsque le programme a une taille supérieure à 8 Koctets (à vérifier à l'aide de P. LEN) il faudra le placer dans plusieurs blocs. Il faudra, lors de ce processus, tenir compte du point suivant : l'instruction PCFF S, 10, 0200H écrit les données présentes des adresses 0200 à $21FD_{HEX}$ vers le bloc 10. Cela nous fait un total de 8+190 octets. En effet, 2 octets d'un bloc de 8 192 octets sont utilisés pour indiquer que le bloc comporte déjà des données. Il faudra partant, pour transférer un programme de 10 Koctets par exemple, utiliser l'instruction suivante :

```
PCFF S, 10, 0200H : PCFF S, 11, 21FEH
```

ou :

```
PCFF S, 10, 0200H : PCFF S, 11, 0200H+8190
```

Un rechargement du programme se fait par le biais de l'instruction :

```
PCFF L, 10, 0200H : PCFF L, 11, 0200H+8190
```

On pourra également utiliser des variables en tant que numéro de bloc ou d'adresse. Supposons que nous voulions effacer les blocs 100 à 120 : pour pourra à cet effet utiliser le petit programme ci-après :

```
100    FOR T=100 to 120
101    PCFF F, T
102    NEXT T
103    END
```

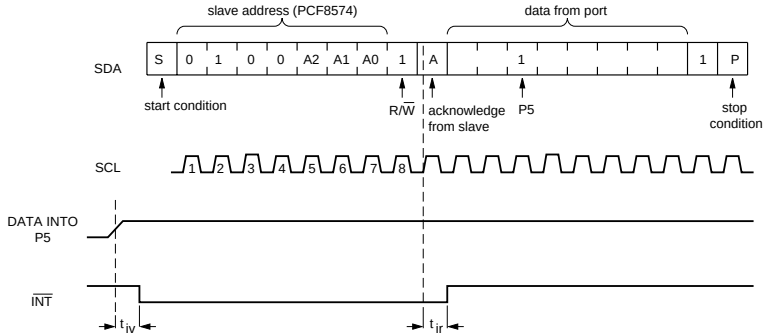
PARCFF.ASM

Il peut arriver qu'il soit impossible, sur un système donné, d'ajouter des instructions pour MCS-BASIC-52 à compter de l'adresse de programme 2000_{HEX}. Le code-source PARCFF.ASM a pour fonction de permettre de placer le programme à n'importe quel emplacement disponible dans la mémoire de programme du système. L'adresse de base utilisée dans le code-source en question est 0B000H, position que l'on pourra bien évidemment modifier en fonction des besoins. L'adresse de début est donnée à l'aide de l'instruction assembleur suivante :

```
START  EQU    0B000H
ORG    START
```

Il est possible de modifier cette attribution et de définir une nouvelle adresse de début de programme. Il va sans dire qu'il faudra, après une telle modification, procéder à un nouvel assemblage du programme-source.
Le fonctionnement et le mode d'emploi sont exactement les mêmes que dans le cas de PARCFFU.ASM. La seule différence est qu'il faudra dans le cas présent remplacer, dans les instructions PCFF évoquées plus haut, les lettres PCFF par un « CALL 0B000H » (ou l'adresse modifiée que vous aurez choisie). On trouvera dans le code-source (disponible, comme le reste des programmes, en anglais eux aussi, sur notre site Internet www.elektor.fr sous la dénomination **EPS020389-11**) les informations sur la manière d'ajouter les variables requises en assembleur.

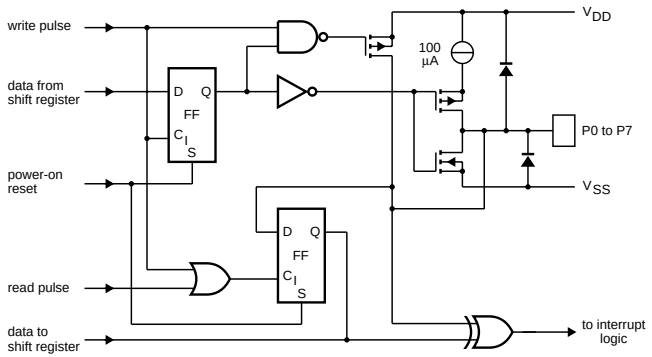
(020389)



vée lorsque les données sur le port retourne dans leur position de sortie ou que les données vers/en provenance du port ayant provoqué l'interruption ont été lues ou écrites. On a remise à zéro au cours du bit d'acquiescement (*acknowledge*) en mode lecture après le flanc montant du signal SCL, ce processus se faisant, en mode écriture, après le flanc descendant de ce même signal (t_{ir} étant écoulé). De nouvelles impulsions d'interruption nées au cours de tels RAZ peuvent être très brèves voire perdues. Chaque changement au niveau des E/S prenant place après une RAZ est détecté et traduit, après le premier flanc montant du signal d'horloge, comme étant une interruption (signal INT).

E/S quasi-bidirectionnelles :

Les E/S quasi-bidirectionnelles peuvent être utilisées soit en entrée soit en sortie, sans qu'il ne soit nécessaire, par le biais d'un signal de commande, d'indiquer quelque direction de données que ce soit. Après mise sous tension les E/S se trouvent au niveau haut. Dans ce mode, seule une source de courant vers VDD est active. Une résistance de forçage au niveau haut (pull up) additionnelle permet des flancs à pente très raide même si les sorties sont fortement chargées. La charge est mise en circuit lors du basculement d'une sortie vers le niveau haut, et déconnectée lors d'un flanc descendant du signal SCL. Les E/S doivent se trouver au niveau haut, avant d'être utilisées en entrées.



PCF8574
Expanseur 8 bits pour le bus I²C
Fonctions spéciales Numérique
ELEKTOR INFOCARTE 5/2003

PCF8574
Expanseur 8 bits pour le bus I²C

Fabricant :

Philips Semiconductors
www.semiconductors.philips.com
www.semiconductors.philips.com/acrobat/datasheets/PCF8574 4.pdf

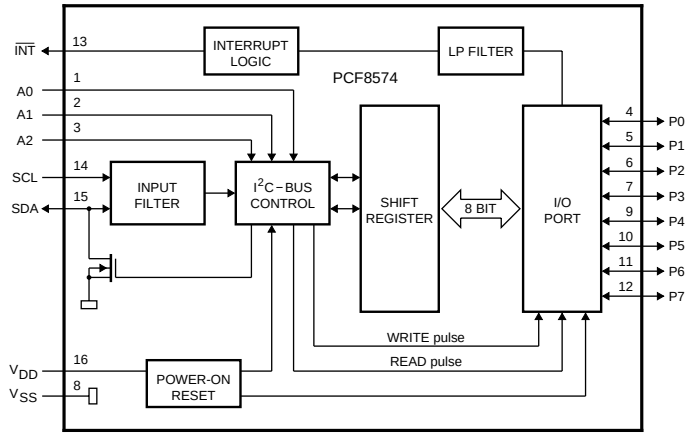
Caractéristiques :

- Tension d'alimentation : 2,5 à 6 V
- Consommation de courant extrêmement faible en veille : 10 µA maximum
- Expanseur de bus I²C vers port parallèle
- Sortie d'interruption à drain ouvert
- Compatible avec la majorité des microcontrôleurs
- Les verrous de sortie peuvent fournir des courants importants
- 3 bits d'adresse sélectables pour les PCF8574 et PCF8574A
- Boîtiers DIP16 ou SO16 ou SSOP20 pour gain d'espace

Description :

Le PCF8574 sert à doter différents microcontrôleurs de ports additionnels. Il offre une possibilité de 8 lignes d'E/S tout en ne nécessitant que 2 lignes de port du microcontrôleur pour le bus I²C. Le circuit intégré comporte un port quasi-bidirectionnel et une interface pour bus I²C. En dépit d'une consommation de courant faible, les verrous de sortie sont en mesure de fournir un courant d'une intensité suffisamment élevée pour permettre une commande directe de LED. Ce circuit une ligne d'interruption (INT), que l'on pourra interconnecter à la logique d'interruption du microcontrôleur. Cette option permet à l'expanseur d'informer le microcontrôleur de l'arrivée de données sans avoir à faire intervenir le bus I²C de sorte qu'il reste esclave (slave) à 100%. Il existe 2 versions de ce composants qui se différencient par leurs domaines d'adressage en mode esclave. 3 des bits d'adresse sont à la discrétion de l'utilisateur, ce qui permet d'adresser individuellement un maximum de 128 lignes d'E/S (Entrée/Sortie) par le biais d'un bus I²C.

Structure interne, boîtiers et brochage :



Symbole	Broche (DIP16/SO16)	Broche (SSOP20)*	Description
A0	1	6	Entrée d'adresse 0 esclave
A1	2	7	Entrée d'adresse 1 esclave
A2	3	9	Entrée d'adresse 2 esclave
P0	4	10	E/S quasi-bidirectionnelle 0
P1	5	11	E/S quasi-bidirectionnelle 1
P2	6	12	E/S quasi-bidirectionnelle 2
P3	7	14	E/S quasi-bidirectionnelle 3
VSS	8	15	Masse de l'alimentation
P4	9	16	E/S quasi-bidirectionnelle 4
P5	10	17	E/S quasi-bidirectionnelle 5
P6	11	19	E/S quasi-bidirectionnelle 6
P7	12	20	E/S quasi-bidirectionnelle 7
$\overline{INT}$	13	1	Sortie d'interruption (active au niveau bas)
SCL	14	2	bus I²C, horloge sérielle
SDA	15	4	bus I²C, données sérielles
VDD	16	5	Plus de l'alimentation

* : les broches non mentionnées ne sont pas utilisées

PCF8574P/AP: DIP16
PCF8574T/AT: SO16
PCF8574S/AS: SSOP20

Adressage :

Start	Adresse esclave								Ack	Type
S	0	1	0	0	A2	A1	A0	0	A	PCF8574
S	0	1	1	1	A2	A1	A0	0	A	PCF8574A

Interruption :

Le PCF8574 possède une sortie d'interruption INT (drain ouvert) que l'on pourra interconnecter à l'entrée correspondante d'un microcontrôleur. On obtient ainsi une sorte de fonction de maître (*master*) indépendante du bus I²C qui pourra, au niveau du système, déclencher une action donnée. On a génération d'une interruption sur chaque flanc descendant d'une ligne de port lorsque le circuit intégré se trouve en mode Input. Après écoulement du temps tiv le signal $\overline{INT}$ est valide.
La logique d'interruption est réinitialisée et réacti-

PCF8574
Expanseur 8 bits pour le bus I²C
Fonctions spéciales Numérique
ELEKTOR INFOCARTE 5/2003

Caractéristiques (V_{DD} = 2,5 à 6 V, V_{SS} = 0 V, T_{amb} = –40 à +85 °C, sauf mention contraire)

Paramètre	Conditions	Symbole	Min.	Typ.	Max.	Unité
ALIMENTATION						
Tension d'alimentation			2,5		6,0	V
Consommation de courant	En service, V_{DD} = 6 V, hors-charge, V_I = ou V_{SS}, f_{SCL} = 100 kHz	I_{DD}		40	100	µA
Courant de veille	Standby, V_{DD} = 6 V, hors-charge, V_I = V_{DD} ou V_{SS}	I_{stb}		2,5	10	µW
Tension de Power On Reset	V_{DD} = 6 V, hors-charge, VI = V_{DD} ou V_{SS}	V_{POR}		1,3	2,4	V
SDA, SCL						
Tension d'entrée niveau bas		V_{IL}	-0,5		+0,3V_{DD}	V
Tension d'entrée niveau haut		V_{IH}	0,7·V_{DD}		V_{DD}+0,5	V
Courant d'entrée niveau bas	V_{OL} = 0,4 V	I_{IL}	3		10	mA
Courant de fuite	V_I = V_{DD} ou V_{SS}	I_L	–1		+1	µA
Capacité d'entrée	V_I = V_{SS}	C_I			7	pF
ENTRÉES D'ADRESSE A0 à A2						
Tension d'entrée niveau bas		V_{IL}	–0,5		+0,3V_{DD}	V
Tension d'entrée niveau haut		V_{IH}	0,7·V_{DD}		V_{DD}+0,5	V
Courant de fuite en entrée	Broche à V_{DD} ou V_{SS}	I_{LI}	–250		+250	nA
E/S						
Tension d'entrée niveau bas		V_{IL}	–0,5		+0,3V_{DD}	V
Tension d'entrée niveau haut		V_{IH}	0,7·V_{DD}		V_{DD}+0,5	V
Courant de sortie niv. haut	V_{OH} = V_{SS}	I_{OH}	30		300	µA
Courant de sortie niv. bas	V_{OL} = 1 V, V_{DD} = 5 V	I_{OL}	10	25		mA
Courant d'entrée maximal adm. par diode de prot.	$V_I \geq V_{DD}$ ou $V_I \leq$ VSS	$I_{IHL(max)}$			±400	µA
Courant d'à coup (flanc)	Haut pour Ackn, V_{OH} = V_{SS}, V_{DD} = 2,5 V	I_{OHt}		–1		mA
Capacité d'entrée/de sortie		C_i,C_o			10	pF
CHRONOLOGIE DES PORT (CL ≤ 100 pF)						
Données de sortie valides		t_{pv}			4	µs
Temps d'établissement des données d'entrée		t_{su}	0			µs
Temps de maintien des données d'entrée		t_h	4			µs
INTERRUPTION (CL ≤ 100 pF)						
Courant de sortie niv. bas	V_{OL} m= 0,4 V	I_{OL}	1,6			mA
Courant de fuite	V_I = V_{DD} ou V_{SS}	I_L	-1		+1	µA
Données de sortie valides		t_{iv}			4	µs
Temporisation de RAZ		t_{ir}			4	µs

Enceintes de PC comme Intercom

A. C. J. Bauer

La plupart des systèmes d'enceintes fournies avec les PC multimédia bas de gamme sont de qualité douteuse et ont vite fait d'être remplacés par « quelque chose de mieux ». Une fois que l'on a installé ses nouvelles acquisitions, le matériel « Low-Fi » termine ses jours dans la chambre des enfants s'il n'est pas abandonné dans un grenier.

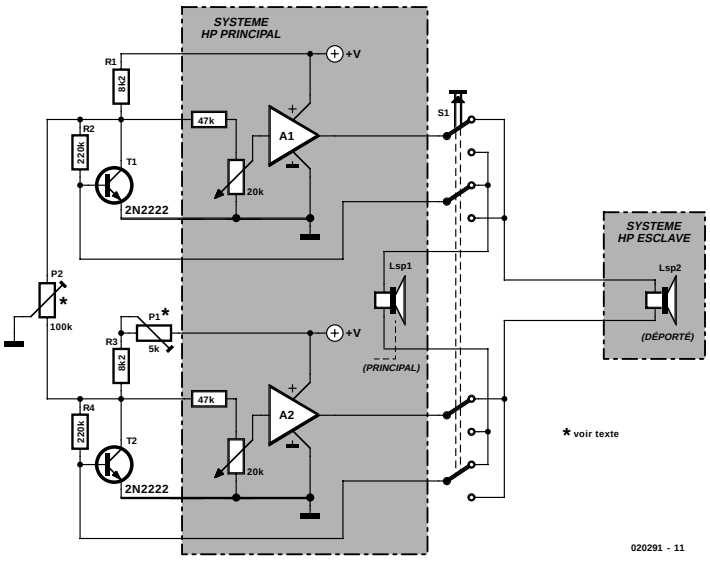


Figure 1.

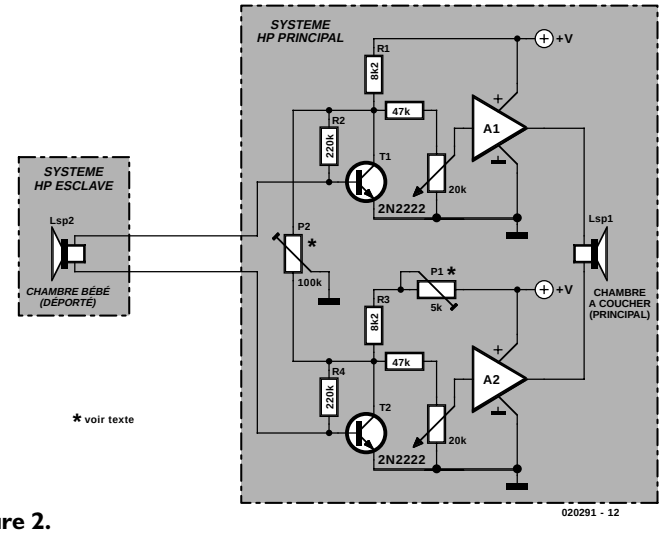


Figure 2.

Le présent petit article a pour but de montrer comment transformer un système de haut-parleur bon marché en un système d'intercom simple mais efficace, surtout qu'il n'aura pas coûté grand chose.

Dans la majorité des cas, l'un des coffrets des enceintes contient une alimentation, un amplificateur stéréophonique et un haut-parleur. L'autre enceinte est un « esclave » qui ne contient rien de plus qu'un haut-parleur.

Système d'intercom

La recette pour convertir un set de haut-parleurs pour PC en un intercom est présentée en **figure 1**. Seules sont à ajouter les parties se trouvant en dehors des zones ombrées. Le haut-parleur esclave fait office de microphone lorsque l'on actionne l'inverseur multi-pôles monté sur ou présent à côté de l'unité centrale. Ce fonctionnement requiert un minimum d'amplification ce qui explique la présence des transistors T1 et T2. Le montage des 2 amplificateurs de l'unité centrale en une configuration symétrique réduit sensiblement le bruit induit par le câble (qui aura inévitablement une certaine longueur) entre les stations de l'intercom. L'ajustable P1 permet de régler au mieux l'équilibre entre les amplificateurs lorsque l'on a affaire à une liaison d'une certaine longueur. L'ajustable P2 est optionnel et ne sera nécessaire que dans le cas où l'on se trouverait en présence de niveaux de bruit extrêmement importants.

Dans la majorité des enceintes, mais pas dans toutes, l'interrupteur marche/arrêt commute la ligne de tension d'alimentation positive plutôt que la tension du secteur. Cette situation permet de dériver la ligne d'alimentation +V nécessaire aux étages à transistor de l'une des bornes de l'interrupteur marche/arrêt.

Bébéphone

Si l'on veut réaliser un bébéphone (*babyphone*) il n'est normalement pas nécessaire de « parler » à station située à l'autre bout du fil, de sorte que l'on pourra modifier le circuit en prenant comme référence le schéma de la **figure 2**. On pourra, si l'on a besoin d'une sensibilité plus élevée, court-circuiter les résistances de 47 kΩ.

(020291-1)

APPLIKATOR est une rubrique servant à la description de composants intéressants récents et de leurs applications; par conséquent, leur disponibilité n'est pas garantie. Le contenu de cette rubrique est basé sur les informations fournies par les fabricants et les importateurs, ne reposant pas nécessairement sur les expériences pratiques de la Rédaction.

Filtre-bouchon à facteur Q élevé

Sans composants à sélecter

Klaus-Jürgen Thiesler

Un filtre-bouchon servant à bloquer une bande de fréquences étroite requiert, normalement, des composants à la tolérance très stricte. Pour peu que l'on utilise un type d'amplificateur opérationnel de Maxim spécial, cet impératif n'est plus de mise.

Dans les circuits de filtre à flanc raide, les tolérances des différents composants mis en oeuvre s'influencent l'une l'autre lorsqu'il s'agit de comportements en fréquence complexes, de sorte qu'il est quasiment impossible d'obtenir des résultats satisfaisants si l'on utilise des composants standards. Le circuit proposé ici transpose à l'intérieur de circuits intégrés faciles à trouver les résistances qui jouent un rôle important sur l'amplitude du filtre et simplifie partant énormément le dessin des pistes du circuit imprimé. Les dits amplificateurs opérationnels sont dotés de résistances internes calibrées au laser et respectent leur valeur nominale à moins de 1‰. S'il nous fallait apparier entre elles des résistances de précision individuelles présentant des valeurs de tolérance absolues en vue d'obtenir une tolérance relative, un tel processus d'appariement non seulement coûterait sensiblement plus cher mais présenterait également une complexité redoutable.

Il est facile de calculer, pour les 2 réseaux RC de la **figure 1**, la fréquence de coupure désirée.

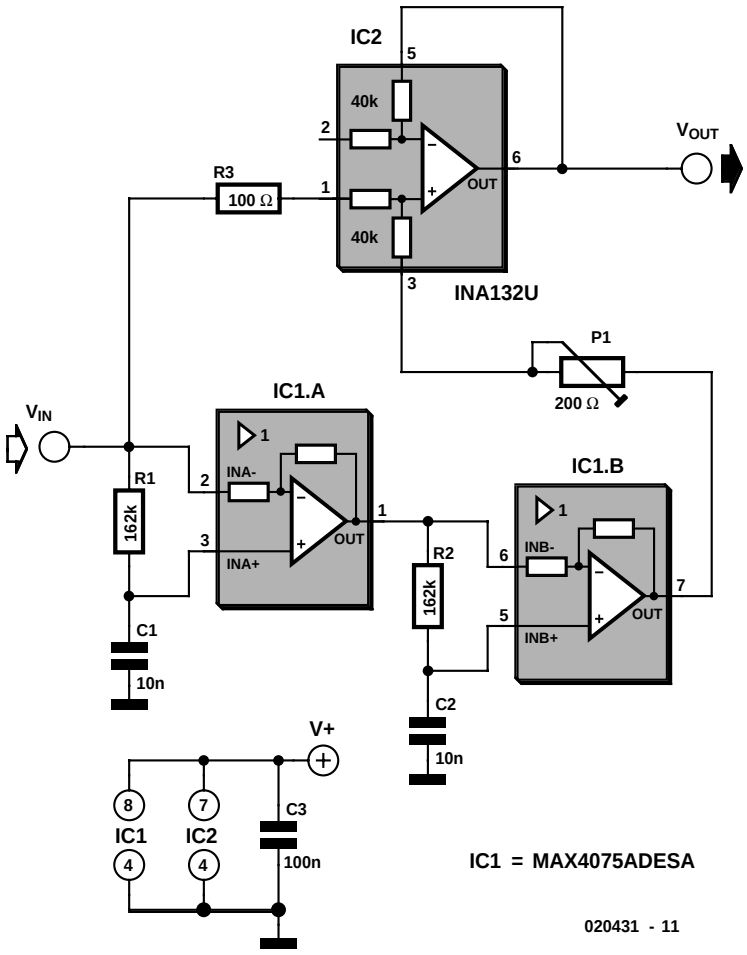


Figure 1. Des amplificateurs opérationnels spéciaux à résistances ajustées au laser.

Répartition de tâches

L'électronique subdivise au niveau de l'amplificateur sommateur IC2, l'amplitude et la fréquence en 2 réseaux RC qui déterminent la fréquence et en 2 réseaux de rétroaction qui agissent sur le niveau du signal; cet amplificateur débarrasse le signal d'entrée de la fréquence à supprimer par un simple déphasage. IC1 intègre une paire d'amplificateurs opérationnels y compris leur réseau de rétroaction. Le MAX4075 existe en 54 versions avec autant de rapports d'amplification (gain) qui vont ainsi de –0,25 à –100 V/V (inverseur) et, en version non inverseuse, de+1,25 à 101 V/V. Le suffixe AD signifie que nous avons utilisé la version inverseuse (G = –1). Ils fonctionnent en filtre passe-bande distinct dont le déphase total est de 180 ° exactement à la fréquence centrale f_0. Les résistances définissant le gain intégrées introduisent une dérive de gain inférieure à 0,1%. Elles sont responsables de la taille de l'amplitude du signal (à la fréquence de coupure) que IC2 à additionné au signal d'entrée. Elles n'ont cependant pas d'influence sur la fréquence de coupure proprement dite. C'est là la fonction des réseaux RC externes qui eux n'ont pas d'effet sur l'atténuation du signal.

Les composants CMS disponibles ont, en règle générale, une plage de tolérance moindre que leurs homologues à pattes. Comme les 2 circuits intégrés concernés n'existent qu'en boîtier SOIC-8 (CMS), nous avons tout intérêt à utiliser, pour les 6 composants passifs, des versions CMS.

L'ajustable P1 permet d'ajuster le filtre de manière à avoir la meilleure élimination de la plage de fréquence désirée.

Circuit RC de résonance série

Il est possible, avec des valeurs de tolérance standard de 1% (0806) pour R1 et R2 et de 10% pour C1 et C2 (céramique X7R), d'obtenir une caractéristique de suppression de fréquence meilleure que celle représentée par la courbe de la **figure 2**. Il est même possible, si tant est que l'on choisisse avec soin les réseaux RC, d'obtenir une fréquence de réjection plus précise.
Sur la broche 3 de IC2 arrive le signal qui a subi, au niveau de la fréquence du filtre coupe-bande, un double déphasage de 90 °. Sa broche 1 est attaquée par le signal d'entrée. Les 2 résistances intégrées assurent le mélange de ces 2 signaux.
IC2 est amplificateur opérationnel différentiel de précision doté lui aussi d'un réseau de résistances de précision élevée étalonné au laser de manière à induire une erreur de ±2‰ au maximum. Il est monté en amplificateur sommateur équilibré dont l'entrée inverseuse en broche 2 reste en l'air.
Pour les fréquences sensiblement inférieures à la fréquence de résonance

$$f_0 = 1 / (2 \cdot \pi \cdot R \cdot C)$$

les condensateurs présentent une impédance élevée, de sorte que les suiveurs de tension inverseurs ne déphasent pas le signal. Pour des

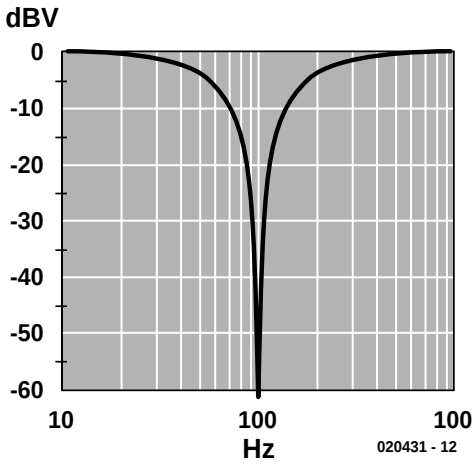


Figure 2. Il est possible d'obtenir ce superbe comportement de blocage à l'aide de résistances à tolérance de 5% et de condensateurs à ±20% non calibrés.

fréquences sensiblement supérieures à f_0 chaque suiveur de tension inverseur déphase son signal d'entrée de 180 °, ce qui se traduit par un déphasage total de 360 ° = 0 °. Les phases de chaque filtre passe-bande se comportent comme un pôle RC unique, c'est-à-dire qu'elles déphasent le signal à chaque fois de 90 ° à la fréquence de résonance.
Les 3 circuits intégrés d'amplification de précision peuvent traiter des signaux dont la fréquence ne dépasse pas 100 kHz, les distorsions qu'ils introduisent sont très faibles. La plage de tension d'alimentation va de 2,7 à 5,5 V, la consommation de courant atteignant 250 µA.

(020431)

Choix de composants CMS standards

Fréquence de coupure désirée f_O [en Hz]	R1 / R2 Valeur la plus proche (E12, ±5%) [en kΩ]	C1 / C2 Valeur choisie (±20 %) [en nF]	Fréquence de coupure obtenue f_0 (à ±0,1 % R + C)	Plage de tolérance théorique maximale de f_0 [en Hz] (avec ±5 % R et ±20 %)
50	300	10	53	42 à 70
60	270	10	59	47 à 77
100	150	10	106	85 à 140
120	56	22	128	100 à 170
400	18	22	401	320 à 530

Impression de dessins de circuits

Travailler adroitement avec le presse-papiers

Harry Baggen

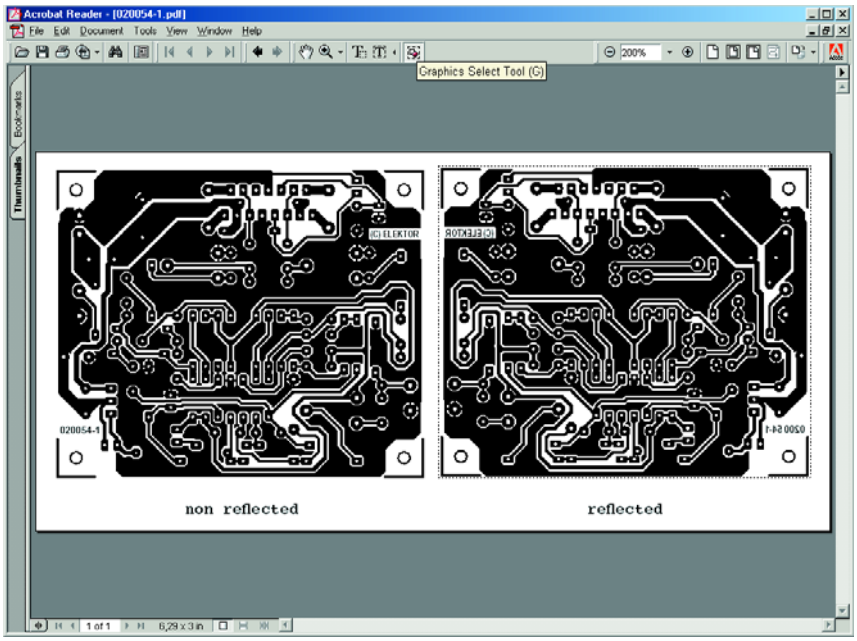
Sur le site d'Elektor tous les dessins de circuits imprimés actuellement disponibles le sont sous la forme de fichiers .pdf. À partir de ceux-ci, on peut facilement en faire soi-même l'impression sur transparent en vue de l'insolation et de la gravure d'un circuit imprimé. Cependant, l'impression à l'échelle correcte donne du souci à pas mal de lecteurs. C'est pourquoi nous donnons à travers cet article quelques trucs permettant d'imprimer ces tracés de circuits imprimés à la taille et à l'emplacement correct, sur papier ou sur film transparent.

Pratiquement tous les tracés de circuits qui sont reproduits dans Elektor sont également disponibles sur notre site (www.elektor.fr) sous forme de fichiers .pdf. De la sorte, tout le monde peut aisément en imprimer le tracé sur un film transparent, pour ensuite procéder soi-même à l'insolation et à la gravure.
Depuis quelques temps, nous faisons toujours figurer dans ces fichiers .pdf le circuit de manière normale et en miroir, de sorte que les hobbyistes peuvent choisir la version qui convient le mieux à leur application (normalement, c'est en plaçant la face qui a reçu l'impression en contact avec le côté « pistes » du circuit que l'on obtient les meilleurs résultats lors de l'insolation et c'est pour cela que l'on doit utiliser une impression en miroir du tracé des pistes).
En outre, la taille du papier dans le fichier .pdf est adaptée à la taille de l'imprimante, de sorte que les impressions dépassant le format A4 ne seront pas réparties sur plusieurs pages.
Le grand avantage de l'utilisation des fichiers .pdf est le fait que le dessin du circuit imprimé est mémorisé sous forme vectorielle (spécifique à Acrobat), grâce à quoi la taille du fichier est minimale par rapport à une netteté optimale des tracés. Malheureusement, peu de programmes reconnaissent ce format, de sorte que l'exportation d'un dessin de circuit pour le traiter dans un autre programme n'est pas si simple. Beaucoup de lecteurs semblent avoir des problèmes avec ces fichiers : le circuit n'est pas imprimé à la bonne échelle ou bien apparaît à un mauvais endroit. C'est surtout avec des films transparents relativement chers qu'il est intelligent de les utiliser de manière optimale et donc de ne pas imprimer les deux dessins de circuit (normal et en miroir), ni non en plein milieu de la feuille (réglage par défaut d'Acrobat Reader). C'est pourquoi nous avons rassemblé quelques astuces grâce auxquelles, et avec aussi peu de ressources que possible (lire : sans programmes supplémentaires), vous pourrez malgré tout obtenir une bonne qualité d'impression.

Sélectionner

Tout commence habituellement avec une version d'Acrobat Reader car c'est grâce à elle que vous ouvrirez les fichiers .pdf téléchargés qui contiennent le dessin du circuit. Nous partirons du principe que la plupart des lecteurs utilisent la version 5 de ce programme (disponible sur le CD annuel d'Elektor et sur le site Web d'Adobe, www.adobe.com). Après l'ouverture d'un fichier .pdf, apparaissent à l'écran les deux dessins du circuit. Normalement, vous n'aurez besoin que d'un seul de ces dessins. Réglez d'abord la taille d'affichage de Reader de façon à ce que le dessin choisi apparaisse dans son entier sur l'écran. Activez le bouton *Sélection d'Affichage (Graphics Select Tool)* au milieu de la barre de boutons en cliquant dessus avec la souris. Le curseur se change alors en une croix et nous pouvons maintenant sélectionner un des deux circuits en plaçant la souris juste en dehors d'un des coins du dessin choisi, en poussant sur le bouton gauche de la souris et, tout en maintenant ce bouton enfoncé, en sélectionnant tout le circuit. Quand vous relâcherez le bouton de la souris, vous verrez apparaître un cadre en pointillés tout autour de ce dessin (**figure 1**). Vous pouvez alors imprimer séparément

1

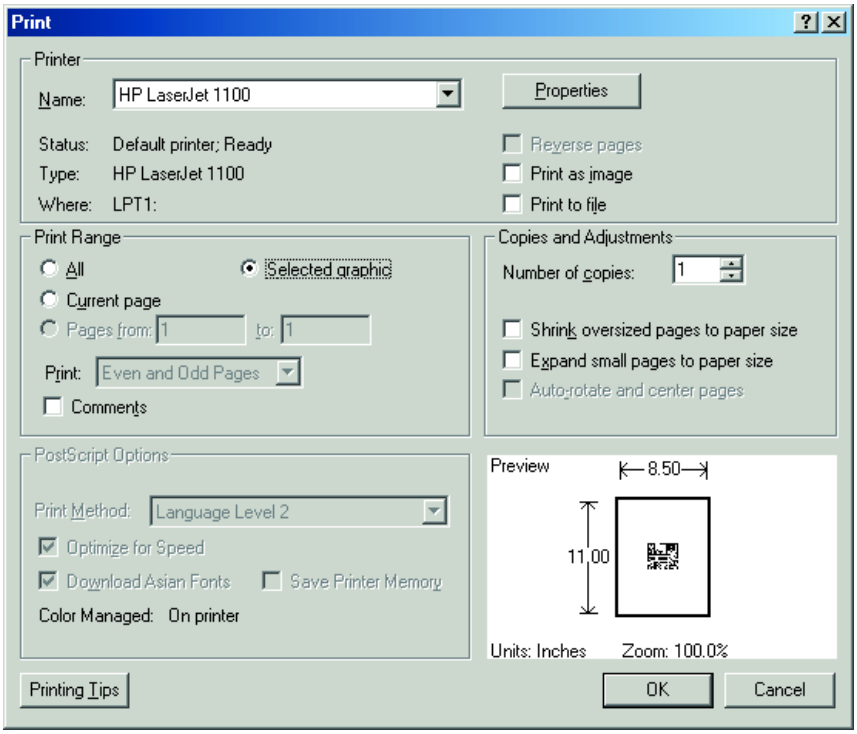


depuis Acrobat Reader le dessin sélectionné ou bien le recopier via le presse-papiers (*clipboard*) vers un autre programme pour traitement supplémentaire ou pour impression.

Imprimer

Nous traiterons d'abord de l'impression depuis Acrobat Reader. Les possibilités d'impression sont fortement dépendantes des réglages que votre pilote d'imprimante offre, c'est pourquoi nous ne décrivons ici que les réglages propres à Acrobat Reader. Cliquez sur *Fichier/Imprimer* (*File/Print*). Il apparaît alors un menu d'impression de Reader indépendant de l'imprimante (**figure 2**). Vous voyez alors dans la fenêtre *Exemple* (*Preview*) comment le dessin sélectionné serait imprimé sur l'imprimante. Prenez soin de vérifier que les options *Réduire les pages trop grandes au format du papier* et *Agrandir les pages trop petites au format du papier* sont désactivées (dans la version anglaise *Shrink oversized pages to paper size* et *Expand small pages to paper size*), sans quoi le dessin ne sera pas représenté à la taille correcte. Il vaut mieux également désactiver *Rotation et centrage automatique des pages* (*Auto-rotate and center pages*).

2



Lorsqu'il imprime le résultat d'une sélection (comme nous venons de le faire), Reader imprime toujours cette sélection en plein centre de la feuille/du film. Si vous souhaitez une impression plus sur le côté de la feuille/du film, vous devrez alors expérimenter quelques-uns des formats de papier figurant dans le pilote de l'imprimante. Choisissez par exemple A6 tout en plaçant un film A4 dans l'imprimante (ce truc ne fonctionnera pas avec toutes les imprimantes).

Il est à conseiller d'effectuer un certain nombre d'essais d'impression sur du papier bon marché avant de placer un film coûteux dans l'imprimante. Contrôlez par la même occasion que le circuit est effectivement imprimé à la bonne taille (comparez avec l'impression dans Elektor).

Via le presse-papiers

La deuxième méthode nous offre nettement plus de possibilités Nous utilisons pour cela un des programmes de traitement d'images disponibles sur l'ordinateur afin de choisir manuellement sur la page l'endroit où une illustration sera imprimée (p.ex. Paintshop Pro ou Picture Publisher), ou bien même le classique traitement de texte Word. Nous copions la sélection que nous avons faite via le presse-papiers (*clipboard*) vers Word ou le programme de traitement d'images.

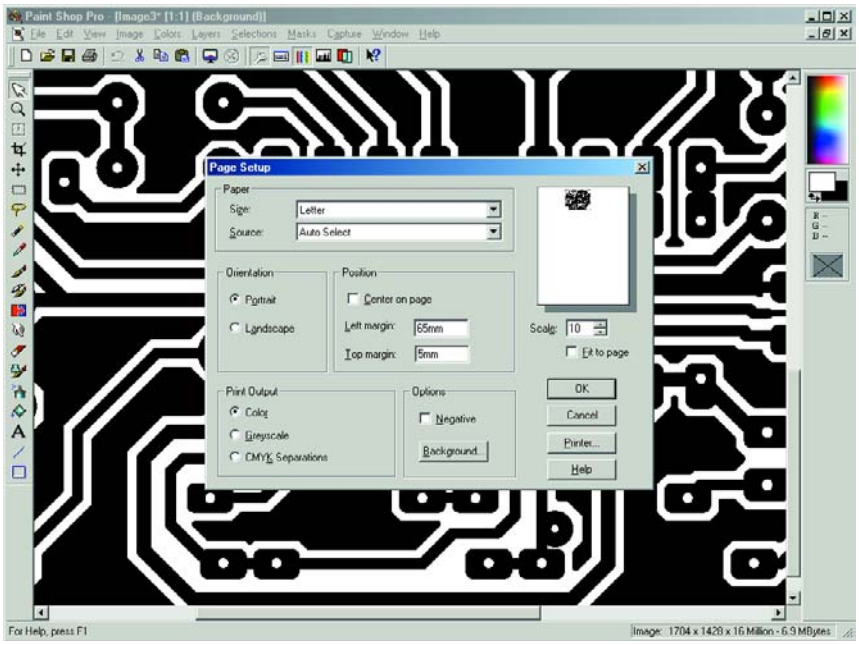
Lors de la copie via le presse-papiers, le dessin (vectoriel) sélectionné est automatiquement converti au format pixel et cela a pour conséquence que la résolution devient trop basse pour obtenir une bonne impression (sans bavures) via un autre programme. Habituellement, on ne peut copier vers le presse-papiers que les sélections visibles à l'écran, ce qui nous limite à la résolution de l'écran. Cela ne vaut toutefois pas pour Acrobat Reader ! Il est ici possible de copier des sélections qui ne sont que partiellement visibles à l'écran. Après avoir sélectionné le dessin comme décrit précédemment, nous réglons l'affichage à l'écran de Reader sur 400 ou 500% (ou plus encore). Il ne reste donc plus qu'une partie du dessin visible à l'écran, mais cela ne fait rien. Copiez maintenant vers le presse-papier via *Editer/Copier* (*Edit/Copy*) (ou Ctrl C sur le clavier). Vous pouvez alors rappeler le dessin dans, par exemple, Paintshop Pro (*Edit/Paste as new image*) ou Word (*Editer/Copier* ou *Edit/Paste*). L'image peut

ensuite être imprimée à partir de ce programme.
Dans Paintshop Pro, il vous faut encore définir précisément à travers *Fichier/Paramètres de page* (*File/Page Setup*) (**figure 3**) la position de l'impression sur la page. Ensuite, vous devez vous occuper de ce que le circuit soit représenté à la taille correcte. Pour un agrandissement initial de 400%, vous devrez imprimer le circuit au quart (1/4) de sa taille affichée, pour 500% à 20%. Une fois que tout cela est au bien au point, on peut imprimer. Le détour par un programme de traitement d'image présente encore l'avantage que vous pouvez modifier quelques petites choses au circuit, comme enlever une connexion ou déplacer une piste. Il est vrai que cela se passe au niveau du pixel mais, grâce à l'agrandissement important, cela donne un résultat parfait à l'impression.
Dans Word, nous devons travailler différemment. Une fois que le dessin du circuit est placé sur une page vide depuis le presse-papiers, on clique une fois sur le dessin afin de sélectionner l'image. On clique ensuite dessus avec le bouton droit de la souris et on choisit *Formatage de l'image (Format Picture*). Sous *Taille* (*Size*, voir **figure 4**) on peut choisir la taille désirée (en fonction de l'agrandissement choisi dans Reader, respectivement 25 et 20%, veillez à ce que *Verrouillage du rapport hauteur/largeur* (ou *Lock aspect ratio*) soit coché et sous *Agencement* vous choisirez *Devant le texte* (*In front of text*).
Ce dernier choix a pour conséquence que vous serez en mesure de placer l'illustration à un endroit quelconque de la page à l'aide de la souris. Après cela, on peut imprimer.
Word n'acceptant pas les demi-pourcent, on peut toutefois préciser les mesures exactes.
De l'une des manières décrites ci-avant, il devrait être possible d'imprimer un dessin de circuit à l'endroit voulu d'une feuille de papier ou de film.

Pour terminer, encore deux astuces.

1) Pour les grands dessins de circuit qui ne tiennent pas sur une page A4, on peut souvent se tirer d'affaire en choisissant la taille de papier *Letter* (un peu plus longue que l'A4) ou en définissant soi-même la taille de papier (*Custom size*). Coupez pour cela manuellement une feuille A3 à la taille voulue et placez là dans le bac d'alimentation de l'imprimante.
2) Les dessins de circuits des anciens articles ne sont représentés qu'en « normal » dans les fichiers .pdf fournis. Au cas ou le pilote d'imprimante n'offre pas la possibilité d'imprimer l'illustration en miroir, vous pouvez utiliser la fonction de miroir présente dans quasi tous les programmes de traitement d'images (Paintshop Pro: *Image/Mirror*). (030065)

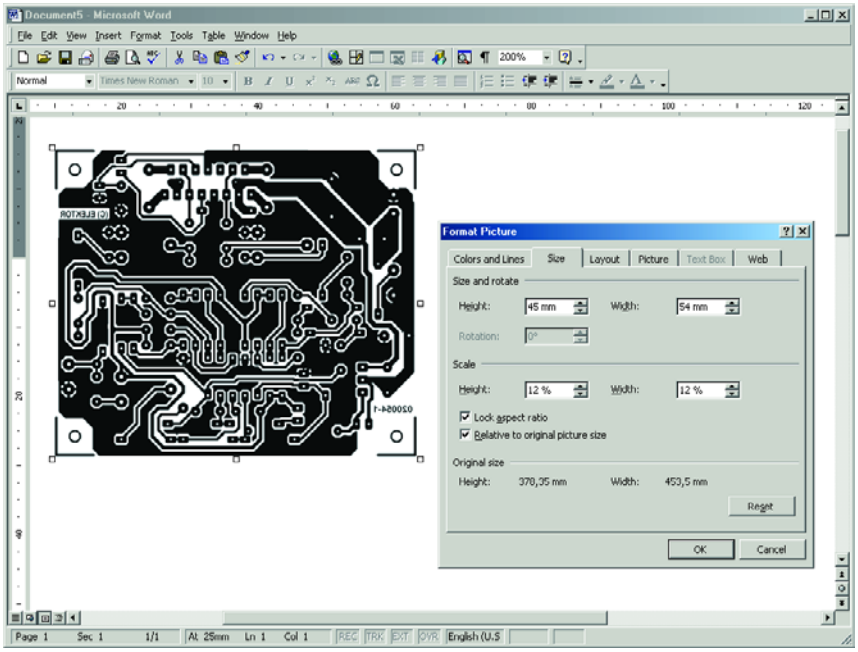
3



Résolution de l'écran

D'après nos expériences, il apparaît que certains programmes tiennent compte de la résolution de l'écran et d'autres pas. Cela peut avoir pour conséquence qu'un dessin qui est agrandi à 400% dans Reader et ensuite réduit à 25% dans Word n'ait pas la taille correcte. Il faut alors expérimenter un petit peu pour trouver la bonne échelle de réduction. Une fois celle-ci trouvée, elle pourra être utilisée les fois suivantes. Souvent, la déviation est la différence entre une résolution d'écran de 96 et 72 dpi, soit un facteur 1,33. Dans Word, on peut heureusement également introduire les dimensions exactes de l'illustration, mais il faudra alors également tenir compte de l'éventuel espace de blanc que vous aurez introduit autour lors de la sélection dans Reader.

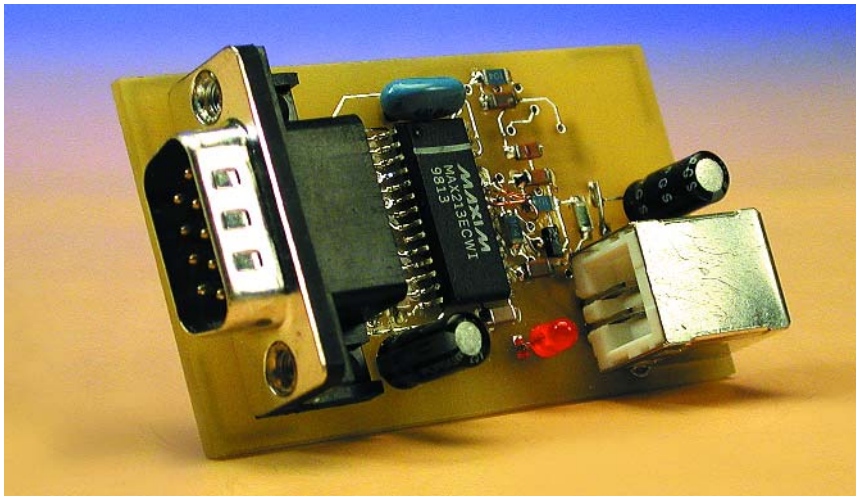
4



Interface USB → RS-232

Solution compacte pour ports manquants

Il devient possible, grâce à un circuit intégré spécialisé de FTDI, de connecter, sans le moindre problème, des périphériques dotés d'une interface RS-232 (sérielle) à un port USB. Une solution optimale lorsqu'un appareil ne possède pas de connexion USB ou que le PC de bureau ou portable n'a plus de port sériel de disponible voire n'en possède tout simplement pas.



S'il a fallu un certain temps au port USB pour se faire connaître, il n'est pas moins devenu, ces toutes dernières années, le port universel que suggère sa dénomination (*Universal Serial Bus*). Son succès est d'ailleurs tel que l'on se trouve confronté actuellement à un problème vu que dans le meilleur des cas les ordinateurs actuels ne comportent plus qu'un unique port sériel si tant est qu'ils en aient encore un. Grâce à au convertisseur USB –> RS-232 décrit dans le présent article il reste possible de brancher un périphérique à interface RS-232 à un port USB. Les pilotes gratuits (pour Win98/ME/2000/XP, Linux et Macintosh) rendent cette interface pratiquement transparente, de sorte que le port USB concerné auquel est connecté le convertisseur se comporte comme le ferait un port COM standard. Le pilote et le circuit intégré sont d'origine FTDI et permettent l'établissement d'une liaison sérielle complète y compris tous les signaux d'acquittement (*handshake*) présents sur une embase RS-232 9 points. FTDI est une société écossaise. Pour plus d'informations à son sujet on pourra faire un tour sur leur site à l'adresse www.ftdichip.com. Notons que c'est OPTIMINFO qui distribue FTDI dans l'Hexagone.

Fonction et structure interne

Il n'est pas nécessaire, en principe, pour la présente application, de s'inquiéter de ce qui se passe à l'intérieur du convertisseur, mais il nous semble qu'il n'est pas mauvais d'avoir une petite idée des fonctions remplies par le circuit intégré et une vue d'ensemble sur le fonctionnement du tout.

Le synoptique simplifié de la **figure 1** permet de jeter un coup d'oeil aux entrailles du FT232AM, le composant utilisé ici. En gros, le FT232AM, de même d'ailleurs que son petit frère le FT245AM, tous 2 de FTDI, est un FIFO USB sériel (FIFO = *First In First Out*) piloté par le PC par le biais d'un port COM virtuel. La différence entre les 2 composants est que le FT232AM intègre, contrairement au FT245AM, un UART qui met à disposition une interface RS-232 (niveaux TTL). Le FT245AM possède quant à lui une interface 8 bits avec lignes d'acquittement qui permet un accès direct au FIFO. Ce second composant tombe à pic lorsque l'on se trouve en présence d'un système à microcontrôleur sans interface sérielle d'origine et que l'on veut partant doter d'une telle interface.

Il faudra se débrouiller soi-même pour l'intégration au système. Au niveau du port USB, ces 2 composants sont identiques, ce qui explique qu'ils utilisent le même pilote. Un émetteur/récepteur (*trans-*

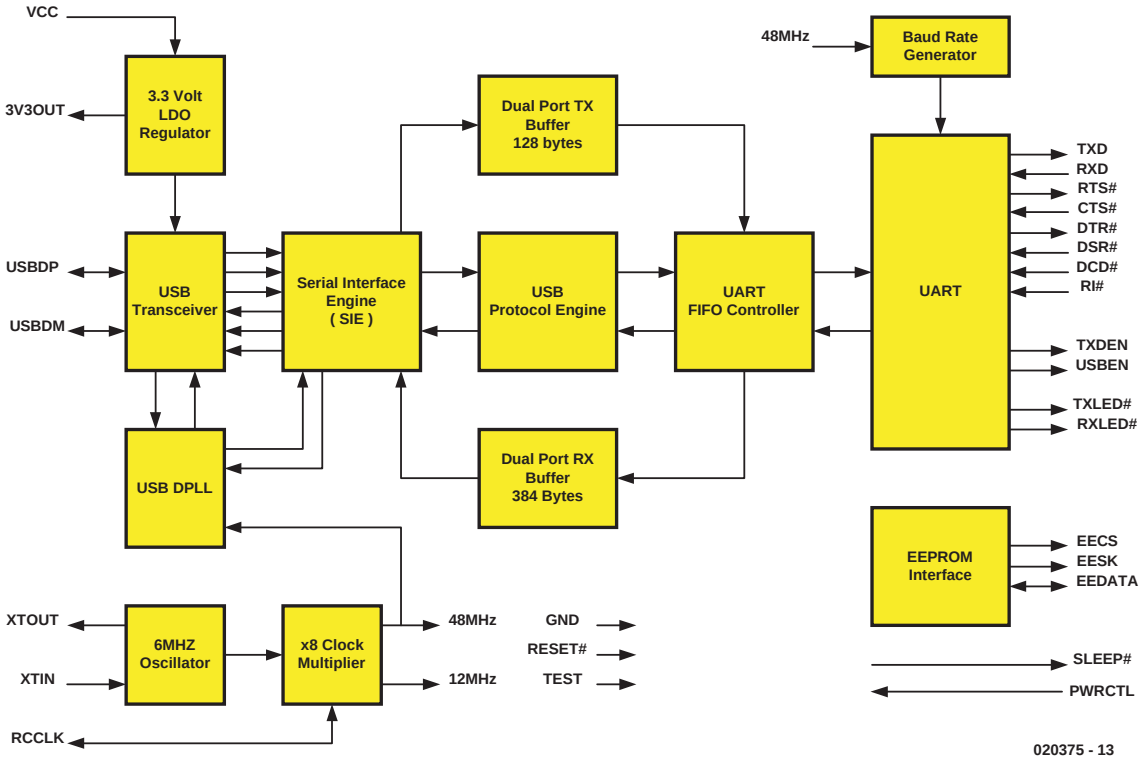


Figure 1. Synoptique simplifié du convertisseur USB ➔ RS-232 du type FT232AM de FTDI.

ceiver) USB à l'entrée constitue l'interface dans laquelle s'emboîte le câble USB avec ses 2 lignes de signal D+ et D– en mode USB 1.1 pleine vitesse (*full speed*). Un régulateur de tension à faibles pertes interne fournit la tension de référence de 3,3 V nécessaire. Cette tension de référence de 3,3 V est disponible en broche 6 vu qu'il faut d'une part la découpler par le biais d'un condensateur et que l'on a besoin, d'autre part, pour le paramétrage du mode pleine vitesse. Sur le schéma de la **figure 3**, c'est là la fonction de la résistance R6 par l'intermédiaire de laquelle la ligne D+ est forcée à +3,3 V. Cet état de choses apprend à l'hôte USB (le contrôleur USB intégré dans le PC) que notre interface désire être adressée comme étant un périphérique pleine vitesse. Dans le cas d'un périphérique faible vitesse (*Low speed*) c'est la ligne D– qui est forcée à +3,3 V au travers d'une résistance.

En aval du *transceiver* USB, le SIE (*Serial Interface Engine*) prend les données à son compte et les convertit du format sériel au format parallèle et inversement.

Le bloc *USB Protocol Engine* traite les informations de commande USB et assure la communication avec le contrôleur USB de l'hôte (le PC) en respect du protocole USB de base (*low level*) et le traitement des instructions pour la définition du paramétrage de fonctionnement de l'UART.

La transmission des données dans les 2 directions (entre le SIE et les registres de l'UART) se fait à chaque fois par le biais de tampons pour l'émission et la réception, tampon TX (transmission) double port à 128 octets dans le premier cas et RX (réception) de 384 octets dans le second. C'est le contrôleur FIFO de l'UART qui pilote le transfert entre les 2 tampons et le registre d'émission/réception de l'UART.

L'UART proprement dit ne diffère guère de ses homologues logés dans les PC; il fournit l'ensemble des signaux requis par l'interface RS-232 ainsi que ceux nécessaires aux normes RS-422 et RS-485. Le générateur de taux de transmission (*baudrate generator*) permet de paramétrer des taux de transmission allant de 300 bauds à 2 Mbauds (jusqu'à 920 Kbauds en RS-232 et 2 Mbauds pour RS-422/RS-485). Il ne faut pas oublier de mentionner l'interface pour EEPROM sérielle prévue pour une 93C46. Le FT232AM fonctionne également en l'absence de cette

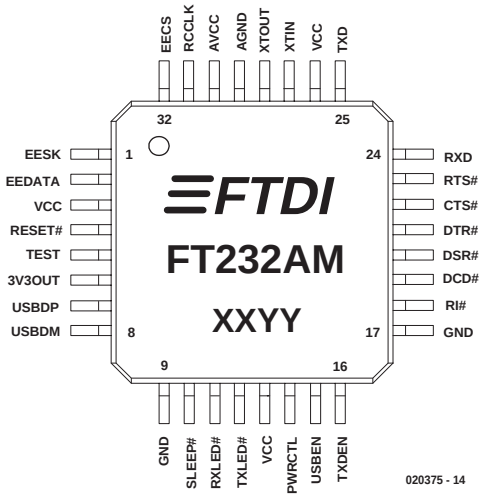


Figure 2. Brochage du FT232AM en boîtier QFP (7 x 7 mm).

mémoire de données non volatile, mais l'interface s'annonce alors comme un périphérique sériel standard. L'EEPROM a cependant l'intérêt de permettre le stockage et la fourniture d'informations spécifiques telles qu'identification de fabricant et de produit (VID et PID pour *Vendor* et *Product IDentification*), numéro de série et autres données similaires. Cette EEPROM devient indispensable lorsque l'on envisage de connecter plusieurs convertisseurs USB –> RS-232 à puce FTDI à un même PC, vu que les pilotes n'installent de port

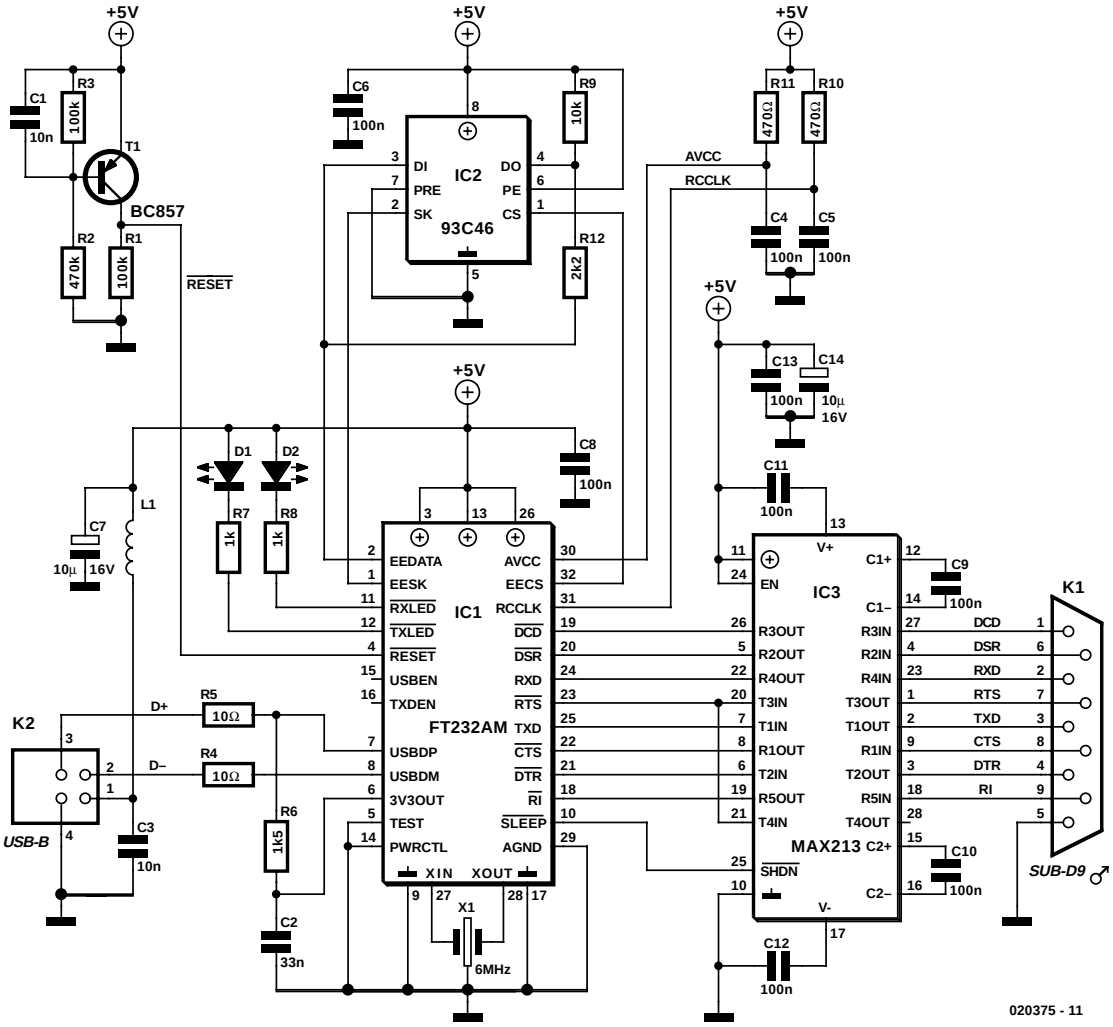


Figure 3. L'électronique de l'interface USB ➔ RS-232.

COM virtuels distincts que dans le cas de numéros de série différents. En l'absence de numéro de série, il n'est possible d'installer qu'un seul et unique port COM Virtuel.

L'électronique et la platine

Le schéma de la figure 3 est on ne peut plus lisible : le transistor T1 et le réseau RC de remise à zéro à la mise sous tension (POR = *Power On Reset*) en haut à gauche, l'EEPROM juste à côté et, en seconde ligne, en bas, de la gauche vers la droite, l'embase USB, le FT232AM, un MAX213 et en bout de schéma, l'embase RS-232, un connecteur sub-D 9 points. L'alimentation du montage se fait par le biais de la broche 1 de l'embase USB (K2) du port USB du PC sous +5 V. Le découplage est un peu plus étoffé que d'habitude, prenant la forme, outre le condensateur C3, d'une petite self, L1, combinée à un condensateur électrochimique, C7. De plus, les différents circuits intégrés sont dotés de leur propre découplage. Comme nous le disions plus haut, la résistance R6 fait office de résistance de forçage au niveau haut (*pull up*) et force la ligne USB D+ à 3,3 V, en vue de signaler à l'hôte USB qu'il a affaire à un périphérique « *full-speed* ». La présence de cette résistance a une seconde fonction, lors de la connexion au port USB du PC de faire reconnaître à ce dernier le branchement d'un périphérique USB.

Pour le plaisir des yeux et les vôtres uniquement, le FT232AM comporte une paire de sorties de commande de LED, qui visualisent l'une l'émission (D1) et l'autre la réception (D2) des données.

Bien que de dimensionnement identique, les réseaux RC R11/C4 et R10/C5 remplissent des fonctions différentes.

R10/C5 pris dans la ligne RCCLK est un réseau de temporisation chargé d'assurer la stabilité d'horloge du FT2322AM, lorsque ce circuit quitte son mode de veille (*sleep*), le réseau R11/C4 servant lui uniquement au découplage de la tension de la ligne AVCC (*Analog VCC*), qui est la tension d'alimentation du multiplicateur d'horloge par 8 interne.

La seule fonction du MAX213 est la tâche classique de convertir, à l'aide de pompes de charges à condensateurs externes, le niveau 5 V des signaux présents côté RS-232 du FT232AM en niveaux aux normes RS-232. Nominalement, cela devrait être du ±12 V, dans la pratique avec ce circuit on a de l'ordre de ±8 V (±10 V au maximum).

Si le schéma est très accessible, cela n'est pas nécessairement le cas de la platine représenté en **figure 4**, en raison de la présence de composants CMS. Il s'agit d'une platine double face à trous métallisés dotée de composants de part et d'autre. Si vous avez déjà une certaine expérience

Liste des composants

Toutes les résistances et condensateurs CMS au format 1206

Résistances :
R1,R3,R10 = 100 kΩ
R2 = 470 kΩ
R4,R5 = 10 Ω
R6 = 1kΩ5
R7,R8 = 1 kΩ
R9 = 10 kΩ
R11 = 470 Ω
R12 = 2kΩ2

Condensateurs :
C1,C3 = 10 nF
C2 = 33 nF
C4 à C6,C8 à C13 = 100 nF
C7,C14 = 10 µF/16 V radial

Inductances :
L1 = BLM31A601S Murata (par exemple chez Farnell 581-094)

Semi-conducteurs :
D1,D2 = LED 3 mm
T1 = BC857
IC1 = FT232AM (FTDI (Optiminfo)-dénommination FT8U232AM)
IC2 = 93C46 (optionnel)
IC3 = MAX213ECWI (Maxim Integrated)

Divers :
K1 = embase sub-D 9 points en équerre encartable
K2 = embase USB encartable (type B)
X1 = -résonateur céramique 6 MHz

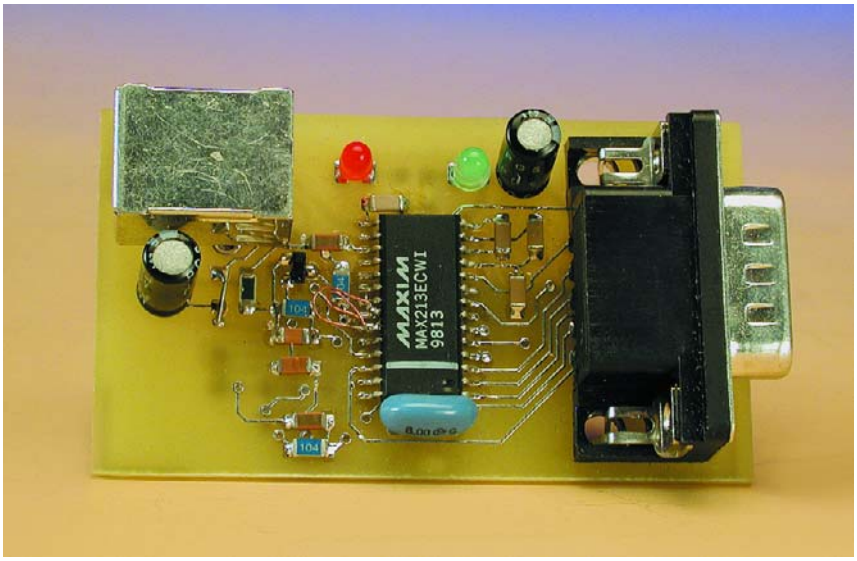


Figure 5. Vu du dessus d'un exemplaire terminé de l'interface...

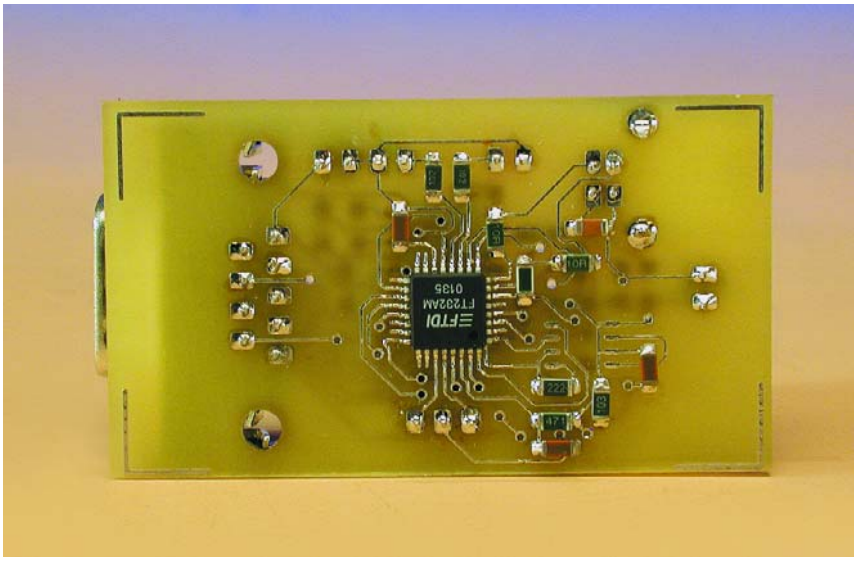


Figure 6. ...et du dessous.

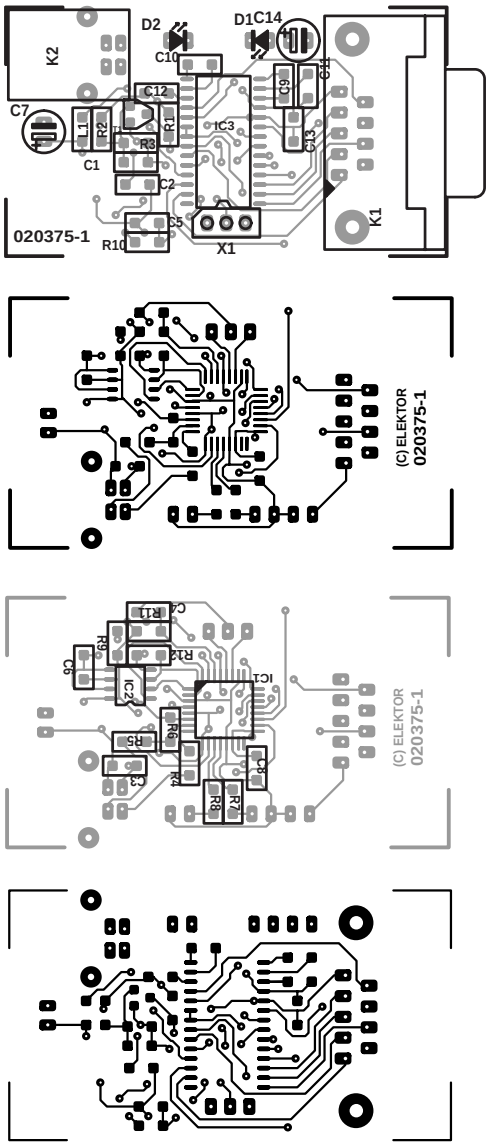


Figure 4. Dessin des pistes et sérigraphie de l'implantation des composants de la platine à « majorité CMS ».

du soudage de composants CMS, il n'y a rien ici pour vous effrayer. Sinon, il n'y a pas non plus de raison de paniquer, Elektor ayant eu la bonne idée de publier 2 articles consacrés au sujet (l'un dans le numéro de mars, l'autre dans le présent numéro, cf. la bibliographie en fin d'article). Si vous êtes débutant en la matière, nous vous recommandons de commencer par la lecture des dits articles et de vous faire la main à l'aide de quelques composants courants sur un morceau de platine d'expérimentation à pastilles pour éviter que votre premier contact avec un projet à CMS ne devienne une grande déception. On commence l'implantation des composants par la mise en place des composants CMS pour passer ensuite aux composants classiques et aux embases USB et RS-232.

La platine reçoit une embase USB de type B (dont la **figure 7** donne le brochage). L'autre variété d'embase (type A) se trouve toujours côté PC ou *Hub*, le type B se trouvant toujours dans l'appareil qui s'annonce au PC. L'embase de type A fournit, par le biais du câble USB, du courant à l'embase de type B présente sur le périphérique, ce courant servant, dans le cas présent, à l'alimentation de la platine de notre convertisseur USB –> RS-232. Les câbles USB sont toujours reliés sans croisement (1:1).

Mise en oeuvre et logiciel

Il faudra, avant de brancher la platine au port USB d'un PC pour la première fois, examiner avec soin l'implantation des composants et leur soudage une fois encore; il est fortement recommandé d'utiliser une loupe à fort grossissement...
Il faudra ensuite télédécharger les pilotes requis depuis le site Web indiqué plus loin. Il existe des pilotes gratuits pour toutes les versions de Windows actuelles, de même que pour le Mac et pour Linux. Il existe, en ce qui concerne les pilotes sous Windows des versions à support PNP (pour mémoire PNP = *Plug&Play*) et d'autres sans support PNP (*without PNP support*), appelés *Non-PNP-driver*. Ces petites différences ont leur importance. Il ne faudra utiliser le pilote avec support PNP que dans le cas où le périphérique sériel relié au PC par le biais du convertisseur USB –> RS-232 a également installé son (ou ses) pilote(s) en mode PNP. En d'autres termes, en cas de doute utiliser les pilotes non-PNP pour éviter tout problème. Les problèmes typiques que l'on rencontre avec les pilotes supportant PNP peuvent être : un *boot* plus long et l'identification erronée du convertisseur USB –> RS-232 comme étant un périphérique « pointeur » ce qui a pour conséquence un dysfonctionnement de la souris. Il existe, pour Windows XP, un outil XPNPNP qui permet de désactivé la fonction *Plug&Play* pour interfaces sérielles FTDI.
Une fois que nous avons trouvé le bon pilote et qu'il a été recopié dans un répertoire nous pouvons débuter l'installation par connexion du convertisseur USB –> RS-232 à l'un des ports USB du PC.
Très peu de temps après, Windows se manifeste signalant qu'il a détecté un périphérique USB -si tel n'était pas le cas il y a un problème au niveau de notre platine USB- (on peut essayer de débrancher le connecteur USB pour le rebrancher ensuite). Si tout se passe comme prévu il nous reste plus qu'à cliquer sur le fichier **FTDIBUS.INF** pour démarrer une installation classique. On découvre ensuite que le PC est doté d'un port

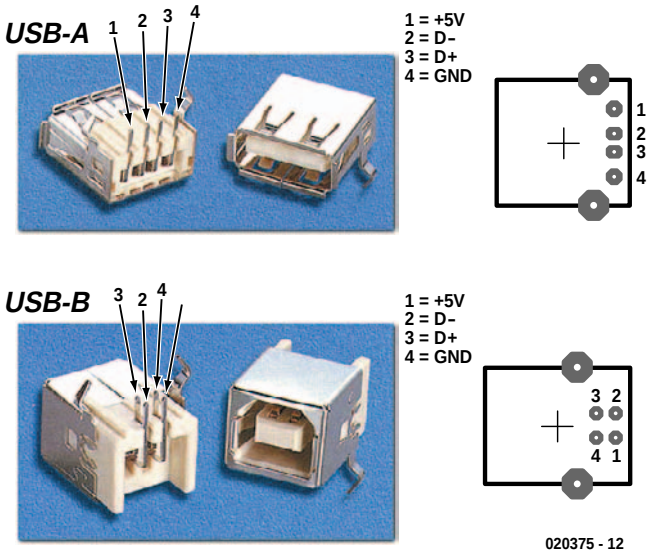


Figure 7. Brochage des 2 versions d'embase que l'on trouve sur les interfaces USB.

sériel additionnel, ce que l'on peut vérifier par **Démarrer** - **Paramètres** - **Panneau de configuration** - **Système** - **Gestionnaire de périphériques**. Si l'on jette à cet endroit un coup d'oeil au dossier **Ports (COM et LPT)**, on devrait y trouver, comme l'illustre la **figure 8a**, un nouveau « **Port USB Sériel (COMx)** ».
On a, lors de l'installation, mise en

Installation

Il est de notoriété publique que le branchement de périphériques USB au PC sont un jeu d'enfant. Le système d'expérimentation reconnaît l'interface et requiert le pilote adéquat. On trouvera ce pilote sur le site Web de FTDI sous le repère « Drivers and Utilities ». Les pilotes dits VCP (Virtual COM Port) veillent à ce que l'interface se comporte comme le ferait un port sériel classique. Il existe, nous le disions, des pilotes pour Windows, MacOS et Linux, sachant que nous voulons utiliser ici la version pour Windows..
Une fois l'installation du pilote effectuée, le port COM se laissera adresser par les applications comme le ferait n'importe quel autre port sériel du système. On pourra, avec les langages de programmation tels que Delphi et C++, utiliser des composantes de port COM tel que Tcomport pour permettre la communication avec l'interface sérielle. Si vous écrivez vos propres programmes il est préférable d'utiliser, à la place des pilotes VCP, les pilotes D2XX « Direct » pour Windows proposés par FTDI. La programmation de l'EEPROM optionnelle implique l'utilisation d'un Direct-Driver !
Les pilotes VCP pour Windows proposés sur le site Web de FTDI existent en 2 versions. La première supporte la reconnaissance automatique (PnP), l'autre version ne disposant pas de cette faculté. Cela n'a rien à faire avec la présente interface (Windows reconnaît cette dernière automatiquement) mais concerne uniquement le périphérique qui viendra se brancher sur cette interface (cf. le paragraphe « Mise en oeuvre » de l'article).
Le télédéchargement des pilotes VCP comporte un fichier .zip avec pilotes pour Windows98, ME, 2000 et XP. On décomprimera l'archive .zip sur son disque dur ou une disquette. On trouvera en outre sur le site Web en question une documentation exhaustive concernant les nouveautés au niveau du logiciel et du processus d'installation (le tout en anglais uniquement).
Windows démarre automatiquement le « New HArdware Wizard » dès que l'on connecte l'interface au PC par le biais d'un câble USB; le SE demande à l'utilisateur sur quel support et où se trouve le pilote. Entrez le répertoire où se trouve le pilote décompacté. Au bout de quelques secondes le système trouve le fichier FTDIBUS.INF et installe les pilotes et le logiciel pour l'interface correspondant.

place de 2 pilotes qui se trouvent en relation l'un avec l'autre. L'un des pilotes met à disposition le port COM virtuel que nous avons découvert dans le Gérant de périphériques sous la forme d'un nouveau port dans les connexions. L'autre pilote fait en sorte que le côté USB du FT232AM fasse son apparition dans le sous-répertoire « **Contrôleur de bus sériel universel** » en tant que périphérique USB (cf. **figure 8b**).

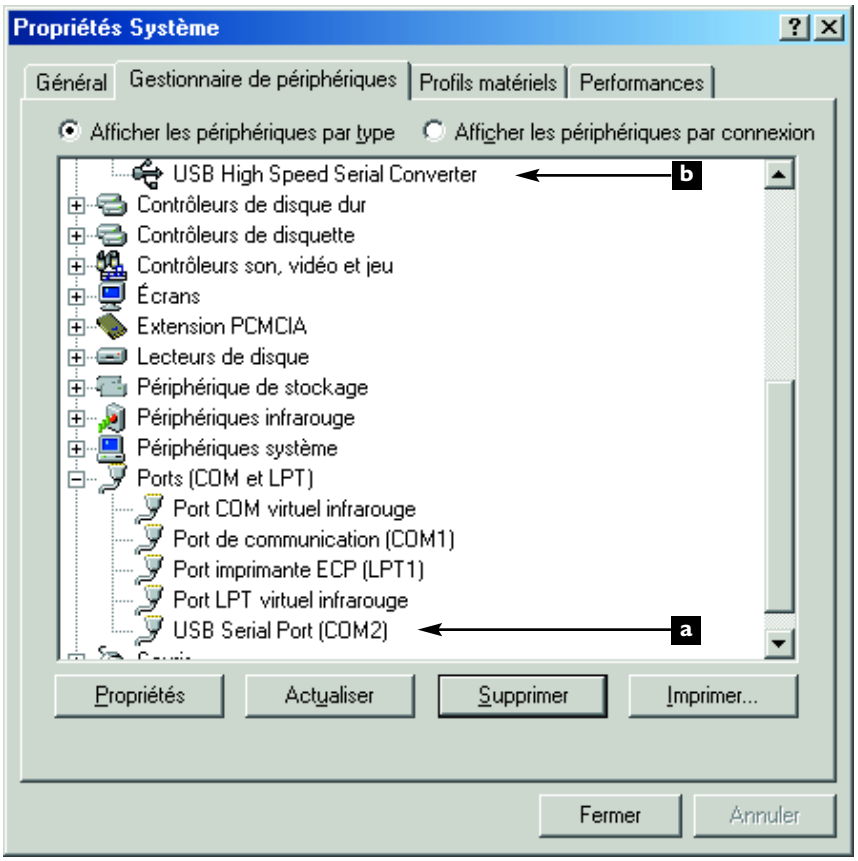


Figure 8. Après l'installation on découvre dans le Gérant de périphériques un nouveau « Port COM sériel USB » (a) et dans le dossier « Contrôleur de bus sériel universel » un nouveau « Convertisseur sériel haute vitesse » (b) nouveau lui aussi.

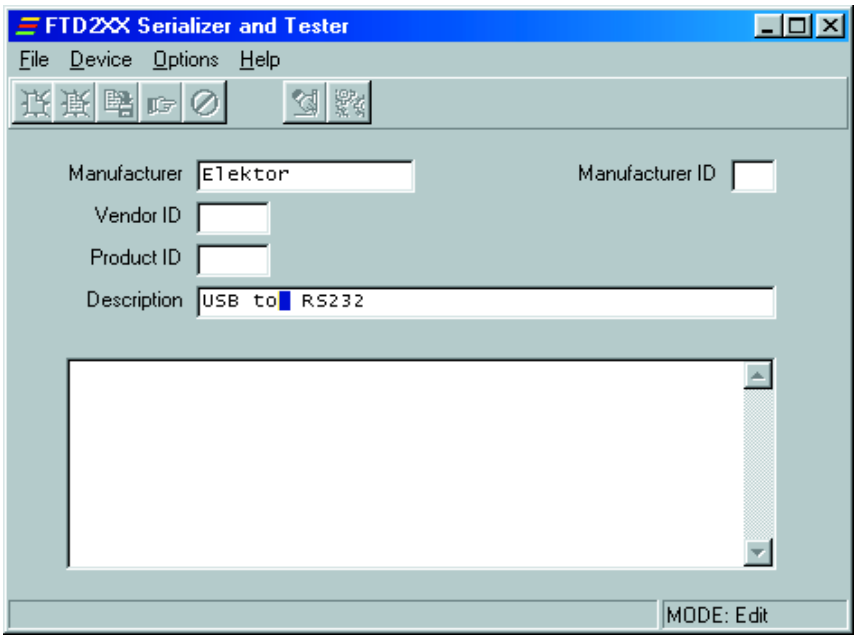


Figure 9. Utilitaire pour la programmation (optionnelle) des PID et VID dans l'EEPROM.

La recopie d'écran de la **figure 9** montre la fenêtre de programmation de l'outil (*tool*) servant à la programmation de L'EEPROM reliée au FT232AM, programme que l'on pourra également télédécharger du site FTDI. Il permet à ceux d'entre vous qui envisageraient, en tant que fabricant, d'implanter la puce FTDI dans un appareil, de programmer leur propre VID et PID. Si vous ne possédez pas de VID et PID propres, vous pouvez soit purement et simplement ne pas vous en inquiéter et ne pas implanter l'EEPROM, soit utiliser les PID et VID de FTDI en tant qu'utilisateur de la puce. Dans le cas du FT232AM on utilisera dans ce cas-là VID = 0403 et PID = 6001. Si vous voulez en savoir plus à ce sujet nous vous recommandons la lecture du manuel de programmation (lui aussi disponible au téléchargement).

Il nous faut cependant, en guise de conclusion, signaler qu'un convertisseur de ce type possède lui aussi ses limites. Il est très important, pour garantir une conversion souple entre USB et RS-232, d'assurer un suivi correct du flux de données, de manière à éviter que le tampon du FT232AM (128 et 384 octets respectivement) n'ait plus une taille suffisante ce qui se traduirait par des pertes d'octets. Ce risque n'existe cependant que si l'on travaille à des taux de transmission élevés sans utiliser d'acquittement.

(020375)

Bibliographie :

– « USB pour tous » de Markus Müller et Christian Ehmer, Elektor n °293 et 294, novembre et décembre 2002
– « UART USB » de Burkhard Kainka, Elektor n °283 et 284, janvier et février 2002
– « Interface USB » de Burkhard Kainka, Elektor n ° 267 et 268, septembre et octobre 2000
– « Des CMS ? Pas de panique ! », Elektor n ° 297 et 298, mars et avril 2003

Télédéchargements

pour le présent projet :

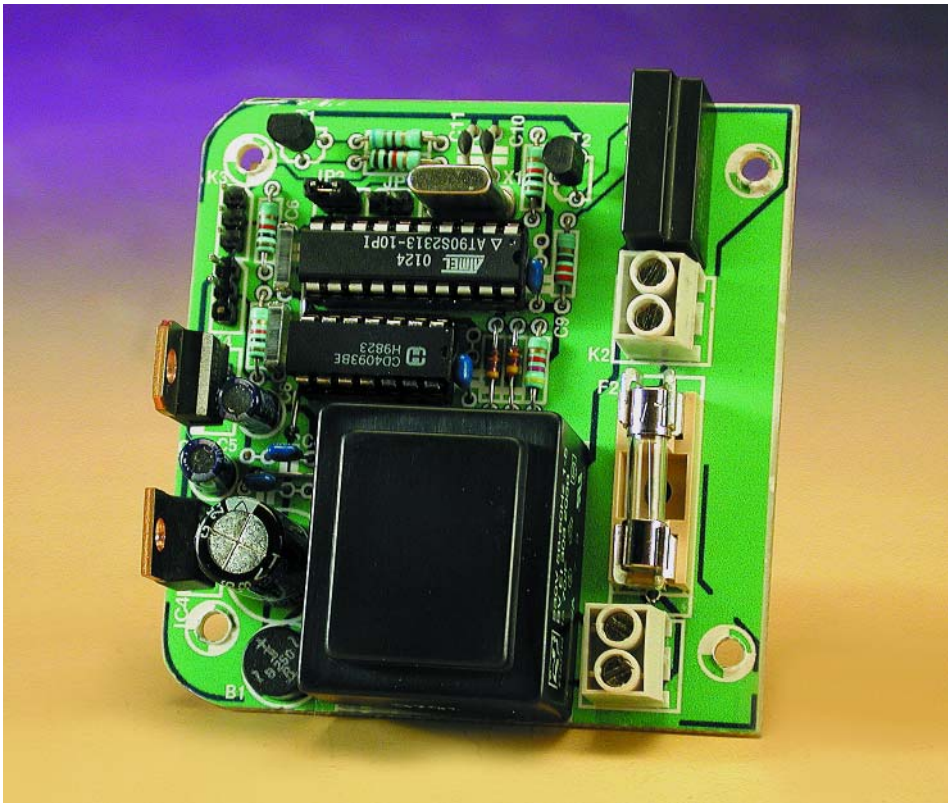
– Dessin des pistes de la platine au format .pdf sous www.elektor.fr/dl/dl.htm
– Fiche de caractéristiques du FT232AM, documentation et pilotes sous : www.ftdichip.com/FTDriver.htm

Noctilum

Éclairage temporisé à AT90S2313

Volker Schmidt

La spécificité de Noctilum, notre éclairage de nuit, est son extinction temporisée. Ce processus est piloté par un microcontrôleur minuscule à compteur 16 bits intégré.



Le cahier des charges était simple : il fallait que Noctilum puisse être allumé et éteint « normalement », mais il devait également comporter une fonction de temporisateur (*timer*) qui éteindrait la lumière au bout d'un certain temps. La commande de Noctilum doit pouvoir se faire par le biais de 2 interrupteurs (voire plus), l'un de ces interrupteurs se trouvant en local sur l'éclairage, l'autre (ou les autres) pouvant être disposé(s) à une certaine distance. L'interrupteur « local » devra en outre être doté de LED qui non seulement en faciliteront la découverte dans le noir mais en outre en visualiseront l'état.

L'électronique

Le processus de commande ayant été confié à un microcontrôleur, la complexité de l'électronique de ce montage n'a rien de bien effrayant. Sur le schéma de la figure 1 on retrouve le microcontrôleur AVR, un AT90S2313 d'Atmel, qui, par le biais de sa ligne de port PD0 et le transistor T2, attaque un relais à semiconducteurs, IC5, un S201S02. Ce relais constitue l'interrupteur à isolation galvanique pris dans la boucle de la charge entre K1 (bornier de la tension secteur) et K2 (lampe). Notre relais comporte en outre un commutateur au passage par zéro de la tension du secteur qui met la charge en fonction à l'instant précis, comme le suggère sa dénomination, du passage par zéro de la tension du secteur. Il nous faut, pour l'alimentation de l'électronique, une paire de tension continues régulées (bien entendu isolées galvaniquement du secteur) de +5 V (tension fournie par IC4) et de +12 V (IC3).

Les diodes D4 et D5 font subir aux 2 interrupteurs connectés aux borniers K3 et K4 une fonction logique « OU » avant de les relier à la ligne de port PB0. Au repos, en l'absence de toute action sur l'un des interrupteurs, l'entrée du trigger de Schmitt inverseur se trouve, par le biais, selon le cas, de R2 ou R3, au niveau bas, et la sortie partant au niveau haut.

De par la présence de D4 et D5, le niveau de sortie ne doit pas correspondre au niveau d'entrée, R4 établissant un niveau haut de 5 V sur l'entrée du microcontrôleur. Les portes font ainsi également office de convertisseur de niveau de 12 V vers 5 V. La question que l'on peut se poser est de savoir pourquoi on a

opté pour de 2 tensions d'alimentation différentes. La raison en est simple. Vu qu'il se peut fort bien que l'on ait des câbles d'une certaine longueur entre la platine du contrôleur et les touches, une tension d'alimentation plus élevée augmente très sensiblement l'immunité aux parasites et cela au prix très acceptable d'un régulateur de tension et de 3 condensateurs.

En cas d'action sur une touche, le niveau de l'entrée de porte concernée bascule au niveau haut, la sortie passant de ce fait au niveau bas. De par la tension de seuil des diodes le niveau bas appliqué à l'entrée du contrôleur ne descend que jusqu'à 0,9 V, valeur que le contrôleur n'a cependant pas de problème à interpréter comme un niveau bas. L'électronique ne comporte pas de dispositif anti-rebond pour les signaux, ce processus étant réalisé logiciellement dans le programme stocké dans le contrôleur AVR.

Le transistor T1 est piloté par le biais des sorties PD1 ou PD2 (ceci en fonction du paramétrage des cavaliers JP1/JP2). Ces cavaliers permettent d'alimenter les LED de l'interrupteur 1. Si on a implanté le cavalier JP1, les LED du boîtier sont allumées en continu et se mettent à clignoter lorsque le temporisateur est activé. En cas d'implantation du cavalier JP2, les LED sont normalement éteintes et se mettent à clignoter lors de l'activation du temporisateur.

La fréquence du quartz fournissant l'horloge au microcontrôleur est de 4,915 2 MHz, valeur qui s'explique par le besoin d'obtenir une fréquence de 10 Hz nécessaire à des fins internes.

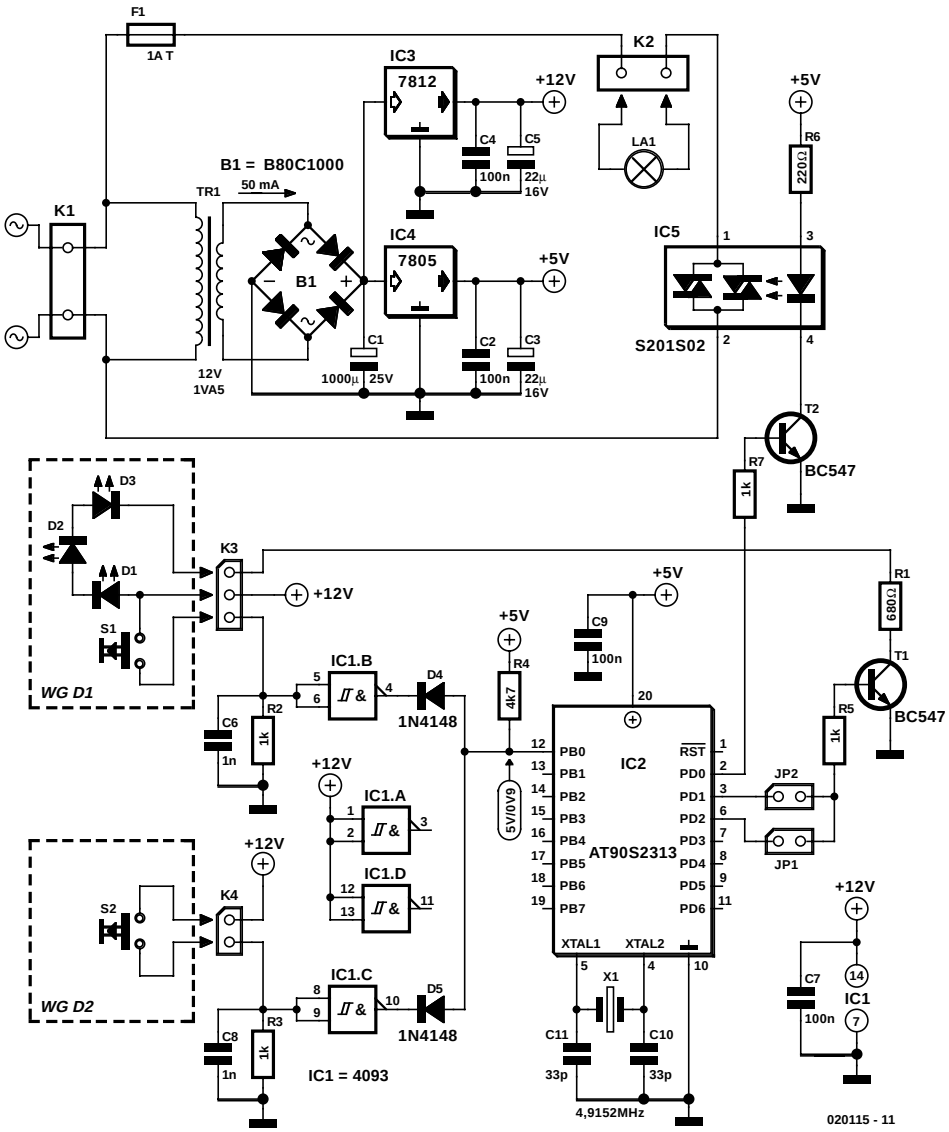


Figure 1. Schéma de Noctilum. Les organes de commande (dans les cadres pointillés)

Le programme

L'auteur a fait appel au Starter Kit STK-500 d'Atmel pour le développement et le test du programme. Pour son écriture il s'est servi de l'assembleur Wavrasm, son test s'est fait à l'aide du programme AVR Studio 3.2, le transfert vers le microcontrôleur se faisant également par le biais de cet environnement.

Le programme de Noctilum, disponible au téléchargement depuis le site d'Elektor (www.elektor.fr) sous la forme du fichier assembleur (en allemand) et de celle du fichier compilé, se compose de 6 modules auxquels nous allons nous intéresser d'un peu plus près, l'intérêt de l'aspect matériel restant relativement limité.

Initialisation (INI)

Dans ce module on définit les paramètres qui déterminent la fonction des sous-ensembles internes du microcontrôleur. Le premier pas consiste à définir le vecteur d'interruption pour le temporisateur Timer1 qui travaille en mode de comparaison (*Compare Mode*). On entre à cet endroit l'adresse du programme d'interruption (étiquette ini). En ce qui concerne Timer1 on valide ensuite, par mise à « 1 » du bit OCIE1A du registre TIMSK, l'interruption de comparaison. Dans le registre TCCR1B on paramètre à 1 024 le prédiviseur de Timer1 et on active l'option CTC. Lorsque le contenu des registres OCR1AL/OCR1AH a atteint la valeur requise on a chargement d'un « 0 » et reprise du processus à son début. On saisit ensuite les valeurs maximales (OCR1AL : 30_{HEX} et OCR1AH : 00_{HEX}). Ce 30_{HEX} correspond à 48 en décimal. Avec le prédiviseur et un quartz de 4,915 2 MHz, la routine d'interruption est appelée toutes les 10 ms. L'étape suivante concerne le paramétrage du registre DD (*Data Direction*) pour les ports B et D. Le port B est défini comme entrée, le port D comme sortie. Pour finir, on met à « 1 » le bit I du registre SREG et on valide les interruptions.

Le programme principal

Le programme ne comporte que 4 instruc-

Paramétrage des SFR du AT90S2313

Paramétrage du Timer1

Paramètre Timer - Interruptions TIMSK

TOIE1	OCIE1A			TICIE1		TOIE0		HEX
0	1	0	0	0	0	0	0	40

Interruption Timer/Counter1 Compare Match activée. On a génération d'une interruption lorsque le contenu du compteur = celui du registre Compare)

Paramètre CTC / Prédiviseur TCCR1B

ICNC1	ICES1			CTC1	CS12	CS11	CS10	HEX
0	0	0	0	1	1	0	1	13

(Timer/Counter1 à 0000, lorsque le contenu du compteur = celui du registre Compare, prédiviseur paramétré pour division par 1 024 (CS12 à CS10))

Charger Registre Compare

OCR1AH 00h
OCR1AL 30H
0030H = 48 dez

(Le Timer 1 travaille en source d'horloge et génère une interruption toutes les 10 ms. Pour ce faire, la fréquence du quartz de 4 915 200 Hz est ramenée, après division par le prédiviseur et Timer1, à 100 Hz)

Parmétrage du pointeur de pile *(stackpointer)*

SPL = DFh

Parmétrage des Ports

Registre DDRD	FFh Port D est une sortie
Registre DDRB	00h Port B une entrée

Parmétrage SREG

Positionner Bit 1, validation des interruptions

tions. Elles appellent les 3 routines INPUT, LOGICOUT et OUTPUT. Lorsqu'elles ont été exécutées, la boucle de programme reprend à son début.

Routine temporisateur pilotée par interruption (SUB TIMER)

La routine d'interruption génère une horloge de base de 10 ms. DELAY s'en sert pour une fonction anti-rebond des boutons-poussoirs. Cette horloge incrémente en outre les registres TAKT10 (dont sont dérivées toutes les grandeurs chronovariables) et TAKT (mesure de la durée d'action sur les boutons). Ce processus se fait toutes les 100 ms lorsque le registre TAKT10 a été incrémenté à 10D. Ce registre est ensuite remis à zéro et le processus reprend à zéro.
On a, parallèlement, génération, dans la routine OUTPUT, d'un signal de seconde utilisé par le dispositif de « visualisation d'état » et, dans la routine LOGICOUT, d'un signal de dizaine de secondes pour le temporisateur de sortie (OUTC). En fonction de l'état des registres SEC1 ou SEC10, on a, dans la routine STEUER, positionnement des indicateurs (*flag*) F1SEC et F10SEC correspondants. L'indicateur F1SEC se trouve à « 1 » pendant 500 ms, avant d'être à « 0 » les 500 ms suivantes. L'indicateur F10SEC est positionné toutes les 10 s avant d'être remis à zéro dans la routine LOGICOUT.

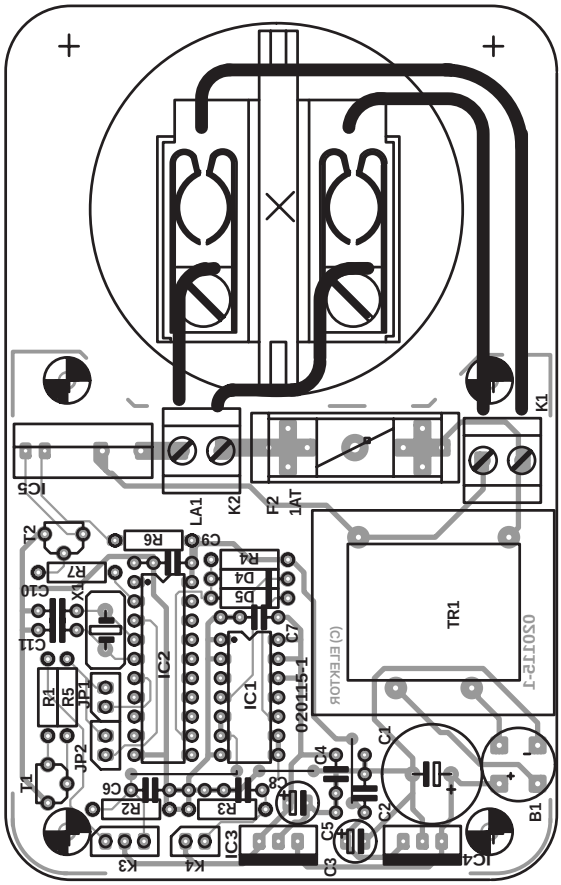
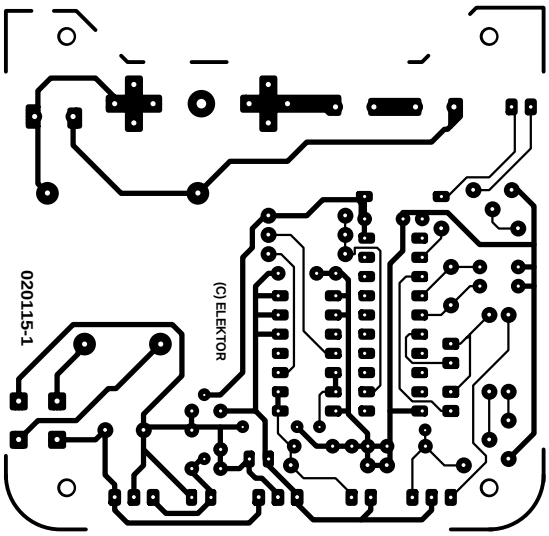


Figure 2. Noctilum est monté dans un coffret secteur à prise « SCHUKO ».



Liste des composants

Résistances :
R1 = 680 Ω
R2,R3,R5,R7 = 1 kΩ
R4 = 4kΩ7
R6 = 220 Ω

Condensateurs
C1 = 1 000 µF/25 V radial
C2,C4,C7,C9 = 100 nF
C3,C5 = 22 µF/16 V radial
C6,C8 = 1 nF
C10,C11 = 33 pF

Semi-conducteurs :
Pont de redressement rond B80C1000
D1 à D3 = LED
D4,D5 = 1N4148
T1,T2 = BC547
IC1 = 4093
IC2 = AT90S2313-10PC (Atmel, programmé **EPS 020115-41**)
IC3 = 7812
IC4 = 7805
IC5 = S201S02 ou S201SE2 (Sharp)

Divers :
F1 = porte-fusible + fusible 1 A retardé
JP1,JP2 = embase autosécable mâle SIL à 2 contacts + cavalier
K1,K2 = bornier encartable à 2 contacts au pas de 5 mm (RM5)
K3 = embase autosécable mâle SIL à 3 contacts ou picots
K4 = embase autosécable mâle SIL à 2 contacts ou picots
S1,S2 = bouton-poussoir à contact travail (cf. texte)
TR1 = transfo 1 x12 V/1VA5 (tel que, par exemple, Hahn BV EI 302 2022)
X1 = quartz 4,915 2 MHz
Boîtier tel que, par exemple, Micro N12 (Bopla)

SUB INPUT
La routine INPUT est activée lorsque l'on force la ligne de port PB0 à la masse. Le flanc descendant né de l'action sur le bouton-poussoir démarre DELAY à des fins d'anti-rebond du bouton. Après le saut de retour de la routine DELAY on a un nouveau test de l'état de la ligne de port pour vérifier si elle est toujours à zéro. Si tel est le cas, le contenu du registre TAKT est recopié dans le registre START et l'indicateur FKENN activé. Sinon, on a retour au programme principal. Lors du relâchement du bouton-poussoir on a recopie du contenu du registre TAKT vers le registre STOP, ceci en vue de déterminer la longueur de l'action sur le bouton-poussoir. Un éventuel dépassement sera détecté par une comparaison entre les registres TAKT et STOP. Si la durée de l'action est inférieure à 1,5 s ($0F_{HEX}$), on aura positionnement (mise à « 1 ») de l'indicateur FKL15, si elle dépasse cette valeur ce sera le cas de l'indicateur FGR15. Ensuite, dans les 2 cas, on a remise à zéro de l'indicateur de reconnaissance de flanc, FKENN.

SUB LOGICOUT
La routine LOGICOUT comporte la logique pour la sortie et la commande du temporisateur de coupure (*sleeptimer*) dans le registre OUTC. Tout au début on vérifie si l'indicateur FLK15 ou FGR15 est positionné. Dans le premier cas on a basculement de l'indicateur OUT du registre STEUER (on a, par son biais, allumage ou extinction de la lampe). Dans le cas contraire, on aura positionnement des indicateurs OUT et OTIML. OTIML constitue le critère de lancement du temporisateur de coupure dans OUTC. Ce registre est incrémenté toutes les 10 s, jusqu'à ce qu'il ait atteint la valeur de la constante OUTIM (180_D). À ce moment-là, OUT et OTIML sont effacés et le programme se retrouve à son état initial.

SUB OUTPUT
OUTPUT pilote la sortie au niveau du port D. Le traitement de cette routine dépend de l'état de l'indicateur OUT. S'il est positionné, on aura saut vers l'étiquette (*label*) OUTPUT 1. Après mise de la ligne PD0 au niveau haut, on procède à l'interrogation de l'état de l'indicateur OUTIML pour déterminer la suite du déroulement du programme. Si cet indicateur est inactif (à « 0 ») la routine OUTPUT se termine par un saut de retour au programme principal.
À l'inverse, s'il est actif (« 1 »), les instructions en assembleur qui suivent se traduisent par l'apparition sur les lignes de port PD1 et PD2 de signaux d'horloge d'une fréquence de 1 Hz déphasés de 180 °.
Si l'indicateur OUT n'est pas positionné, les lignes de port PD0 à PD2 sont mises à leur état initial (PD0 et PD1 à « 0 », PD2 à « 1 ») et la routine se termine.

SUB DELAY
La routine DELAY se charge de l'anti-rebond au niveau des boutons-poussoirs. Elle utilise à cet effet le compteur 10 ms TAKT10. Lors de son appel, on a recopie de son état dans le registre DELAY, valeur augmentée de 2 (20 ms). Si le contenu est supérieur à 10, on soustrait 10 du contenu de DELAY. Cela est nécessaire vu que le compteur TAKT10 est remis à zéro lorsque son contenu atteint 10.
La boucle de programme qui suit est exécutée le nombre de fois nécessaire jusqu'à ce que les contenus de DELAY et TAKT10 soient identiques. Lorsque cette condition est atteinte on a un saut de retour vers le programme principal.

Aspect mécanique de la réalisation

L'électronique prendra place dans un petit boîtier à prise et fiche secteur intégrées. Il faudra cependant découper quelque peu la platine si on veut arriver à la glisser dans le boîtier mentionné dans la liste des composants. L'implantation des composants ne devrait pas poser de problème et le montage fonctionner du premier coup pour peu que l'on n'ait pas fait d'erreur au niveau de la polarité des composants, de leur identification et que l'on n'ait pas oublié de mettre en place les 3 ponts de câblage.
On pourra, mais cela n'est pas indispensable, mettre les interrupteurs dans leur propre boîtier. Les câbles de liaison seront alors glissés par le dessous du boîtier (ne pas oublier la bride anti-arrachement) et connectés aux borniers K3 et K4. Il nous paraît cependant judicieux de mettre le bouton-poussoir S1 et les LED dans le boîtier secteur enfichable.

(020115)

Téléchargements

À la page www.elektor.fr/dl/dl.htm
Vous trouverez les fichiers correspondants à ce projet pour les télédécharger :
– Programme de l'éclairage de nuit en assembleur et en code compilé (source et fichier hexadécimal)
– Dessin de la platine au format .pdf.