ELEKTOR

ELEKTOR

ÉLECTRONIQUE et MICRO-INFORMATIQUE

N° 293
NOVEMBRE 2002
mensuel
Belgique 5,80 €
Suisse 10,40 FS
Canada 6,90 $Can.
DOM surface 6,04 €
DOM avion 7,55 €
ISSN 0181-7450
www.elektor.fr

ELEKTOR à EDUCATEC du 20 au 23/11

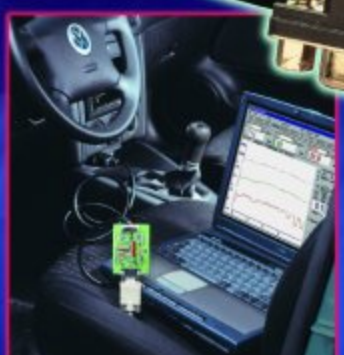
DIAGNOSTIC AUTO

par adapteur de réalisation perso

Pico-API

Émulateur d'EPROM

- Alarme pour moto
- Les radars routiers
- Tube Box
- Ports sériels sous Windows



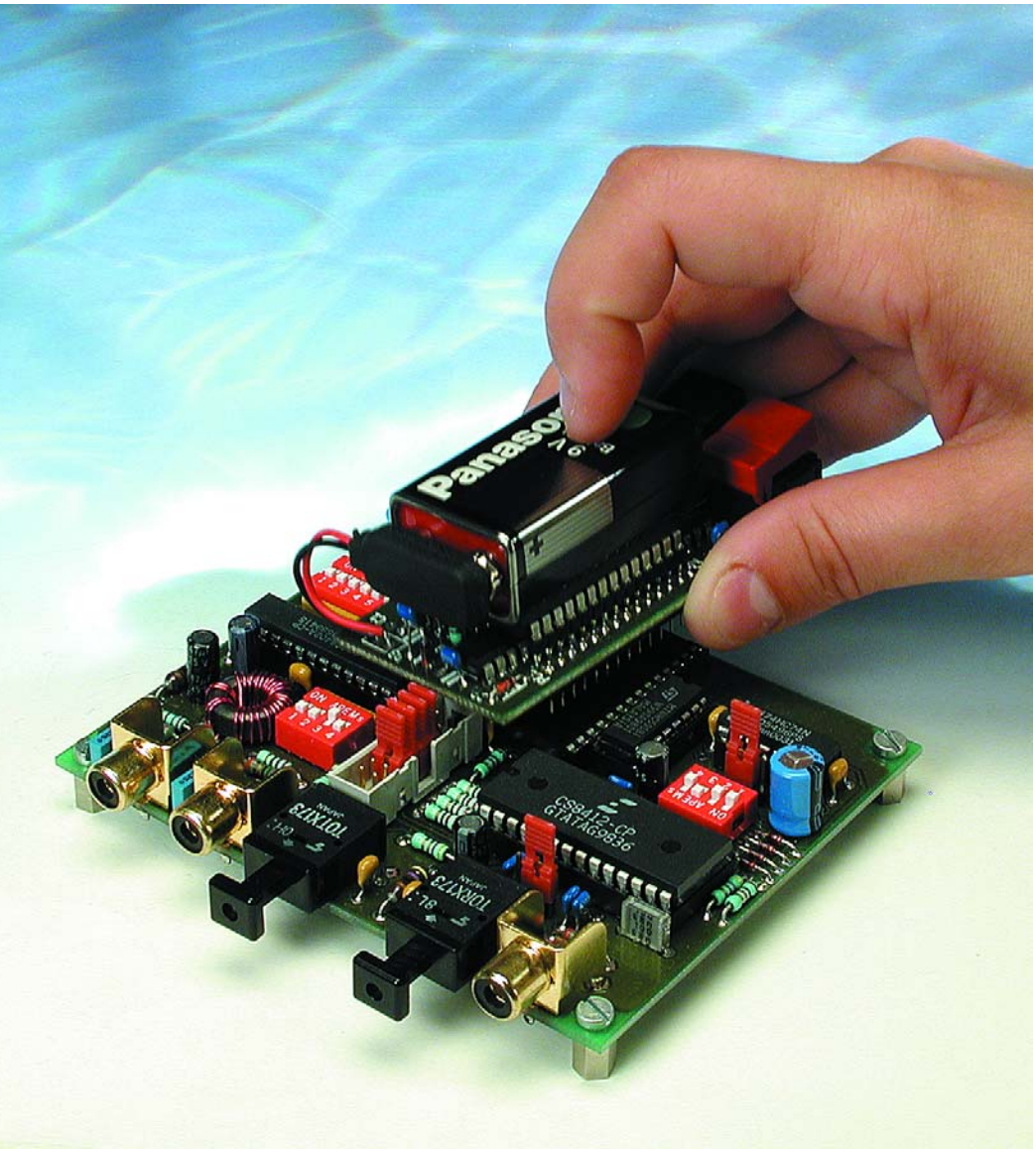
M 01531 - 293 - F: 5,00 €

Émulateur d'EPROM

Constitue une EPROM de substitution parfaite

Projet : Paul Goossens

À la différence de la plupart de ses homologues, l'émulateur décrit dans le présent article simule une 27C256 à la perfection, à tel point qu'il se laisse programmer dans n'importe quel programmateur d'EPROM et qu'il pourra remplacer l'original dans n'importe quelle réalisation.



Nous n'avons pas la prétention de vous apprendre qu'il est extrêmement pratique, lorsque l'on doit développer un programme pour un montage comportant une EPROM, de disposer d'un émulateur d'EPROM. Une EPROM d'imitation de ce type évite d'avoir à parcourir, après chaque modification du programme, le fastidieux cycle effacement aux ultra-violets suivi d'une reprogrammation. La durée d'effacement atteint en règle générale de l'ordre de 20 minutes, attente et perte de temps dont on se serait fort bien passé. Si l'on tient à tester à tout prix une nouvelle version de programme pendant cette temporisation imposée, il reste la solution d'utiliser une seconde EPROM mais cette approche est loin d'être pratique, sans même parler des risques de substitution qu'elle comporte.

On a vu apparaître, au fil des ans, nombre d'émulateurs d'EPROM sur le marche, Elektor ne voyant pas être en reste en a également proposé diverses versions dans ses colonnes. La plupart du temps, si ce n'est pas pratiquement toujours, ces émulateurs sont reliés au support de l'EPROM correspondant par le biais d'un connecteur terminant un morceau de câble plat. Il faut éviter que cette liaison ne dépasse une certaine longueur en raison des capacités parasites qui pourraient naître sur les bus de données et d'adresse, ce

qui se traduirait par d'éventuelles réflexions à ce niveau. On aura compris qu'il est souvent délicat, pour cela, de connecter un émulateur d'EPROM au système-cible.
La dite approche présente un autre inconvénient, il est en effet pratiquement impossible de programmer ce type d'émulateur par le biais d'un programmateur d'EPROM classique. Il est bien souvent nécessaire d'utiliser un programme spécifique livré à cet effet avec l'émulateur.

L'émulateur présenté ici ne présente aucun des inconvénients cités plus haut. Il se laisse programmer dans un programmateur d'EPROM et l'implantation de l'ensemble du montage dans le circuit-cible à l'endroit où se trouve normalement l'EPROM se fait sans la moindre difficulté.

L'EPROM

Normalement, la programmation d'une EPROM se fait à l'aide d'un programmateur d'EPROM. La plupart des programmateur commencent par vérifier que le composant qu'on leur « confie » est totalement vide (on parle du test de virginité). À cet effet, il entreprend la lecture de tous les octets et s'assure qu'ils ont tous pour contenu la valeur 0xFF. Si ce n'est pas le cas, il en déduit que l'EPROM n'est pas vierge, déduction qu'il affiche par le biais d'un message avant de croiser purement et simplement les bras.
Certains programmateurs modernes disposent d'une capacité de détecter le type et la marque de l'EPROM que l'on vient de mettre dans le support FIN (à **F**orce d'**I**nsertion **N**ulle), ce qui le met en demeure d'utiliser l'algorithme et les niveaux de tension de programmation les mieux adaptés. Cette fonction d'auto-information prend la forme d'une application de la tension de +12 V sur la broche A9 avant de procéder à la lecture de certaines adresses de l'EPROM. Ces adresses additionnelles ne peuvent pas être

Caractéristiques

- Émule une 27C256
- Peut être programmé par n'importe quel programmateur d'EPROM capable de programmer une 26C256
- Protection électrique empêchant la lecture de l'identificateur composant (DEVICE-ID)
- Se laisse facilement effacer
- Compact
- Aisé à mettre en oeuvre

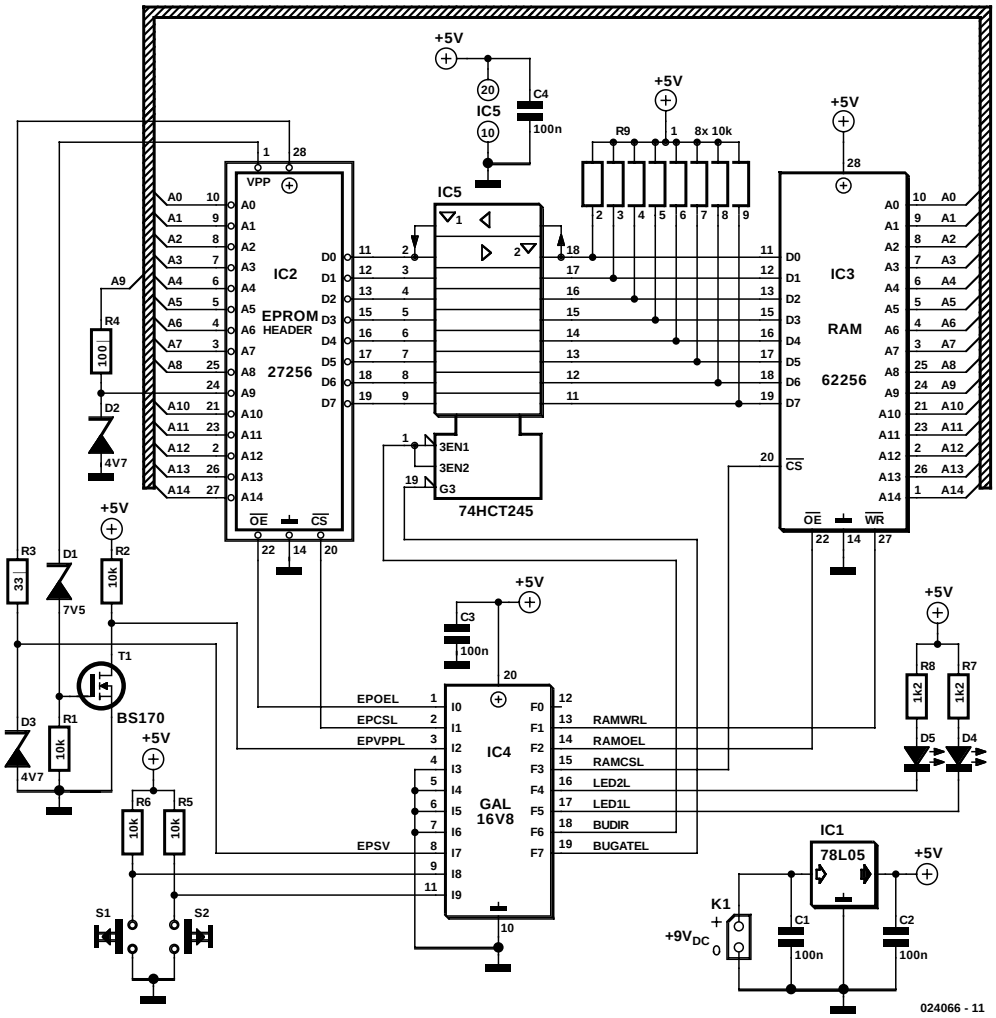


Figure 1. La caractéristique la plus marquante du schéma de l'émulateur est sa simplicité. Les contacts de « IC2 » servent à établir la connexion de l'EPROM artificielle.

programmées et sont uniquement sélectées par l'application d'une tension de +12 V à la ligne A9.
S'il s'est avéré que l'EPROM est bien vierge, le programmateur passe à l'étape suivante, la programmation proprement dite. Pour cela on commence par forcer l'EPROM à l'état de programmation par la mise de l'entrée V_{pp} à une tension relativement élevée. Le niveau de cette tension est fonction du type d'EPROM concerné et peut varier entre 12, 5 et 21 V. L'étape suivante consiste, dans la plupart des cas, par une augmentation de la tension d'alimentation de +5 vers +6 V. Un coup d'oeil sur la partie supérieure de la **figure 4** proposée un peu plus loin.
L'application d'un niveau bas à l'entrée $\overline{CS}$ de l'EPROM lorsque cette dernière se trouve en mode de programmation se traduit par la programmation dans l'EPROM des signaux présents sur les lignes de données à l'adresse définie par le bus d'adresses. L'application d'un niveau bas à l'entrée $\overline{OE}$ permet la lecture de la donnée présente à l'adresse définie à cet instant. Dans la plupart des cas, un programmateur moderne vérifie immédiatement après la programmation d'un octet que la programmation du dit s'est faite correctement dans l'EPROM. Si c'est bien le cas, il programme l'octet suivant. La longueur de l'impulsion de programmation est fonction de l'algorithme de programmation utilisé de même d'ailleurs que la reprise ou non du cycle de programmation au cas où l'octet concerné n'aurait pas été programmé correctement.
En fonctionnement normal (Vpp et Vcc = 5 V) l'EPROM réagit de façon normale aux signaux CS et OE. Cela signifie que ce n'est que lorsque les signaux CS et OE sont actifs (mise au niveau bas de la ligne correspondante) que le contenu de l'EPROM correspondant à l'adresse appliquée sera placé sur le bus de données.

L'électronique

Le schéma de l'émulateur d'EPROM reproduit en **figure 1** ne comporte guère plus de 3 circuits intégrés, un régulateur, un transistor FET et une paire de LED indicatrices. En ce qui concerne IC2 il ne s'agit pas en fait d'un circuit intégré, mais d'une paire de barrettes sécables à 14 contacts qui serviront à établir le contact avec le support destiné à recevoir l'EPROM émulée par la présente électronique.
IC3 est un circuit de RAM (*Random Access Memory*) du type 62C256. La liste des com-

```
*IDENTIFICATION
    024066;
*TYPE
    GAL16V8;

*PINS

% Pin-assignment for input signals %

    EPVP    = 8,        % Detection of 5V supply                %
    /EPVPP  = 3,        % Low active detection of VPP > 12V     %
    /EPOE   = 1,        % Input for /OE from EPROM-socket       %
    /EPCS   = 2,        % Input for /CS form EPROM-socket       %
    /SW1    = 9,        % Input for switch S1                   %
    /SW2    = 11,       % Input for switch S2                   %

% Pin-assignment for output signals %

    /RAMCS.t   = 15,    % output for /CS of RAM IC3             %
    /RAMOE.t   = 14,    % output for /OE of RAM IC3             %
    /RAMWR.t   = 13,    % output for /WR of RAM IC3             %
    /BLANK.t   = 16,    % BLANK-state output for LED D5         %
    /PROGRAM.t = 17,    % PROGRAM-detect output for LED D4      %
    /BUGATE.t  = 19,    % output for /G of buffer IC 5          %
    BUDIR.t    = 18;    % output for DIR of buffer IC 5         %

*BOOLEAN-EQUATIONS
   RAMCS.e   = VCC;     % These lines make sure that all used   %
   RAMOE.e   = VCC;     % output-lines of the GAL are           %
   RAMWR.e   = VCC;     % constantly driven                     %
   BLANK.e   = VCC;
   PROGRAM.e = VCC;
   BUGATE.e  = VCC;
   BUDIR.e   = VCC;

% Switch to BLANK-state is S1 is pressed and hold this state    %
% Until the EPROM programmer wishes to program the EPROM        %
% OR the user wishes to switch to normal operation by pressing  %
% S2                                                            %
% This signal also drives LED D5 to indicate the BLANK-mode     %
   BLANK   = BLANK*/PROGRAM*/SW2+SW1 ;

% Detection of a programming pulse to drive LED D4              %
   PROGRAM = EPVPP*EPCS*/EPOE;

   % CS for the RAM : Continuous during programming, else equal %
   % to CS on the ROM                                          %
   % Only active during presence of external power to save the %
   % battery                                                   %
   RAMCS= EPVPP*EPVP+/EPVPP*EPCS*EPVP;

   % OE for the RAM : During programming if OE on the ROM is   %
   % active AND Cs of the ROM is inactive                      %
   RAMOE= EPVPP*EPOE*/EPCS + /EPVPP*EPOE*/BLANK;

   % WR for the RAM : During programming if ROM-OE is inactive %
   % AND ROM-CS is inactive, else if in BLANK-mode AND ROM-OE  %
   % is active. This means that reading while in blank-mode    %
   % causes the RAM to be written. Pull-ups make sure that the %
   % code 0xFF will be written                                 %
   RAMWR= EPVPP*EPCS*/EPOE + /EPVPP*BLANK*EPOE;

   % The DIR-signal for the 74HCT245. When this pin is active, %
   % Data is transferred from the socket to the RAM-IC         %
   % Otherwise, the datapath is FROM the RAM to the socket     %
   BUDIR = EPVPP*/EPOE;

   % The GATE-signal for the 74HCT245. If it is active (low)   %
   % the drivers are active, otherwise the A and B pins are    %
   % in High-Z state                                           %
   % It has to be active even in the BLANK-mode, so it wil force%
   % 0xFF on the external databus during blank-check!          %

   % BUGATE is only active when external power is supplied to  %
   % save power from the battery whenever the device is not    %
   % required to be active                                     %

   BUGATE=EPVP*EPVPP*EPOE + EPVP*EPVPP*EPCS  + EPVP*/EPVPP*EPOE;
*END
```

Figure 2. Listage du programme présent dans IC4, la GAL.

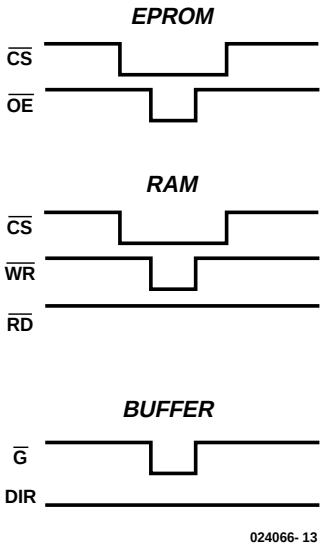


Figure 3. Chronodiagramme des signaux pendant le mode « Blank ».

posants prévoit une version 70 ns, mais il n'y a aucune contre-indication à utiliser un modèle plus rapide. Nous déconseillons cependant d'utiliser un modèle de RAM plus lent.

IC4 est une GAL du type 16V8, un circuit de logique programmable, programmée de façon à convertir les signaux externes en signaux compréhensibles par IC3. Ce circuit intégré a besoin, pour remplir sa tâche de conversion, outre de certaines connexions au support d'EPROM, également des informations de présence (ou d'absence) d'une tension externe sur le contact Vcc et d'une tension de >12 V sur le contact V_{pp}. La combinaison de la résistance R3 et de la diode D3 protège la GAL contre des niveaux de tension élevés pouvant être appliqués au contact V_{cc} de l'EPROM. Cette précaution est nécessaire vu que très souvent, lors de la programmation, la tension d'alimentation est rehaussée, passant de +5 vers +6 V. L'électronique centrée sur le transistor T1 sert à la détection de la présence (ou non) de la tension de programmation sur le contact V_{pp} du support de l'EPROM. La diode zener D1 entre en conduction en cas de présence de la tension de programmation.

La tension sur la grille de T1 est alors suffisante pour faire passer ce transistor en conduction. En association avec la résistance R1, la diode zener D1 abaisse la tension à 7,5 V. Cet abaissement de la tension empêche T1 de devenir passant lorsque la tension présente sur le contact V_{pp} est de 5 V.

IC5 est un circuit intégré tampon chargé, vous l'avez sans doute deviné, de tamponner les signaux de donnée sur leur trajet entre le bus de données et l'EPROM. La présence de ce composant est nécessaire au cours du mode Blank (test de virginité) de l'émulateur. Lorsqu'il se trouve dans ce mode, l'émulateur doit produire, en permanence, lors de chaque opération de lecture, la valeur 0xFF. La GAL se charge de faire en sorte que le circuit de commande (*driver*) soit actif dans le sens B –> A et qu'il n'y ait pas lecture de l'EPROM (plus exactement, on donne une instruction d'écriture au circuit de RAM, nous y reviendrons un peu plus loin). En raison de la présence des résistances de forçage au niveau haut (*pull up*) du réseau R9, IC5 ne « voit » que des « 1 » sur l'entrée B de sorte qu'il les place sur le bus de données externe.

Vu que certains programmateurs d'EPROM sont dotés de résistances de forçage au niveau bas (*pull down*), il est indispensable d'intégrer le tampon IC5 dans le montage vu que sinon, en fonction du programmateur utilisé, il faudrait choisir, pour R9 une valeur de résistance tellement faible que l'émulateur pourrait se comporter comme une source de tension non-négligeable

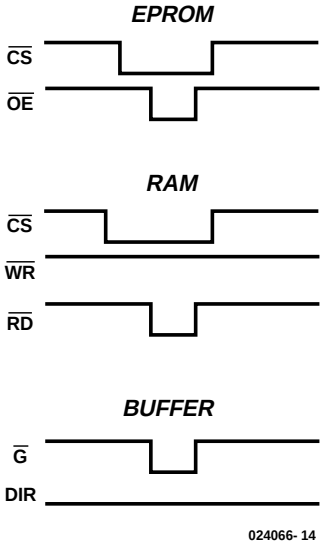


Figure 4. Les signaux en cours de programmation.

par rapport au montage-cible.
Le bouton-poussoir S1 a pour fonction de mettre l'émulateur en mode Blank, S2 servant au contraire à quitter cet état manuellement.
Il nous reste, avant d'en avoir terminé avec l'électronique, à parler des composants qui entourent le régulateur de tension IC1. Il s'agit de rien de plus que d'un régulateur 5 V standard chargé d'abaisser à cette valeur la tension fournie par la pile de 9 V.

La GAL

Si l'on fait abstraction de IC3, la GAL peut se targuer d'être le composant le plus important de ce montage. Comme nous le disions plus haut, la GAL intègre toute la logique requise pour la conversion des signaux externes en signaux dont IC3 a besoin pour pouvoir fonctionner comme il faut.
Nous vous proposons, en **figure 2**, le listage du contenu de la GAL. Il ne s'agit que d'une logique très rudimentaire, mais si nous l'avions remplacée par des circuits intégrés standard le montage aurait pris un embonpoint qui l'aurait rendu inutilisable.
La GAL connaît 3 états distincts, à savoir : les modes « Blank Check », « Normal » et « Program ». Le fonctionnement de la GAL change d'un mode à l'autre. L'utilisateur peut choisir manuelle-

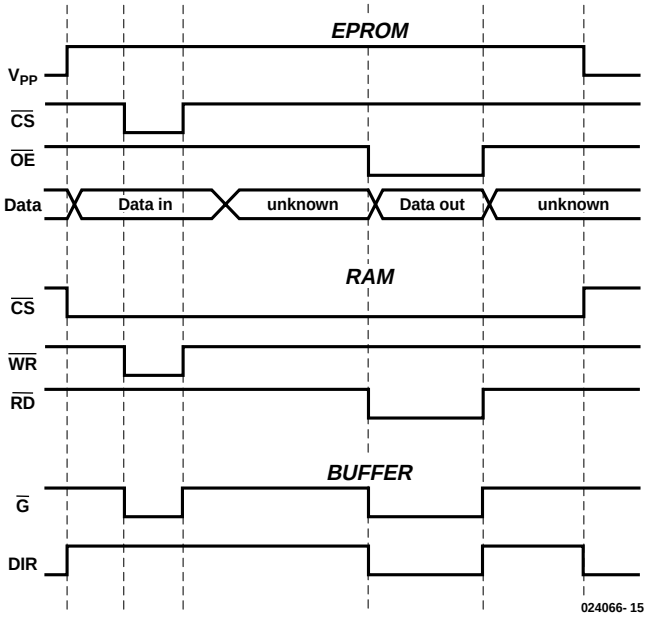


Figure 5. Les signaux en mode « Normal ».

Liste des composants

Résistances :
R1,R2,R5,R6 = 10 kΩ
R3 = 33 Ω
R4 = 100 Ω
R7,R8 = 1kΩ2
R9 = réseau de 8 résistances de 10 kΩ

Condensateurs :
C1 à C4 = 100 nF

Semi-conducteurs :
D1 = diode zener 7V5
D2,D3 = diode zener 4V7
D4 = LED à haut rendement (*high-efficiency*) jaune
D5 = = LED à haut rendement (*high-efficiency*) rouge
T1 = BS170 ou BS107
IC1 = 78L05
IC3 = 62256-70CP
IC4 = GAL 16V8 programmée **EPS024066-31**
IC5 = 74HCT245

Divers :
IC2 = embase mâle à 2 rangées de 14 contacts
S1,S2 = bouton-poussoir (Digitast)
K1 = connecteur à pression pour pile 9 V

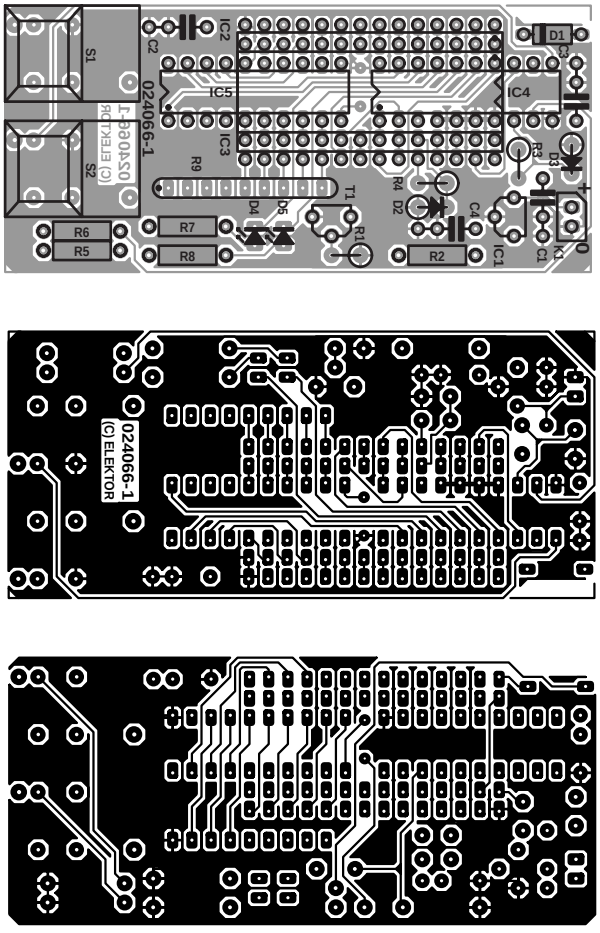
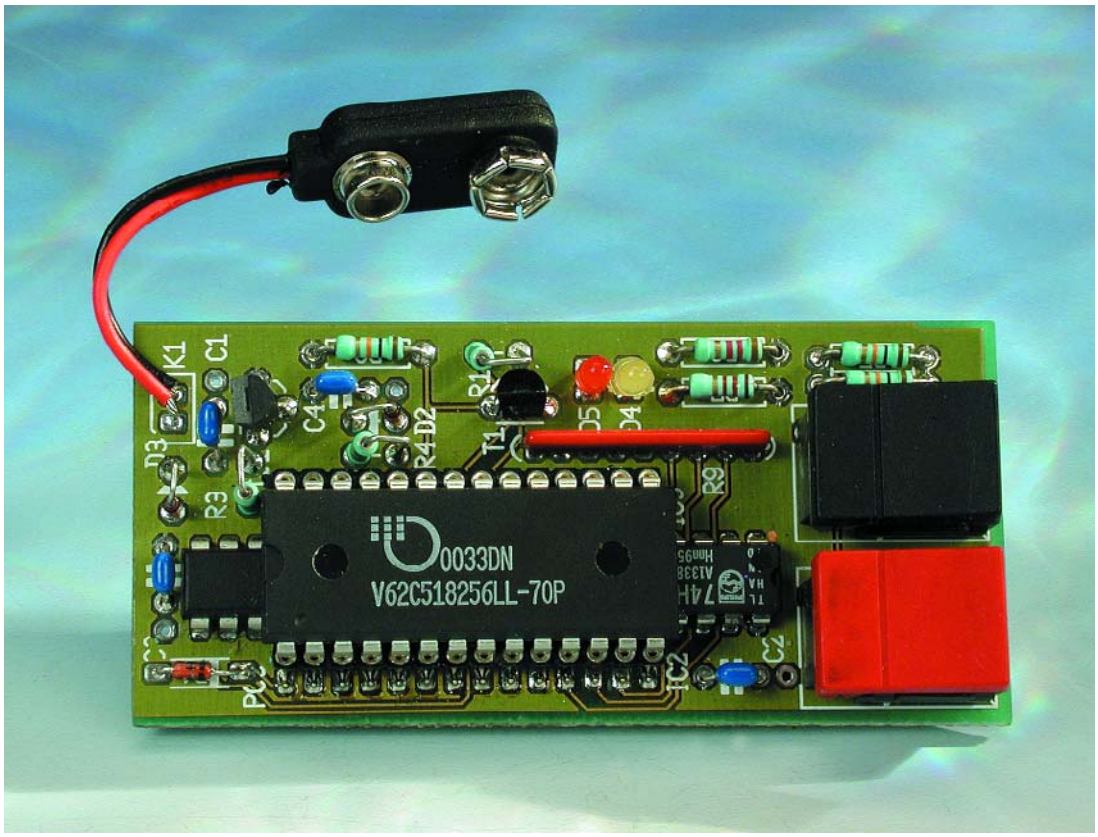


Figure 6. Le montage en gigogne de IC3 par-dessus IC4 et IC5 permet de réaliser un montage très compact.

ment le mode « Blank Check » et « Normal ». Une seule pression sur le bouton S1 fait passer la GAL en mode « Blank Check ». Il est possible alors d'effacer le contenu de l'émulateur en faisant exécuter un test de virginité (*blanck check*) par le programmateur. Ce mode reste en fonction jusqu'à ce que l'utilisateur actionne le bouton-poussoir S1 ou que le programmateur se mette à programmer l'émulateur.

Une unique pression sur S2 fait passer l'émulateur en mode « Normal ». L'émulateur se comporte à partir de là comme une EPROM normale. Dès que le programmateur se met à programmer l'émulateur (Vpp passe à >12,5 V), l'émulateur passe automatiquement en mode « Program ». Dès la fin de l'opération de programmation (Vpp est repassé à = 5 V) l'émulateur revient automatiquement en mode « Normal ».



Mode « Blank Check »

Ce mode remplit une double fonction, à savoir, d'une part, simuler une EPROM vierge, et ensuite à vider effectivement la RAM de son contenu. Le chronodiagramme de la **figure 3** montre la forme des signaux au cours du mode « Blank Check ». Comme nous le signalions plus haut, la grande majorité des programmateurs s'assurent que l'EPROM confiée à leurs soins est bien vide avant d'en effectuer la programmation. Dès lors qu'ils se sont assurés de la virginité de l'EPROM, la majorité des programmateurs met à profit cette caractéristique pour augmenter la vitesse de l'opération de programmation. Comme le programmateur sait que chaque adresse de l'EPROM (vierge) comporte la valeur 0xFF, il peut se passer de programmer les adresses où il aurait fallu programmer un 0xFF. Ces sauts lui font gagner du temps. Cela implique cependant que l'émulateur d'EPROM soit effectivement totalement « vide » avant qu'il ne soit programmé.

La technique classique de contrôle de virginité d'une EPROM consiste à la lecture de toutes ses adresses et à une comparaison des données lues avec la valeur 0xFF. Le programmateur génèrera partant toutes les adresses possibles et produire, pour chacune des adresses, une impulsion de lecture. Nous pouvons mettre ce processus à contribution pour vider « l'EPROM ». La conversion de cette impulsion de lecture produite par le programmateur en une impulsion d'écriture destiné au circuit intégré de RAM on pourra programmer chacune des adresses. Toutes les lignes de données sont forcées au +5 V par le biais du réseau de résistances R9, ce qui permet de remplir IC3 de données 0xFF partout.

Le programmateur lui-même a émis une instruction de lecture de sorte qu'il s'attend à trouver des données (0xFF de préférence) sur le bus de données. C'est là la tâche du tampon IC5. L'entrée DIR reste au niveau bas, ce qui signifie que les signaux A sont montés en sortie et que les signaux B le sont en entrée. Ces signaux B sont eux aussi forcés au +5 V par le biais du réseau R9 de sorte que le tampon placera les données souhaitées (0xFF) sur le bus de données.

Programmation

Le signal $\overline{\text{EPVpp}}$ permet à la GAL de détecter une tension de programmation (= 12,5 V). Dès qu'elle voit apparaître une tension de ce type, la GAL fera en sorte que le comportement de l'émulateur soit celui d'une EPROM en mode de programmation. Une EPROM se trouvant dans cet état réagit différemment aux signaux $\overline{\text{CS}}$ et $\overline{\text{OE}}$. En fait, on pourrait fort bien, dans cet état, changer la dénomination de ces signaux en $\overline{\text{WR}}$ et $\overline{\text{RD}}$. On remarque en outre qu'il n'y a plus de signal supplémentaire pour la sélection de l'EPROM. Cela n'est d'ailleurs pas nécessaire en cours de programmation vu que l'EPROM n'a pas à partager les bus de données et d'adresses avec d'autres composants de mémoire. Dans ce mode, l'EPROM se trouve pour ainsi dire sélectée en permanence (cf. le chronodiagramme de la figure 4).

Au cours d'une impulsion de programmation le circuit intégré tampon doit transférer les données du programmateur d'EPROM vers IC3. Ceci s'obtient par la mise au niveau haut du signal DIR. Pendant une impulsion de lecture (le signal OE est actif) le tampon doit

taponner les données dans l'autre direction ce qui explique qu'il faudra que la GAL force alors l'entrée DIR au niveau bas.

Mode normal

La **figure 5** rend le chronodiagramme des signaux lorsque l'émulateur d'EPROM se trouve en mode « Normal ». Un coup d'oeil montre à l'évidence que la GAL n'a pas, dans ce mode, à remplir de fonctions délicates ou complexes.
La GAL remplit, pour finir, une fonction d'économies d'énergie lorsque l'émulateur n'est pas en service. Par le biais de la paire R3/D3 IC4 détecte la présence d'une tension d'alimentation externe. En l'absence d'une telle tension on pourra fort bien forcer IC5 et IC3 à l'inactivité. IC4 se charge de veiller à ce que les signaux G et CS ne puissent pas devenir actifs, ce qui se traduit par une consommation moindre de IC3 et IC5 lorsque le montage n'est pas utilisé. Une caractéristique intéressante vu que l'alimentation de notre émulateur se fait par le biais d'une pile.

L'alimentation

L'ensemble du montage est alimenté en permanence par une pile compacte de 9 V.
C'est à dessein que nous n'avons pas choisi d'alimenter le montage à partir de la ligne +5 V de l'EPROM. En effet, lors de la programmation cette tension peut atteindre +6 V, ce que les circuits intégrés de l'émulateur n'apprécient pas nécessairement tous. Si vous voulez quand même utiliser cette tension au lieu d'une pile, il vous faudra prendre les mesures requises en vue de stabiliser la dite tension à +5 V. On pourrait envisager d'utiliser un convertisseur CC/CC, mais les avantages présentés par cette approche sont largement écrasés par ses inconvénients. Le montage ne peut manquer de prendre de l'embonpoint et devient plus encombrant. À cela s'ajoute qu'il est impossible de supprimer totalement la pile vu que le montage ne se trouve pas relié en permanence à la tension d'alimentation externe de sorte qu'IC3 perdrait le contenu de sa mémoire dès que l'émulateur d'EPROM quitte le programmateur d'EPROM pour aller s'implanter dans le circuit-cible.

La réalisation

Nous vous proposons, en **figure 6**, le dessin des pistes et la sérigraphie de l'implantation des composants de la platine qui permet de faire de l'émulateur d'EPROM un montage compact. Cette étape de réalisation ne devrait guère vous poser de problème. Comme d'habitude, nous commencerons par l'implantation des composants de l'épaisseur relative la plus faible, les résistances montées horizontalement. On passera ensuite à la mise en place des autres composants discrets, des circuits intégrés et des boutons-poussoirs. Cette partie de l'implantation des composants appelle 2 remarques. La première concerne le connecteur de l'émulateur identifié par un « IC2 » sur la sérigraphie. Il s'agit en fait d'une paire de rangées de 14 contacts devant ressortir par le dessous de la platine ce qui explique qu'ils soient montés par le dessous; il est préférable d'effectuer ce montage avant de mettre les circuits intégrés en place. Autre détail marquant, le circuit de RAM, IC3, vient se monter au-dessus des circuits IC4 et IC5. Cette disposition implique de mettre d'abord IC4 et IC5 sur la platine avant que IC3 ne vienne pendre place sur un support rehausseur. Ce support pourra prendre la forme d'une paire de barrettes de 14 contacts en tulipe. Il va sans dire que les circuits IC4 et IC5 sont à monter directement sur la platine vu que sinon il faudrait monter IC3 sur un échafaudage de 2 épaisseurs de barrette. Attention à l'implantation de IC3 : ce circuit intégré doit se trouver, vu du dessus décalé vers la droite par rapport aux rangées de contact de « IC2 » (les boutons-poussoirs se trouvant en bas).
Il reste à évoquer quelques détails pratiques. On pourra penser à coller la pile de 9 V sur le dos de IC3 à l'aide d'un morceau de film autocollant double face de manière à réaliser un ensemble compact. Il peut être intéressant voire nécessaire de limer le bord de la platine du côté de la diode D1 de manière à faciliter le passage du levier du support FIN du programmateur d'EPROM. En pratique, nous avons constaté qu'il suffisait de lever partiellement ce levier pour pouvoir glisser les contacts de l'émulateur d'EPROM dans le support de sorte qu'il n'est pas toujours nécessaire d'effectuer cette opération de limage.

Les derniers pas

Une fois la réalisation du montage terminée et que l'on y a connecté la pile, l'émulateur d'EPROM est prêt à être utilisé. Il est cependant recommandé de procéder à quelques tests de bon fonctionnement. Il est facile de vérifier le bon état de la GAL par une action sur S1.
On devrait voir alors s'allumer la LED rouge D5, une pression sur la touche S2 devrait la faire s'éteindre. Mettez l'émulateur d'EPROM dans le programmateur et appuyez sur S1. Demandez au programmateur d'effectuer un test de virginité. Si tout se passe comme prévu, le programmateur annonce que l'EPROM est vierge (*blank*). Appuyez ensuite sur S2 pour faire passer l'émulateur en mode « Normal ». Relancez un test de virginité. Ce test également devrait se passer normalement. Cela prouve que chaque emplacement de mémoire de IC3 contient effectivement la valeur 0xFF.
On pourra, comme dernier test, vérifier la fonction de programmation. On choisira, pour cela, un fichier de programmation quelconque dans le cadre du logiciel de programmation et demandera au programme de programmer l'EPROM. ; la LED D4 doit d'allumer. Il restera à effectuer une vérification pour s'assurer que le contenu de l'émulateur (la RAM) correspond bien à celui du fichier de programmation que l'on a transféré.
Si vous deviez rencontrer un problème, prenez le temps de vérifier l'ensemble de votre réalisation. Les aspects importants auxquels il faut faire attention est l'orientation des circuits intégrés, la polarité des diodes et la valeur correcte des composants (cf. la liste des composants et le schéma). Assurez-vous de l'absence de court-circuit ou de mauvais contact au niveau des soudures.
Il est judicieux en outre de monter, dans le support destiné à l'EPROM du montage-cible, un support FIN, ce qui facilitera les opérations de mise en place et d'extraction de l'émulateur d'EPROM sans risque d'usure pour le support du montage. Ce support FIN permet normalement de se mettre à l'abri de phénomènes bizarres dus à un mauvais contact entre l'émulateur et le montage-cible. Les supports bon marché en particulier ont la mauvaise habitude de s'user rapidement et de perdre le ressort de leurs contacts, ce qui est loin de faciliter les recherches en cas de problème.

(024066)

La guerre des formats DVD

DVD–RW vs DVD+RW

Harry Baggen

Les graveurs de DVD arrivent ! Après le succès du graveur de CD, il semblerait, à première vue, que le graveur DVD ait la prétention, à court terme, de le remplacer dans le PC. Un DVD possède une capacité de 4,7 Goctets soit 7 fois celle d'un CD ! Plus d'une dizaine de fabricants proposent des modèles de graveurs DVD, le prix de certains d'entre eux étant déjà inférieur à 500 E. Le seul problème non résolu pour le moment est l'impossibilité de définir un standard commun.

Le DVD (*Digital Versatile Disc*) peut se targuer, au cours des 2 dernières années, d'une popularité à la croissance exponentielle. Il se substitue très rapidement à la bande VHS. Tous les films et les concerts les plus récents sont disponibles aujourd'hui sur DVD, et d'une qualité d'image et de son (Surround bien souvent) remarquable. Cela tient à cette capacité incroyable de 4,7 Goctets (par couche) !

Maintenant que le DVD est devenu un media courant, l'étape logique suivante est celle du DVD enregistrable, à l'image de ce que nous connaissons depuis des années déjà dans le cas du CD.

Cela fait déjà quelque temps que l'on y travaille. On vit apparaître les premiers graveurs de DVD en 1997, mais à un prix exorbitant. De nos jours leur prix a atteint un niveau que l'on peut qualifier d'« acceptable », vu que l'on trouve actuellement un graveur de DVD à intégrer dans un PC pour moins de 500 €. Le prix de DVD inscriptibles se situe entre 5 et 15 € pièce. Qu'est-ce qui nous empêche encore d'acheter un graveur de ce type pour nous en servir comme super-système de sauvegarde de données voire pour y mettre les films de vous vacances au format MPEG2 ? Pas grand chose, n'est-ce pas ?

Mais les choses sont moins simples qu'on ne pourrait le penser à première vue. À l'image de ce qui s'est passé avec les bandes vidéo où plusieurs systèmes se sont bagarrés pendant de longues années, nous nous trouvons aujourd'hui en présence de 2 camps défendant chacun son format DVD. Dans l'un d'entre eux on trouve quelques fabricants qui respectent la seule norme officielle définie jusqu'à présent en ce qui concerne les DVD inscriptibles. Cette norme a été définie par le **DVD-Forum** [1] qui définit tous les standardS DVD importants. Ce forum a défini les standard des disques connus sous les dénominations de DVD–RW et DVD–R, le suffixe RW signifiant qu'il s'agit d'un disque réinscriptible et le R d'un disque ins-

criptible 1 seule fois (suffixes que l'on retrouve d'ailleurs dans le cas des CD). Pioneer et Masushita (Panasonic) sont les 2 partisans de ce format et fabriquent de ce fait des graveurs respectant ce standard.
Mais depuis l'an dernier un certain nombre de « gros » fabricants ont créé un nouveau format DVD qui se distingue du précédent par la présence d'un « + » au lieu du « – » : on a ainsi les DVD+RW et DVD+R. Le + semble sous-entendre que ce nouveau format présente un certain nombre d'avantages par rapport au format « – ». Il devrait être compatible avec les lecteurs et châssis DVD existants et devrait être utilisable tant pour de la vidéo que pour des données. Ce standard intègre en outre un certain nombre de caractéristiques fort intéressantes. On dispose ainsi d'une possibilité de choix, lors de la gravure, entre une vitesse linéaire constante (CLV, comme cela est le cas avec les CD ordinaires) et une vitesse angulaire constante (CAV), ce qui se traduit dans le cas de données, par une possibilité d'obtention de temps d'accès plus courts. Ce standard connaît de plus un dispositif de gestion de défauts (*defect managment*) pour une fiabilité accrue et une fonction de formatage rapide (*quick formatting*) permettant une mise en oeuvre rapide de disques vierges. Autre caractéristique, un collage sans pertes (*lossless linking*) qui rend possible l'utilisation de taux de bits (*bitrates*) variables.
On trouve, dans le camp des « + » la quasi-totalité des grands noms tels que Philips, Sony, HP, Yamaha, Ricoh, Dell et bien d'autres pour ne citer que ceux-là. Ces fabricants ont constitué la **DVD+RW Alliance** [2]. Autre site intéressant, le site non-officiel **DVDplusRW.org** [3] où l'on trouve quantité d'informations et de nouvelles du camp « + ». Détail piquant, la plupart de ces fabricants sont également membres du DVD-Forum, mais ils ne sont pas encore arrivés à imposer l'acceptation de ce format pour en faire un standard. De ce fait, on se trouve en présence de deux types différents de graveurs utilisant chacun leur propre type de DVD qui ne sont pas, au niveau de l'écriture, interchangeables.
Ceux d'entre nos lecteurs qui voudraient en savoir plus quant aux détails techniques de ces 2 formats pourront jeter un coup d'oeil aux « **White Papers** » de **l'Alliance DVD+RW** [4] et lire le **Technical Guide** de **Pioneer** [5] (qui concerne lui bien évidemment le DVD-RW).
Au niveau de la compatibilité ces 2 formats ne différentient guère. DVD+ prétend être plus compatible, mais différents tests effectués par des magazines micro importants bien équipés nous apprennent qu'ils sont très proches. Il semble que les disques DVD–R et DVD+ fonctionnent pratiquement dans tous les lecteurs et modèles PC modernes, les résultats des exemplaires réinscriptibles étant un peu moins bons. On trouvera, sur Internet, chez **VCDHelp** [6] une liste de compatibilité exhaustive.
Qui sortira vainqueur de ce combat de Titans ? Le format « + » étant épaulé par un grand nombre des fabricants connus, semble avoir les meilleures chances. Les supporters de la version « – » sont en minorité ce qui est un inconvénient majeur (tout comme ce fut le cas pour Sony qui tenta en vain de défendre contre vents et marées son système Betamax). Il semblerait en outre que Microsoft ait affiché son intérêt pour le format DVD+ que cette société envisage d'intégrer dans les prochaines versions de Windows. Il n'est pas exclu que ce soit là un coup de pouce du « destin » décisif.
D'ici là, il nous faudra faire notre propre choix : notre (prochain) graveur de DVD serait-il du type « + » ou du type « – » ?

(025060)

Pioneer

PIONEER R&D
Technical Guide
DVD Technical Guide — Back to Index

- Chapter 1 DVD Overview
- Chapter 2 Physical Format of Read-Only Discs
- Chapter 3 Read-Only Disc File Format
- Chapter 4 Video Format
- Chapter 5 Audio Format
- Chapter 6 DVD-R and DVD-RW
  - 6.1 History of Specification Publications
  - 6.2 Basic Concept
  - 6.3 Basic Specifications
  - 6.4 Features of the Specifications
  - 6.5 Conclusion
- Chapter 7 DVD-RAM

Chapter 6 DVD-R and DVD-RW

6.3 Basic Specifications

This section will describe the basic specifications that are common to 4.7 GB DVD-R and DVD-RW discs.
As explained previously, the basic playback specifications of DVD-R and DVD-RW discs after recording are the same as those for DVD-ROM discs. As shown in the table below, the reflectivity of DVD-RW discs differs from that of single-layer DVD-ROM discs (but is the same as that of dual-layer DVD-ROM discs). With that exception, other parameters such as recording capacity, density (track pitch, minimum pit length), and recorded signal playback quality follow suit with the parameters of single-layer DVD-ROM discs.

Table 1 Comparison of Basic Playback Specifications with Those of DVD-ROM Discs

DVD standard	single-layer DVD-ROM	DVD-R	DVD-RW	dual-layer DVD-ROM
Laser Wavelength	635 / 650 nm			
Objective lens NA	0.60			
Reflectivity	45 to 85 %		18 to 30 %	
Modulated amplitude	0.60 min.			
Data track form	Single spiral track			
Track pitch	0.74 μm			
Tracking method	DPD (Differential Phase Detection)			
Minimum pit length	0.40 μm			0.44 μm
Data modulation	8 / 16, RLL(2,10)			
Error correction	RS-PC (Reed-Solomon Product Code)			
Channel bit rate	26.16 Mbps			
Scanning velocity	3.49 m/s (CLV)			3.84 m/s (CLV)
User data capacity	4.70 Gbytes / side			4.25 Gbytes/layer

As shown in Figure 3, the recording tracks (grooves) "wobble" at a fixed frequency, and address pits called Land Pre-Pits are positioned between the recording tracks. (The details of this structure will be explained later.) These two types of addressing are used during recording to control disc rotation and generate the recording clock, as well as providing information such as recording address, which is necessary in the recording process.
After recording, the disc Information Area, the playback region, has exactly the

Adresses Internet :

[1] www.dvdforum.com/forum.shtml
[2] www.dvdrw.com/
[3] www.dvdplusrw.org/
[4] www.dvdrw.com/whitepapers.html
[5] www.pioneer.co.jp/crdl/tech/dvd/6-3-e.html
[6] www.vcdhelp.com/dvdplayers.php

Carte de développement pour PIC universelle

Un outil flexible pour l'amateur et l'enseignement

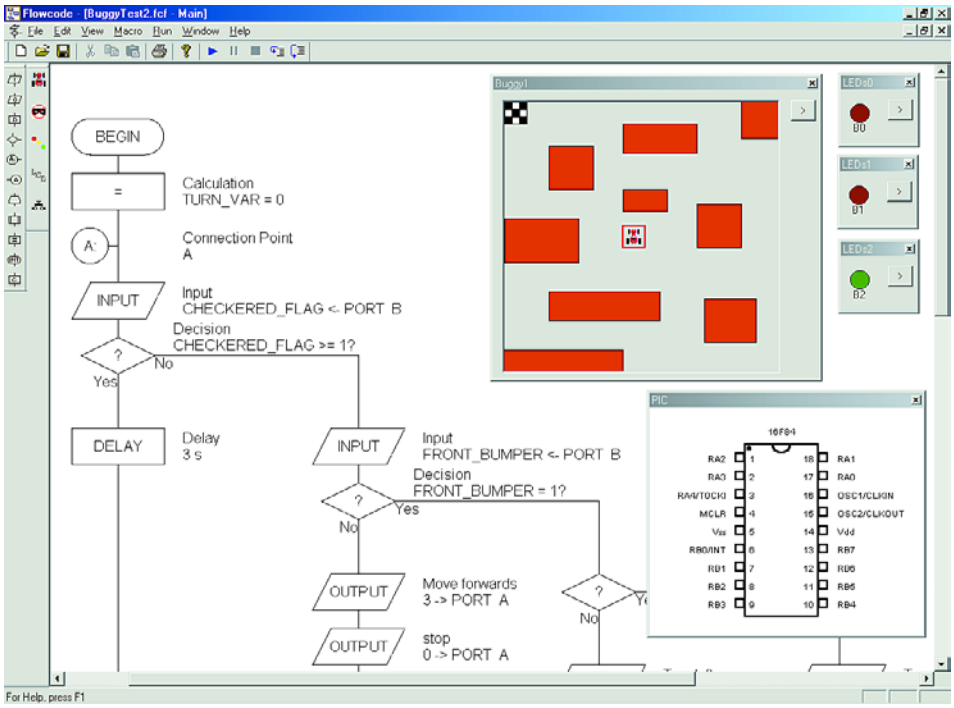
Harry Baggen

Matrix Multimedia, une société anglaise, propose toute une panoplie de produits électroniques à valeur (éducative) ajoutée. Nous avons eu l'occasion d'examiner de plus près l'un de leurs produits : une carte pour microcontrôleur PIC pour laquelle il existe différents sets de programmes.

Il n'est sans pas nécessaire de vous présenter les microcontrôleurs PIC : il s'agit de l'une des familles de microcontrôleurs les plus utilisés en raison de leur prix très abordable, de leur flexibilité et de leur programmation aisée. Les établissements d'enseignement de l'électronique les ont également bien souvent pris dans leur programme d'étude.
Matrix Multimedia, qui est distribué en France par Multipower, est renommé pour ses produits multi-

média et ceux ayant trait à l'électronique; ils ne s'adressent pas uniquement à l'Éducation mais également aux amateurs voulant en savoir plus et aimeraient le faire de façon plaisante.

L'un des produits qui devrait, pensons-nous, intéresser les lecteurs d'Elektor est la carte de développement pour microcontrôleur PICmicro (version 2). Ce petit système de développement permet en effet de réaliser rapidement et de tester facilement des montages à base de microcontrôleurs de la famille des PIC.
La carte comporte un certain nombre d'accessoires qui la rendent universelle. On y trouve, par exemple, des supports pour les PIC à 8, 18, 28 et 40 broches (en standard, c'est un 16F84 qui accompagne cette carte de développement). La programmation devient une opération simple qui se fait par le biais d'un connecteur Centronics que l'on enfiche dans le port parallèle d'un PC. Tous les ports du PIC sont aisément accessibles par l'intermédiaire de bornier (à vis).
La carte est agrémentée d'un nombre impressionnant de boutons-poussoirs et de dispositifs de visualisation : 13 touches permettent de paramétrer les niveaux des entrées des ports A et B, un nombre identique de LED (13) visualisant l'état de chacun des ports (de sortie).
On découvre en outre sur la carte un affichage de 4 afficheurs 7 segments à LED ainsi qu'un affichage LCD à 2 lignes de 16 caractères. Des interrupteurs permettent de les mettre en et hors-fonction. En ce qui concerne les capteurs, on trouve une photo-résistance (LDR) sur la platine, une paire de connexions spéciales pouvant se voir connectées des capteurs numériques et analogique (Matrix propose différents capteurs tout faits dont un détecteur de mouvement).
La vitesse de travail du PIC pourra être fonction de la fréquence du quartz utilisé, mais rien n'interdit non plus d'utiliser un oscillateur RC dont la fréquence sera ajustée à la valeur requise par le biais d'un potentiomètre.
La carte a la taille d'une cassette VHS et, logiquement, est proposée dans un étui pour cassette vidéo. Elle n'est pas accompagnée d'une



alimentation, mais comporte le jack d'alimentation permettant la connexion d'un adaptateur secteur standard voire un porte-pile dotée de 4 piles R6.

De par la présence des différents affichages, touches, interrupteurs et autres connecteurs, cette carte convient à nombre d'applications destinées à expérimenter avec les microcontrôleurs PIC. Le logiciel de programmation PPP qui accompagne la carte est facile à utiliser et possède une fonction de vérification.

Pour le support de cette carte, Matrix Multimedia propose 3 CD-ROM qui concernent la programmation, respectivement, en assembleur, en C et en Flowcode. Ce dernier CD-ROM accompagnait la carte de développement. Nous n'avons pas pu résister à l'envie de l'essayer.
Flowcode pour microcontrôleurs PICmicro est un langage de programmation qui permet de créer un programme pour le PIC à partir d'un ordinogramme. Il n'est pas nécessaire que l'utilisateur ait la moindre notion de langage machine. Il est même possible, grâce aux macro(-instructions), de réaliser des montages relativement complexes. Il est même possible d'enfouir dans le code du C ou de l'assembleur sous la forme de macros. Les signaux d'entrée et de sortie sont visualisés ainsi d'ailleurs que le matériel (boutons, LED, moteurs) avant de simuler sur son PC le programme créé, le résultat de la manipulation des boutons étant visualisé au niveau des sorties. Le CD-ROM comporte un certain nombre d'exemples complets et finalisés.
Ce CD-ROM constitue un complément utile pour la carte de développement et est à recommander si l'on a encore que peut (ou pas) d'expérience dans la programmation des PIC.

Nous venons d'apprendre en toute dernière minute que Multipower dispose d'une version française de ce produit. Pour plus d'information au sujet des produits Matrix Multimedia disponibles en France, un petit tour à l'adresse : www.multipower-fr.com/ s'impose. On pourra y décharger une version de démonstration de Flowcode (4 Moctets).

Si d'autres produits de Matrix Multimedia vous intéressent, vous trouverez toutes les informations voulues (en anglais) à : www.matrixmultimedia.co.uk.

(020280)

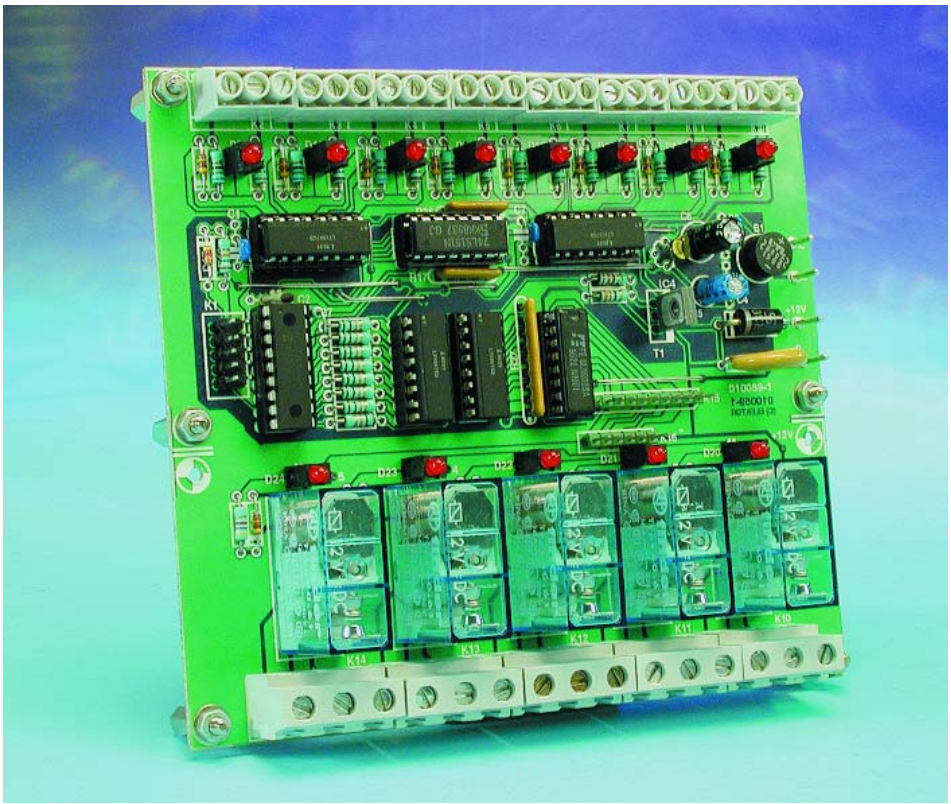
Prix de la carte de développement V2 : 249 €
Prix du CD-ROM Flowcode pour microcontrôleurs PIC : 79 € (monoposte)

Pico-API

Microcontrôleur ou automate programmable ?

Prof. Dr.-Ing. Hermann Gebhard, DF2DS

Depuis toujours ces 2 modes de pilotage ont été très proches l'un de l'autre. Avec la variante proposée dans cet article les différences deviennent encore plus floues. Pico-API (API pour Automate Programmable Industriel) ne requiert que des composants standard, les outils de développement de programme nécessaires sont disponibles gratuitement au téléchargement depuis Internet pour une utilisation non-commerciale.



Il n'en reste pas moins que l'on fait souvent la différence entre les automates programmables (aussi connus sous l'acronyme PLC pour *Programmable Logic Controller* et en RFA sous celui de SPS pour *SpeicherProgrammierbar Steuerung* = automate à mémoire programmable) industriels et ceux qui ne le sont pas. La première caractéristique de ce type de matériel destiné à l'industrie est qu'il travaille, en règle générale, à une tension de 24 V. À cela s'ajoute que ces installations sont extrêmement robustes et protégées contre une inversion malencontreuse de polarité de la tension d'alimentation, les crêtes de tension et les courts-circuits.

Il existe une autre différence, au niveau du logiciel cette fois : un API est piloté par un programme moniteur chargé de la coordination des processus internes. On a, entre autres, traitement d'interruptions, suivi des temporisateurs logiciels et pilotage des protocoles des différentes interfaces (souvent RS-232 ou bus CAN).

L'exécution du programme d'applications (le programme utilisateur) se fait de façon cyclique à intervalles réguliers. Il est possible, de ce fait, de projeter le comportement de l'automate dans le temps et ainsi de le prévoir. L'une des conditions primordiales est alors de ne pas faire appel à des attentes actives -la seule possibilité de l'utilisateur est de s'assurer, au cours de chaque cycle du programme, si un événement donné a

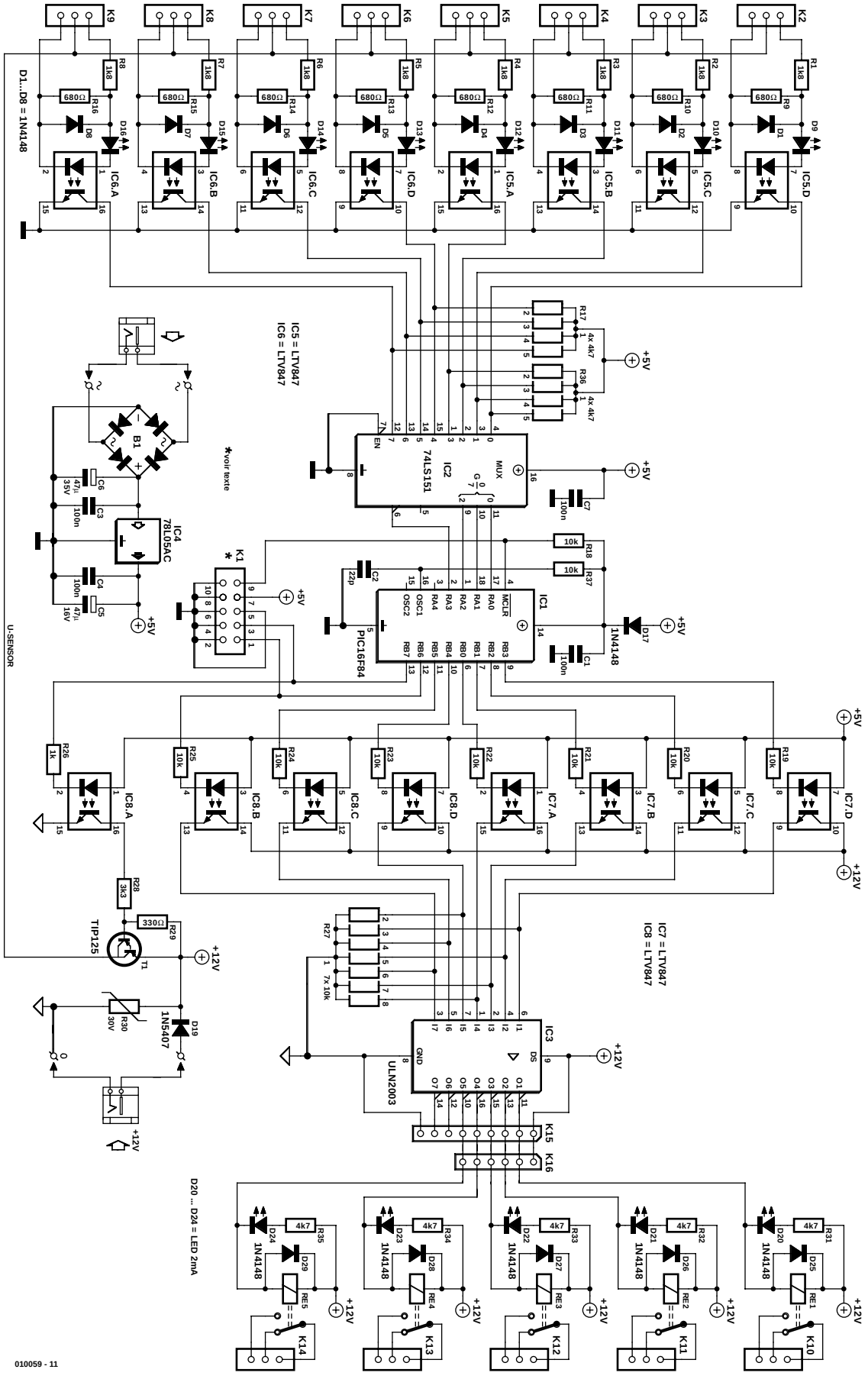


Figure 1. Un microcontrôleur à entrées et sorties isolées.

bien eu lieu.
La programmation d'un API peut faire appel à diverses méthodes telles que les plans de câblage (technique d'interconnexion de schémas) ou listes d'instructions (qui ressemble beaucoup à un fichier-source en assembleur). Le programme utilisateur est transféré dans la mémoire Flash de l'API, son exécution démarrant automatiquement dès la mise sous tension de l'automate.

Microcontrôleur plutôt qu'API ?

Il est difficile de donner une réponse universelle à cette question. Si les systèmes à microcontrôleur sont sensiblement plus flexibles ils requièrent également en règle générale une certaine expérience de la programmation et plus de discipline si on veut les utiliser en tant qu'automate programmable. Ils offrent alors indubitablement plus de possibilités vu que l'on dispose alors de toutes les instructions en assembleur. On a en outre, presque toujours, à sa disposition un compilateur pour langage de haut niveau, pour C ou BASIC, voire dans certains cas pour Pascal ou un autre langage de programmation exotique.
Nous ne pensons pas qu'il faille apprendre à un concepteur-développeur que de par l'utilisation d'un compilateur pour langage de haut niveau il devient quasiment impossible de faire donner au microcontrôleur le maximum de chacun de ses cycles d'horloge. Il n'en reste pas moins que l'on devra préférer la programmation en langage de haut niveau vu que la recherche d'erreur et les modifications de programme ultérieures et les adaptations en deviennent bien plus simples qu'en assembleur.

Le Pico-API

Pico-API est un système pouvant s'adresser aux deux domaines d'application évoqués. Le coeur de cette réalisation est un microcontrôleur de l'écurie Microchip [1], un classique PIC16F84. Ce contrôleur possède 13 ports numériques, dispose de 64 octets d'EEPROM et d'une mémoire de programme en Flash de 1 024 mots de 14 bits. Cette quantité peut sembler dérisoire à première vue lorsque l'on connaît les quantités de mémoire dont disposent les PC actuels, mais elle suffit pour de nombreuses applications vu la structure spécifique et extrêmement efficace des instructions de ce processeur.
Si le contrôleur n'a pas à remplir de tâches à la chronologie « pointue » on pourra même se passer de quartz et générer le signal d'horloge à l'aide de l'oscillateur RC interne du processeur (épaulé par la paire RC R37/C2).
Un coup d'oeil au schéma de la **figure 1** permet de constater que le sous-ensemble centré sur le processeur ne cache pas de secret. Les

Liste des composants

Résistances :
R1 à R8 = 1kΩ8
R9 à R16 = 680 Ω
R17,R36 = réseau SIL de 4 résistances de 4kΩ7
R18 à R25 = 10 kΩ
R26 = 1 kΩ
R27 = réseau SIL de 7 résistances de 10 kΩ SIL (voire à 8 résistances)
R28 = 3kΩ3
R29 = 330 Ω
R30 = Varistor 30 V/600 mW
diamètre 15 à 17 mm tel que, par exemple, BC-Components 2322 5953006
R31 à R35 = 4kΩ7
R37 = 10 kΩ (4kΩ7)*

Condensateurs :
C1,C3,C4,C7 = 100 nF céramique
C2 = 22 p
C5 = 47 µF/16 V axial
C6 = 47 µF/35 V axial

Semi-conducteurs :
B1 = B80C1500 modèle rond

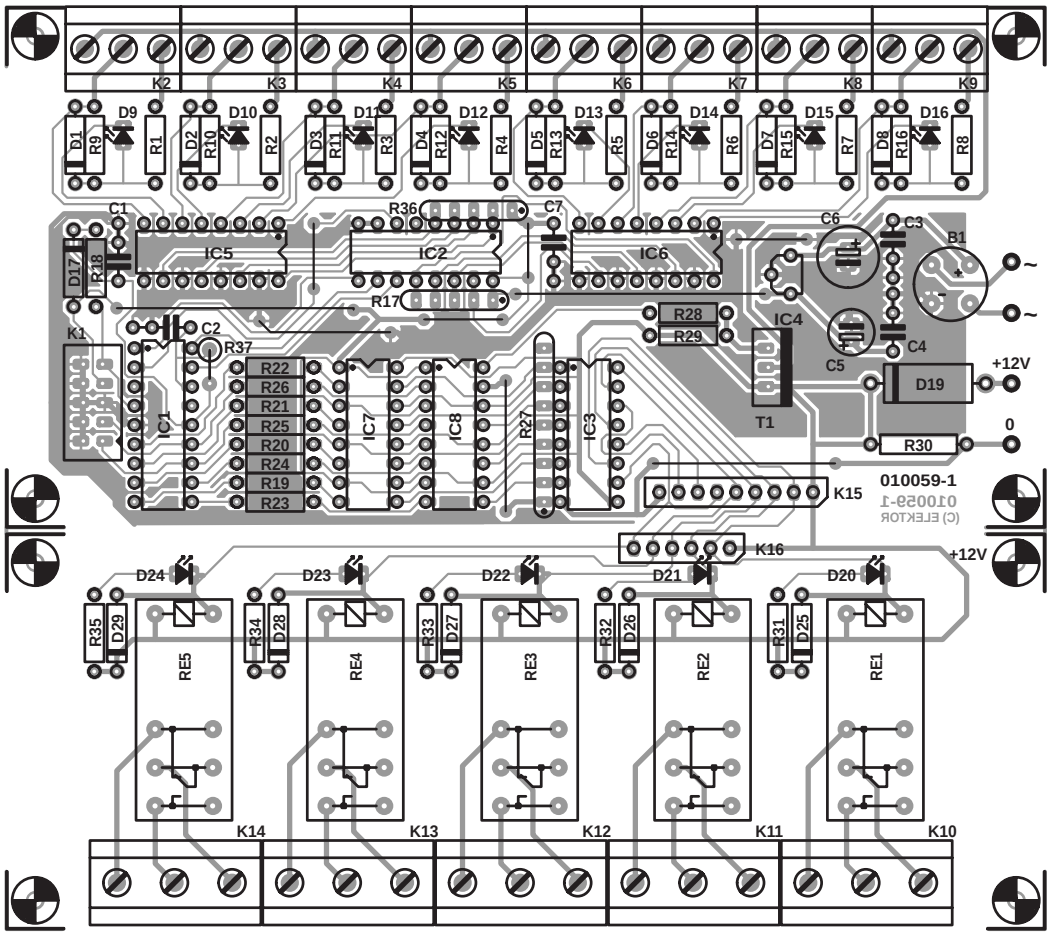
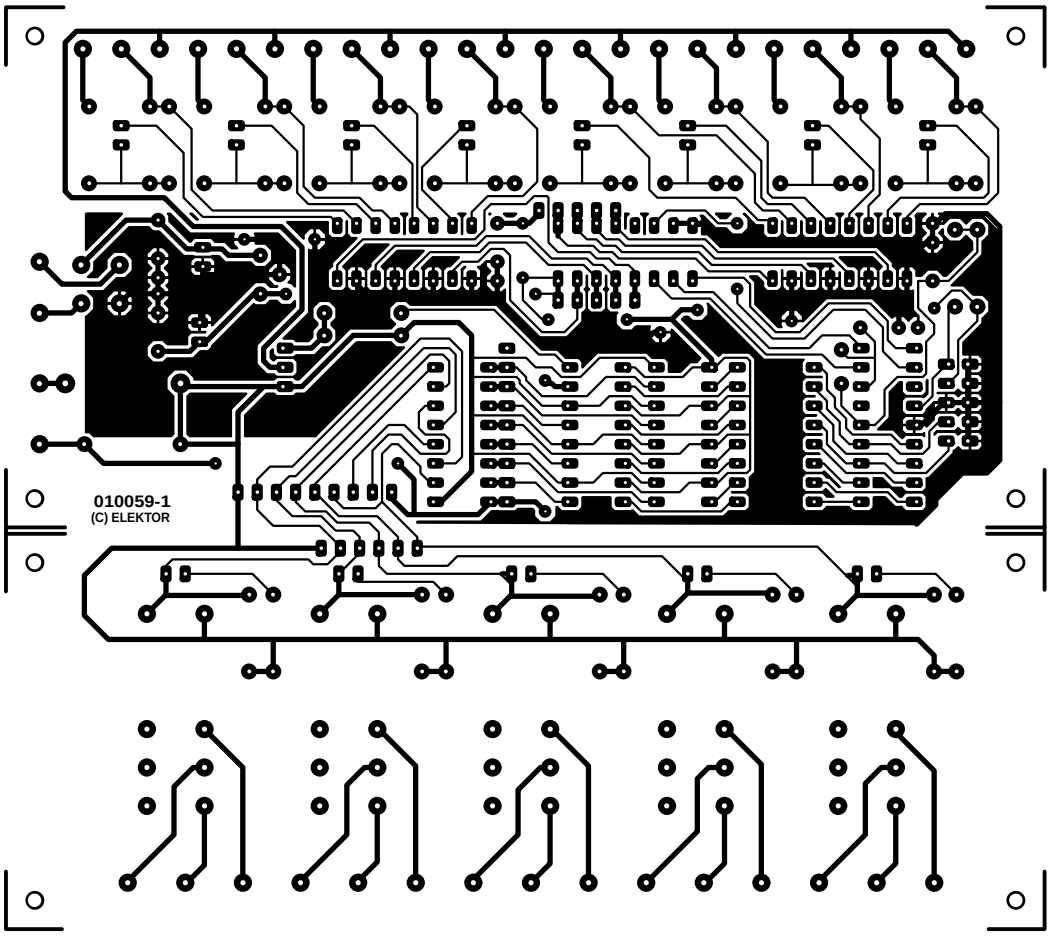


Figure 2. La partie réservée aux relais pourra être détachée de l'API proprement dit.

entrées attaquent, par le biais du multiplexeur IC2, l'une après l'autre le port RA3 programmé en entrée. La sélection des entrées se fait par l'intermédiaire des ports RA0 à RA3. La sortie inverseuse du multiplexeur compense l'inversion introduite par l'opto-coupleur.
Les entrées proprement dites sont ensuite, en concordance avec les normes de conception d'API, mises hors-potentiel par le biais d'opto-coupleurs. Les diviseurs de tension évitent la circulation d'un courant trop élevé au travers des opto-coupleurs en cas d'application d'une tension d'entrée élevée (entre 10 et 30 V dans le cas présent). La diode prise en tête-bêche par rapport à la diode intégrée dans l'opto-coupleur remplit une fonction importante. Elle protège la LED de l'opto-coupleur en cas d'inversion de la polarité de l'entrée, sachant que les LED IR ont une tension inverse maximale faible (6 V au maximum dans le cas du PC847 ou LTV847), ce qui les fragilise face à une manipulation erronée.
La ligne *U-SENSOR* (au bas du schéma), une sortie pilotée par le microcontrôleur, mérite que l'on s'y intéresse de plus près. Cette sortie permet de ne fournir de courant aux capteurs que si cela est nécessaire. Cette mesure d'économie d'énergie est sensible en particulier dans le cas d'applications alimentées par pile.
De même, les circuits de commande (driver) de sortie sont isolés de l'API proprement dit par le biais d'opto-coupleurs. La tension de sortie désirée pourra être appliquée sans autre forme de procès ou arrière-pensée. Cette tension est en règle générale de 12 ou de 24 V. De par la présence de IC3, un ULN2003, la tension de sortie peut atteindre jusqu'à 50 V, sachant cependant que l'intensité de courant totale au travers des sorties ne doit pas dépasser 500 mA.
Le fait qu'il faille pouvoir commuter des charges plus importantes et travailler avec la tension du secteur explique la présence, en aval du circuit de commande, de relais encartables capables de commuter jusqu'à 16 A sous 250 V en alternatif. Ceci devrait suffire pour la quasi-totalité des applications, sachant que rien ne vous oblige à implanter les relais si vous vous satisfaites de la puissance du ULN2003.

D1 à D8,D17,D25 à D29 = 1N4148
D9 à D16,D20 à D24 = LED faible courant
D19 = 1N5407
IC1 = PIC16F84A-04/P
IC2 = 74LS151
IC3 = ULN2003
IC4 = 78L05AC
IC5 à C8 = LTV847 (Liteon), ILQ621 (Infinion) ou PC847 (Sharp)

Divers :
K1 = embase mâle à 2 rangées de 5 contacts (HE-10)
K2 à K9 = bornier à vis encartable à 3 contacts au pas de 5 mm (RM5)
K10 à K14 = bornier à vis encartable à 3 contacts au pas de 7,5 mm (RM7,5)
K15 = embase autosécable mâle SIL à 1 rangée de 9 contacts
K16 = embase autosécable mâle SIL à 1 rangée de 6 contacts
RE1 à RE5 = relais 16 A/250 V~ (Finder 40.61 bobine 12 V CC 220 Ω, Omron G2R-1-E 12 V CC ou Schrack RP310012)



Le logiciel

Tout matériel piloté par processeur ne saurait faire mieux que ce que lui permet de faire son programme, programme dont la qualité dépend indubitablement des capacités des outils de développement mis en oeuvre. L'une des principales raisons du choix du PIC16F84 est l'existence de très bons outils de développement au prix le plus abordable possible (gratuit de préférence). Le fabricant propose, comme la plupart de ces concurrents, un set de logiciels pour le développement de programmes. Le MPLAB [4] de Microchip puisque c'est de lui qu'il s'agit, comprend un assembleur, un éditeur de lien (linker) et un simulateur qui permet de vérifier sur un PC la cor-

rection logique du programme. Ce set de programmes permet de réaliser nombre de projets, nous n'en voulons pour preuve que le nombre impressionnant d'applications proposées sur le site de Microchip.
Si vous voulez programmer le PIC en langage de haut niveau il vous faudra disposer d'un compilateur adéquat. Il existe heureusement 2 bons compilateurs C pour les PIC, à savoir C2C de Pavel Baranov [5] et CC5X de Knudsen Data [6]. Ces 2 compilateurs se différencient à peine au niveau des instructions spécifiques au contrôleur mais sont à part cela de même intérêt. À partir du code-source en C, on produit un fichier-source en assembleur qui est ensuite assemblé automatiquement avant d'être converti en un fichier binaire (au format Intel-Hex). Il ne reste plus qu'un problème à résoudre :

Comment transférer le programme dans le PIC ?

La popularité des microcontrôleurs PIC tient pour une bonne part à son excellente « programmabilité ». Le processus de programmation des PIC étant sériel, l'interface requise à cet effet est relativement simple, sachant qu'il était même possible, en principe, de se contenter d'un simple câble. Même si vous êtes habitué à plus de luxe, la complexité du matériel nécessaire reste très acceptable.
Dans son numéro double de 1998 [7], Elektor a décrit un système de développement à faible coût. La petite carte de développement décrite dans cet article met à disposition une alimentation stable, un oscillateur d'horloge commutable à fréquence variable ainsi qu'un champ d'expérimentation à pastilles sur lequel pourront prendre place, par exemple, des LED servant à la visualisation de l'état des lignes de port.
Le repère [3] de la bibliographie donne les références d'une autre minuscule platine de programmation avec toute la documentation requise. On trouvera sur le site Internet les programmes (DOS) correspondants.

Réalisation et mise en oeuvre

Grâce à la platine dont on retrouve le dessin des pistes et la sérigraphie de l'implantation des composants en **figure 2**, la réalisation de Pico-API devrait être une affaire rapidement menée à bien. La platine est du type simple face sans composant CMS. Si vous n'avez pas besoin des relais, vous pourrez, à la hauteur des connecteurs K15 et K16, découper la platine qui leur est réservée et mettre l'API dans un boîtier de dimensions plus compactes. On peut également fort bien envisager de placer la platine des relais dans un boîtier distinct. Il faudra alors rétablir l'interconnexion des 2 platines par le biais des embases K15 et K16. Il faudra, en cas de mise en oeuvre des relais, que les sorties respectent les normes fixées pour les appareils de classe II.
On pourra bien entendu monter les circuits intégrés sur supports à condition que ces derniers soient de bonne qualité. Il faudra, pour éviter que les câbles ne puissent se détacher, placer des brides anti-arrachement au niveau des borniers des entrées et des sorties.
Elektor ne propose pas de programme spécifique pour la programmation de Pico-API, mais un petit programme de test. Le dit programme a été créé à l'aide des outils gratuits CC5X et MPLAB et transforme, si le montage fonctionne correctement, Pico-API, au niveau des LED D20 à D24, en une sorte de chenillard aller-retour. On a ensuite clignotement de D24 et visualisation de l'état des entrées. Si D24 est éteinte, les LED D20 à D23 représentent les 4 bits de poids faible des entrées, si D24 est allumée, ces LED correspondent aux 4 bits de poids fort.
On peut imaginer nombre de possibilités d'application du Pico-API. L'exemplaire « original » sert à piloter le système de rotation d'un mât d'antenne de réception radio-amateur. On pourrait, dans le même ordre d'idées, penser à la commande de volets roulants ou de rideaux de vitrine, voire l'automatisation de modèles techniques ou de jouets.

(010059)

Bibliographie et liens

[1] Microchip www. microchip. com

[2] Fiche de caractéristiques disponible, entre autres sous www.sharpmeg.com/products/opto/pdf/pc847x.pdf

[3] Adaptateur pour programmateur Madsen www.jdm.homepage.dk/newpic.htm

[4] Environnement de développement pour microcontrôleurs de Microchip, MPLAB : www.microchip.com/1010/pline/tools/picmicro/devenv/mplabi/plab5x/9019/index.htm

[5] Baranov, Pavel : C2C-Compiler www.geocities.com/SiliconValley/Network/3656/c2c/c.html

[6] B Knudsen Data CC5X www.bknd.com/cc5x/index.shtml

[7] Système de développement à faible coût pour PIC, Elektor n° 241, juillet/août 1998, page 64 et suivante

Programmation des µC AVR

Un jeu d'enfant grâce à BASCOM-AVR

Mark Alberts

Le présent article se targue d'avoir pour mission de montrer combien il est facile aujourd'hui de travailler avec des microprocesseurs. Pour peu que l'on opte pour un processeur AVR et que l'on se mette au BASIC, un langage de programmation qu'il n'est sans doute pas nécessaire de vous présenter, le succès est assuré.

La famille des microcontrôleurs AVR comprend plusieurs membres. Le AT90S2313 en est un au prix très abordable. Avec ses 20 broches il possède 2 Koctets de mémoire de programme en Flash. L'avantage de ce type de mémoire est qu'il est possible de le reprogrammer au moins 1 000 fois. Un autre avantage tout aussi important sinon plus est qu'il permet de se passer de programmateurs spécifiques et d'un système d'effacement aux ultra violets (UV) : il suffit d'un câble de 3 sous (ou faudrait-il dire centimes d'euro) que l'on branche au port imprimante de son PC pour pouvoir programmer ce microcontrôleur.

Le AT90S2313

Prenons le temps, en guise d'introduction, d'examiner les caractéristiques les plus importantes du AT90S2313 (figure 1):

– Il possède 2 ports : PORTB et PORTD.
– Mémoire interne : 128 octets
– Mémoire EEPROM : 128 octets
– Mémoire de programme en Flash : 2 Koctets

L'un des aspects marquants est l'utilisation de 15 des 20 broches en tant que ports d'entrées/sorties. Ce sont eux qui offrent au processeur ses capacités de mesure et de commande. Un port comporte en règle générale 8 broches (bits), mais dans le cas du AT90S2313 la ligne PD7 n'est pas accessible de l'extérieur sous la forme d'une broche physique.
Chaque broche peut être utilisée en entrée ou en sortie. Si l'une des broches est montée en sortie, elle peut se voir attribuer un niveau logique, soit un « 1 » soit un « 0 ». Si le processeur peut procéder ainsi individuellement pour chacune des broches, il est également possible de travailler par groupes de broches. Les broches PD0 à PD6 constituent ainsi ensemble le port D (PORTD). Il suffit alors d'attribuer une valeur à PORTD pour, d'un coup d'un seul, faire changer le niveau de toutes les broches concernées.
La plupart des broches possèdent une seconde fonction. Cette fonction alternative est donnée entre parenthèses. (INT0) PD2 signifie qu'il s'agit de la broche PORTD.2 et que sa seconde fonction est INT0. Nous nous limiterons, dans le cadre de cet article, aux fonctionnalités standard des broches !

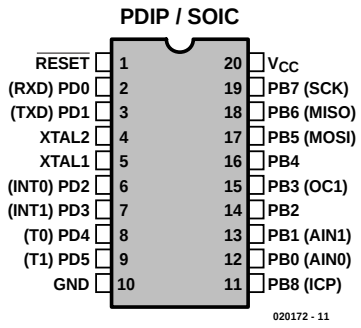


Figure 1. Brochage du AT90S2313.

Listage 1

```
CONFIG PORTB = OUTPUT ' utiliser les broches du port B en sor-
   ties
PORTB.0 = 1 ' mettre la ligne PORTB.0 au niveau haut
PORTB.0 = 0 ' remettre la ligne PORTB.0 au niveau bas
PORTB = 255 ' mettre toutes les lignes au niveau haut
PORTB = &B10101010 ' mettre les lignes 1,3,5 et 7 au niveau haut
END
```

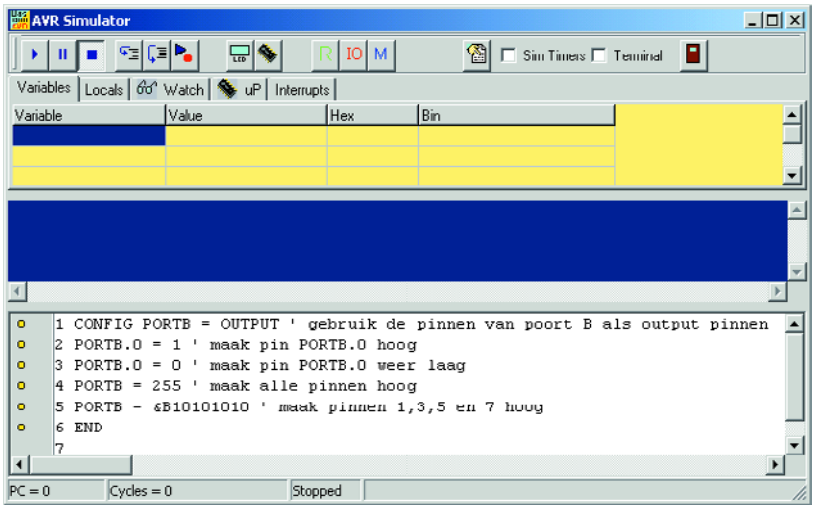


Figure 2. La fenêtre du simulateur de BASCOM-AVR.

Travailler avec BASCOM-AVR

Il nous faut, avant de pouvoir nous mettre au travail, commencer par installer le logiciel de programmation. Le programme en question s'appelle BASCOM-AVR (*BASic COMpiler*); il est présent sur le CD-ROM Elektor Freeware & Shareware 2002 mais également disponible au téléchargement sur le site de MCS Electronics. À l'adresse
www.mcselec.com/elektor.htm
vous pouvez voir quels sont les fichiers à télécharger et comment installer le logiciel. Le logiciel utilisé est une version de démonstration parfaitement fonctionnelle.
La seule limitation se situe au niveau de la taille du code processeur généré, sachant que cette dernière est limitée à 2 Koctets. Le hasard veut que c'est très précisément la taille du code que l'on peut stocker dans un AT90S2313 !
Démarrer BASCOM-AVR et choisissez dans le menu Files l'option New. Dans la fenêtre qui s'ouvre alors nous allons saisir notre premier programme (**listage 1**).
On sauvegarde le programme par File > Save As en lui donnant le nom (classique) de **sample1.bas**.
Nous venons d'écrire notre premier programme. Le microcontrôleur n'en saisira goutte vu qu'il ne comprend que ce que l'on appelle le code-objet. Pour obtenir ce code-objet il nous faut compiler le programme BASIC. Nous allons pour ce faire appuyer sur la touche « F7 ».
Si vous ne voulez pas apparaître de message d'erreur c'est que vous avez saisi votre programme sans la moindre erreur. En cas d'apparition d'un message d'erreur il vous faudra corriger le programme et le recompiler.
Avant de transférer le code-objet ainsi généré dans le microcontrôleur (programmer ce dernier) nous allons en effectuer une simulation. Ce n'est qu'une fois que nous serons satisfaits du programme testé en simulation que nous le transférerons vers le microcontrôleur.
Une fois que l'on n'a plus d'affichage de message d'erreur, on pourra appuyer sur la touche « F2 », action qui se traduira par l'ouverture de la fenêtre du simulateur.
Le programme entré au clavier est visualisé au bas de la fenêtre (**figure 2**). Dans la partie supérieure de l'écran nous trouvons les boutons de simulation. Un clic sur le bouton « LCD » ouvre la fenêtre de simulation matériel : on voit apparaître dans le haut de la fenêtre du simulateur matériel (**figure 3**) un affichage LCD. En bas à gauche on découvre les ports PORTB et PORTD. Nous reviendrons à la LED verte. Nous allons maintenant parcourir le programme pas à pas. Une action sur le bouton pas à pas (step <) se traduit par l'exécution invisible de code. Ce code sert à préparer le microcontrôleur pour qu'il puisse ensuite remplir sa fonction. Une nouvelle action sur le bouton < se traduit par l'exécution de la première ligne du programme. cette ligne 1 n'a pas pour conséquence de traitement visible. Une nouvelle action sur le bouton se traduit par l'exécution de la ligne de code numéro 2. Nous voyons alors la LED 0 du port B s'allumer.
En effet, par l'instruction « PORTB.0 = 1 » nous avons demandé la mise à « 1 » de la broche 0 de PORTB. La présence d'un « 1 » logique sur l'une des broches se traduit, au niveau du simulateur, par l'allumage de la LED correspondante. Le résultat de l'étape suivante ne devrait pas vous surprendre : la LED s'éteint.
La ligne de code numéro 4 fait passer toutes les broches à « 1 » par une écriture vers PORTB. Cette instruction permet donc de faire changer de niveau l'ensemble des broches en une seule fois.
La ligne 5 est intéressante car elle permet de constater qu'il est également possible d'utiliser la notation binaire pour indiquer la valeur de PORT. Il nous faut dans ce cas-là introduire la notation binaire par un &B suivi de la totalité des valeurs de bits. Le bit numéro 1 correspond à la broche 7, le dernier à la broche 0 (cf. la recopie d'écran de la figure 3). « PORTB=&B1001 » par exemple fait passer uniquement les broches 0 et 3 au niveau haut !
Ayant pu constater que notre programme fonctionne, nous pouvons le programmer dans le microcontrôleur. Pour cela il faut actionner la touche « F4 ». Une fois la programmation effectuée, nous allons constater que les LED impaires, 1, 3, 5 et 7 vont s'allumer. Vu la rapidité d'exécution du programme par le microcontrôleur il est impossible de voit le résultat de l'exécution des lignes 2 à 4 !
Il va sans dire que nous pouvons fort bien remplacer les LED par des relais ou toute autre électronique de sortie.
Vous vous demandez sans doute à quoi peuvent bien servir les LED vertes de la fenêtre de simulation. Le microcontrôleur AVR est également capable de déterminer le niveau logique présent à une broche lorsque le port en question est défini comme étant une entrée. Nous allons baptiser nos ports PINB et PIND. Nous supprimons l'instruction END de fin de notre listage et ajoutons les lignes suivantes à notre programme :

Listage 2

```
CONFIG PORTB = INPUT
PORTB = 255
DO
   PRINT PINB
LOOP
```

Recompilons le programme (F7) et appuyons à nouveau sur la touche « F2 » pour lancer le simulateur.
Appuyons ensuite sur la touche RUN (>). Nous verrons apparaître au centre de l'écran la valeur de PINB. Elle est de « 0 ». Nous pouvons, par un clic sur les LED vertes de PINB, modifier les niveaux de PINB. Cela se traduit par un changement immédiat de la valeur affichée.
Notre montage comporte un interrupteur connecté à la ligne PINB.0.
Nous allons entrer un nouveau programme :

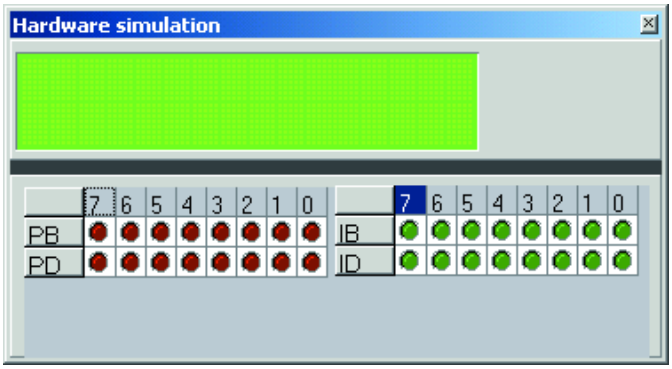


Figure 3. Le simulateur matériel (hardware) comporte une partie affichage LCD à 2 rangées de 16 caractères et en dessous une visualisation des niveaux d'entrée et de sortie des 2 ports.

Listage 3

```
CONFIG PINB.0 = INPUT  : CONFIG
  PINB.2 = OUTPUT : PORTB.0 = 1
DO
     PORTB.2 = PINB.0
LOOP
```

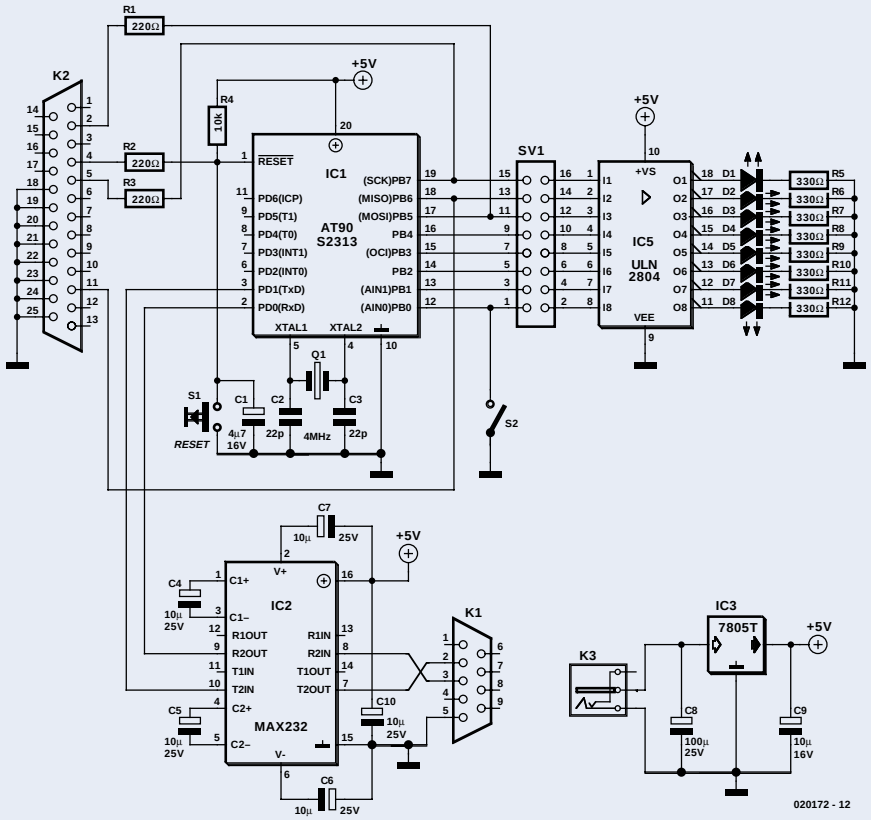
Compilons le programme et transférons-le dans le microcontrôleur par une action sur « F4 ».
Une action sur l'interrupteur S1 se traduit par l'extinction de la LED. En effet, la broche 2 de PORTB est forcée au même niveau que celui de la broche d'entrée 0 de PINB.
Le fichier d'aide de BASCOM explique dans le détail (en anglais) le fonctionnement des différentes instructions utilisées.

(020172)

L'auteur de cet article, Mark Alberts, est le père spirituel des compilateurs BASCOM pour les processeurs 8051 et AVR. On pourra trouver de plus amples informations à ce sujet sur son site sis à : www.mcselec.com.

Exemple de test

Le schéma représenté ici pour être mis à contribution pour procéder à différents essais et expérimentations.
On réalise, par le biais du connecteur imprimante (LPT), X2, un programmateur qui a été baptisé SEP (Sample Electronics Programmer).
Les résistances-série servent à la protection des lignes. Notons qu'il est recommandé de connecter cette électronique par le biais d'une carte d'entrées/sorties spécifique lui étant réservée.
Le connecteur Sub-D à 9 contacts, X1, constitue, épaulé par IC2, un MAX232, une entrée et sortie sérielle que l'on pourra connecter à l'interface sérielle (port COM) du PC.
Pour l'exemple utilisé plus haut, nous avons connecté un interrupteur à la ligne 0.SV1 est une embase autosécable mâle à 2 rangées de 8 contacts sur lesquels pourront être implantés des cavaliers. IC5, Le ULN2804, est requis par la présente application. Normalement ce circuit intégré n'est pas nécessaire si on inverse la polarité des LED et qu'on les force, par le biais d'une résistance de limitation de courant, au pôle positif de la tension d'alimentation, VCC Si l'on relie l'alimentation du ULN à +12 V, il devient possible de commander des relais. Ce schéma a pour seule fonction d'être un exemple minimaliste. Nous ne doutons pas que vous ayez, en tant qu'amateur inconditionnel des AVR, nombre de schémas simples pour ce microcontrôleur dans vos documents, schémas auxquels vous pourrez faire appel.



Ports sériels sous Windows

Et logiciels Delphi pour s'inspirer

Créer une fonction d'entrée/sortie dans un logiciel qui opère sous Windows, avouons-le, c'est encore une entreprise hasardeuse pour nombre de programmeurs. Essayons d'y voir plus clair en utilisant un port RS-232 en parfaite compatibilité avec Windows.

Dès que nous voulons mettre en service un quelconque composant dans un ordinateur, il nous faut un bout de programme pour le faire tourner. Du temps des ordinateurs domestiques travaillant avec le DOS comme système d'exploitation, écrire pareil logiciel était aisé. Un seul programme était en activité à un moment donné, donc pas d'interférence possible avec un autre qui aurait voulu s'approprier le même matériel simultanément. Celui qui voulait faire les choses à la perfection s'arrangerait en outre pour restituer le matériel, à l'issue de l'exécution de son programme, dans les conditions où il l'avait trouvé.

Tout a bien changé, avec l'apparition sur PC de Windows. D'un coup, on pouvait exécuter plusieurs tâches en même temps. Rien de bien nouveau pour les professionnels, mais à la maison, cela tenait de la magie !

Oui, mais l'utilisateur occasionnel qui venait à peine de se familiariser avec la gestion directe de son matériel se retrouvait en rade ! Qu'arrive-t-il, en effet, quand deux programmes tentent d'accéder à un périphérique au même moment ? On peut parfois s'arranger pour que l'événement soit impossible, mais avec l'apparition de nouveaux langages, on a perdu le contrôle direct sur les instructions d'E/S. Il fallait vraiment ruser pour accéder aux instructions d'E/S du processeur lui-même.

Nous allons, dans cet article, montrer que programmer selon ses désirs une interface sérielle sous Windows n'est pas aussi pénible que d'aucuns le pensent. Non seulement vous insufflez à vos logiciels personnels un look professionnel, mais en outre, ils tourneront sans faux pas, en harmonie avec les nouvelles évolutions de Windows.

Tableau I

Fonctions utiles de l'API pour la commande de l'interface sérielle

FileCreate	Ouverture de fichier
EscapeCommFunction	Commande des drapeaux DTR et RTS
GetCommStatus	Demande de l'état des entrées et du tampon de réception
ReadFile	Lecture du tampon de réception
WriteFile	Écriture dans le tampon de sortie vers le port sériel
GetCommState	Lecture des paramètres courants du port sériel
SetCommState	Réglage des paramètres du port sériel
GetCommMask	Lecture du masque d'événement. Il indique à Windows quand il faut créer un « event »
SetCommMask	Définition du masque d'événement
ClearCommError	Fournit des informations sur la dernière erreur rencontrée dans l'interface sérielle

Systèmes d'exploitation

Tirons les choses au clair : un système d'exploitation n'est rien d'autre qu'un (gros) logiciel qui sert de chef d'orchestre à tous les composants d'un ordinateur. Et parmi ces composants, on trouve, pêle-mêle, une souris, un clavier, un écran, des mémoires, un contrôleur audio, etc. Souvent, on les appelle des sources, sans trop s'inquiéter de savoir s'il s'agit d'organes d'entrée ou de sortie.

À l'égard du programme d'application, le système d'exploitation a pour mission de le dispenser de la connaissance précise de l'architecture de l'ordinateur sur lequel il opère. Le programme d'application informe simplement le système d'exploitation de ce qu'il compte faire, à charge pour ce dernier de s'occuper du reste. C'est précisément ce qui permet à un logiciel donné de fonctionner sur des plates-formes différentes sans aucune adaptation. L'unique condition pour bien exécuter les fonctions prévues, c'est la conformité du système d'exploitation sur les différents ordinateurs.

Programmes pilotes

La conduite effective du matériel, ce sont des pilotes qui s'en chargent, des programmes de gestion, les célèbres *drivers*. Le système d'exploitation doit disposer d'un pilote pour chaque composant de l'ordinateur. Il s'entendra au mieux avec n'importe quel nouveau matériel, à condition que le fabricant lui fournisse un pilote adéquat.

Les API

Ne confondons pas les pommes et les poires, une API (*Application Programming Interface*) est une interface qui définit la manière dont un logiciel d'application peut communiquer avec le système d'exploitation. Les compilateurs proposent souvent des librairies et des fichiers de liaison qui se chargent d'appeler une API. Sinon, il faudra se plonger dans la documentation du système d'exploitation. La plupart du temps, d'ailleurs, le SDK (*Software Development Kit*), que l'on peut télécharger gratuitement, fournit aussi l'API.
Comme on le voit à la **figure 1**, le logiciel d'utilisation ne communique qu'avec l'API du système d'exploitation. Nul besoin, pour le programmeur, de connaître en détails la cuisine interne du système d'exploitation, pour autant que l'API fasse exécuter ce qui lui est demandé. L'API s'assure également que le *kernel* (noyau) traite les commandes. Dès qu'un programme demande l'intervention d'un périphérique, le *kernel* fait appel au pilote correspondant, qui assurera lui-même la conduite du matériel.
La communication entre pilote et *kernel* ne transite pas par l'API utilisée, mais par la SPI (*System Programming Interface*).

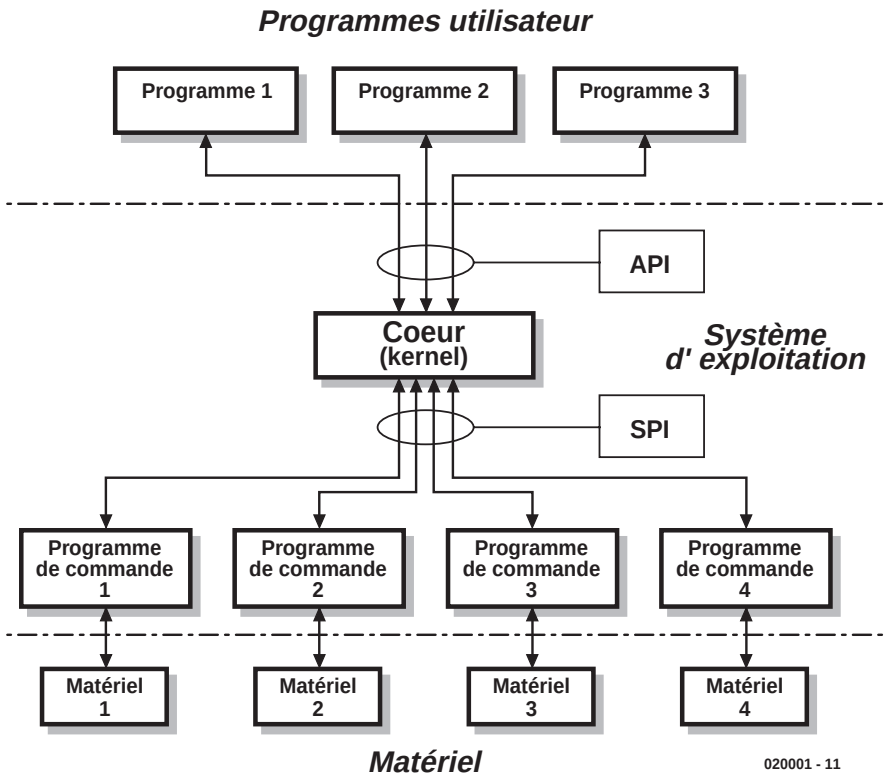


Figure 1. Structure simplifiée d'un système informatique.

Tableau 2

Les principaux champs dans un enregistrement du type DCB.

DCBLength	La longueur de DCB.
BaudRate	Débit de transmission en bauds.
Binary	Binary Liaison binaire (doit toujours être TRUE en Windows 95).
Parity	Avec ou sans parité.
fOutxCtsFlow	Positionne le drapeau CTS (prêt à émettre) pour le transfert de données. Le programme n'émet aucune donnée sur le port sériel tant que CTS est inactif.
fOutxDsrFlow	Idem que fOutxXCtsFlow, mais pour le drapeau DSR (modem prêt).
fDtrControl	Positionne le drapeau DTR (ordinateur prêt). Peut contenir les valeurs suivantes [1] : *DTR_CONTROL_DISABLE : DTR devient bas dès qu'un port est ouvert.* *DTR_CONTROL_ENABLE : DTR devient haut dès qu'un port est ouvert.* *DTR_CONTROL_HANDSHAKE : DTR utilisé pour la mise en liaison (handshake).*
fDsrSensitivity	Force le programme à ignorer les données reçues si DSR est bas.

[1] On peut commander le drapeau DTR au moyen de la fonction EscapeCommFunction, par exemple, sauf si l'on a choisi l'option DTR_CONTROL_HANDSHAKE.

DLL

Windows travaille avec des fichiers appelés DLL. Ce sont des bibliothèques accessibles à tous les programmes. La plupart des fonctions de Windows se composent essentiellement de ces DLL. Quand un programme utilise une DLL, elle s'unit au programme par un lien dynamique, d'où son nom : *Dynamically Linked Library*.
Pour aller voir quel genre de fonction Windows exporte dans une DLL, il existe, dans la plupart des versions de ce système d'exploitation, un petit programme qui n'a l'air de rien, mais peut rendre de grands services, c'est « Aperçu rapide » (quickview.exe). Il permet d'ouvrir la DLL et présente une série d'informations sur elle, en particulier les noms des fonctions qu'elle contient. De quoi se faire une bonne idée des possibilités d'une DLL.
On peut installer ce programme par : Panneau de configuration – Ajout – Accessoires – Aperçu rapide. On peut alors l'activer en cliquant sur un fichier avec le bouton droit de la souris.

Listing 1

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TForm1 = class(TForm)
    GroupBox1: TGroupBox;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    RadioButton3: TRadioButton;
    RadioButton4: TRadioButton;
    GroupBox2: TGroupBox;
    RadioButton5: TRadioButton;
    RadioButton6: TRadioButton;
    RadioButton7: TRadioButton;
    RadioButton8: TRadioButton;
    RadioButton9: TRadioButton;
    Button1: TButton;
    Button2: TButton;
    Label1: TLabel;
    Label2: TLabel;
    CheckBox1: TCheckBox;
    CheckBox2: TCheckBox;
    CheckBox3: TCheckBox;
    CheckBox4: TCheckBox;
    CheckBox5: TCheckBox;
    CheckBox6: TCheckBox;
    Button3: TButton;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    GroupBox3: TGroupBox;
    RadioButton10: TRadioButton;
    RadioButton11: TRadioButton;
    RadioButton12: TRadioButton;
    GroupBox4: TGroupBox;
    RadioButton13: TRadioButton;
    RadioButton14: TRadioButton;
    RadioButton15: TRadioButton;
    Label7: TLabel;
    Edit1: TEdit;
    Label8: TLabel;
    Button4: TButton;
    Edit2: TEdit;
    procedure FormActivate(Sender: TObject);
```

L'interface sérielle

Beaucoup de montages personnels utilisent l'interface sérielle. Ils contiennent la plupart du temps l'un ou l'autre processeur sur lequel est intégré un port sériel. Rien de plus simple, alors, que d'envoyer des commandes ou d'échanger des informations par ce canal. Et même si l'on n'a pas l'usage d'un contrôleur, quelques circuits logiques permettent de se construire une bonne interface sérielle, pour preuve le DCI-PLC paru dans le numéro de juin 2001 d'Elektor.
Des sources numériques telles que port imprimante, interfaces sérielles ou appareillages d'E/S, Windows et la plupart des autres systèmes d'exploitation les considèrent à priori comme des fichiers spéciaux. Ceci veut dire que le logiciel commence par ouvrir le « fichier », avant de pouvoir l'utiliser. Une fois ouvert, divers traitements y sont possibles. Vous trouverez dans le **tableau 1** les fonctions applicables sur un port sériel.
Il y a une décision importante à prendre avant de commencer à programmer un port sériel ; on a en effet le choix entre deux options.
Lors de l'ouverture du fichier, il faut préciser OVERLAPPED (superposé) ou NON-OVERLAPPED. Avec la première option, Windows garde la latitude de traiter le fichier à l'arrière-plan, tandis que le logiciel continue à s'occuper de son travail. Avec l'option NON-OVERLAPPED, l'exécution du logiciel est momentanément interrompue, le temps que Windows exécute l'opération sur le port sériel. Après quoi, le logiciel retourne à l'accomplissement de ses tâches.
L'avantage du mode OVERLAPPED, c'est que le programme n'est pas interrompu inutilement lors de chaque opération sur le port sériel. Mais l'inconvénient majeur, c'est qu'on n'est pas sûr que l'action a réussi. À nous de créer un « gestionnaire d'événement » que Windows appellera après l'achèvement de chaque opération sur le port sériel. Le risque est grand, malheureusement, d'introduire dans le logiciel des erreurs bien difficiles à débusquer, entre autres le blocage complet et le basculement en série. Mieux vaut réserver ce mode aux programmeurs familiers des tâches

à exécuter en parallèle.
Le principal avantage du mode NON-OVERLAPPED est de pouvoir vérifier si l'action a été couronnée de succès et s'éviter toute angoisse à ce sujet. Dans les programmes témoins en Delphi présentés dans cet article, nous n'avons pas employé l'options NON-OVERLAPPED, par souci de simplicité.

Delphi

Comme illustration, nous avons rédigé (**listage 1**) un programme simple en Delphi. Il fait appel aux principales fonctions API utiles sur un port sériel.
Quand l'utilisateur clique sur le bouton OPEN, la routine « Button1Click » s'active et elle ouvre un fichier dont le nom correspond au port COM sélectionné (COM1, COM2, etc.)
Mais ouvrir le fichier ne suffit pas. Nous devons encore régler les paramètres du port sériel. Pour commencer, nous allons demander l'état actuel de la situation à l'aide de « GetCommState ». C'est une routine de Windows qui copie dans le registre « dcbCom » les paramètres actuellement en vigueur du port sériel. Le **tableau 2** reprend la liste des constantes les plus intéressantes de ce registre. Il nous faut remplir les termes qui correspondent à nos souhaits et finalement rendre actifs ces paramètres par « SetCommState ». Le port sériel est à présent ouvert selon nos prescriptions.
Notre programme d'exemple permet de saisir un texte et de l'envoyer, d'un clic sur le bouton. La routine qui se charge du transfert du texte s'appelle « Button4Click ». Sa structure est directe et elle utilise la routine « Writefile » de Windows pour l'expédition.
Il est aussi possible de recevoir des caractères par le port sériel et de demander l'état des autres entrées à l'aide du logiciel d'expérimentation. On clique sur le bouton « Update input » pour appeler la routine « UpdateClick ». Tout d'abord, c'est la routine Windows « GetModemStatus » qui est mise en action. Elle copie l'état actuel des broches d'entrée dans une variable, « State » dans ce cas-ci. Les instructions suivantes s'occupent de répercuter à l'écran les nouvelles valeurs reçues.

```
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure RadioButtonComPort(Sender: TObject);
    procedure RadioButtonBaudRate(Sender: TObject);
    procedure CheckBox1Click(Sender: TObject);
    procedure CheckBox2Click(Sender: TObject);
    procedure UpdateClick(Sender: TObject);
    procedure ComboBox1KeyPress(Sender: TObject; var Key: Char);
    procedure RadioButtonParityClick(Sender: TObject);
    procedure RadioButtonStopBitsClick(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Edit2KeyPress(Sender: TObject; var Key: Char);
  private
    { Private declarations }
    hPort       : LongInt;      {handle for the serial port}
    dcbCom      : TDCB;         {record which holds the properties for the}
                                {opened COM-port}
    Open        : Boolean;      {Is COM-port open or not?}
    ComPort     : Integer;      {The COM-port number}
    BaudRate    : LongInt;      {desired Baudrate}
    Parity      : Byte;         {Parity}
                                {use only the constant declared in Windows}
                                {NOPARITY EVENPARITY ODDPARITY! }
    StopBits    : Byte;         {Nr of stopbits StopBits}
                                {use only the constant declared in Windows}
                                {ONESTOPBIT, ONE5STOPBITS or TWOSTOPBITS !}
    ReadBuffer  : string;

  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormActivate(Sender: TObject);
begin
  ComPort:=1;                    {Set the startup settings}
  RadioButton1.Checked:=true;
  BaudRate:=9600;
  RadioButton6.Checked:=true;
  Parity:=NOPARITY;
  RadioButton10.Checked:=true;
  StopBits:=ONESTOPBIT;
  Radiobutton13.Checked:=true;

  Open:=false;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  hPort:=CreateFile (PChar('COM'+IntToStr(ComPort)),
    GENERIC_READ or GENERIC_WRITE,0,nil,OPEN_EXISTING,
    FILE_ATTRIBUTE_NORMAL,LongInt(0));
  if (hPort = LongInt(INVALID_HANDLE_VALUE)) then
    MessageDlg ('Error opening port COM'+IntToStr(ComPort)+' : '+
    #13+#10+SysErrorMessage(GetLastError), mtError,[mbOk],0);

  if (hPort  LongInt(INVALID_HANDLE_VALUE)) then
  begin
    if GetCommState (hPort,dcbCom) then
    begin
      dcbCom.Baudrate:=BaudRate;
      dcbCom.ByteSize:=8;
      dcbCom.Parity:=Byte(Parity);
      dcbCom.Flags:=0;
      SetCommState (hPort, dcbCom);
    end;
  end;
```

```
  if (hPort  LongInt(INVALID_HANDLE_VALUE)) then
  begin
    Open:=true;
    CheckBox1Click(Self);
    CheckBox2Click(Self);
    UpdateClick (Self);
    Button1.Enabled:=false;           { Open-button disabled }
    Button2.Enabled:=true;            { Close-button enabled }
  end;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  CloseHandle (hPort);       { Close the FileHandle  }
  Button2.Enabled:=false;    { Close-button disabled }
  Button1.Enabled:=True;     { Open-button enabled   }
end;

procedure TForm1.RadioButtonComPort(Sender: TObject);
begin
  if (Open=true) then Button2Click(Self);  { Close Handle }
  with Sender as TRadioButton do
  begin
    ComPort:=(Sender as TRadioButton).Tag;
  end;
end;

procedure TForm1.RadioButtonBaudRate(Sender: TObject);
begin
  if (Open=true) then Button2Click(Self); {Close handle }
  with Sender as TRadioButton do
  begin
    BaudRate:=(Sender as TRadioButton).Tag;
  end;
end;

procedure TForm1.CheckBox1Click(Sender: TObject);
var command : integer;
begin
    if CheckBox1.Checked=true then
      command:=SETDTR
    else
      command:=CLRDTR;
  if (EscapeCommFunction (hPort,command)=false) then
      MessageDlg ('Error changing signal : '+
      #13+#10+SysErrorMessage(GetLastError), mtError,[mbOk],0);
end;

procedure TForm1.CheckBox2Click(Sender: TObject);
var command : integer;
begin
    if CheckBox2.Checked=true then
      command:=SETRTS
    else
      command:=CLRRTS;
  if (EscapeCommFunction (hPort,command)=false) then
      MessageDlg ('Error changing signal : '+
      #13+#10+SysErrorMessage(GetLastError), mtError,[mbOk],0);
end;

procedure TForm1.UpdateClick(Sender: TObject);
var State : Cardinal;
    State2 : TCOMSTAT;
    Error  : DWord;
    chRead : DWord;
    avail  : DWord;
begin
  ReadBuffer:='';
  if (GetCommModemStatus (hPort,State)=false) then
      MessageDlg ('Error retreiving ModemStatus : '+
      #13+#10+SysErrorMessage(GetLastError), mtError,[mbOk],0)
  else
```

La deuxième partie de cette routine sert à lire les caractères éventuellement obtenus. On appelle tout d'abord la routine « ClearCommError », qui fait davantage que de gommer les erreurs de transmission. Elle complète un message du type TCOMSTAT. L'information la plus précieuse, dans ce cas, c'est la quantité de caractères restants dans le tampon d'entrée. Si leur nombre n'est pas nul, le programme appelle la routine Windows « ReadFile » et les caractères reçus seront affichés à l'écran.
Le reste du programme s'oriente vers la convivialité de l'interface vers l'utilisateur, pour lui permettre de désigner le port sériel, d'en choisir un autre, etc.

HTML et E/S

Que vient faire le langage HTM dans notre programmation ? C'est que les pages HTML autorisent également l'exécution de programmes. On peut utiliser les langages Script pour animer les pages. Les deux principaux sont JavaScript et VBScript. Nous recommandons de choisir JavaScript, parce que la plupart des butineurs Internet (*browser*) y adhèrent. Au début, ses possibilités restaient assez limitées, mais après que Microsoft ait lancé les composants ActiveX, il est désormais possible de faire de grandes choses dans une page Web à l'aide de JavaScript et de VBScript, par exemple.
Un composant ActiveX très intéressant, c'est le Microsoft « Communication Control ». Pour tout dire, il commande un port sériel. Comme les composants ActiveX s'associent à Internet Explorer, il est donc parfaitement possible de diriger un port sériel au départ d'une page Web !
Pour utiliser ce composant dans une page Web, il y a une déclaration préalable à faire :

```
<OBJECT
classid=clsid:648A5600-
2C6E-101B-82B6-
000000000014 id=MSComm1>
</OBJECT>
```

Avec elle, le composant ActiveX est placé sur la page Web sous le nom MSComm1. Vous ne pouvez pas le voir sur la page parce qu'il ne s'agit

pas d'un composant graphique. Ses propriétés, on peut les fixer à l'aide d'une commande PARAM ou directement par le script.
Si elles proviennent de la commande PARAM, elles entreront en vigueur aussitôt que le butineur aura placé le composant sur la page Web. Les commandes PARAM ne sont pas obligatoires. À défaut, ce sont les valeurs initiales qui sont prises en compte. Encore faut-il savoir lesquelles sont effectivement prévues.
Quelques lignes de JavaScript nous permettront de déterminer les propriétés et de les modifier. Les programmeurs qui ont une certaine expérience en langage OOP s'en servent volontiers. L'application HTML de l'exemple 1 vous indique les deux manières de procéder.

Autres langages et environnements

Le programme en Delphi décrit dans cet article, nous aurions aussi bien pu le rédiger en C++ Builder, en Visual Basic ou d'autres langages. Celui qui décide d'écrire son propre logiciel dans l'une de ces langues de programmation peut utiliser exactement les mêmes fonctions qu'en Delphi.
Il faut d'ailleurs remarquer que Delphi, parmi d'autres, accepte directement l'API de Windows, alors qu'elle n'est pas documentée dans les rubriques d'aide. Même si votre environnement de conception (pour Windows) ne décrit pas ces routines, il y a de grandes chances pour que vous puissiez malgré tout utiliser telle quelle l'API de Windows, pour peu que vous connaissiez les noms des routines. Au besoin, on peut aussi employer le composant ActiveX qui figure dans la page HTML de l'exemple.

(020001)

Liens dignes d'intérêt :

www.codeguru.com

www.programmersheaven.com

www.microsoft.com

```
  begin
    if ( (state and MS_CTS_ON)0) then
      CheckBox3.Checked:=True
    else
      CheckBox3.Checked:=false;
    if ( (state and MS_DSR_ON)0) then
      CheckBox4.Checked:=True
    else
      CheckBox4.Checked:=false;
    if ( (state and MS_RLSD_ON)0) then
      CheckBox5.Checked:=True
    else
      CheckBox5.Checked:=false;
    if ( (state and MS_RING_ON)0) then
      CheckBox6.Checked:=True
    else
      CheckBox6.Checked:=false;
  end;
  ClearCommError (hPort,Error,@State2);
  if (Error 0) then
      MessageDlg ('Error retreiving CommError : '+
      #13+#10+SysErrorMessage(GetLastError), mtError,[mbOk],0);
  Avail:=State2.cbInQue;
  chRead:=0;
  if (avail>20) then avail:=20;
  begin
    setLength (ReadBuffer,avail+1);
    ReadFile (hPort,PChar (ReadBuffer)^,avail,chRead,nil);
    Edit2.Text:=ReadBuffer;
  end;
end;

procedure TForm1.ComboBox1KeyPress(Sender: TObject; var Key: Char);
begin
  Key:=Chr(0);
end;

procedure TForm1.RadioButtonParityClick(Sender: TObject);
begin
  if (Open=true) then Button2Click(Self); {Close handle }
  case (Sender as TRadiobutton).Tag of
  1 : Parity:=NOPARITY;
  2 : Parity:=EVENPARITY;
  3 : Parity:=ODDPARITY;
  else
  end;
end;

procedure TForm1.RadioButtonStopBitsClick(Sender: TObject);
begin
  if (Open=true) then Button2Click(Self); {Close handle }
  case (Sender as TRadioButton).Tag of
  1 : StopBits := ONESTOPBIT;
  2 : StopBits := ONE5STOPBITS;
  3 : StopBits := TWOSTOPBITS;
  end;
end;

procedure TForm1.Button4Click(Sender: TObject);
var Written : DWord;
begin
  Writefile(hPort, PChar (Edit1.Text)^, Length (Edit1.Text), Written, nil);
  if (WrittenLength(Edit1.Text)) then
      MessageDlg ('Error writing : '+#13+#10+SysErrorMessage(GetLastError),
      mtError,[mbOk],0);
end;

procedure TForm1.Edit2KeyPress(Sender: TObject; var Key: Char);
begin
  Key:=#0;
end;

end.
```

Pistes & Co.

Tuyaux pour réalisateurs de cartes

Michael Möge

Les programmes actuels de conception de circuits imprimés – dont certains sont même gratuits – permettent de créer des cartes au tracé merveilleux pour le bricolage, les petites séries ou les prototypes. Mais une fois le travail sur PC terminé vient le moment moins agréable de la gravure.

Le commerce spécialisé en électronique offre un grand choix de kits de gravure et d'accessoires qui ne sont malheureusement pas tous bon marché et parfois d'un emploi quelque peu compliqué. Ça commence par le typon : même la photocopie ou l'impression laser sur des transparents coûteux pour rétroprojecteur n'est pas assez opaque, il faut placer au moins 2 feuilles identiques l'une sur l'autre pour effectuer l'insolation. Une copie sur pellicule rigide exige un petit laboratoire photographique. La gravure par perchlorure de fer ou persulfate d'ammonium est une leçon de patience à base de plaques chauffantes, saleté et quelques ratages. N'oublions pas que les déchets devront être éliminés. La personne qui veut obtenir une bonne gravure et ne pas risquer d'empoisonnement devrait savoir ce qu'elle fait. Il est déconseillé de manipuler les produits chimiques au petit bonheur.

Le typon

Une imprimante à jet d'encre permet d'obtenir une impression opaque et nette sur des transparents. Nous avons fait de bonnes expériences avec les transparents de Conrad (transparents OH 3 pour imprimante à jet d'encre). Ils possèdent une texture très fine et sont entraînés de façon très précise par l'imprimante grâce à la feuille de papier contrecollée au dos. Placez le transparent avec sa feuille au dos dans l'alimentation en papier (par exemple d'un Epson Stylus Color 660 ou d'un Canon S450). Veillez avec un soin tout particulier à ce que le côté collé du transparent soit exactement positionné à la butée du mécanisme d'entraînement ET que l'ensemble repose sur la butée fixe longitudinale. Passez le transparent au sèche-cheveux après l'impression, puis imprimez une seconde fois. Il suffit normalement de 2 passages d'impression. Il est crucial de bien sécher entre les passages. Il faut donc faire preuve de patience. Si le transparent a été placé avec soin, même l'impression de pistes larges de 0,2 mm sera parfaitement superposée. L'impression est devenue d'un beau noir et suffisamment opaque. Un dernier séchage de 24 heures avant l'insolation. La texture des transparents des autres fabricants est trop grossière, ou des déformations empêchent le transparent d'être correctement entraîné ou alors il n'y a pas de feuille au dos.

Insolation et révélation

Pour l'insolation, il suffit d'utiliser une vieille lampe bronzante (lampe 1 000 W UV à vapeur de mercure). À une distance d'environ 50 cm et en recouvrant de verre la plaque à insoler, l'auteur a obtenu de bons résultats au bout d'une petite minute d'insolation. Le temps d'insolation optimal dépend de la source lumineuse, du typon, et de la qualité du matériau de base. On devrait donc tout d'abord graver une plaque d'essai : prenons-en une vierge et insolons-la « par tranche ». On l'a préalablement recouverte d'un carton que l'on retire par morceau toutes les 15 secondes. Le processus de gravure permettra de déterminer le temps d'insolation le plus favorable. Portez des lunettes de protection contre les UV (ou ne fixez pas le champ lumineux) !

Il est facile de préparer soi-même le bain de révélation. On peut se procurer de l'hydroxyde de sodium (soude caustique NaOH) en droguerie pour quelques euros par kilo. Pour préparer le bain de révélation, dissoudre 2 g de NaOH dans 0,4 l d'eau tiède. La révélation peut être effectuée à la température ambiante. La laque photosensible colore la solution en bleu-vert.

Réglage de l'imprimante

Papier	Photo Quality Glossy Film (Pas de transparent Ink-Jet !)
Couleur	noir
Réglages définis par l'utilisateur	finesse 720 dpi, pas de demi-tons, luminosité –25 %, contraste +25 %

Figure 1. Finesse de la structure superficielle, indéformable et feuille de papier collée au dos : le transparent pour rétroprojecteur idéal.

La gravure

Pour obtenir les meilleurs résultats, on peut toujours faire confiance au bon vieux procédé basé sur le chlorure de cuivre avec acide chlorhydrique et eau oxygénée.

Toutefois, et bien qu'il soit utilisé pour la production en série, il pose des problèmes de sécurité dans le

Figure 2. Insolation grâce à une bonne vieille lampe pour bronzage.

Conseils professionnels lors du bavardage en ligne sur les platines

C'est à un expert, l'un des 2 directeurs de l'entreprise Bungard Elektronik spécialisée dans la fabrication de cartes que nous devons l'existence du forum de discussion *www.batronix.com* (en allemand malheureusement, mais on y trouve également quantité d'informations intéressantes en anglais dans les domaines de la gravure de cartes de circuits imprimés ainsi que de nombreuses propositions de réalisations et de dessins de platines). On y trouve de nombreux conseils et tuyaux sur la fabrication de carte par les mordus de l'électronique. Il vaut la peine de visiter ce forum. On trouve d'autres contributions à l'adresse *http://news.cadsoft.de/forum.htm* (en anglais), où l'on peut poser ses questions.

D. Bungard
Salut, je ne puis m'empêcher d'exprimer un certain embarras maintenant que j'ai pigé quels sont vos problèmes de cartes. La plus grande partie de ce que vous écrivez est super, mais il ne faudrait peut-être pas oublier quelques petites choses. Au sujet des cartes :

1. Insolation :
Les UV-C produits par un effaceur d'EPROM ne conviennent pas !!!. Les lampes à bronzer (à tubes) – no problem. Support noir, (double) pochette de film maintenue par 2 bandes de matériau de carte, plaque de verre par-dessus (l'idéal : verre de cristal d'un vitrier qui laisse mieux passer les UV), lampe à bronzer, et en avant pour environ 2 min.
Nous insolons nos plaques à la lumière UV-A superactinique, c'est-à-dire dont la longueur d'onde ≥ m. UV-B n'est JAMAIS inclus dans les insoleuses des professionnels. La couleur de la lumière est indiquée sur les tubes fluorescents. TL 20 W 05 par exemple est OK. 6 de ces tubes par côté = 120 W donnent un temps minimum d'insolation garanti de 2 min. pour une plaque (= notre appareil Hellas, interdistances 10 cm, voir ci-dessous).
Les tubes type XX yy W 08 ou 09 conviennent tout aussi bien. Les 2 derniers chiffres de la désignation (05, 08, 09) désignent la longueur d'onde. 08 et 09 sont les types utilisés dans les lampes à bronzer le visage (Philips, minuterie mécanique comprise).
Ah oui, j'ai failli oublier : Pour que l'éclairage de la carte soit uniforme, il faut que la distance entre les tubes et entre ceux-ci et la plaque soit à peu près la même, ou alors faire appel à des réflecteurs ! Les lampes « Nitraphot » sont OK, mais elles ont besoin d'un temps de mise en marche – il leur faut environ 15 minutes pour atteindre le rendement lumineux optimum. Le temps d'insolation de nos plaques est méchamment long, 7 min. ou plus, même à 50 cm. Et plus le temps d'insolation de base est long, plus grandes sont aussi les déviations qui peuvent se produire par rapport au typon : 10 % sur 7 min sont en fait, vus de façon absolue ,différents de la « norme » de 10 % sur 2 min ! Les projecteurs de studio convain-

cas d'applications « domestiques ». Il faut toujours porter des lunettes adéquates de protection contre les acides lors de travaux faisant appel à des acides ou des solutions alcalines.
Ne manquez pas de lire l'opinion du « gourou des cartes », Dietmar Bungard, dans l'encadré.

Pour un bain de gravure destiné à environ 5 cartes au format européen, utiliser une solution composée de 340 ml d'acide chlorhydrique à 6 % à laquelle on ajoute 160 ml d'eau oxygénée à 15 % (l'eau oxygénée est une solution aqueuse de peroxyde d'hydrogène). La gravure est achevée au bout de 10 à 15 minutes à température ambiante. L'acide chlorhydrique peut être obtenu dans les pharmacies, les drogueries ou en fûts de 10 l dans les grandes surfaces de matériaux de construction

quent par leur seule puissance, mais la chaleur engendrée dans le typon et la plaque peut être une source de problèmes.
Avantage des sources lumineuses ponctuelles : il est possible de réduire la largeur des pistes et la distance entre elles. La meilleure source ponctuelle, mais qui dépend des caprices du temps : le soleil. Cinq minutes en mai, plaque et typon sous un verre pesant assurant un bon contact, et c'est parti (et pratiquement gratuit). Mais gare à la pluie… ou la plaque sera finie en novembre (ne perdons pas de vue l'aspect financier…)
Une insolation trop longue est préférable à une insolation trop courte, tout au moins si la plaque est de bonne qualité. Une insolation trop longue ne pose pas de problèmes avec un bon typon. Encore un bon truc, il pourrait être de moi, insolation progressive en retirant la feuille de protection par bande et en insolant chaque bande n pendant X secondes. La bande n, qui est complètement révélée après 60 s, idéalement 40 s, plus 1 pas pour le film, donne un temps d'insolation minimum n*X (noter le temps, qui reste relativement constant).

2. Développement (révélateur) :
Le révélateur ne peut que difficilement sauver ce qui a été massacré lors de l'insolation. Les plaques Bungard aiment les révélateurs puissants, de 13 à 30 g de NaOH par litre, mais s'il vous plaît, uniquement à température ambiante (sécurité : éclaboussures de solution brûlante = on en prend plein la vue !!!).
Conserver le révélateur non utilisé dans un vieux bidon de plastique (quantité plus importante : même concentration, bien fermer le bidon, sinon le révélateur perdra son efficacité par absorption du gaz carbonique de l'atmosphère). Ne prélever que la quantité nécessaire, éliminer après emploi par dilution dans l'eau. Une solution à 1 % de NaOH est comme ce qui sort du lave-vaisselle !
Une insolation trop courte laisse sur le cuivre une couche de laque photosensible qui n'a pas été exposée assez longtemps. Cette couche rappelle parfois la structure du tissu de verre textile ; lors de la révélation, cet endroit prend une teinte de rouge-brun à violet, ce qui est logique : cette couleur signale qu'à cet endroit, le cuivre est hors limites pour l'agent de gravure. Test : après avoir révélé la plaque et l'avoir rincée à l'eau claire, la plonger brièvement dans l'agent de gravure. Le cuivre des parties révélées de la plaque doit se colorer immédiatement. Sinon, insolation insuffisante !
Derniers secours : rincer la plaque à l'eau du robinet, la sécher prudemment mais à fond (à cause de l'insoleuse), si possible à l'air (comprimé), puis insoler toute la surface (pratiquement impossible de repositionner le film) pendant 20 % du temps initial. Révéler de nouveau, répéter le test. Ce processus paraît compliqué, mais qui renoncerait à récupérer une plaque coûteuse ?
L'effet : la quantité de laque photosensible des pistes n'est plus que de 80 %, ce qui est suffisant pour la gravure, tandis que la couche isolante sur le reste de la plaque a disparu. Opération sauvetage platine réussie !
Ne manquons pas de signaler que nos cartes peuvent toujours être insolées plusieurs fois ! On peut même abuser de cette possibilité pour réaliser une sorte d'arrêt de la soudure ou l'impression des composants au dos. Par ailleurs, pour celles et ceux qui ont entendu parler de gradation du contraste : en mettant les choses au mieux, on peut insoler et révéler nos platines à l'aide de photocopies sur papier de machine à écrire : insolation (relativement) courte, révélateur 2 fois plus fort, autres trucs en réserve. Tout cela encore une fois sur le « pouvoir couvrant du typon ».

3. Gravure :
Ce qui ne joue pas lors de la gravure a sa source dans l'insolation ou, rarement, dans la révélation. Mais PLEASE, ouvrez bien grandes vos oreilles pour ne pas manquer ce que je vais vous dire : les lois sur l'environnement qui conseillent d'éviter les déchets spéciaux rendent le persulfate de sodium ou d'ammonium INDÉSIRABLES. La gravure est très mauvaise, les produits se décomposent au moindre regard, ne parlons pas de l'utilisation, et la largeur des pistes qu'il est possible d'atteindre est à hurler !
Les coûts d'élimination sont 10 fois plus élevés que ceux du perchlorure de fer, $FeCl_3$. Pourquoi donc ces persulfates ? Parce que, contrairement au $FeCl_3$, la diffusion d'air ne les fait pas mousser et il est donc possible d'utiliser un aquarium ou tout autre bac similaire.
Au moins 90 s de temps de gravure pour du cuivre de 35 mm avec une solution à 45°C à peu près fraîche dans une graveuse À PULVÉRISATION. Temps de gravure maximum : 180 s, puis il faut remplacer le perchlorure de fer qui peut contenir jusqu'à environ 5 fois plus de cuivre que le persulfate de sodium. De toute façon 80 % au plus de déchets spéciaux en moins, exclusivement dus au cuivre de la gravure (mais pas à l'agent de gravure) !! Encore une fois, plus systématiquement : le perchlorure de fer dissout de 5 à 10 fois plus de cuivre par litre que le persulfate de sodium. Dans le cas du processus à pulvérisation, il permet de graver des contours 10 fois plus rapidement avec une précision supérieure à 0.1 mm. Et il ne fait pas de trous dans les fringues. Le détachant RX3 aide à ôter les taches de « rouille » brunes.
Il existe un agent de gravure encore mieux adapté, mais, pour des raisons de sécurité, il est réservé à des professionnels : du chlorure de cuivre mélangé à beaucoup d'eau, un peu d'acide chlorhydrique et encore moins d'eau oxygénée. Pas pour les utilisateurs en chambre !!!
Ne jamais entreposer de sachet ouvert ! Le NaOH attire l'humidité de l'air comme un aimant, formant une pulpe super agressive qui peut causer la cécité, donc : acheter du révélateur spécial pour 1 l d'eau, dans laquelle on le dissoudra, garder le liquide dans un récipient fermé, ou alors acheter des boîtes de 250 g pour y puiser selon ses besoin avec une cuillère à mesurer et les mettre SOIGNEUSEMENT sous clé.
EN CLAIR : À l'état pur, en granules ou sous forme de perles, le perchlorure de fer est moins dangereux pour votre état de santé , si l'on en croit les données de sécurité, que le NaOH. Il ne faut pas mélanger les torchons avec les serviettes, mais je ne peux m'empêcher d'ajouter un commentaire : en solution aqueuse préparée selon les règles, le révélateur est plutôt inoffensif, voyez les lave-vaisselles. Qui s'est jamais donné la peine de lire attentivement les avertissements sur les comprimés pour lave-vaisselle ? Ou qui utilise le « bon » révélateur SENO 4007 ? Et qui savait que le FeCl3 est utilisé copieusement dans la préparation de l'eau potable ? Il est vrai que dans ce cas il ne contient pas de cuivre !!!
Mais je ne voudrais pas trop prêcher pour ma paroisse ; je répondrai volontiers aux intéressés dans la mesure du temps disponible. En espérant vous avoir appris quelque chose.

Cordialement
Dietmar Bungard
Assistance technique
Bungard Elektronik
Internet : www.bungard.de

Pour les questions spéciale, un courriel à support@bungard.de. Réponse selon disponibilité professionnelle.

Abrégé de chimie

Réactions principales :

H_2O_2
$H_2O + O$ (accélérée par la présence de Cu)

$Cu + O$
CuO

$CuO + 2\ HCl$
$CuCl_2 + H_2O$

Diverses réactions accessoires :

$2\ H_2O_2$
$2\ H_2O + O_2$
$2\ HCl + H_2O_2$
$2\ H_2O + Cl_2$
$H_2O_2 + Cl_2$
$2\ HCl + O_2$

(les maçons s'en servent pour éliminer les restes de chaux sur les murs).

L'eau oxygénée est disponible en pharmacie, en droguerie, ou chez un coiffeur. Entre le bain de révélation et le bain de gravure, bien rincer la plaque et la pincette de plastique à l'eau courante.
Lorsque la plaque révélée est placée dans la solution de gravure, les parties à absorber prennent immédiatement une teinte rouge distincte. De petites bulles se forment sur les surfaces recouvertes par la laque. La plaque devrait être légèrement bougée pendant toute la durée de la gravure. Il faut éviter que de l'air pénètre dans la solution, ce qui causerait une décomposition prématurée de l'eau oxygénée. Il n'est pas nécessaire ni même utile de chauffer le bain. La solution de gravure se colore légèrement en bleu-vert (chlorure de cuivre) mais ne forme pas de boues. Si l'efficacité de la gravure diminue, réactivez la solution en ajoutant un peu d'eau oxygénée. Si les surfaces de cuivre précédemment rouges deviennent distinctement blanchâtres, il est possible de régénérer temporairement le bain avec un peu d'acide chlorhydrique.
Une fois la gravure terminée, bien nettoyer encore une fois la plaque à l'eau courante. Ne pas verser simplement l'eau de rinçage à l'égout, mais la recueillir et l'éliminer dans les règles. Essuyer les restes de laque photosensible au moyen d'acétate d'éthyle, de diluant universel (rapide et pas cher), d'alcool (pas très rapide), d'acétone (très performant) ou bien plus simplement avec un torchon à nettoyer les casseroles.
La gravure produit du chlorure de cuivre et de l'eau. Cela provoque le dégagement d'oxygène et d'un peu de chlore. L'eau oxygénée est la principale substance décomposée. Il est possible de régénérer avec succès de « vieilles » solutions de gravure en ajoutant 20 % d'eau oxygénée (H_2O_2) ou 20 % d'acide chlorhydrique (HCl). Le succès est moins certain avec des produits dilués. Mais les produits chimiques concentrés ne devraient être utilisés que par des spécialistes.
Un bain encore utilisable peut être conservé dans une bouteille qui ne doit pas être hermétique (l'eau oxygénée se décompose en eau et en oxygène gazeux). Un bain doit être réactivé à l'eau oxygénée avant d'être réutilisé.

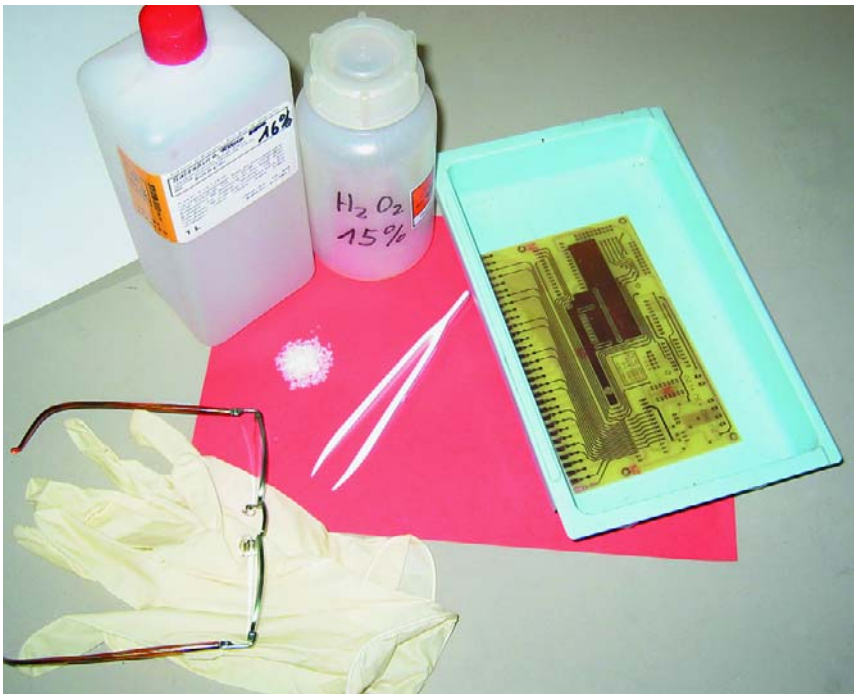


Figure 3. Accessoires de gravure.

Élimination

Il faut laisser un bain de gravure utilisé dégazer pendant quelques jours. L'eau oxygénée se décompose en eau et oxygène. Il ne reste plus alors que du chlorure de cuivre, de l'acide chlorhydrique et de l'eau. Cette solution ne forme pas de boues et ne flocule pas ; elle peut donc être mise sans problème dans un bidon de plastique. Un mélange de chlorure de cuivre et d'acide chlorhydrique peut être remis en tant que pur produit chimique à l'emplacement de réception des déchets spéciaux. Il est déconseillé, à cause du danger, de neutraliser à la soude caustique. On obtient de l'eau salée, des boues de cuivre et divers gaz. Vu la violence de la réaction, mieux vaut laisser ce genre d'élimination à des chimistes qualifiés.
La concentration des produits chimiques mentionnés n'est pas très dangereuse, mais même des acides dilués peuvent causer des dégâts. Il va sans dire que personne ne boira ces solutions. Mais il est nécessaire de porter des lunettes de protection et des gants. On peut facilement laver les éclaboussures sur la peau et les habits, mais pas dans les yeux. De faibles quantités d'oxygène et de chlore se dégagent au cours du travail. Il devrait aller sans dire qu'il ne faut ni manger, ni boire et ni fumer pendant ce genre de travail. Assurer une bonne aération ; ouvrir la fenêtre est un bon début.
Le matériau pour cartes de l'entreprise Bungard donne d'excellents résultats. Ces cartes sont un peu plus coûteuses que les produits sans marque, mais elles valent leur prix. Elles sont indispensables lorsque la structure est fine, d'autres exécutions conviennent aussi lorsqu'elle est plus grossière.

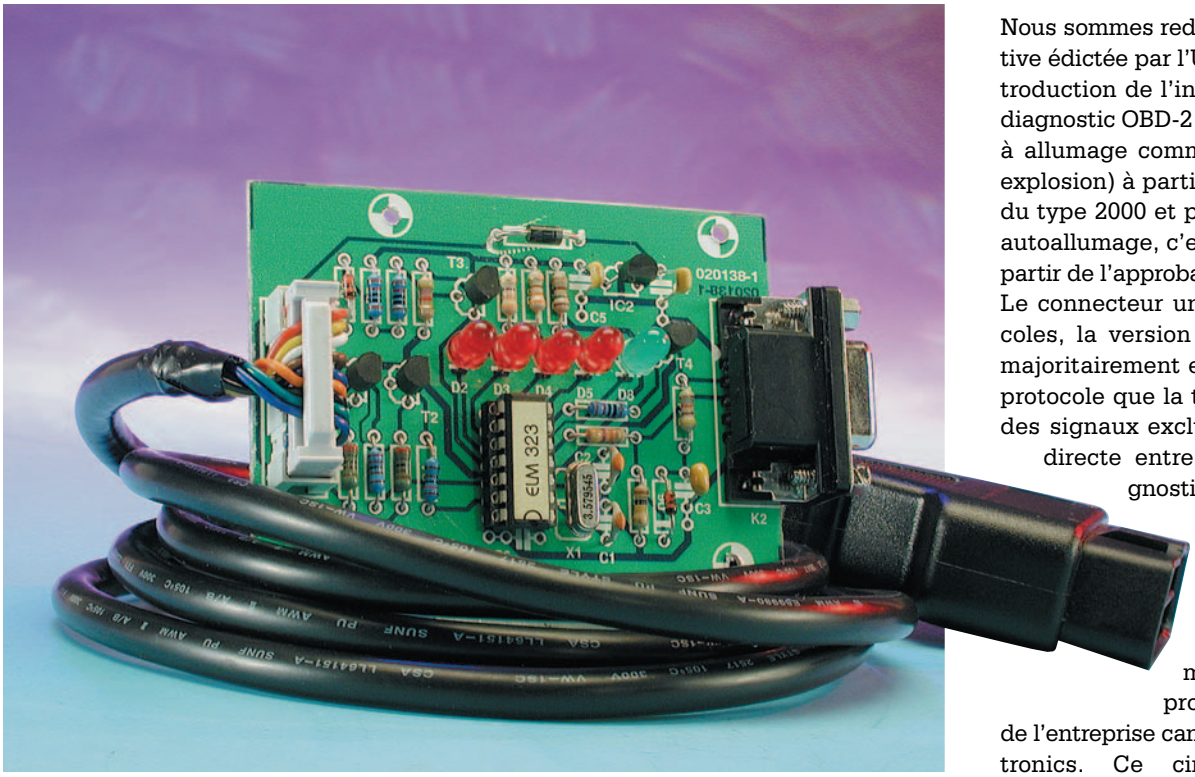
(020099)

Adaptateur pour diagnostic de véhicule

Interface entre la prise OBD-2 pour diagnostic de véhicule et le port sériel d'un PC

Projet : Gerhard Müller

Comme déjà mentionné dans l'article d'introduction du numéro précédent sur les systèmes de diagnostic pour véhicules de la deuxième génération (OBD-2), les véhicules équipés d'un moteur à explosion mis sur le marché depuis janvier 2001 doivent disposer d'une interface de diagnostic unifiée. Cette interface n'est pas directement compatible avec celle d'un PC. Tant le matériel (niveau des signaux) que le logiciel (protocole) doivent être adaptés, ce dont se charge l'interface à microcontrôleur présentée ici.



Nous sommes redevables à la directive édictée par l'UE en 1998 de l'introduction de l'interface unifiée de diagnostic OBD-2 pour les véhicules à allumage commandé (moteurs à explosion) à partir de l'approbation du type 2000 et pour les moteurs à autoallumage, c'est-à-dire Diesel, à partir de l'approbation du type 2003. Le connecteur unifié offre 3 protocoles, la version ISO étant utilisé majoritairement en Europe. Tant le protocole que la tension du niveau des signaux excluent toute liaison directe entre la prise de diagnostic du véhicule et le port sériel d'un ordinateur.

L'interface présentée ici est basée sur le microcontrôleur préprogrammé ELM323 de l'entreprise canadienne Elm Electronics. Ce circuit intégré à

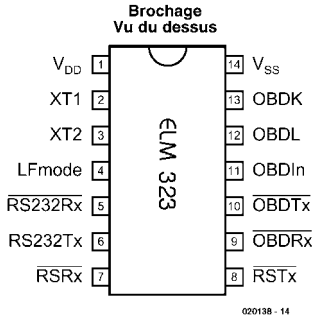


Figure 1. Brochage du ELM232.

14 broches ne nécessite que peu de composants externes pour interpréter les signaux OBD et les convertir en caractères ASCII. Cela permet d'utiliser pour afficher et enregistrer les données n'importe quel PC, ordinateur portatif ou assistant numérique personnel (ANP) muni d'une interface sérielle et comportant un programme d'émulation de terminal. On peut aussi bien sûr envisager un programme de diagnostic plus convivial. Nous examinerons dans le prochain numéro les conditions nécessaires pour réaliser soi-même un tel programme et nous présenterons aussi un programme déjà disponible permettant d'effectuer les fonctions de base de lecture et d'interprétation des codes d'erreur, d'effacer ceux-ci et de saisir en temps réel diverses données des capteurs. Ce programme sera disponible sous forme de code source C, ce qui permet de le faire fonctionner facilement sous divers systèmes d'exploitation (Linux, BeOS, QNX) car le compilateur gcc utilisé est disponible gratuitement. Outre le code source, il sera possible de télécharger une version compilée pour PC Windows.

Le convertisseur

Le microcontrôleur ELM323 est spécialement conçu pour offrir aux amateurs un accès peu coûteux au système de diagnostic OBD2. Par souci de simplicité, les fonctions comme le protocole de transfert RS-232 et le changement du nombre de bauds ne sont pas implémentées. L'ELM323 est conçu pour le protocole ISO-9141 10,4 kHz utilisé principalement par les constructeurs des pays européens et asiatiques.

Caractéristiques techniques du ELM323

Valeur limites absolues :

Température de stockage :	–65 à +150 °C
Température de service sous tension :	–40 à +85 °C
Tension sur V_{DD} référée à V_{SS} :	0 à +7,0 V
Tension sur n'importe quelle autre broche (référée à V_{SS}) :	–0,6 à (V_{DD} + 0,6 V)

Caractéristiques électriques

Sauf indication contraire, toutes les valeurs mentionnées sont relevées à une température de service de 25 °C et une tension d'alimentation de 5 V. Pour de plus amples informations cf. la Note 1) ci-après.

Caractéristique	Minimum	Typique	Maximum	Unité	Remarque
Tension d'alimentation, V_{DD}	4,5	5,0	5,5	V	
Taux de montée de V_{DD}	0,05			V/ms	Cf. remarque 2
Consommation moyenne, I_{DD}		1,0	2,4	mA	Cf. remarque 3
Entrée – niveau bas	V_{SS}		0,15 V_{DD}	V	
Entrée – niveau haut	0,85 V_{DD}		V_{DD}	V	
Sortie – niveau bas			0,6	V	Courant (drain) = 8,7 mA
Sortie – niveau haut	V_{DD} - 0,7			V	Courant (source) = 5,4 mA
Courant d'entrée, RS-232 broche Rx	0,5		0,5	mA	Cf. remarque 4
Taux de transmission RS-232		9600		Baud	Cf. remarque 5

Remarques :

1) Ce circuit intégré est développé sous la forme d'un microcontrôleur à noyau enfoui du type PIC16C505 de Microchip Technology Inc.
2) La valeur spécifiée doit être respectée pour un fonctionnement correct de la réinitialisation (reset) à la mise sous tension. Une tension d'alimentation à la croissance trop faible peut se traduire par des problèmes de réinitialisation.
3) Circuit intégré seul sans courant de charge.
4) Les valeurs représentent le flux de courant au travers des diodes de protection lors de l'application de tensions élevées à l'entrée RS-232 Rx (broche 5) au travers d'une résistance de limitation de courant. Les courants indiqués sont les valeurs maximales continues.
5) Taux de transfert nominaux en cas d'utilisation du quartz de 3,58 MHz dans l'oscillateur à quartz. Le transfert vers et en provenance du ELM232 se font sous le paramétrage 8 bits de données, sans parité et 1 bit d'arrêt (8N1).

La sortie sérielle fonctionne à 9 600 bauds. Elm Electronics offre aussi des circuits intégrés pour les protocole VPW et PWM qu'emploient de préférence les fabricants américains.
Le texte encadré « Caractéristiques techniques » contient les spécifications principales du ELM323. Le brochage du circuit intégré est indiqué dans la **figure 1** et la structure interne reproduite dans le schéma fonctionnel de la **figure 2**. Les fonctions suivantes sont associées aux broches :

V_{DD} (broche 1)
Côté positif de l'alimentation : point où la tension du circuit doit être la plus élevée (valeurs indiquées dans les caractéristiques techniques). Un circuit interne assure la réinitialisation du microcontrôleur lors de sa mise sous tension (*Power-up reset*).

XT1 (broche 2) et **XT2** (broche 3)
Un quartz de 3,579 545 MHz est raccordé à ces broches (fréquence de synchronisation de chrominance NTSC). Normalement, un condensateur (valeur type 27 pF) est raccordé entre chaque broche et la masse du circuit (V_{SS}).

LFmode (broche 4)
Cette entrée sert à sélectionner le changement de ligne standard (LF = *Line Feed*) lors de la mise sous tension ou d'une réinitialisation du système. Si l'entrée se trouve au niveau haut, les lignes de caractères envoyées par l'ELM323 se terminent par un retour chariot (CR = *Carriage Return*) et un changement de ligne. Les lignes ne sont terminées que par un CR lorsque l'entrée se trouve au niveau bas. Le logiciel peut aussi modifier ce comportement par l'envoi des commandes AT appropriés (ATL0 ou ATL1).

$\overline{\text{RS-232Rx}}$ (broche 5)
Le signal RS-232 émis par un ordinateur peut être directement raccordé à cette broche pour autant qu'une résistance limitatrice de courant (valeur type > 47 kΩ) soit placée en série. Du côté interne, l'entrée est raccordée à une diode de protection et à un trigger inverseur de Schmitt pour la conformation des signaux.

RS-232Tx (broche 6)
Broche d'émission ou de sortie des données RS-232. Le niveau du signal est compatible avec la majorité des circuits intégrés d'interfaçage et peut fournir un courant suffisant pour n'utiliser qu'un transistor PNP comme interface si désiré.

Sorties du circuit d'attaque des LED
(broches 7, 8, 9 et 10)
Ces 4 broches sont à l'état bas lorsque l'ELM323 envoie ou reçoit des données RS-232 ou OBD, sinon à l'état haut. Ces sorties peuvent directement attaquer des LED à travers des résistances de protection.

OBDIn (broche 11)
Les données OBD sont disponibles sur cette broche à logique positive en fonction de l'état actif de la ligne K d'OBD. En l'absence de trigger de Schmitt, il faut que le signal OBD passe par un circuit tampon pour minimiser les périodes de transition des circuits CMOS internes.

OBDL (broche 12) et **OBDK** (broche 13)
Il s'agit des signaux de sortie actifs à l'état haut destinés à attaquer le bus OBD par des transistors NPN externes. La transmission

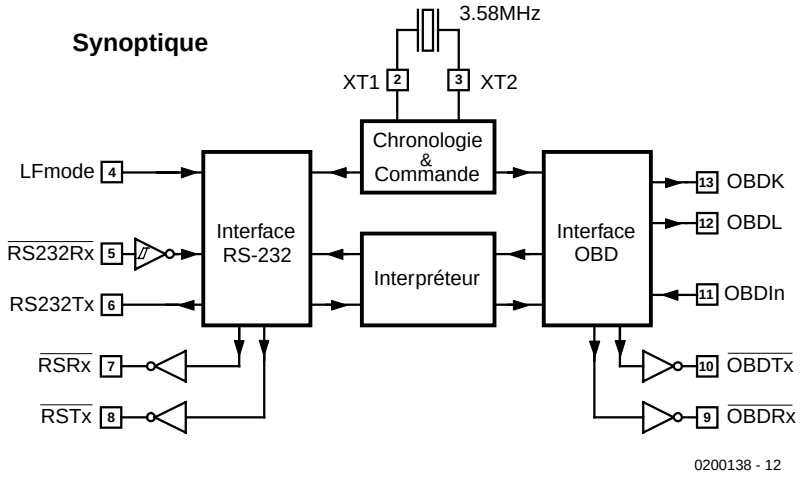


Figure 2. Blocs fonctionnels du convertisseur OBD-2/RS-232.

des données est normalement effectuées par la ligne K, mais la norme exige que la ligne L soit aussi implémentée pour initialiser correctement le bus. La suite dans la section suivante.

V_{SS} (broche 14)
Connexion de masse (le point le plus négatif du circuit).

Circuit d'interfaçage

La norme SAE (*Society of Automotive Engineers*) J1962 spécifie qu'une prise standard de diagnostic doit être située à proximité du siège du conducteur de chaque véhicule compatible OBD. La forme et le brochage de la prise à 16 contacts ont déjà été décrits dans l'article d'introduction à OBD du fascicule précédent. Le circuit décrit ici est simplement connecté à cette prise ; il ne faut donc faire subir aucune modification au véhicule.
Mais la fiche J1962 (**figure 3**) correspondant à la prise du véhicule n'est pas si facilement disponible. La liste de pièces indique toutefois où se procurer un kit permettant d'assembler une fiche de ce type. Lors des tentatives de réalisation qui suivront, il faut bien entendu éviter à tout prix d'endommager l'interface OBD du véhicule. Les risques de court-circuit inhérents aux fiches téléphoniques RJ11 nous font par

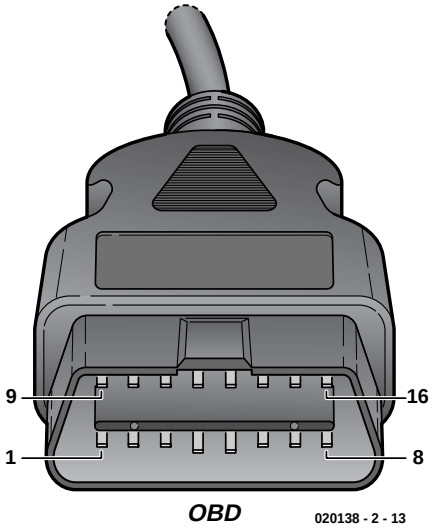


Figure 3. Fiche spéciale à 16 contacts pour raccordement à la prise de diagnostic du véhicule.

exemple déconseiller formellement celles-ci.
Le circuit de l'interface OBD/RS-232 avec l'ELM323 est reproduit dans la **figure 4**. Le circuit est alimenté par la fiche OBD de la prise de diagnostic du véhicule. La tension de bord (tension nominale de la batterie 14,4 V) est disponible à la broche 16, la masse à la broche 5 de la fiche. La broche 16 est raccordée par une diode de protection contre l'inversion de polarité à IC2, un régulateur de tension 78L05 qui stabilise la tension de fonctionnement du circuit à 5 V et par sa présence, protège le reste de l'électronique. La présence de la tension de 5 V est indiquée par la LED D8.
Les 2 liaisons restantes avec le véhicule (broches OBD 7 et 15) sont 2 lignes de données décrites dans les normes ISO 9141 et ISO 14230. La broche 7 de la prise est désignée par « sortie K » et la broche 15 par « sortie L » dans la norme ISO 9141-2. Nous nous servirons des termes « ligne K » et « ligne L » du système de bus OBD. Les 2 transistors NPN dont les résistances collecteur de 510 Ω faisant office de résistances de charge sont prescrites par les normes assurent la terminaison des lignes.
Le circuit reçoit les données par la ligne K (broche 7 de la prise OBD). Elles sont inversées par le transistor PNP T3 avant de parvenir à la broche 11 du ELM323. La valeur de commutation relativement élevée (4 V) de cet étage à transistor améliore le rapport signal/bruit par rapport à l'entrée CMOS de l' ELM323 dont le seuil de commutation est de 2,5 V. L'étage à transistor assure aussi la conformation d'impulsion et son amplification améliore la pente du signal de données.
Une interface RS-232 très simple du côté PC du circuit assure la liaison entre RxD (broche 2) et TxD (broche 3) de l'embase Sub-D 9 broches (K2) et les broches 5 et 6 du ELM323. Ce circuit « emprunte » la tension de l'ordinateur hôte pour atteindre les niveaux RS-232 maximaux sans l'apport d'une tension négative auxiliaire.
Les données sont envoyées directement par l'ordinateur à l'entrée Rx du circuit intégré (broche 5) par une résistance limitatrice de courant de 47 kΩ. R13 maintient un niveau défini (bas) à la broche 5 lorsque l'ordinateur n'est pas raccordé.
L'envoi des données RS-232 à partir de la broche 6 (sortie Tx du circuit intégré) est effectuée par le transistor PNP T4 qui commute entre +5 V et la tension négative emmagasinée dans le condensateur C3 ; le niveau alterne donc entre +5 V (haut) et environ –5,1 V (bas). Le condensateur est chargé à une tension négative par la ligne TxD de l'ordinateur. Ce couplage RS-232 joue parfaitement son rôle malgré son extrême simplicité.
En reliant la broche 4 du ELM232 à V_{DD} (+5 V), on active le mode LF, de sorte qu'un caractère CR sera automatiquement envoyé avec chaque Line Feed (LF).
Les 4 LED raccordées aux broches 7 à 10 indiquent si des données sont envoyées ou reçues du côté OBD ou RS-232. Une résistance de protection commune suffit pour chaque paire de LED (Tx et Rx) car il est

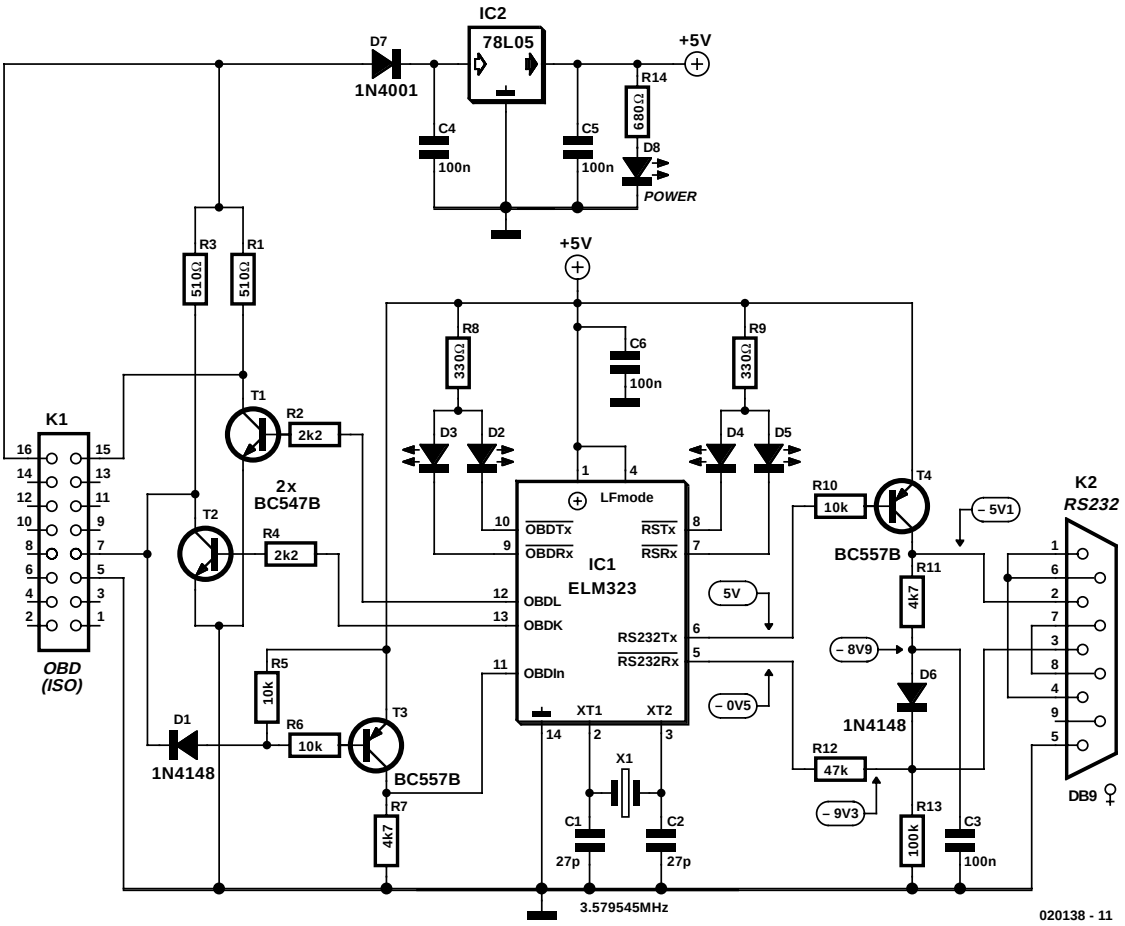


Figure 4. Circuit d'interfaçage pour raccordement direct à n'importe quelle prise de diagnostic compatible OBD-2.

impossible d'envoyer et de recevoir simultanément : l'ELM323 ne fonctionne pas vraiment en mode multitâche. Le bus OBD peut toutefois se trouver dans une phase d'initialisation au cours de laquelle des données sont envoyées par et vers le bus RS-232, de sorte qu'il est indispensable d'équiper les 2 groupes de LED de résistances séparées.
Le quartz entre les broches 2 et 3 révèle que l'ELM232 a été développé dans un pays où l'on utilise la télévision couleur NTSC : il s'agit en effet d'un quartz (bon marché) dont la fréquence correspond à celle de la sous-porteuse couleur NTSC. Les 27 pF des condensateurs de charge sont des valeurs type qui peuvent varier en fonction des caractéristiques du quartz.

Liste des composants

Résistances :
R1,R3 = 510 Ω
R2,R4 = 2kΩ2
R5,R6,R10 = 10 kΩ
R7,R11 = 4kΩ7
R8,R9 = 330 Ω
R12 = 47 kΩ
R13 = 100 kΩ
R14 = 680 Ω

Condensateurs :
C1,C2 = 27 pF
C3 à C6 = 100 nF

Semi-conducteurs :
D1,D6 = 1N4148
D2 à D5 = LED rouge
D8 = LED verte
D7 = 1N4001
T1,T2 = BC547B
T3,T4 = BC557B
IC1 = ELM323 *)
IC2 = 78L05

Divers :
K1 = embase à 2 rangées de 8 contacts (HE-10)
K2 = embase Sub-D à 9 contacts femelle encartable en équerre
X1 = quartz 3,579 545 MHz, 32 pF parallèle
connecteur 16 bornes aux normes OBD2 *)

*) Source pour le ELM323 et le kit de connexion OBD :
Küster Datensysteme (KDS)
Geibelstraße 14
D 30173 Hannover
Tél. : +49 511/886059
Fax. : + 49 511/8093329
E-Mail: OBD-Service@KDS-Online.com

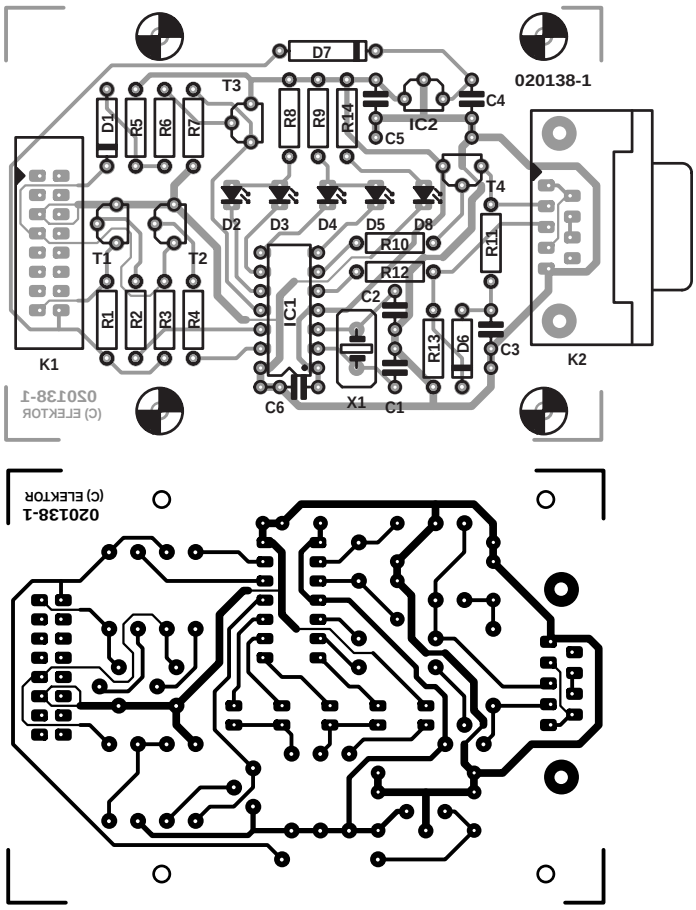


Figure 5. Carte de montage du circuit d'interfaçage.

Montage et test

Le montage sur la carte de la **figure 4** est aussi simple que le circuit lui-même. La carte simple face ne comporte pas de cavaliers. Permettez-moi d'insister sur un point ayant souvent causé des malentendus : la carte de la liaison RS-232 doit être équipée d'une embase Sub-D 9 broches. Les conducteurs du câble RS-232 tout à fait ordinaire utilisé (câble de prolongation) sont interconnectés 1:1. Ne pas utiliser de câble faux modem à conducteurs croisés !
Les sources d'approvisionnement en composants spéciaux (ELM323 et connecteur OBD) sont indiquées dans la liste de pièces. Procéder à la fin aux contrôles routiniers mais indispensables du montage (tous les composants correctement placés et soudés), des soudures et des pistes. Ne jamais connecteur la carte au véhicule lors du premier essai. Il suffit de disposer d'une table d'essai (alimentation secteur 12 V ou, faute de mieux, batterie 9 V) et d'un ordinateur muni d'une interface sérielle à proximité. Lorsque la tension d'alimentation est appliquée à la broche 16 (+12 V) et à la broche 5 (masse), la LED verte (affichage pour +5 V) doit s'allumer, les LED rouges doivent s'allumer brièvement l'une après l'autre. On peut maintenant vérifier la tension de 5 V par mesure de précaution.
Relions la carte au PC. Les lignes RxD et TxD de l'interface sérielle sont soumises à présent aux tensions appliquées par le PC. Les valeurs caractéristiques données dans le schéma peuvent être comparées aux valeurs mesurées. Les valeurs –0,5 V et +5 V (tensions aux broches 5 et 6 du ELM323) ne devraient pas présenter de déviation majeure, tandis que la tension à la broche 3 de K2 n'est autre que la tension de la ligne TxD. La valeur de cette tension dépend des circuits intégrés d'interfaçage utilisés dans le PC et peut dévier significativement de la valeur type de –9,3 V

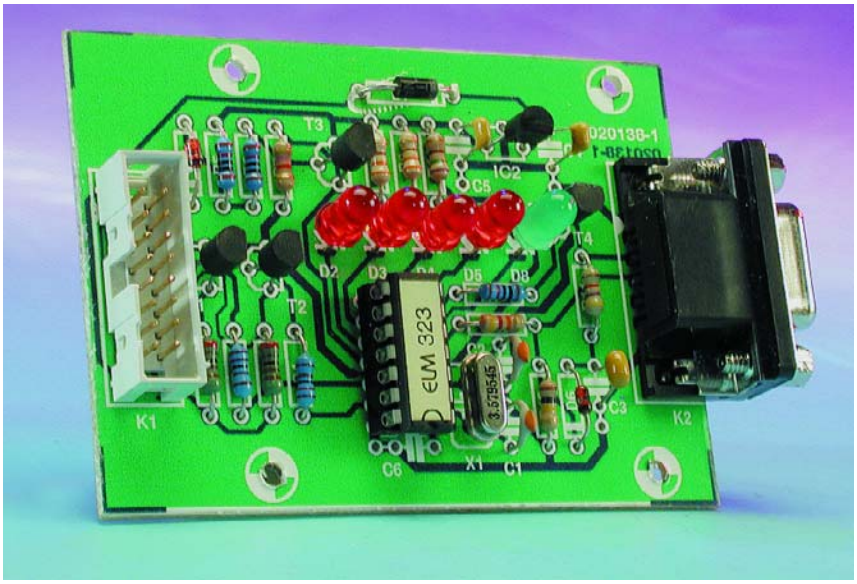


Figure 6. Carte prototype montée. Les LED rouges indiquent l'émission et la réception de données aux E/S.

indiquée. La plage de déviation peut atteindre de –3 V à –12 V. Cette valeur a une influence directe sur la tension de C3 qui est plus positive que la tension TxD, la différence étant celle de la tension aux bornes d'une diode (chute de tension aux bornes de D6, environ 0,4 à 0,6 V). Si la tension sur TxD est égale à -9,3 V, la valeur aux bornes de C3 sera celle indiquée dans le schéma, soit –8,9 V. Des déviations par trop notables lors des essais sont l'indice d'une erreur qu'il s'agit de corriger. L'interface sérielle du PC devrait survivre à un court-circuit car la norme RS-232 prescrit une limitation de courant.

Des tests de fonctionnement plus poussés de l'interface sérielle nécessitent la présence d'un logiciel approprié sur le PC. On peut bien entendu se servir du programme Windows mentionné au début. Pour qui ne veut pas attendre aussi longtemps et ne recule pas devant quelques réglages, il est aussi possible d'avoir immédiatement recours à un programme de terminal comme HyperTerminal pour vérifier le fonctionnement de la transmission entre le PC et l'ELM323. Si Windows a été installé sans programme d'émulation de terminal, HyperTerminal peut être téléchargé gratuitement à partir de http://www.hilgraeve.com.

Lancer Hyperterminal avec les réglages de l'interface suivants : Débit 9 600 bauds, 8 bits de données, pas de bit de parité, 1 bit d'arrêt et pas d'acquittement (h*andshake*) (protocole de contrôle, donc pas de acquittement hardware ni d'acquittement XON/XOFF). Pour résumer : 9600,8N1

Si l'interface correctement branchée est mise sous tension, les 4 LED rouges s'allument d'abord l'une après l'autre, puis le message suivant

ELM323 v1.0

>

apparaît sur l'écran. Ce message révèle la version du circuit intégré, mais il signifie aussi que celui-ci fonctionne, que le nombre de bauds est correct, et que la transmission du circuit intégré à l'entrée Rx du PC s'effectue correctement. Le caractère « > » qui suit le message constitue l'invite d'entrée (*Prompt*) du ELM323 indiquant que l'interface se trouve en état d'attente, prête à accepter des caractères du port RS-232.

Les informations envoyées par l'ordinateur peuvent être destinées à l'usage interne du ELM323 ou à être traitées puis envoyées au bus OBD. L'ELM323 peut déterminer rapidement la destination des caractères reçus en analysant la suite qu'ils forment. Les commandes internes du ELM323 commencent toujours par les caractères « AT » comme dans le cas des modems, alors que les commandes destinées au bus OBD ne doivent contenir que les codes ASCII des chiffres hexadécimaux (de 0 à 9 et de A à F). Pour effectuer un essai, entrer par exemple la commande ATE1 suivie d'un CR. Si l'on n'obtient pas de réponse OK, vérifier la connexion à la masse (broche 5) et s'assurer qu'aucun « Handshake » n'est actif.

Utilisation

Le plus simple est d'utiliser le programme Windows mentionné au début et présenté dans le prochain fascicule. Il permet de ne plus se soucier des détails de la communication entre l'interface et le PC d'une part et avec l'interface OBD-2 du véhicule d'autre part.

Pour qui voudrait toutefois tester plus à fond les fonctions et les possibilités disponibles, il est tout à fait possible de continuer à se servir du programme de terminal sur le PC et de s'initier aux secrets de l'interface OBD-2 grâce à une documentation plus abondante à télécharger sur le site web d'Elektor.

Les points suivants sont traités :

– Communication entre le PC et l'ELM323
– Commandes AT
– Initialisation du bus OBD
– Commandes OBD
– Modes de test de diagnostic
– Lecture et évaluation des codes d'erreur
– Effacement des codes d'erreur
– Messages d'erreur du ELM232.

Cette description détaillée est destinée avant tout aux lecteurs désireux de réaliser leurs propres applications avec l'interface et de faire appel à des techniques spéciales. Des conseils supplémentaires pour les programmateurs amateurs, ci-inclus des exemples de code, sont aussi prévus dans la contribution finale du prochain fascicule où l'accent sera mis sur l'aspect logiciel.

(020138-2)

Sources :

Cette contribution est basée sur une fiche de données en traduction autorisée de l'entreprise Elm Electronics, Canada. Pour télécharger la fiche de données originale :
www.elmelectronics.com/dsheets.html

NdlR :

La Rédaction regrette qu'il y ait eu une erreur dans le premier article en l'attribuant à Mr Haas, alors que son auteur était Mr Müller, source de cet article également.
Rendons à César ce qui est à César !

Alarme pour moto

À double détecteur, protection et télécommande

Projet : Goswinning Visschers

texte : Sjef van Rooij

Ce système d'alarme à base d'une paire de PIC16F84 est relativement facile à réaliser et à installer soi-même. Comparé à un système du commerce, le coût est très abordable, d'autant plus que les fonctionnalités n'appellent aucune critique !

Il suffit de lire les journaux pour se convaincre d'une augmentation explosive du nombre de vols de mots et autres scooters. Bien souvent, des bandes organisées volent, pour ainsi dire sur commande, certaines types de motos. Tout en haut de la liste de souhaits de ces malfrats on trouve des modèles exclusifs de Ducati ou de Harley-Davidson, mais ils ne crachent pas non plus sur les versions les plus chères de BMW, ainsi d'ailleurs que la Goldwin de Honda dont le prix approche les 14 000 €. Ce n'est pas pour autant que les possesseurs de motos ou de scooters sont à l'abri d'une mauvaise surprise, car cette catégorie aussi voit nombre de ses sujets passer (illégalement) d'une main à l'autre. Désolant mais vrai.

Il semblerait heureusement que la police commence à mieux maîtriser les faits en mettant en cartes les zones sensibles, mais cela ne vous interdit pas de bien surveiller votre bien précieux et si vous y tenez comme à la prunelle de vos yeux, une installation d'alarme peut être une nécessité incontournable.
Le prix d'une installation d'alarme fiable est malheureusement loin d'être négligeable. On a vite fait, si l'on veut un système tant soit peu correct, d'avoir dépensé quelques centaines d'euros et cela sans même tenir compte des frais d'installation. Lorsque l'on se trouve confronté à de telles sommes, on y réfléchit à deux fois et on ne manque pas de se poser la question s'il n'y a pas moyen de faire moins cher...

Il va sans dire qu'il existe aussi des installations moins chères mais il s'avère bien souvent que ce « prix d'ami » est directement proportionnel à la fiabilité du système. Pour peu que l'on ait quelques notions d'électronique on en arrive rapidement à une réalisation personnelle. D'un point de vue électronique, la complexité d'une alarme fonctionnant correctement n'a rien d'effrayant, d'autant plus qu'une réalisation personnelle et son montage présentent l'avantage de pouvoir adapter au mieux l'installation à la moto ou au scooter en question.
L'avantage majeur de notre approche se situe cependant au niveau des coûts, l'installation devrait pouvoir être réalisée pour de l'ordre d'une centaine d'euros !

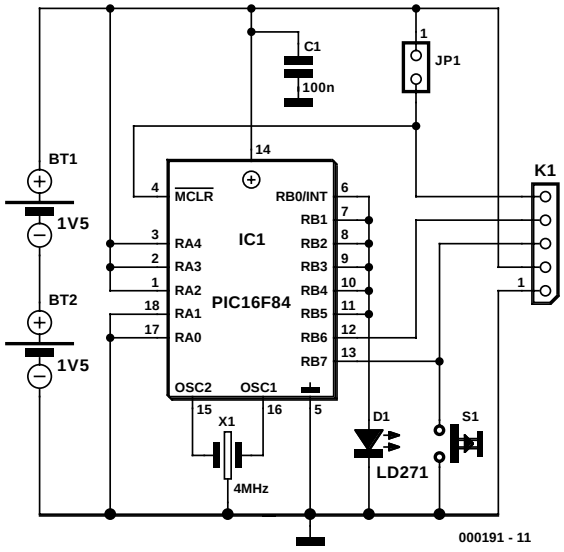


Figure 1. La télécommande se résume en fait quasiment à un microcontrôleur PIC (programmé) et à une LED d'émission IR.

Le cahier des charges

Quelles sont les exigences que l'on est en droit de poser à une alarme anti-vol pour moto ou scooter fiable? En fait, il existe 3 aspects importants:

– Le montage doit comporter un capteur sensible capable de détecter le moindre mouvement. Une alarme pour moto n'a que faire de détecteurs à infrarouge ou à ultra son tels que nous les connaissons des installations domestiques. Ce dont nous avons besoin est un capteur en mesure de détecter tout mouvement mécanique ou des vibrations.
– Il se doit bien évidemment de posséder un signal d'alarme tant optique qu'acoustique. La réalisation de cette fonction ne pose pas de gros problème.
– Il faut en outre que l'alarme soit relativement bien protégée contre des tentatives de sabotage inévitables. Cet aspect est moins évident, mais il faudra impérativement en tenir compte.
– Il faut également que le seul propriétaire puisse activer et mettre l'alarme hors-fonction par l'entrée du code correct. Et cela de préférence à l'aide d'une télécommande.

Plan d'attaque

Quels sont les aspects du cahier des charges établi plus haut à prendre le plus de poids lors de la réalisation pratique et quelle approche avons-nous adoptée ?

Nous avons choisi de travailler avec 2 types de détecteurs de mouvement différents. On pourra penser que nous avons poussé un peu loin notre besoin de perfection, mais ce choix ne grève notre budget que très peu et ne vaut-il pas mieux, de mettre l'accent sur la sécurité ? Un détecteur au mercure, dont nous avons d'ailleurs utilisé 2 exemplaires, sert à la détection de tout mouvement latéral de la moto. Une version modifiée d'élément piézo-électrique sert ici de capteur de vibrations; la sensibilité de ce dernier en particulier est très élevée au point qu'il est possible de l'ajuster pour qu'il réagisse très rapidement lorsque l'on bouscule la moto, même très légèrement.

Nous en arrivons ensuite à la protection anti-sabotage. Nous avons pris 2 précautions à ce niveau. Une photo-résistance (LDR) déclenche l'alarme lorsque l'on essaie, sans précaution préliminaire, d'enlever le siège arrière ou le couvercle latéral derrière lequel se trouve montée l'électronique de l'alarme. Une sécurité supplémentaire évite en outre que l'on ne puisse saboter l'alarme par la mise en court-circuit des lignes d'alimentation au niveau des clignotants.

Le déclenchement de l'alarme se traduit par la mise en fonction des clignotants de la moto et la mise en circuit d'une sirène piézo qui, si elle est compacte, n'en fait néanmoins pas énormément de bruit. L'activation de l'alarme se fait à l'aide d'un minuscule émetteur IR, ce qui implique bien évidemment la présence, au niveau de l'alarme elle-même, du module de réception IR correspondant.

Le « coeur » du montage est un PIC16F84 dans lequel a été stocké le programme. Certains de nos lecteurs ne manqueront pas de se demander s'il n'aurait pas été possible de réaliser cette alarme sans avoir besoin de microcontrôleur. Nous ne pouvons bien entendu répondre à cette question silencieuse que par l'affirmative. Il n'en reste pas moins que cette électronique doit remplir un certain nombre de fonctions : il faut, pour commencer, traiter le signal en provenance des détecteurs, détecter le signal émis par la télécommande et le convertir en un signal, selon le cas, d'activation ou de mise en sommeil de l'installation, il faut en outre fournir le signal de commutation des clignotants et de mise en fonction de la sirène, sachant que ce signal d'alarme doit être coupé au bout de 20 s, durée limite légale de nombreux pays européens, et que le nombre maximum de déclenchements successifs de l'alarme est fixé 3. Il reste pour terminer à faire en sorte que l'installation soit réactivée une fois écoulé le processus d'alarme en cours.

Comme ce problème d'encombrement est encore plus aigu en ce qui concerne l'émetteur de la télécommande, nous avons également doté cette dernière d'un PIC16F84, en version CMS cette fois. Les tâches de ce microcontrôleur sont bien plus limitées puisqu'il lui suffit d'assurer la génération du signal impulsionnel de 36 kHz de caractéristiques correctes nécessaire au fonctionnement de la LED IR d'émission.

Il est possible d'obtenir les 2 PIC programmés auprès des adresses habituelles; cependant, si vous voulez les programmer vous-même (et que vous disposez des outils requis), nous mettons les fichiers source et .hex à votre disposition sur notre site Internet d'où vous pourrez les télécharger gratuitement.

Mini-émetteur IR

Il est plus que temps que nous nous intéressions enfin au schéma. Nous allons commencer, pour nous mettre dans le bain, par l'émetteur de la télécommande vu que son électronique se résume, outre le PIC CMS, à très peu de choses. La **figure 1** en donne le schéma.

Le résonateur X1 fournit la fréquence de 4 MHz requise, D1 est la LED infrarouge d'émission et S1 fait office de bouton marche/arrêt. L'embase K7 sert à la program-

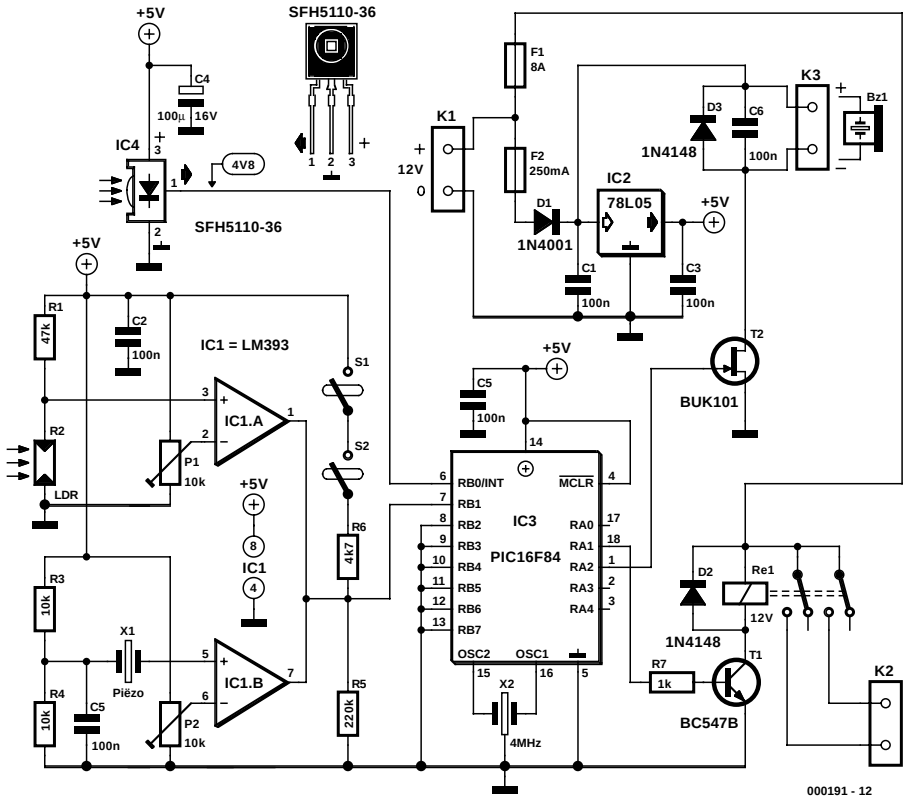


Figure 2. Au coeur du circuit de l'alarme proprement dite on trouve également un PIC. Sur sa droite on découvre le récepteur IR et les détecteurs, sur sa droite les dispositifs produisant le signal d'alarme.

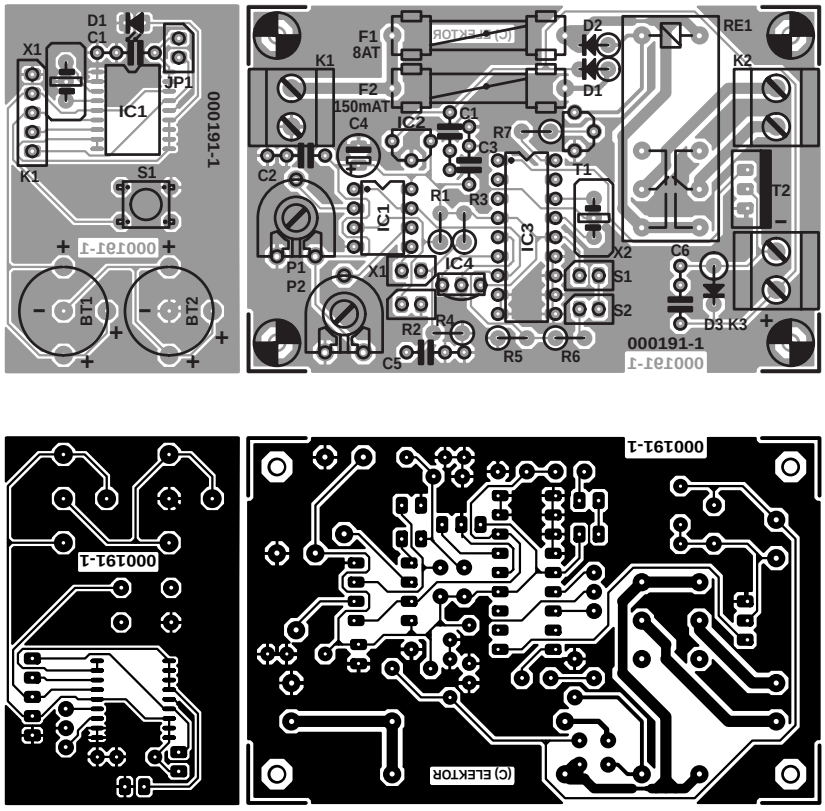


Figure 3. Les platines de l'émetteur et de l'alarme sont proposées d'une seule pièce. Il vous faudra les détacher à l'aide d'un trait de scie.

mation du PIC, vu que sinon la programmation d'une version CMS devient un calvaire, à moins de disposer de l'adaptateur adéquat. Il faudra, pour la programmation, enlever momentanément le cavalier JP2 (qu'il ne faudra pas oublier de remettre en place une fois la programmation terminée.

Le programme stocké dans IC1 est très simple. Une action sur S1 fait sortir le PIC de son état de sommeil (*sleep*), le processeur émettant ensuite le code d'impulsions prédéterminé par le biais de ses sorties RB0 à RB5. Le PIC retombe ensuite dans son mode de sommeil. Le code IR est envoyé en direction du récepteur de l'alarme par le biais de la LED D1. Les sorties RB0 à RB5 ont été prises en parallèle de manière à pouvoir fournir le courant nécessaire à la LED.

Bien que le niveau de tension d'ali-

Liste des composants de l'émetteur

Condensateurs :
C1 = 100 nF

Semi-conducteurs :
D1 = LED IR tel que, par exemple, CQW13 (Conrad 184551-89)
IC1 = PIC16F84-04/SO CMS (programmé **EPS 000191-41**)

Divers :
JP1 = embase SIL mâle à 2 contacts + cavalier
K1 = embase SIL mâle à 1 rangée de 5 contacts
S1 = bouton-poussoir miniature
X1 = résonateur 4 MHz à 3 broches
BT1,BT2 = pile-bouton1,5 V + porte-pile encartable 16 x 1,6 mm (tels que, par exemple, CR1616)
boîtier : par exemple, Teko 11121 (Conrad 543287)

Liste des composants de l' alarme

Résistances :
R1 = 47 kΩ
R2 = LDR
R4,R4 = 10 kΩ
R5 = 220 kΩ

mentation requis par le PIC16F84 soit, dit la fiche de caractéristiques, de 5 V, il apparaît que ce composant fonctionne, dans la pratique, fort bien à 2,8 V. Ceci explique que la source d'alimentation de l'émetteur prenne la forme d'une paire de piles-bouton de 1,5 V prises en série. Vu que la consommation du PIC en mode sommeil est insignifiante, une paire de piles devrait durer au moins un an.

Le détail de l'électronique de l'alarme

La **figure 2** nous donne le schéma de l'alarme proprement dite. Un premier coup d'oeil suffit à se convaincre que la mise en oeuvre d'un PIC présente au moins l'avantage de simplifier sensiblement les choses.

Le contrôleur, IC3, « trône » au centre du schéma; son « pouls » lui est fourni par le résonateur X2, qui travaille à une fréquence de4 MHz. Il n'y a pas grand chose à ajouter au sujet du PIC, vu que c'est le programme qui y est stocké qui se charge de pratiquement tout. Nous avons vu plus haut quelles étaient les différentes fonctions qu'il avait à remplir. Il nous faut cependant mentionner que 2 des sorties (non utilisées), ont été mises à contribution pour la détection d'erreurs : RA0 donne l'état de l'alarme (un « 0 » indique qu'elle sommeille, un « 1 » lorsqu'elle est activée), la sortie RS3 passant, elle, à « 1 » lors de la réception d'un signal IR.

Le récepteur IR intégré, IC4, que l'on retrouve dans la partie supérieure gauche du schéma, capte le signal émis par l'émetteur. Est-il bien nécessaire de préciser que ce composant devra être monté de façon à ce qu'il soit en « contact visuel » avec l'émetteur et on pourra le relier à la platine par le biais d'un petit morceau de câble. Le signal émis par la télécommande capté est transmis à l'entrée RB0 du PIC, qui reconnaît le code et, sur commande, change l'état de l'alarme : si elle se trouvait en « stand-by » elle est activée et inversement passe à l'état de veille si elle était activée. Chaque action sur le bouton poussoir se traduit par un changement d'état. Chaque changement est visualisé par un double « clin d'oeil » des clignotants. Lors d'un retour en mode de stand-by, d'éventuels clins d'oeil supplémentaires indiquent le nombre de déclenchements de l'alarme au cours de la période « d'activation » qui a précédé.

En dessous de IC4 nous trouvons l'électronique du sous-ensemble de détection; il se compose des interrupteurs à mercure S1/S2 et une paire de circuits de comparateurs IC1.A et IC1.B qui concernent respectivement la LDR et l'élément piézo-électrique. Le circuit est conçu de manière à ce que l'entrée RB1 passe au niveau « bas » lorsqu'il y a lieu de déclencher l'alarme.

En cas de mouvement de la moto, l'interrupteur S1 et/ou S2 s'ouvre et la ligne RB1 se trouve forcée au niveau bas par le biais de la résistance R5. Si la photo-résistance R2 est frappée par de la lumière ou que l'élément piézo-électrique X1 se met à produire une tension sous l'effet de vibrations, la sortie à collecteur ouvert de, respectivement IC1.A ou IC1.B devient passante, ce qui se traduit à nouveau par un forçage au niveau bas de la ligne RB1. L'ajustable P1 permet de jouer sur la sensibilité de la LDR, celle de l'élément piézo se laissant ajuster par le biais de l'ajustable P2.

En cas d'alarme, le FET T2 est mis en conduction par la ligne RA2, ce transistor activant à son tour la sirène piézo connectée au bornier K3. Nous avons utilisé, pour T2, un type de FET acceptant d'être connecté directement à la sortie d'un circuit intégré TTL et capable de commuter une intensité de 5 A. Il n'a partant pas le moindre problème avec les quelque 140 mA que consomme la sirène. Si nous avons opté pour un FET cela tient au fait qu'il ne coûte guère plus cher qu'un relais mais qu'il est bien moins encombrant que ce dernier.

Depuis R1, on a en outre, en cas d'alarme, activation, par le biais de T1, des feux clignotants de la moto reliés au bornier K2. Lors de chaque changement d'état de l'alarme (d'un état actif à un état de sommeil et vice-versa) la ligne RA1 génère un bref signal qui se traduit par un double clignotement des feux (chiffre auquel s'ajoutent les clignotements signalant une excitation de l'alarme au cours de sa période «active»). Les diodes D2 et D3 éliminent toute tension inductive potentielle pouvant se manifester.

L'alimentation se résume à un régulateur de tension 5 V, IC2, qui dérive sa tension d'entrée de la batterie de la moto ou du scooter par l'intermédiaire du bornier K1. D1 évite qu'une interversion malencontreuse des lignes d'alimentation (le + et la masse) n'ait de conséquence désastreuse sur l'électronique du montage. Comme nous le disions plus haut, nous avons prévu, pour des raisons de sécurité, une double protection par fusible. S'il venait l'idée à un voleur imaginatif d'ouvrir les clignotants pour y court-circuiter la tension d'alimentation, le fusible F1 rend l'âme, mais le fusible F2 reste intact lui. Le montage reste partant fonctionnel, l'alarme pouvant encore être donnée par le biais de la sirène.

Une note « rassurante » en ce qui concerne l'alimentation : au repos l'ensemble du circuit ne consomme que 12 mA, quel que soit le mode, actif ou en sommeil dans lequel il se trouve. Il n'y a donc pas de risque d'épuiser rapidement la batterie.

Une platine double

Vu que, par définition, l'alarme et la télécommande ne peuvent se passer l'une de l'autre,

R6 = 4kΩ7
R7 = 1 kΩ
P1,P2 = 10 kΩ

Condensateurs :
C1 à C3,C5,C6 = 100 nF
C4 = 100 µF/16 V radial

Semi-conducteurs :
D1 = 1N4001
D2,D3 = 1N4148
T1 = BC547
T2 = BUK101
IC1 = LM393
IC2 = 78L05
IC3 = PIC16F84-04/P (programmé **EPS000191-42**)
IC4 = SFH5110-36

Divers :
K1 à K3 = bornier à 2 contacts au pas de 5 mm
RE1 = relais12 V bipolaire 8 A, (tel que, par exemple, Schrack RT424012)
S1,S2 = interrupteur au mercure (Conrad 700452)
BZ1 = sirène 12 V telle que, par exemple Alecto
X1 = élément piézo (Conrad 712930)
X2 = résonateur 4 MHz à 3 broches
F1 =porte-fusible + fusible 8 A(T)
F2 = porte-fusible + fusible 250 mA (T)

Figure 4. Ce mini-boîtier paraît avoir été conçu tout spécialement à l'intention de la platine de l'émetteur.

nous les avons mises toutes deux sur une même platine dont on retrouve le dessin des pistes en **figure 3**. Il vous faudra les séparer d'un trait de scie. Vous aurez alors en main 2 platines compactes, la plus petite d'entre elles, l'émetteur, ayant la taille d'une boîte d'allumettes.

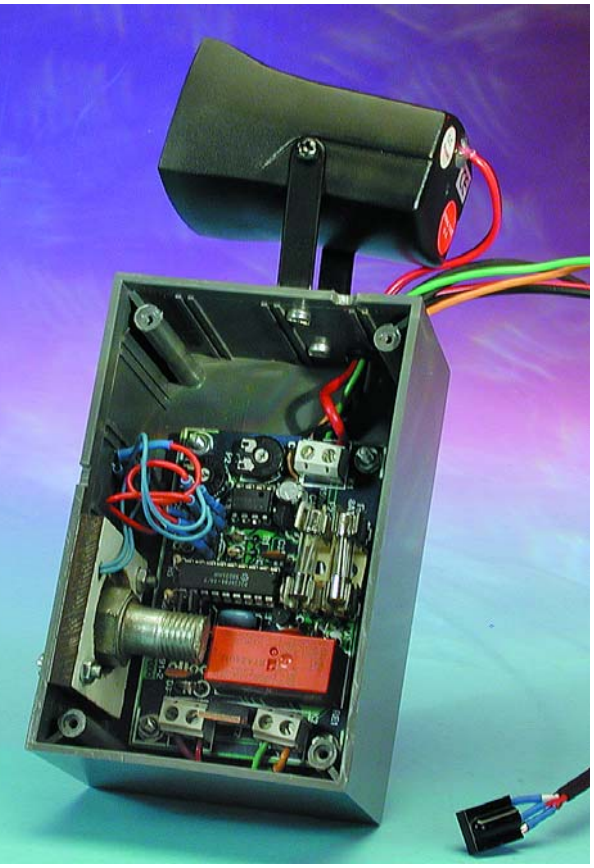


Figure 5. Vue plongeante dans notre prototype de l'alarme. La technique de montage dépendra beaucoup du type de moto ou de scooter devant être doté de cette alarme.

La platine de l'émetteur ne comporte que le minimum indispensable de composants de sorte que sa réalisation est une affaire de quelques minutes. Seule l'opération de soudure du circuit intégré CMS exige une attention particulière et requiert un fer à souder à pointe fine. Si vous êtes débutant en la matière, il est préférable de commencer par positionner le composant exactement à la position prévue et de l'y fixer à l'aide d'une goutte de colle. On soude ensuite, avec grand soin, l'une des pattes du composant, vérifie une nouvelle fois son positionnement, soude une patte diamétralement opposée à la première, s'assure une dernière fois du positionnement correct de la puce avant de souder le reste des pattes, une à une en vérifiant à la loupe la qualité de la soudure. On pourra éliminer tout excédent de soudure à l'aide d'un morceau de tresse à dessouder. On vérifiera soigneusement, en fin d'opération de soudure, l'absence de court-circuit produit par des restes de soudure ! L'écartement des broches est très faible...

S'il devait se faire que vous n'ayez pas de support pour pile au lithium vous pourrez utiliser du fil de câblage souple pour réaliser une paire d'armatures de soutien. Le résultat est pratiquement aussi bon. Notons au passage que les piles doivent être positionnées dans les supports leur pôle positif tourné vers le haut.

La réalisation de la platine principale ne constitue pas une opération sujette à problèmes. La sérigraphie de l'implantation des composants représentée en figure 3 donne toutes les informations nécessaires, l'identification des différents points de connexion étant clairement indiquée. L'implantation des composants standard ne devrait pas poser de problème. La partie de cette étape de réalisation de la platine principale concerne moins cette dernière que les composants qui ne prennent pas place sur le circuit imprimé mais qui y sont reliés par le biais de morceaux de fil de câblage aux points de connexion prévus. Nous y reviendrons un peu plus loin.

Mise en boîtier

La platine de l'émetteur est tellement compacte qu'il est possible de l'implanter, même dotée de ses piles au lithium, dans une boîte d'allumettes. Il va sans dire qu'il existe des boîtiers plus solides conçus tout spécialement à l'intention des télécommandes, solution qui aura notre préférence. Nous avons utilisé un boîtier de Teko, disponible entre autres chez Conrad (cf. **figure 4**), mais rien ne vous interdit d'opter pour un boîtier PP22 de Supertronic (Selectronic). IL suffira de percer, sur l'avant du boîtier, un petit trou destiné à la diode d'émission IR. Le boîtier utilisé comporte à l'origine un orifice ovale dans lequel vient se glisser un bouton-poussoir « soft-touch » à l'aplomb duquel nous avons disposé l'interrupteur à pression S1. La mise de la platine dans le coffret est l'affaire de quelques minutes.

Venons-en à la platine principale. La technique adoptée pour son montage dépend de la place dont on dispose sur la moto (ou le scooter). Dans bien des cas, on ira voir du côté du siège passager, l'espace disponible à cet endroit variant cependant d'un modèle de moto à l'autre. On pourrait fort bien imaginer de ne pas utiliser de boîtier mais de glisser le montage dans un morceau de gaine thermo-rétractable « géante » qui servirait d'isolation avant de le mettre et de le fixer directement dans le compartiment de la boîte à outils. Nous avons mis notre prototype dans un boîtier standard simple (cf. **figure 5**). Nous aimerions placer quelques remarques concernant sa réalisation et son câblage.

IL va sans dire qu'il est important de ne pas faire d'erreur au niveau de la polarité de la tension d'alimentation, du récepteur IC4 et de la sirène. On pourra substituer au récepteur IR SFH5110-36 utilisé ici le cas échéant un SFH505; il faudra se rappeler dans ce cas-là que le brochage de ce dernier type de récepteur est inversé de 180 ° par rapport à celui du SFH5110-36. Attention donc ! Il faudra veiller, lors de la connexion des feux clignotants au bornier K2, à bien différencier la gauche de la droite sachant que sinon vous courrez le risque de voir les clignotants

gauche et droit clignoter de concert simultanément.
Il est évident qu'il faudra monter IC4 de façon à ce qu'il puisse voir le signal IR en provenance de l'émetteur. Cela implique de percer un petit orifice dans l'arrière de la carrosserie de la moto; si l'on procède à un montage en retrait de la diode, de manière à lui éviter de détecter des signaux qui ne lui sont pas destinés.
Il faudra utiliser du câble à double âme blindée (audio) pour relier IC4 à la platine, ceci en vue d'éviter le risque de capture ou de génération de signaux parasites.
Les interrupteurs au mercure S1 et S2 devront être montés de façon à établir le contact lorsque la moto ou le scooter se trouve dans sa position de parcage normale (béquille). Il se peut que vous ayez besoin de quelques corrections de positionnement avant d'avoir trouvé les positions correctes de ces 2 interrupteurs.
Sur notre prototype, nous avons monté la LDR R2 sur le côté du boîtier. Il est important que cette photorésistance soit « illuminée » lorsque, selon le cas, on soulève le siège arrière ou que l'on ouvre le compartiment dans lequel est installé le montage.
Il nous reste à parler du montage de l'élément piézo-électrique X1 :

Détecteur de vibrations

Le piézo-élément (bon marché) mis en oeuvre ici s'est avéré être un excellent détecteur de vibrations très efficace, à tel point que nous avons sérieusement envisagé, au cours de développement, de nous passer des interrupteurs à mercure. SI vous optez pour cette solution (pas d'interrupteurs au mercure) il faudra procéder à une petite modification. Il faut en effet accroître la sensibilité du piézo-élément.
On colle l'élément à son boîtier (n'en coller que le bord, de manière à laisser une liberté de mouvement à la partie active de l'élément !) puis on dote l'autre face d'un poids de quelque 60 g que l'on y colle. L'inertie de masse de ce petit poids accroît très sensiblement la déformation de l'élément induit par les vibrations, ce qui augmente la sensibilité du capteur à un point tel qu'il peut être considéré comme l'un des éléments essentiels de toute l'installation d'alarme. Nous avons utilisé, en tant que poids additionnel, un boulon M10 raccourci à une longueur de 25 mm. Cette technique fonctionne parfaitement.
Autre solution pour les amateurs de perfection : ils pourront monter le piézo-élément dans un support plastique qu'ils auront réalisé eux-mêmes à l'aide d'une fraise avant de fixer le tout au coffret à l'aide de 2 vis ( ). Sachant qu'il nous fallait effectuer un certain nombre d'expériences sur notre prototype nous avons opté pour cette solution.

L'étalonnage

Une fois que l'on a terminé la réalisation de l'ensemble du montage, que l'on y a connecté les clignotants et la sirène, on commencera par tourner les ajustables P1 et P2 à fond vers la gauche avant de connecter la tension de batterie au bornier K1.
Une fois que l'on a connecté la tension d'alimentation l'installation d'alarme se trouve en veille (*standby*). Dès réception du code envoyé par l'émetteur de la télécommande et vérification de sa correction par le PIC16F84, on aura un double clignotement des clignotants, l'alarme étant ensuite « armée » (active). On pourra contrôler cet état par vérification de la présence d'un niveau logique haut sur la ligne RA0 de IC3.
ON pourra ensuite jouer avec précaution sur les ajustables P1 et P2 et les tourner progressivement vers la droite jusqu'à ce que l'on ait trouvé la sensibilité souhaitée. Attention à ne pas exagérer car il peut arriver que l'on rende, par action sur P2, l'élément piézo tellement sensible que l'alarme soit sur le point de se déclencher pour peu qu'un connaisseur ne fasse rien de plus que de montrer la moto du doigt !

(000191)

Changer le code ?

En pratique, le fait que tous les réalisateurs potentiels de ce montage aient en principe le même code infrarouge pour mettre l'alarme en fonction et la désactiver est bien entendu loin d'être idéal. Ceux d'entre nos lecteurs qui ont la possibilité de programmer les PIC ou ont au nombre de leurs connaissances quelqu'un qui le puisse, pourront bien évidemment remédier à cet état de faits. Il leur est en effet possible de modifier le code d'origine pour le remplacer par leur propre code. On aura besoin pour cela d'un assembleur et d'un programmateur de PIC.

Dans la rubrique Téléchargements de notre site (ww.elektor.fr) il suffit de cliquer sur le numéro du magazine dans l'année concernée, ici 292, pour trouver le code-source des 2 contrôleurs PIC. Leurs dénominations sont respectivement 000191-41 et 000191-42.
Il faudra, dans les 2 fichiers, rechercher les lignes « code 1 » et « code 2 »; il suffira ensuite de remplacer les nombres que l'on y repère, « F2 » et « 45 » par de nouvelles valeurs qui doivent être comprises entre « 00 et FF ». Il va sans dire que les valeurs doivent être les mêmes pour les 2 contrôleurs. Il ne reste plus ensuite qu'à assembler les codes-source modifiés et à les transférer dans les 2 composants à l'aide d'un programmateur adéquat.

Notons que nous avons décrit un mini-programmateur pour PIC16F84/16C84 à réaliser soi-même dans notre numéro double de juillet/août 2000 en page 85.

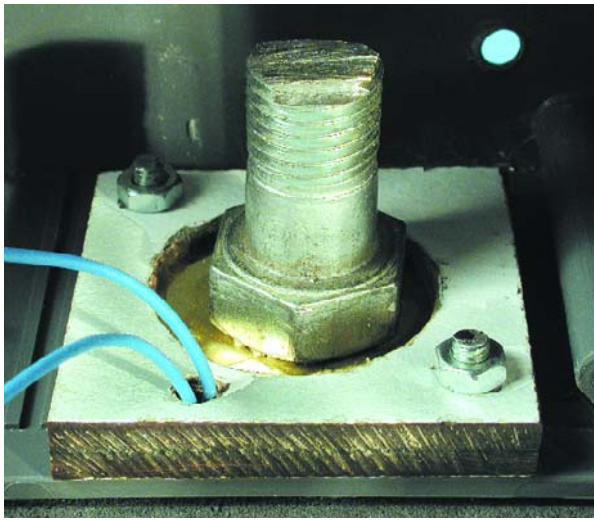


Figure 6. L'élément piézo est doté ici du contrepoids additionnel dont il est fait mention dans l'article avant d'être collé dans un support en plastique.

Tout ce que vous avez toujours voulu savoir sur Les radars

Sans oser le demander…

Benoît Bouchez

Redoutés de bien des automobilistes (que celui qui n'a jamais commis d'excès de vitesse lance la 1ère pierre…), les cinémomètres (plus couramment appelés 'radars') sont des équipements électroniques dont le fonctionnement reste malgré tout méconnu de nombre d'électroniciens. Dans cet article, nous allons présenter de façon détaillée le fonctionnement des curieux engins, et mettre à mal un certain nombre d'idées reçues sur les moyens de se prémunir de leur action.

Remarque importante :

nous ne nous intéresserons ici qu'aux radars hyperfréquences, et non aux radars « laser », récemment apparus et qui pourraient faire, le cas échéant, l'objet d'un prochain article.

(Source: Applied Concepts, Inc. / Stalker Radar)

Un peu d'histoire…

Les travaux en radioélectricité menés dans les années 1930 ont démontré que les ondes HF avaient la capacité de se réfléchir sur les obstacles. Toutefois, cette propriété apparaît de plus en plus nettement au fur et à mesure que la fréquence s'élève (au-delà de 100 MHz). Ce n'est qu'à la fin des années 1930 qu'il fut possible de produire facilement des ondes SHF (f > 1 GHz), grâce à la mise au point du *magnétron* (toujours utilisé dans les fours à micro-ondes actuels, soit dit en passant).

La première application pratique de ce système fut le *Radio Detection And Ranging* (Mesure de Distance et Détection par Radio), connu par la suite sous l'acronyme de RADAR. Le principe du radar est l'émission à intervalles réguliers de trains d'onde SHF. Lorsque ces ondes rencontrent un obstacle suffisamment volumineux, une partie d'entre elles est renvoyée vers l'émetteur, où elles peuvent être captées par une antenne.

Dans les systèmes primitifs, on synchronisait le départ du balayage horizontal d'un tube cathodique avec l'émission de la salve, le signal de retour capté par l'antenne étant envoyé aux amplificateurs verticaux. Si l'antenne de réception captait un signal en retour (train d'onde renvoyé par un obstacle), on observait donc un pic sur l'écran, situé plus ou moins sur la droite du tube selon la distance de l'obstacle (voir **figure 1**). Ce système permettait donc de détecter tout obstacle (avion) placé dans le volume de propagation des ondes (détection), mais également de connaître avec précision la distance entre l'antenne et l'obstacle,

connaissant la vitesse de propagation des ondes et le temps de balayage horizontal du tube cathodique (mesure de distance), d'où ce sigle de R.A.D.A.R (formulé sous la forme d'un nom commun ensuite).
Les radars furent par la suite améliorés afin de pouvoir surveiller non pas uniquement la zone située devant le couple émetteur/récepteur, mais tout le volume avoisinant, simplement en les montant sur un support rotatif (les tubes cathodiques balayant en deux dimensions cette fois)

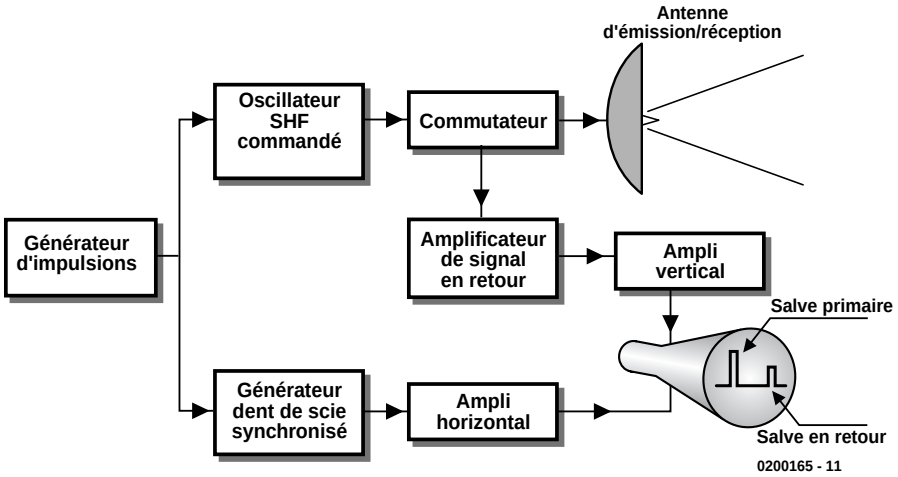


Figure 1 : Principe de fonctionnement des radars de première génération (1940).

Et voici l'effet DOPPLER…

En tant que tel, l'appareil décrit dans les lignes précédentes est incapable de donner la vitesse de déplacement de l'obstacle détecté, sauf à mesurer le mouvement de l'écho, mais la mesure ainsi obtenue est grossière.
Prenons un véhicule mobile, qui produit un signal sonore à fréquence constante (une voiture avec son moteur maintenu à régime constant, par exemple). Si vous montez dans la voiture, vous ne constaterez aucun changement dans la fréquence du bruit émis.
Si vous vous placez maintenant sur le bord de la route, et que vous écoutez passer la même voiture dans les mêmes conditions, vous constaterez que la hauteur (fréquence) du bruit du moteur augmente lorsque le véhicule se rapproche, pour diminuer lorsque le véhicule vous aura dépassé et s'éloignera (écoutez donc la retransmission d'un grand prix de F1 lors du passage des voitures).
Notons que ce phénomène est réciproque : si vous passez en voiture devant quelqu'un qui crie sur le bord de la route, vous remarquerez que la hauteur du son augmente en s'approchant de la personne, pour diminuer en s'éloignant.
Lorsque la distance entre la source du signal et le capteur reste constante, la hauteur du signal capté reste constante elle-aussi.
L'effet Doppler (découvert par le physicien du même nom) n'est ni plus, ni moins que cela, et s'exprime par la relation décrite en **figure 2** :

$$f_M = 2 * v * f_E * \cos(\alpha/c)$$

où

f_M est la fréquence du signal reçu

v la vitesse de déplacement du mobile

f_E la fréquence émise par l'émetteur

α l'angle formé entre l'émetteur et l'axe de déplacement du mobile

c la vitesse de propagation du signal dans le milieu ambiant (300 000 km/s pour les ondes radio, 340 m/s pour les ondes sonores)

On s'aperçoit donc que pour mesurer la vitesse de déplacement d'un mobile, il suffit juste d'envoyer vers lui un signal de fréquence fixe, et de mesurer la fréquence du signal reçu en retour.
C'est sur ce principe que sont basés les cinémomètres, plus connus sous le nom de *radars*, bien qu'ils n'aient pas grand chose à voir avec les systèmes décrits dans la première partie de l'article…
On notera en passant, que plus l'angle formé entre l'axe du faisceau et l'axe de déplacement du mobile est faible, plus la sensibilité du dispositif augmente. C'est pour cela que les antennes des cinémomètres sont orientées dans l'axe de la route, et non pas en travers ! C'est aussi pour cette raison que peu de radars sont capables de travailler dans les courbes, car l'angle formé entre le véhicule et le faisceau de mesure varie en permanence, ce qui est source d'erreur.

De la théorie à la pratique !

Maintenant que nous avons vu comment l'effet Doppler permet de mesurer la vitesse de déplacement d'un mobile, nous allons jeter un coup d'oeil sur les réalisations industrielles, rencontrées sur le bord des routes.
Le point de départ de tout cinémomètre (pour ne pas les appeler radars) est un générateur SHF, capable d'émettre un flux d'ondes dans une direction privilégiée. Le précédent chapitre a montré que la sensibilité de l'appareil dépend directement de la fréquence du faisceau de mesure. La fréquence exacte dépend du constructeur, mais se situe le plus souvent entre 2 GHz et 15 GHz. Selon les appareils, l'oscillateur SHF peut être formé d'un oscillateur à diode Gunn à cavité résonante ou d'un oscillateur transistorisé avec un amplificateur de sortie. La puissance émise par ces oscillateurs n'est pas très élevée (souvent moins de 100 mW), mais l'émission est renforcée par l'ajout d'une antenne qui privilégie une direction de diffusion.
Le capteur d'onde réfléchie est le plus sou-

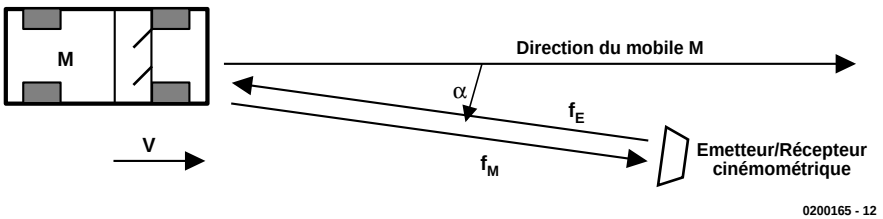


Figure 2 : équation de Doppler.

vent basé sur une diode Schottky, installée dans le foyer d'une antenne (le plus souvent, c'est la même antenne qui sert à l'émission et à la réception), qui joue le rôle de mélangeur avec l'onde émise et de détecteur.
Le signal en sortie du capteur est amplifié, puis mis en forme par un circuit de conditionnement analogique, avant d'être injecté dans le circuit de mesure, qui n'est autre qu'un fréquencemètre.
En sortie du fréquencemètre, un circuit à microprocesseur se charge de la mise à l'échelle des mesures, de leur affichage, ainsi que de contrôler si la vitesse mesurée dépasse un seuil préréglé, pour prévenir les forces de l'ordre restées près de l'appareil du passage d'un contrevenant, voire déclencher un appareil photographique accompagné de son flash…
Bref, le principe de base d'un cinémomètre à hyperfréquence (**figure 3**) n'est pas des plus complexes (nous disons bien : le principe. La réalisation pratique n'est pas à la portée de tout le monde, notamment en ce qui concerne la tête hyperfréquence).

Et ça marche bien ?

Maintenant que nous savons comment ça marche, il est légitime de s'interroger sur la fiabilité des mesures de ces appareils. Soyons clairs : il ne s'agit pas ici d'amener les lecteurs à commettre des excès de vitesse et à adopter des comportements irresponsables, mais d'envisager le problème sous un angle technique, afin de mettre en évidence les limites des cinémomètres à hyperfréquence. Ceci nous permettra aussi de faire la part entre les faits avérés et les 'ragots', colportés un peu partout par des individus peu au fait de la technologie électronique. Plutôt que de rentrer dans des considérations techniques obscures, nous répondons ici aux interrogations les plus courantes.

Fonctionnement sous la pluie ou dans le brouillard : contrairement à une idée largement répandue, les radars fonctionnent parfaitement sous la pluie ou dans le brouillard (ce n'est pas pour rien qu'on guide les avions au radar par mauvais temps !). Dans le cas de la pluie, celle-ci tombe verticalement (enfin, c'est comme cela chez nous !), donc à la perpendiculaire du faisceau, donnant un effet Doppler nul (cos $90° = 0$, donc $f_M = 0$). Une grande quantité de pluie tombant en diagonale (rafales de vent) peut toutefois altérer le rapport signal/bruit du détecteur, et l'empêcher de fonctionner correctement. Dans ce cas, les mesures sont purement et simplement rejetées par le processeur final.
Quant au brouillard, comme il ne se déplace pas dans le faisceau (ou trop lentement), il est simplement invisible pour le détecteur et ne perturbe en rien les mesures.

Distance de mesure : la distance à laquelle un radar est capable de mesurer la vitesse d'un véhicule dépend de deux facteurs : la puissance apparente d'émission de l'oscillateur SHF, et la sensibilité du détecteur. Nous avons vu que les puissances d'émission étaient généralement faibles, mais l'ajout d'une antenne permet d'augmenter la puissance rayonnée apparente. Au niveau du détecteur, le principal problème est posé par le rapport signal/bruit, qui n'est pas des meilleurs avec les diodes Schottky. Ici encore, on peut sérieusement augmenter la sensibilité en faisant appel à des antennes. Alors que les premiers radars étaient limités à une détection à une vingtaine de mètres, les nouveaux modèles équipés de détecteurs ultra-sensibles sont capables d'effectuer des mesures à plusieurs centaines de mètres, donc bien avant que l'on puisse les apercevoir depuis la voiture !

Vitesse de réaction : comme tout appareil basé sur une mesure de fréquence, les cinémomètres ont un temps de réaction, nécessaire pour l'établissement de la mesure. De plus, la plupart des appareils ne font pas une, mais plusieurs mesures, afin de rejeter d'éventuelles erreurs. Les modèles les plus anciens demandent environ une demi-seconde avant de confirmer la mesure. Les modèles actuels réagissent en moins d'un dixième de seconde, ce qui laisse peu de chance au conducteur peu respectueux des limites de vitesse pour réagir après avoir vu le cinémomètre, avant que celui-ci ne délivre son verdict. Sachez que certains cinémomètres sont maintenant équipés de DSP (processeur de signaux numériques), qui utilisent un algorithme spécifique de mesure de fréquence, avec un temps de réponse très court, donnant des mesures extrêmement rapides.

Emission permanente : contrairement à ce que pourrait laisser croire la présentation théorique du début d'article, un radar n'a pas besoin de laisser fonctionner son oscillateur en permanence. Il lui suffit de l'activer suffisamment longtemps avant de procéder aux mesures pour le laisser se stabiliser. Ainsi, la plupart des radars actuels fonctionnent par rafale aléatoire ou ne se mettent en service qu'à l'approche d'un véhicule.

Discrimination : lorsque plusieurs véhicules franchissent le faisceau radar à des vitesses différentes, il en résulte un signal Doppler formé de plusieurs composantes fréquentielles. La plupart des radars actuels ne peuvent pas discriminer ces composantes et rejettent la mesure qu'ils considèrent comme erronée. Il existe toutefois de nouveaux systèmes d'analyse, basés sur des DSP (l'auteur de cet article a travaillé sur l'un de ces systèmes), capables de mesurer simultanément les vitesses de plusieurs véhicules franchissant le faisceau. Dans ce cas, seul un véhicule masqué par un autre peut encore échapper au cinémomètre.
Bref, les cinémomètres sont devenus des instruments d'une précision et d'une fiabilité redoutables (au moins pour ceux qui considèrent les limitations de vitesse comme réservées aux autres conducteurs), qu'il est extrêmement difficile de perturber !

Un soupçon d'illégalité…

L'être humain, et plus particulièrement l'homo automobilis, est ainsi fait que, dès qu'on lui impose une restriction, il tente de la contourner par tous les moyens. Le domaine des cinémomètres n'y fait pas exception, et nombre de techniciens (plus ou moins doués…) se sont attelés à la mise au point de systèmes susceptibles de contrer ces appareils.
Avant tout, soyons clairs : dans la majorité des pays européens, la détention, et à plus forte raison, l'utilisation de dispositifs destinés à empêcher les mesures par cinémomètres sont **formellement interdites**. Le non-respect de ces dispositions légales peut coûter très cher à leur auteur (annulation du permis de conduire, saisie du véhicule, poursuite en justice, etc.)
Malgré tout, certains conducteurs pensent que le jeu en vaut la chandelle, et n'hésitent pas à prendre le

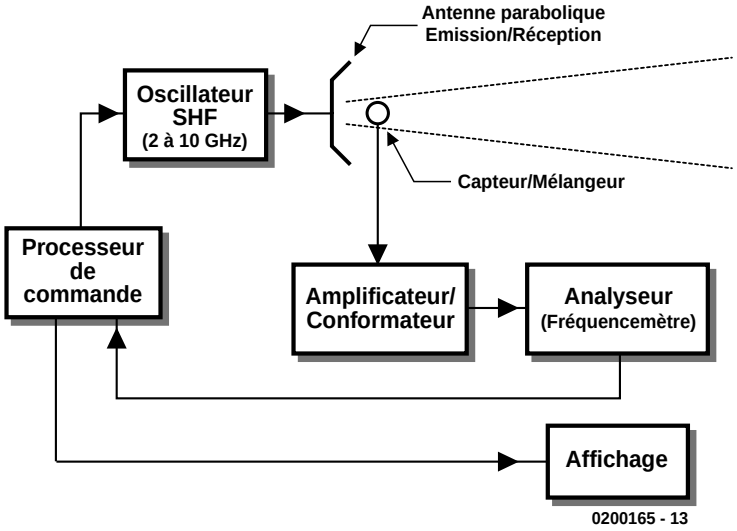


Figure 3 : Principe de base d'un cinémomètre hyperfréquence.

risque de s'équiper en appareils de contre-mesure.
Il existe en gros deux catégories d'appareils « anti-radars » : les brouilleurs et les détecteurs.
Les brouilleurs sont tout simplement de petits oscillateurs SHF, chargés de renvoyer vers le cinémomètre un 'faux' faisceau de mesure, qui va perturber le circuit de mesure et l'empêcher de fournir des informations cohérentes à l'analyseur de fréquences. Outre le fait que ces engins sont d'une efficacité toute relative (la plupart des radars ont des circuits accordés peu sensibles aux fréquences perturbatrices : il faut donc que le brouilleur ait une fréquence proche de celle du cinémomètre, et chaque modèle utilise une fréquence qui lui est propre…), les radars ont des circuits capables de détecter de telles perturbations, et de prévenir les forces de l'ordre. Les brouilleurs sont donc surtout le meilleur moyen de se faire prendre !
Les détecteurs, eux, sont de simples récepteurs SHF. Ils sont donc par définition indétectables. Largement répandus aux USA (où ils sont autorisés), on peut les trouver facilement en vente par correspondance sur Internet, voire en vente libre dans certains pays d'Europe (où leur vente est légale, mais pas leur utilisation !). Bien souvent, ce sont des circuits assez simples, formés d'un simple détecteur de micro-ondes, suivi d'un comparateur déclenchant une petite alarme. Bref, rien de bien complexe (leur prix de vente ayant plutôt tendance à faire croire le contraire !)

Il est assez simple de concevoir un détecteur à large bande, capable de détecter un faisceau d'onde compris par exemple entre 2 et 10 GHz, ce qui couvre la majorité des radars actuels. Toutefois, si vous avez la malchance de tomber sur un cinémomètre qui utilise un oscillateur hors de la gamme de mesure de votre détecteur (ou un modèle à mesure optique par laser), ce dernier n'est d'aucune utilité.
Deuxième problème : pour qu'un détecteur détecte quelque chose, il faut qu'il y ait quelque chose à détecter (logique, non ?). Les anciens radars émettaient en permanence, ce qui simplifiait les choses (la propagation des ondes facilitant encore le travail du détecteur). Mais les nouveaux modèles n'émettent plus que par intermittence, de façon aléatoire ou par rafales très brèves, ce qui diminue très nettement les chances de détection. Certains modèles (comme le Mesta208, largement répandu en France) sont même encore plus vicieux, car ils ne se mettent en marche que lorsqu'un véhicule passe à portée de détection. En effet, ce cinémomètre surnommé par les automobilistes 'la petite boule verte' en raison de sa forme et de sa couleur quelque peu… militaire est équipé d'un détecteur optique situé sur le dessus de l'appareil, qui 'voit' littéralement les véhicules passer dans son faisceau. Dès que ça bouge devant l'appareil, celui-ci s'active immédiatement.
Troisième problème : quand un détecteur de radar s'active, c'est qu'il « sent » un faisceau radar. C'est donc que le radar fait son travail au même moment. Bref, quand le détecteur réagit (le conducteur est censé freiner à ce moment pour perturber les mesures), il est déjà bien souvent trop tard (n'oubliez pas que le temps de réaction des humains à un stimulus externe est de l'ordre d'une demi-seconde –dans cet intervalle de temps, la plupart des radars ont eu le loisir de faire quatre ou cinq mesures). La difficulté de détection est encore accrue par le fait que les faisceaux SHF sont très directifs et étroits, ce qui diminue encore la zone de détection potentielle. Certains utilisateurs de détecteurs ont toutefois découvert qu'ils pouvaient contourner cette difficulté grâce au fait que les faisceaux se diffusent au contact des autres véhicules.
Dernier problème : la plupart des radars actuels sont capables d'effectuer des mesures en approche (par devant) aussi bien qu'en éloignement (par derrière). Or, de nombreux détecteurs de radars ont une sensibilité restreinte à une direction particulière. Il faudrait donc équiper chaque véhicule avec un détecteur à l'avant et un autre à l'arrière pour envisager tous les cas !

Conclusion

Comme dit le proverbe, à force de jouer avec le feu, on finit par se brûler les doigts. Comme cet article l'a démontré, les cinémomètres sont devenus des instruments fiables, très difficiles à perturber et quasiment impossibles à détecter.
Savoir détecter un faisceau SHF est une chose, le faire de telle façon que l'on puisse contrer les mesures d'un cinémomètre est en une autre. Bref, même si la plupart des détecteurs vendus à l'heure actuel fonctionnent effectivement et savent signaler la présence d'un cinémomètre, ce n'est pas pour autant qu'ils soient 'utilisables', le radar ayant de toute façon de grandes chances de réagir bien avant le conducteur en infraction.
Même si les cinémomètres ne sont pas toujours placés aux endroits où ils se justifieraient le plus (ce qui ne fait malheureusement que contribuer à leur réputation de « pompes à euros »), il est à notre avis plus intelligent de respecter les limitations de vitesse (c'est aussi ça le respect des autres), que d'essayer de contourner à tout prix la loi.
À bon entendeur…

(020165)

USB pour tous

Guide d'« auto-développement » (1)

par Markus Müller et Christian Ehmer

La connexion de matériel à l'Universal Serial Bus (USB) recèle de nombreux avantages… et obstacles. Le plan que nous vous fournissons vous permettra d'éviter les obstacles parsemant la voie du développement USB.

Le matériel développé « maison » pour un PC est connecté en règle générale à l'interface sérielle RS232. Cette interface vieille de plusieurs décennies est extrêmement bien documentée et d'une utilisation simple. Elle ne présente bien entendu pas que des avantages : il n'est possible de raccorder un appareil à l'interface RS232 qu'avant l'initialisation du PC et, d'autre part, il est difficile de lui soutirer (un peu) d'énergie pour alimenter le circuit de l'application.
Ces inconvénients disparaissent avec l'interface USB. Mais, dans la plupart des cas, et cela s'applique aussi aux bricoleurs sur PC les plus endurcis, les utilisateurs ne connaissent pas ou connaissent mal le bus USB. Mais, après avoir lu cette contribution, vous serez en mesure de connecter l'appareil USB que vous avez réalisé à un PC. Le clou : presque tout est disponible gratuitement sur Internet.
La réalisation d'un circuit périphérique pour USB comporte 3 étapes.

1. Celle du matériel et d'un programme tournant dans la puce USB
2. Celle du programme communicant avec la puce USB et affichant ses réactions.
3. Celle du pilote (fichier SYS) établissant la communication entre le port USB et le programme.

Nous procéderons donc pas à pas. Mais voyons donc tout d'abord ce qu'on peut dénicher sur le marché. Si de nombreux fabricants proposent des puces USB, leur nombre est inversement proportionnel au niveau d'assistance. Une exception, heureusement : la société Cypress Semiconductor (www.cypress.com).

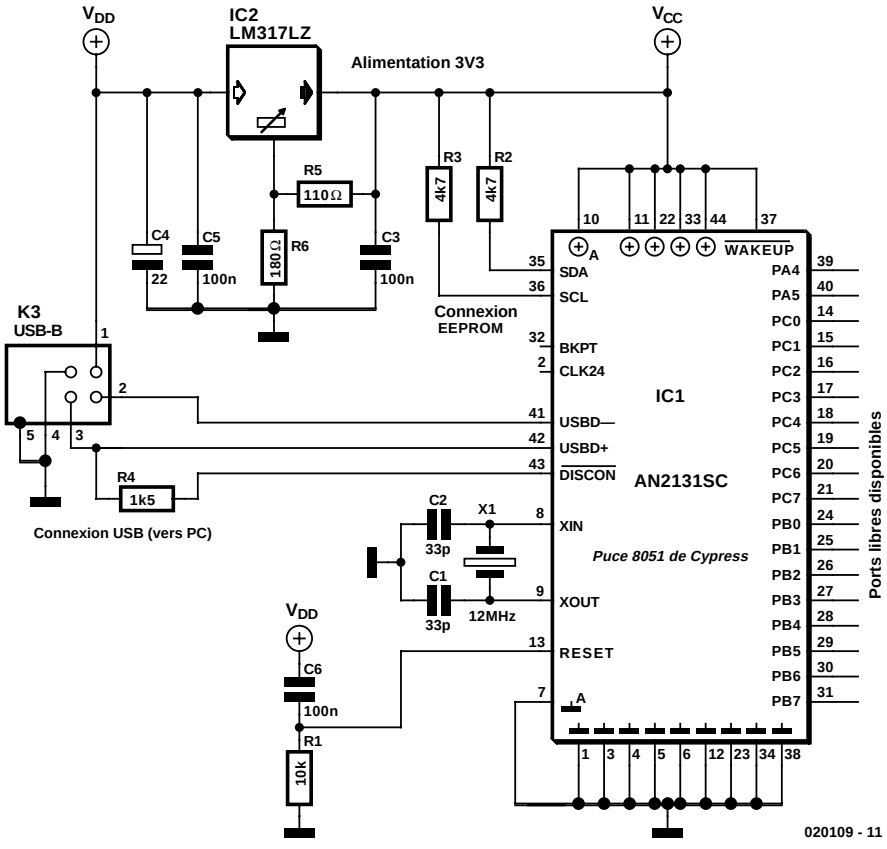


Figure 1. Configuration minimale de fonctionnement de la puce Cypress AN2131SC.

Circuit de base pour réalisateur USB

Cypress fabrique le contrôleur standard USB AN2131SC que nous utilisons. Les caractéristiques techniques de ce microcontrôleur sont données dans l'encadré ci-contre et vous pouvez charger la fiche de données sous :

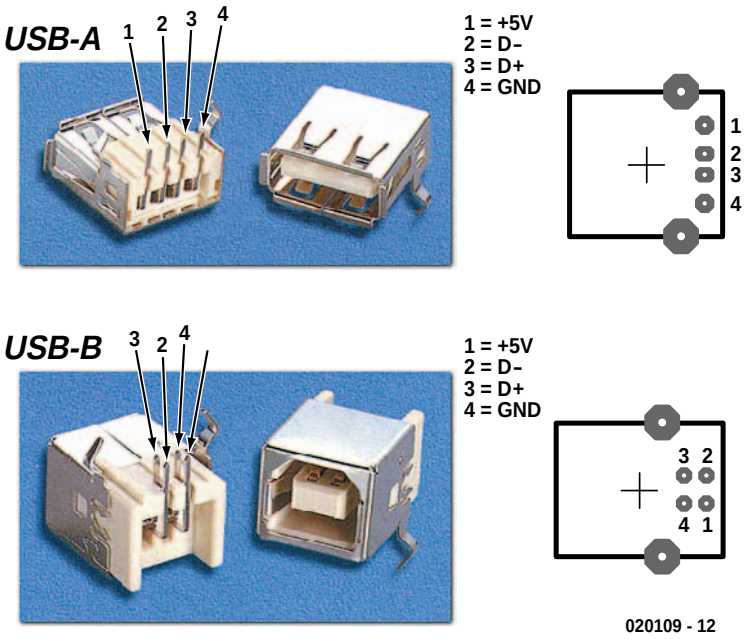


Figure 2. Brochage du connecteur USB.

www.cypress.com/cfuploads/img/products/AN2131SC.pdf.

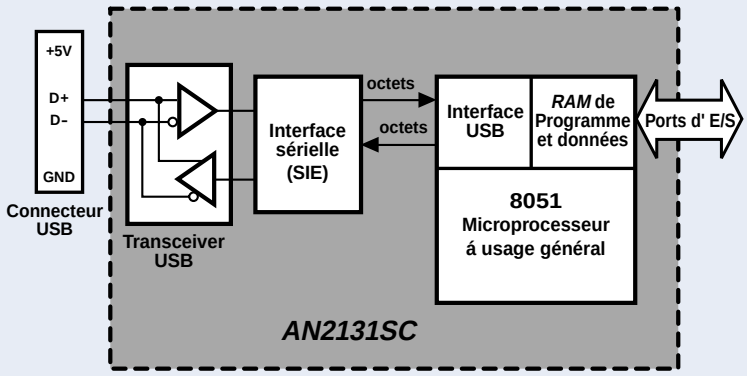
Comme le montre la **figure 1**, le connecteur mâle est, avec le contrôleur USB, le deuxième composant spécial du circuit de base. Il en existe 2 versions : type A et type B. Le type A se trouve toujours du côté du PC ou du concentrateur, tandis que le type B fait toujours partie de l'appareil communiquant avec le PC. La prise de type A peut fournir du courant tandis que la prise de type B en absorbe. Notre circuit ne fait donc appel qu'à des prises de type B (**figure 2**). Les câbles USB sont toujours reliés 1:1. Le bus USB peut fournir 500 mA au plus. C'est pourquoi la puce Cypress doit tout d'abord transmettre la valeur du courant désiré : en effet, si le PC ne dispose pas de réserve de courant suffisante, il met de lui-même l'appareil hors-circuit.

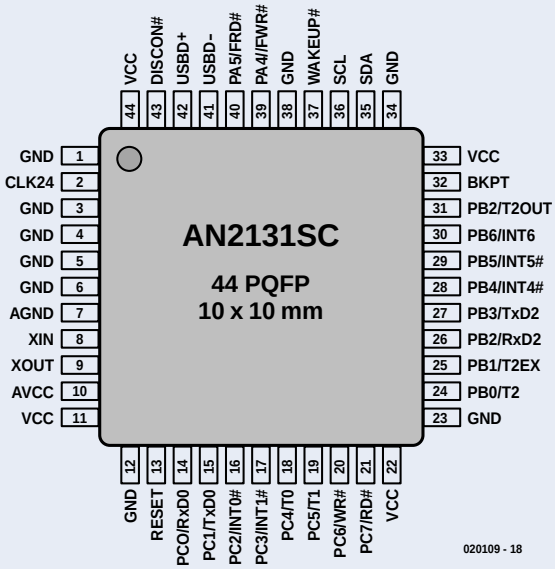
La puce AN2131SC envoie automatiquement la valeur « 100 mA » par sa logique de communication interne. Si le matériel nécessite un courant plus élevé, il est nécessaire d'écrire

Le contrôleur USB

Le microcontrôleur sur 1 puce AN2131SC contient un noyau processeur 8051 avec une interface Full-speed USB. Le contrôleur dispose de 8 Koctets de RAM, de 3 compteurs/temporisateurs, de 2 interfaces sérielles et d'entrées d'interruption. On dispose en tout de 18 lignes E/S librement programmables.

Une EEPROM sérielle peut être raccordée comme mémoire de programme, ou un programme pour l'exploitation normale peut être chargé dans la RAM par le bus USB. D'autres composants externes sont pratiquement superflus. Le AN2131 est alimenté par une source de 3, V, la fréquence d'horloge du port USB est dérivée à partir de l'oscillateur de 12 MHz. Un doubleur de fréquence engendre 24 MHz pour la CPU. Le jeu optimisé d'instructions permet de traiter les instructions machines à une vitesse quadruple.





Les temporisateurs 0 et 1 comportent tout deux 8 bits, le temporisateur 2 comporte 16 bits. Le signal à l'entrée du port destinée à un compteur/temporisateur ne doit pas dépasser la fréquence de 2 MHz. L'interface sérielle peut transmettre à 9 600 bauds quand le temporisateur 0 ou 1 est utilisé. On grimpe à 38 400 bauds avec le temporisateur 2. Pour atteindre un nombre de bauds encore plus élevé, il faut appliquer une autre fréquence aux entrée temporisateur.

Le composant peut réagir de façon appropriée à 5 interruptions différentes, actives à niveau haut ou bas. Les lignes I^2C doivent être placées à 3,3 V par une résistance de 4,7 kΩ et l'entrée de réinitialisation doit être munie d'un réseau POR (Power-On Reset). Il suffit alors de brancher le tout au port USB pour que le PC ne puisse plus se retenir d'installer un pilote d'appareil.

La deuxième version du contrôleur rend superflues les mises à jour du logiciel car le programme correct est toujours chargé dans la puce USB. Grâce à la re-numération intégrée, l'interface USB peut s'identifier auprès du système d'exploitation avec de nouveaux VID et PID (Vendor ID/Product ID).

soi-même le logiciel de communication USB.
On peut raccorder une EEPROM sérielle aux lignes SDA et SCL de la puce Cypress. L'EEPROM peut contenir uniquement l'ID, ou une ID et un programme. Si l'EEPROM ne contient pas de programme, la partie USB et le CPU sont complètement découplés l'un de l'autre. L'ensemble des communications s'effectue par la RAM. Des instructions du pilote USB permettent de placer le CPU en mode de réinitialisation. Un programme est finalement chargé à partir de l'adresse 0 de la RAM. Le bit de réinitialisation est effacé et le CPU exécute le programme à partir de l'adresse 0. Le pilote USB permet d'avoir accès sans restriction à toute la RAM, même lorsque la puce 8051 est en fonctionnement. On n'a même pas besoin de programmer une seule ligne de code : la partie USB du contrôleur se charge de la tâche.
Cette configuration de base constitue un bon point de départ pour réaliser votre propre circuit. Faites toutefois attention à l'attribution des ports. Les connexions sont programmables à volonté ; on peut y raccorder des temporisateurs, des lignes d'interruption et même 2 interfaces sérielles. Si vous incluez des puces munies d'une sortie d'interruption, n'oubliez pas de l'utiliser. Même la programmation en C en sera considérablement facilitée.

Outils du Net

Avant de donner le départ de la programmation, examinons quelques programmes et outils se trouvant sur Internet :

1. Le **kit de développement EZUSB** de **Cypress** comporte un compilateur C de Keil ainsi que des modèles de code aidant à réaliser des pilotes USB. Pour le trouver, introduisez la désignation de la puce dans le moteur de recherche local, puis allez au kit de développement AN2131-DK001 (**figure** 3) et chargez les
   EZ-USB Family Complete Tools
   qui, dans leur version actuelle (début juillet) V2.52.701, ne font pas moins de 63,153 Moctets.
2. Téléchargez aussi le manuel (**EZ-USB Technical Reference Manual**) (1,48 Moctets) à partir de la même page.
3. Il faut encore télécharger **BinTerm,** (787 Koctets) disponible gratuitement sur la page d'accueil de l'auteur www.mmvisual.de. BinTerm permet de charger par l'USB dans la puce Cypress les programmes créés en C et de commander la plage de RAM tout entière. BinTerm comporte les pilotes d'appareils nécessaires.

Installez le kit de développement et ne changez surtout pas les chemins d'installation. Il faut choisir l'option personnalisée (*Custom*) au début de l'installation pour que le compilateur C de Keil soit installé. Lors de l'installation du compilateur C de Keil uVision2, ne pas modifier les chemins d'installation pour que les exemples programmés avec des chemins absolus continuent à fonctionner.
Vous trouverez des conseils supplémentaire sur le programme Keil sous www.keil.com. Attention : la démonstration d'évaluation sur la page d'accueil de Keil ne fonctionne pas correctement avec la puce Cypress. Le code produit par les autres compilateurs, Rigel51 par exemple, est environ 7 fois plus gros que celui du compilateur de Keil.

Figure 3. Page de téléchargement du kit de développement.

Un grand nombre de réalisateurs de logiciel ne jurent que par le langage assembleur. Il est très incommode de tout programmer en assembleur, mais il est parfaitement possible de ne coder ainsi que certains sous-programmes et de les intégrer au code C. Le code assembleur est lié au type de CPU, tandis que le code C peut être utilisé avec n'importe quel CPU moyennant un très petit nombre de modifications.
BinTerm ne doit pas être installé : on crée sous *C:\Programme* un nouveau répertoire nommé *BinTerm* et on y copie purement et simplement tous les fichiers. Le programme **BinTerm.exe** est associé au bureau par un raccourci. Le bouton droit de la

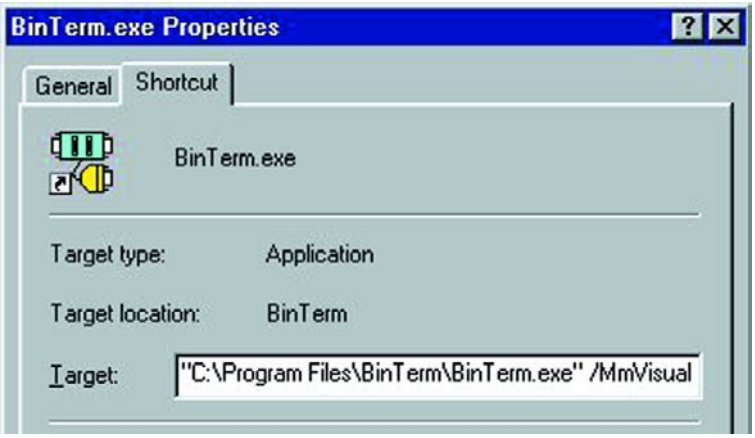


Figure 4. Introduction du paramètre de démarrage supplémentaire.

souris permet de modifier le raccourci sous *Propriétés* :
Ajoutez le paramètre de démarrage */MmVisual* dans le champ *Destination* pour qu'un onglet supplémentaire *USB-Test* apparaisse dans le programme BinTerm (**figure 4**). Si votre version de Windows ne le permet pas, vous pourrez ajouter ce paramètre par le biais du processus *Démarrer Exécuter* à la fin de la ligne de lancement du programme BinTerm.

jet. Choisissez *Close* pour fermer la boîte de dialogue.
Une nouvelle rubrique, *cMain.c*, est apparue sous *Source Group 1*. Ouvrez l'éditeur en cliquant 2 fois. L'éditeur reconnaît la nature du code (source C) et met la syntaxe en évidence. La distinction effectuée par le compilateur entre majuscules et minuscules est une caractéristique particulièrement irritante du langage C.

disquette contenant les pilotes de l'appareil USB. Lorsque le nouvel appareil USB est raccordé, le système d'exploitation installe un pilote, permettant ainsi au logiciel BinTerm de communiquer avec la puce USB Cypress.
Désactivez la communication interne entre BinTerm et l'appareil USB en activant la coche *stop query* (Stopper l'interrogation) sous *Controlling BinTerm* (Commande de BinTerm). Le bouton à 3 points à côté du bouton *send new program* (Transmettre un nouveau programme) permet d'ouvrir le fichier *Ram-*

```
xdata unsigned char BYTE_IN     _at_    0x0100; // Octet, en entrée (Adresse RAM)
xdata unsigned char BYTE_OUT    _at_    0x0101; // Octet, qu'il va falloir écrire (Adr. RAM)
void main()
{  while (1)     // Boucle sans fin
   {     BYTE_OUT = BYTE_IN;     // Recopier Octet de BYTE_IN vers BYTE_OUT
   }
}
```

Code C pour le AN2131SC

Et c'est parti ! Démarrons le programme *Keil uVision2*, l'environnement C pour le 8051. Un nouveau projet sera placé sous *Project > New Project*. Introduisez *RamTest* comme nom de fichier et confirmez en cliquant sur OK. L'étape suivante consiste à choisir le processeur (*Select Device for Target*) : *Cypress Semiconductor > EZUSB AN21XX > OK*. La fenêtre de gauche est celle du projet (*Project Window*) dans laquelle le programme a placé la cible *Target 1* avec *Source Group 1*. Ouvrez le menu déroulant dans *Target 1* avec le bouton droit de la souris, puis cliquez sur *Options for Target Target 1*. Le dialogue d'entrée convivial affiché permet de positionner toutes les options de compilation et d'édition de liens. Choisir la coche *Create HEX-File* dans l'onglet *Output*. Conclure le dialogue en cliquant sur OK.
Il faut créer un nouveau fichier vierge au moyen de l'option de menu *File > New* avant de pouvoir écrire le code. Enregistrez ce fichier sous le nom de *cMain.c* dans le répertoire de projet utilisé au moyen de *File > Save As*. Et, pour terminer, choisissez *File > Close* pour fermer le fichier.
Le fichier se trouve à présent dans le projet, mais n'y est pas encore attaché. Choisissez l'option *Source Group 1* dans *Project Window* et cliquez sur le bouton droit de la souris ; choisissez *Add Files to Group Source Group 1*. Choisissez *Add* pour ajouter le fichier *cMain.c* proposé au pro-

Créons à présent un petit programme pour la puce Cypress. Écrivez les lignes suivantes dans l'éditeur :
F7 permet de générer le projet. Le compilateur devrait afficher *0 Errors, 0 Warnings* si le code a été entré sans fautes de frappe.

Mode de fonctionnement du programme

Les 2 premières lignes servent é créer 2 variables attribuées à un emplacement fixe en mémoire. Le processeur lance le programme par la fonction *main*. La boucle *while(1)* { } se répétera sans fin, ou tout au moins jusqu'à la prochaine réinitialisation. Le sous programme a reçu la tâche particulièrement importante de copier le contenu de l'adresse 0x100 en 0x101.
La génération lancée avec F7 a créé plusieurs fichiers dans le répertoire du projet. Le fichier avec l'extension *HEX* peut être chargé dans la puce Cypress et démarré en se servant de BinTerm.
Lancez *BinTerm* avec l'option */MmVisual* que vous avez installée précédemment sur le bureau. Choisissez l'onglet *USB-Test*. Cette fonction du programme permet de commander complètement la puce Cypress. Raccordez un appareil USB équipé d'une puce Cypress. Le nom d'appareil du matériel doit être choisi selon que vous utilisez un espion USB ou une puce Cypress sans EEPROM.
BinTerm peut à présent créer une

Test.hex engendré par le compilateur et de le charger dans la puce USB. Pour terminer, cliquez sur le bouton *start USB program* pour activer la puce Cypress.

Test du processeur 8051

Le contrôle de la plage de RAM offre, comme le montre la **figure** 5, la possibilité d'entrer des nombre et une représentation en hexadécimal. *write in the address* (Écrire dans une adresse) permet d'écrire directement un octet dans la RAM du processeur 8051 en fonctionnement. Dans cet exemple, nous écrivons l'octet 0x02 à l'adresse 0x100. Cliquez sur la touche d'envoi *send* pour écrire l'octet. *read from address* (Lire de l'adresse) permet d'indiquer l'adresse de départ à partir de laquelle les *Bytes* (ici 3 octets à partir de l'adresse 0x100) seront lus. L'option *auto-read every second* (Lecture automatique chaque seconde) met automatiquement l'affichage à jour.
Nous avons à présent chargé le petit exemple dans la puce Cypress et démarré. Comme vous le voyez, la puce fonctionne sans un accroc. L'octet est copié de l'adresse 0x100 à l'adresse 0x101. Mais comment se fait-il que *BinTerm* communique avec la puce Cypress, bien que l'on n'ait pas écrit une seule ligne de commande de l'interface USB, que l'on n'ait même pas positionné un seul registre spécialisé ?
La partie USB de la puce, assistée par le pilote Cypress, fonctionne de façon totalement autonome . Tout se passe comme si un second microcontrôleur gérait l'interface USB. Cette partie autonome a en outre toujours accès à toute la plage RAM du 8051. Le processeur peut même être réinitialisé et redémarré au moyen du registre 0x7F92. Chaque adresse RAM peut être écrite ou lue à n'importe quel moment. Nous avons tiré parti de

cette caractéristique dans l'exemple. Même si vous ne savez que vaguement ce que sont C et un microcontrôleur, vous venez de programmer le second avec le premier !
La page d'accueil de Keil vous permet de télécharger un nombre quasi illimité d'exemples de code. Mais rien ne vous empêche de chercher ailleurs sur Internet.

Mon Programme – tout aussi simple ?

Les exemples de programmes sont écrits en Delphi qui offre l'environnement de programmation pour usage professionnel le plus efficace et allie la convivialité de Visual Basic aux classes et aux vérifications du type de C. Delphi peut même être téléchargé gratuitement sur Internet à partir de www.borland.com. Delphi se trouve aussi souvent sur le CD du magazine *Der Entwickler*.

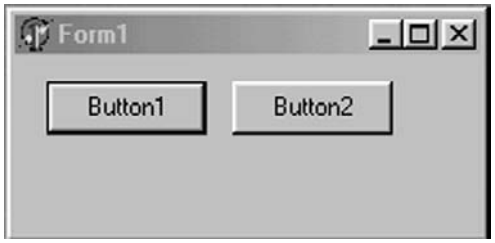


Figure 6. Form1 avec 2 boutons.

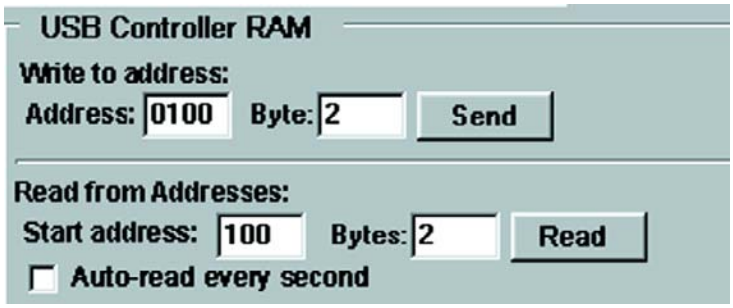


Figure 5. Contrôle de la plage de RAM.

Mais revenons à nos moutons : vous devez tout d'abord connaître la hiérarchie de l'appareil USB jusqu'au niveau de notre programme. Nous avons un appareil raccordé à l'interface USB. Le codage des résistances des lignes de données D+ et D- permet au PC d'identifier la vitesse à laquelle fonctionne l'appareil USB. Le pilote de Windows demande finalement à l'appareil le nom du vendeur et l'ID du produit. Si Windows connaît déjà cet appareil, le pilote spécifique du fabricant l'appareil est chargé, sinon Windows cherche un pilote approprié dans sa base de données et propose de l'installer. Le pilote spécifique ne communique jamais directement avec l'appareil USB, mais seulement avec le pilote Windows. Windows peut gérer plusieurs appareils par l'entremise d'un concentrateur (HUB) raccordé à l'interface USB. Le programme d'application ne peut établir une communication avec l'appareil USB que par l'entremise du pilote spécifique du fabricant. Les commandes Windows *CreateFile* et *DeviceIoControl* dans *Kernel32.dll* sont utilisées dans ce but.

Pour créer une nouvelle application sous Delphi, utilisez *Filer >New > Application*. Un formulaire vierge (Form1) apparaît. Placez-y 2 boutons de type *TButton*. Ajoutez le type suivant dans le programme après la déclaration « *Uses* » et juste avant le premier bloc « *Type* » :

```
type _VENDOR_REQUEST_IN = record
   bRequest : Byte;
   wValue : Word;
   wIndex : Word;
   wLength: Word;
   direction : Byte;
   bData : Byte;
end;
```

Intercaler le code suivant, après la ligne comportant le mot « implementation » :

```
Function WrRAM(Adr: Word; Dat: Byte): Boolean; // Ecrire un octet dans l'adresse RAM
var USBDeviceHandle: THandle;
   USBTemplateHandle: THandle;
   nBytes: DWord;
   MyRequest: _VENDOR_REQUEST_IN;
  USBError: Boolean;
   pDevice: PChar;
begin
   pDevice := PChar('\\.\ezusb-0'); // Avec BinTerm: '\\.\binterm-0'
   USBError := False;
   myRequest.bRequest := $A0;
   myRequest.wValue := Adr;
   myRequest.wIndex := 0;
   myRequest.wLength := 1;
   myRequest.direction := 0;
   myRequest.bData := Dat;
```

```
   USBTemplateHandle := 0;
   USBDeviceHandle := CreateFile (pDevice, Generic_write, File_Share_write, nil, open_existing,
                                  0, USBTemplateHandle);
   If USBDeviceHandle = INVALID_HANDLE_VALUE Then
        USBError := True;
   Result := DeviceIoControl(USBDeviceHandle, $00222014, @myRequest, 10, nil, 0, nBytes, nil);
   CloseHandle (USBDeviceHandle);
end;

function RdRAM(Adr: Word): Byte; // Lire un octet présent à l'adresse RAM
var USBDeviceHandle: THandle;
   USBTemplateHandle: THandle;
   nBytes: DWord;
   MyRequest: _VENDOR_REQUEST_IN;
   Buffer: Array [1..2] of Byte;
   USBError: Boolean;
   pDevice: PChar;
begin
   pDevice := PChar('\\.\ezusb-0'); // Avec BinTerm: '\\.\binterm-0'
   USBError := False;
   USBTemplateHandle := 0;
   myRequest.bRequest := $A0;
   myRequest.wValue := Adr;
   myRequest.wIndex := 0;
   myRequest.wLength := 1;
   myRequest.direction := 1;     // Read
   myRequest.bData := $00;
   USBDeviceHandle := CreateFile (pDevice, Generic_write, File_Share_write, nil, open_existing,
                                  0, USBTemplateHandle);
   If USBDeviceHandle = INVALID_HANDLE_VALUE Then
        USBError := True;
   DeviceIoControl(USBDeviceHandle,$00222014,@myRequest,10,@Buffer,SizeOf(Buffer),nBytes,nil);
   CloseHandle (USBDeviceHandle);
   RdRAM := Buffer[1];
end;
```

Il faut ensuite effectuer un double clic sur la touche 1 et ajouter le code suivant :

```
procedure TForm1.Button1Click(Sender: TObject);
begin
   WrRAM($100, 1);
end;
```

On fera de même pour la touché 2 :

```
procedure TForm1.Button2Click(Sender: TObject);
begin
   Button2.Caption := IntToHex(RdRAM($101), 2);
end;
```

Fonctionnement du programme : L'appareil USB est piloté par le programme *RamTest.hex*. Si vous avez retiré entre-temps la fiche de l'appareil, rechargez le programme au moyen de *BinTerm*. *BinTerm* peut tourner comme tâche de fond qui interroge la RAM pour observer et contrôler celle-ci. *BinTerm* ne bloque pas l'appareil USB.

Lancez le programme Delphi avec F9 à partir du compilateur. Cliquez sur le bouton *Button1* pour écrire l'octet 1 à l'adresse 0x100 de la RAM de l'appareil USB. Le bouton *Button2* permet de lire et d'afficher l'adresse 0x101 (**figure 6**).

Données internes de l'appareil USB

La fonction *WrRAM* permet d'écrire des données dans la RAM de la puce Cypress. *RdRAM* lit un octet à l'adresse indiquée. Le nom de l'appareil (correct dans le cas du fonctionnement sans EEPROM) est sauvegardé dans la variable *pGeraet*. Si un espion des données USB de *BinTerm* est utilisé, remplacez le nom *'\\.\ezusb-0'* par *'\\.\binterm-0'*. *CreateFile* crée un fichier virtuel de liaison avec le pilote de l'appareil. La commande *DeviceIoControl* permet de communiquer directement avec le pilote de l'appareil. Un enregistrement (*Record*) de type *_VENDOR_REQUEST_IN* sert de contrôle de déroulement. *CloseHandle* termine la liaison avec le pilote de l'appareil. Comme la liaison est fermée après chaque demande, il est possible de simuler un accès partagé du matériel par plusieurs programmes – l'appareil n'en saura rien.

Par ailleurs, pour le cas où vous réécririez les fonctions ci-dessus, le tout fonctionne aussi en Visual Basic ou C++.

(020109-1)

La seconde partie de cet article sera dévolue à la création d'un pilote d'appareil USB. Point n'est besoin pour cela de connaissances détaillées de C !

Tube Box

Des diodes pour un son « tube » plus vrai que nature

Projet : Manfred Radler

Le module « Tube Box » tire d'un amplificateur à semi-conducteurs sa sonorité « tube » caractéristique, telle que se doit de l'avoir tour son de guitare digne de ce nom.

Les amplificateurs à tube pour guitare que l'on utilise sur scène ou en studio tirent leur sonorité caractéristique de l'entrée en écrêtage des tubes lorsqu'ils sont confrontés à une surmodulation (niveau de signal trop important). La situation qui, dans le cas d'un amplificateur à semi-conducteurs, se traduit par une sonorité insupportable (et peut même en entraîner un trépas prématuré), est, dans le cas d'un amplificateur à tubes, le mode de fonctionnement standard. Ceci présente cependant un inconvénient : il faut, si l'on veut faire entrer un amplificateur à tubes en surmodulation, tourner le bouton de volume en butée, ce qui peut être très gênant voire insupportable pour les autres membres du groupe voire les voisins du dessous (ou du dessus...)

Grâce au Tube Box, module qui vient s'intercaler entre la guitare et l'amplificateur à tubes, ceci n'est plus nécessaire. Ce minuscule appareil produit la distorsion recherchée du signal d'entrée de l'amplificateur. Le réglage de notre Tube Box fait appel à quelques potentiomètres et une paire d'interrupteurs, l'ensemble de ces organes offrant des possibilités de réglage très efficaces et étonnamment universelles, de sorte que ce « générateur de distorsion » convient à pratiquement l'ensemble de la bande recouverte par la musique de guitare, du blues rond et chaud au Heavy Metal « crissant ».

En amont et en aval des diodes

Le coeur de notre électronique, dont on retrouve le schéma en **figure 1**, la source de distorsion proprement dite, repose sur IC2.A. Les deux paires de diodes montées en tête-bêche, D1 à D4, sont à l'origine de la distorsion. Ces diodes produisent un écrêtage du signal, moins brutalement que ne le feraient des amplificateurs à semi-conducteurs, mais tout en douceur tout comme on en a l'habitude avec les étages à tube. Lorsque l'amplitude du signal devient trop importante, les LED D5/D6 se mettent également de la partie et réduisent le gain de l'amplificateur opérationnel. La combinaison de ces 2 types de distorsion se rapproche de près de la distorsion typique induite par des tubes sachant que les diodes en situation

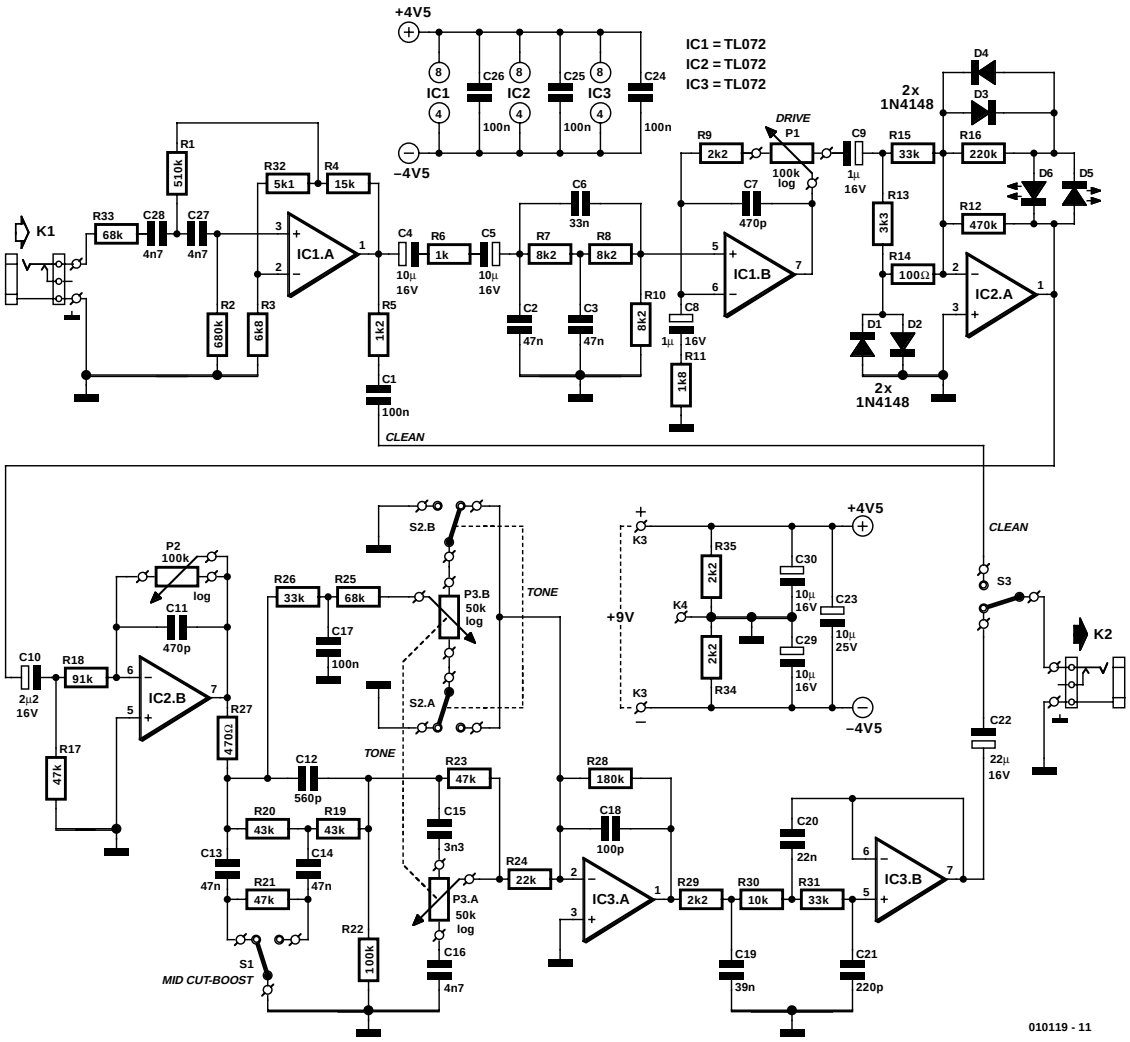


Figure 1. L'électronique du notre Tube Box se résume à quelques filtres associés à un étage de distorsion.

d'écrêtage font apparaître des harmoniques additionnelles tant paires qu'impaires (ce qui donne un son plein et agréable). La palette sonore est bien riche.

Il est temps maintenant de nous intéresser d'un peu plus près à l'électronique située en aval de l'étage générateur de distorsion. Il faut, si l'on veut que l'étage de distorsion puisse travailler le plus naturellement du monde, préparer en conséquence le signal d'entrée. Le trajet du signal en provenance de la guitare arrive, par le biais d'une entrée à haute impédance dotée d'un filtre, tout d'abord à l'amplificateur opérationnel IC1.A. Cet amplificateur opérationnel fait office d'amplificateur-tampon et évite que le circuit ne constitue une charge trop importante pour l'élément micro de guitare caractérisé lui aussi par une impédance élevée.

À la sortie, le signal suit un double trajet : il arrive d'une part, par le biais d'un filtre, à l'interrupteur à pédale S3, et de l'autre à IC1.B au travers d'un filtre. Cet amplificateur opérationnel rehausse le signal d'un facteur 5 environ. Cet interrupteur à pédale permet de ponter l'électronique de distorsion et de transmettre le signal de la guitare pratiquement intact vers l'embase de sortie.

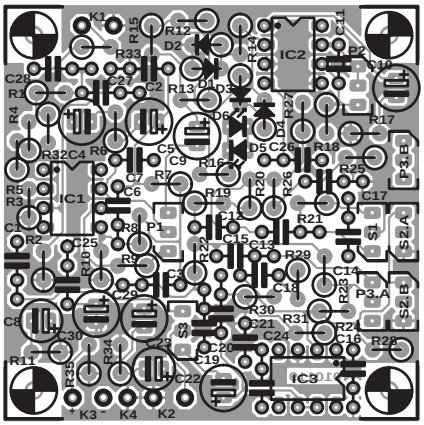
Au cours de son trajet vers l'étage de distorsion, le signal doit traverser un filtre passe-bande, en aval duquel on trouve à nouveau un amplificateur-tampon. La fonction du filtre passe-bande est de rehausser la partie du spectre de fréquences intéressant. P1 permet de jouer sur le facteur d'amplification (le gain) de IC1.B. Ce potentiomètre permet de régler non seulement le volume mais également le niveau de distorsion. Plus on ouvre le potentiomètre, plus le facteur de distorsion du signal arrivant au niveau de IC2.B augmente.

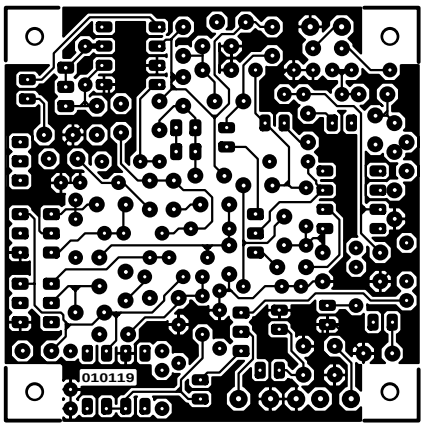
On découvre, sur le schéma, un autre potentiomètre qui permet de régler le volume de manière à ce que le signal de sortie au niveau de l'embase K2 ait le même niveau, que le signal ait traversé l'ensemble de l'électronique ou qu'il ait été ponté par l'interrupteur S3. Ceci permet de ne pas être confronté, lorsque le musicien passe du mode distorsion au mode linéaire (sans distorsion), à des sauts de volume intempestifs.

Il va sans dire qu'il est également possible de régler P2 de façon, par exemple, à ce que le signal distordu ait un volume légèrement supérieur à celui du signal non traité de manière à ce que le guitariste soit, lors d'un solo, quelque peu plus présent.

Nous découvrons ensuite un réglage de tonalité relativement conséquent comportant différentes possibilités de réglage. L'interrup-

teur S1 permet de décider si la plage de médium doit être atténuée ou amplifiée. On pourra, par exemple, envisager une atténuation lorsque l'on joue de la musique à fort niveau (*Metal*), alors que dans le cas de Blues plus « doux » il pourrait être bon de rehausser la plage de médium de sorte que le soliste soit mis plus en avant.
Le réglage de tonalité présente une structure





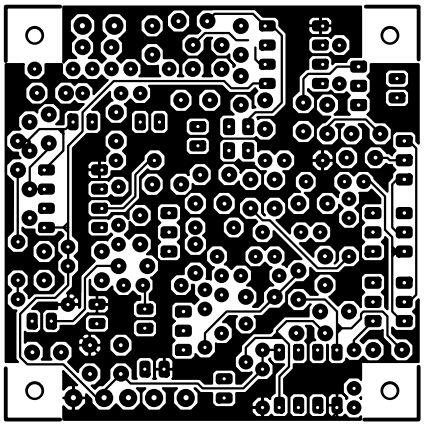
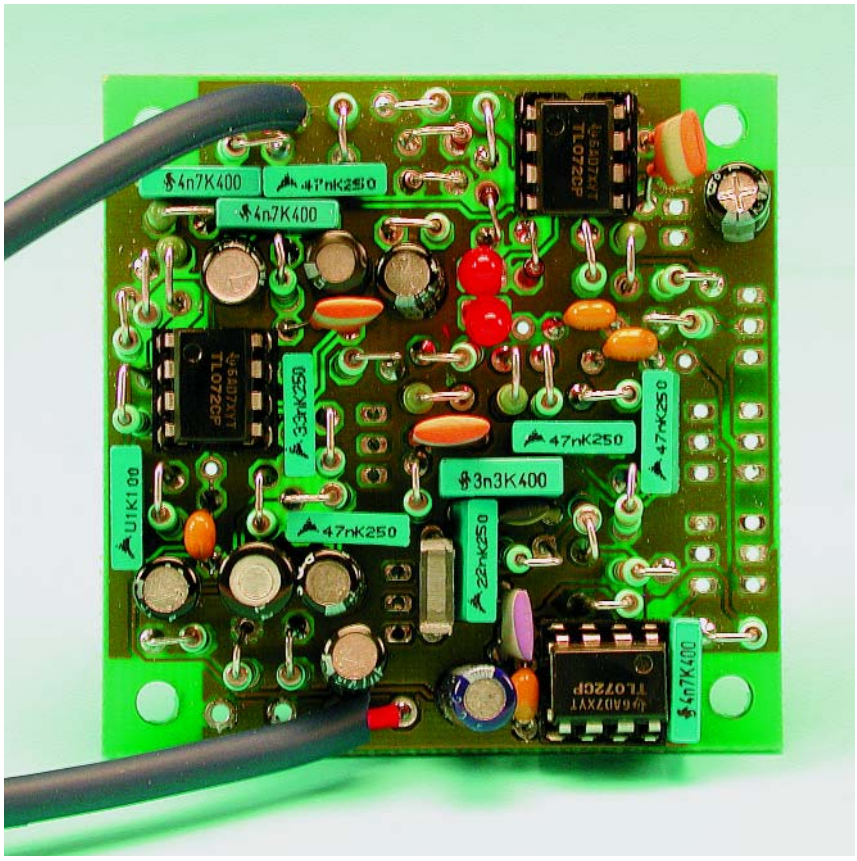


Figure 2. Dessin des pistes et sérigraphie de l'implantation des composants de la platine compacte conçue à l'intention de cette réalisation.



Liste des composants

Résistances :
R1 = 511 kΩ
R2 = 680 kΩ
R3 = 6kΩ8
R4 = 15 kΩ
R5 = 1kΩ2
R6 = 1 kΩ
R7,R8,R10 = 8kΩ2
R9,R29,R34,R35 = 2kΩ2
R11 = 1kΩ8
R12 = 470 kΩ
R13 = 3kΩ3
R14 = 100 Ω
R15,R26,R31 = 33 kΩ
R16 = 220 kΩ
R17,R21,R23 = 47 kΩ
R18 = 91 kΩ
R19,R20 = 43 kΩ
R22 = 100 kΩ
R24 = 22 kΩ
R25,R33 = 68 kΩ
R27 = 470 Ω
R28 = 180 kΩ
R30 = 10 kΩ
R32 = 5kΩ1
P1,P2 = potentiomètre 100 kΩ log
P3= potentiomètre stéréo 50 kΩ log

Condensateurs :
C1,C17,C24 à C26 = 100 nF
C2,C3,C13,C14 = 47 nF
C4,C5,C29,C30 = 10 µF/16 V vertical
C6 = 33 nF
C7,C11 = 470 pF
C8 = 1 µF/16 V vertical
C9 = 1 µF/16 V vertical
C10 = 2µF2/16 V vertical
C12 = 560 pF
C15 = 3nF3
C16,C27,C28 = 4nF7
C18 = 100 pF
C19 = 39 nF
C20 = 22 nF
C21 = 220 pF
C22 = 22 µF/16 V vertical
C23,C29,C30 = 10 µF/16 V vertical

Semi-conducteurs :
D1 à D4 = 1N4148
D5,D6 = LED rouge
IC1 à IC3 = TL072-CP*

Divers:
K1,K2 =embase jack 6,35 mm châssis femelle *
K3 = picots
S1 = inverseur unipolaire
S2 = inverseur bipolaire
S3 = interrupteur à pédale FS-40 (Monacor 01.0260)
Boîtier tel que, par exemple, 1590B (Hammond) *

quelque peu inhabituelle, mais très efficace. Le double potentiomètre P3 permet un réglage simultané des aigus et des graves, la position du double inverseur S2 servant à définir le mode adopté, soit en parallèle soit en opposition : on aura partant, dans l'une des positions de S2, rehaussement/atténuation simultanément des aigus et des graves, soit rehaussement des aigus et atténuation des graves ou l'inverse.

En aval du réglage de tonalité on trouve un amplificateur/tampon. Il ne reste plus, au signal, qu'à traverser un autre filtre passe-haut tamponné avant d'arriver, au travers de l'interrupteur à pédale S3, à l'embase de sortie.

L'alimentation de notre montage de distorsion pourra se faire par le biais d'un adaptateur secteur fournissant une tension régulée de 9 V que l'on applique au connecteur K1. On peut également envisager une alimentation par pile compacte de 9 V. Il est préférable, si l'on choisit ce second mode d'alimentation, d'utiliser non pas des TL072, mais un quadruple amplificateur opérationnel compatible broche à broche rail-à-rail du type OP470 sensiblement moins gourmand. Votre pile n'en durera que plus longtemps.

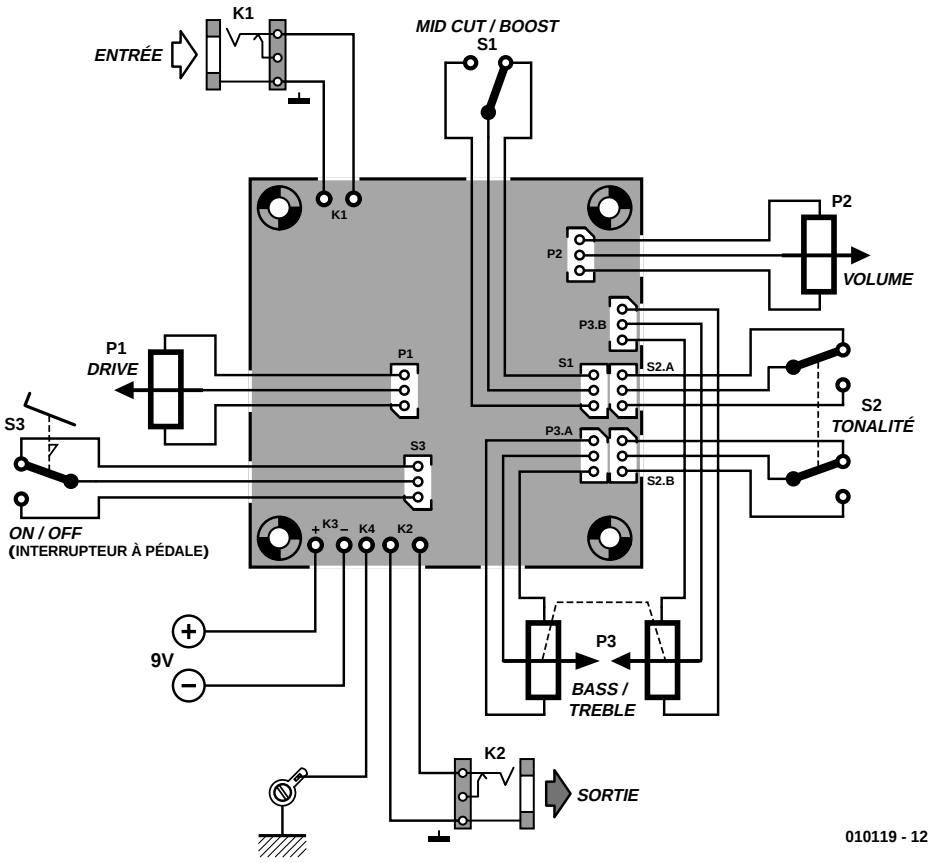


Figure 3. Plan de câblage des organes de commande connectés à la platine.

Construction et essais

En raison des dimensions compactes de la plastique dont nous vous proposons le dessin des pistes et la sérigraphie de l'implantation des composants en **figure 2**, la densité d'implantation est relativement élevée. La plupart des composants sont montés verticalement, ce qui explique que l'on commencera de préférence par l'implantation des composants de petite taille pour terminer par les plus gros. On mettra partant d'abord les diodes et les LED en place avant de passer à l'implantation des supports pour circuit intégré et des condensateurs, hormis les condensateurs électrochimiques, les liaisons câblées vers les potentiomètres et les interrupteurs pour terminer avec la mise place des résistances, des condensateurs électrochimiques, des picots d'entrée et de sortie du signal, de la tension d'alimentation et de la masse (blindage). La **figure 3** vous propose le plan de câblage des différents organes de commande. La platine trouvera place dans un boîtier métallique de dimensions adéquate qui le mettra à l'abri des signaux parasites. Pour la même raison il faudra également relier la masse du circuit (K4) au boîtier. Cette liaison pourra se faire par la mise en place d'un oeillet dans le fond du boîtier. Nous avons utilisé, pour notre prototype, un boîtier 1590B (Hammond) dont les dimensions sont insuffisantes pour permettre la mise en place d'une pile. Pour des raisons de place nous avons remplacé les embases-jack châssis de l'entrée et de la sortie, par quelques centimètres de câble de scène blindé, conducteurs aux extrémités desquels nous avons soudé les 2 fiches-jack 6,3 mm femelles. Il ne faudra pas oublier, si l'on adopte cette solution, de prévoir un dispositif anti-arrachement pour ces câbles.

Il faudra, si l'on opte pour la mise en place d'embases-châssis dans le coffret, choisir inévitablement un boîtier de dimensions plus « confortables ». Il ne sera pas nécessaire de prévoir dans ce cas-là de liaison spécifique par oeillet vers la masse, sachant de ces sont les jacks non-isolés par rapport au boîtier qui remplissent cette fonction.

Outre sa fonction de blindage, un boîtier métallique présente un second avantage : de par son poids, il rend l'appareil très stable de sorte qu'il n'y aucune crainte à avoir de poser le Tube Box sur ces planches, fenêtre sur le monde qu'est une scène, pour ensuite s'en servir.

Pour le tester, on connecte le Tube Box à l'instrument de musique, positionne le potentiomètre de volume au minimum et on gratte une corde de la guitare. Si l'on n'entend rien, il faudra appuyer sur l'interrupteur à pédale qui permettra au signal de contourner le circuit de distorsion. Une nouvelle action sur la pédale pour mettre le Tube Box en fonction et jouer ensuite sur P2 pour obtenir le volume désiré. Il reste ensuite à jouer sur P1 pour choisir le niveau de distorsion voulu, et de mettre S1, S2 et S3 dans la position correspondant au réglage de tonalité le plus satisfaisant.

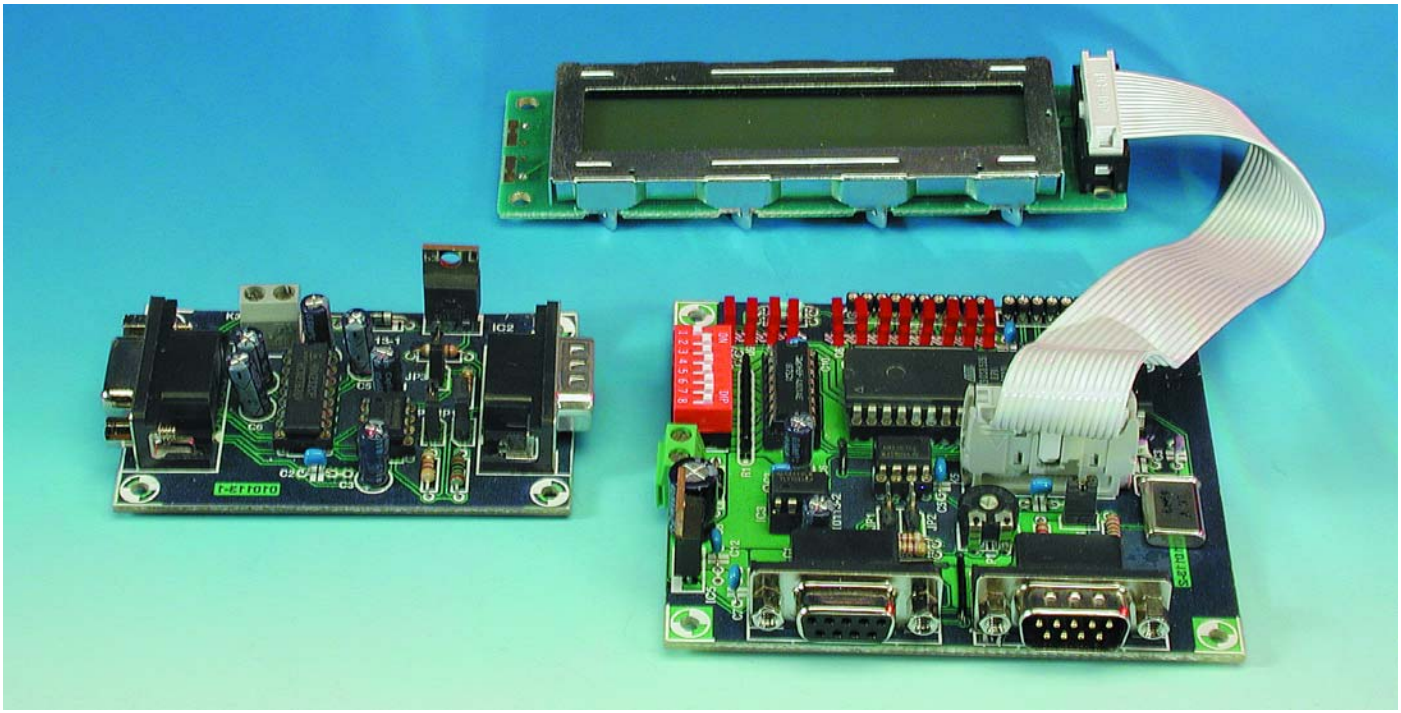
(010119)

Bus DCI

Un bus domestique RS-485 qui dessert 64 postes

Ivo de Coninck

Ce système de bus universel, piloté par PC et capable de desservir 64 postes secondaires offre de nombreuses possibilités. Si sa destination première n'est pas de couvrir de longues distances, l'horizon de ce bus domestique ne se limite pas aux murs de la maison. Comme chaque poste dispose de 8 entrées et de 8 sorties individualisées, ainsi que d'un écran LCD, il pourra étendre ses services aux écoles ou aux entreprises moyennes : communication ou sécurité intérieure, entre autres.



Avant de nous embarquer dans la technique, dégageons les grandes lignes du système.
Il s'agit donc d'un système de bus sériel complet, avec protocole propre, capable d'un grand rayon d'action et susceptible de dialoguer avec un maximum de 64 postes de travail via une interface RS-485.
Le matériel nécessaire se résume à un convertisseur simple de RS-232 à RS-485 et des postes en nombre suffisant, équipés chacun d'un processeur. Du logiciel, il en faudra un pour le PC et un autre pour animer le poste de travail : vous pouvez les télécharger depuis notre site (www.elektor.fr). Le processeur du poste est également disponible complètement programmé. Le logiciel du PC travaille sous Windows 95, 98 et ME, il s'accompagne d'un programme-témoin qui permet, pour chacun des postes, de vérifier et d'influencer l'état de toutes les lignes d'entrée-sortie, de comman-

der l'affichage et de lui envoyer 2 x 20 caractères.

Survol du système

Tout commence par un logiciel qui pilote, à un débit de 115 200 bauds, le port sériel du PC, auquel on branche un convertisseur de RS-232 à RS-485. C'est un montage relativement simple à base de MAX232 et de UART SN75LBC184 et nous voilà en présence d'un bus RS-485 capable de traiter sur pied d'égalité 64 postes.

Dans chacun de ces postes, on trouve un processeur AT90S8515, un 4021 et un autre SN75LBC184, mais également des interfaces à huit bits en entrée et en sortie ainsi qu'un branchement pour écran LCD à deux lignes. Chaque poste a son adresse, comprise entre 0 et 63. Dans le processeur du poste, un logiciel gère un certain nombre de tâches : la communication avec le PC, la commande des sorties, la lecture des entrées, la transmission du texte vers l'affichage et la lecture de l'adresse programmée par cavaliers.

Voyons comment se déroule la communication. Le logiciel chargé dans le PC, le chef de réseau, peut sélectionner un poste par son adresse, puis successivement commander les huit sorties, envoyer les données à afficher et enfin demander l'état des huit entrées. Cela prend environ 25 ms.

Il peut recommencer illico la même procédure avec un autre poste. Mais s'il veut sélectionner de nouveau le même secondaire, cela prendra 0,8 s de plus, parce qu'il faut au préalable rafraîchir les données à afficher.

RS-485

Les transmissions dans le réseau décrit ici utilisent une paire torsadée blindée et utilisent le protocole RS-485. Lorsqu'il s'agit d'échanger de petites quantités de données sur de longues distances, l'interface RS-485 s'indique particulièrement. Elle utilise un réseau symétrique auquel on peut relier plusieurs émetteurs et plusieurs récepteurs.

La norme RS-485 (TIA/EIA-485-A) spécifie les caractéristiques électriques du bus, des émetteurs et des récepteurs. Elle fournit en outre des suggestions sur la manière de câbler

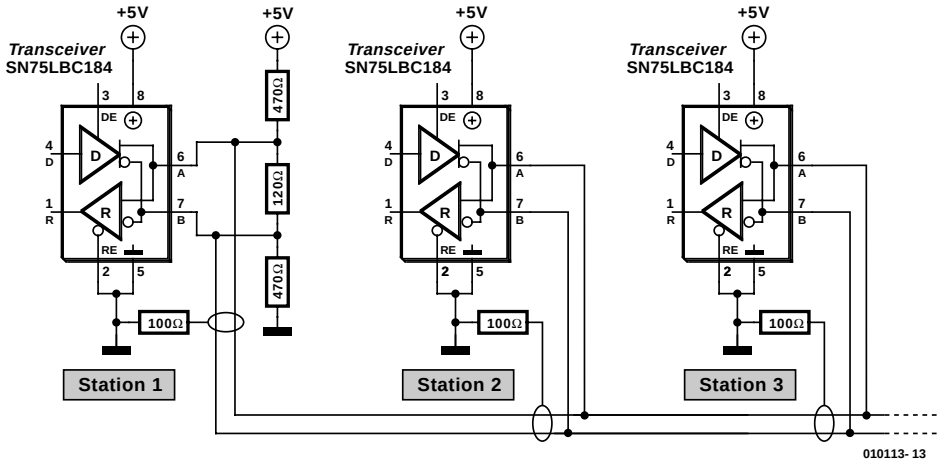


Figure 1. Exemple de réseau RS-485.

et de terminer le réseau, mais de brochage ou de logiciel, il n'en est pas question.

Un réseau RS-485 peut totaliser 256 postes si l'on utilise des récepteurs à haute impédance. Sa longueur peut atteindre 1 200 m pour un débit de 10 Mbps. Sur de plus grandes distances, on peut intercaler des amplificateurs pour régénérer le signal et, de là, repartir vers un nouveau réseau. Lors des essais, nous n'avons pas cherché à battre de record, mais sur 75 m, tout fonctionnait à la perfection.

Comme nous le disions plus haut, la spécification RS-485 ne souffle mot du protocole à utiliser, en pratique, on fait souvent appel à celui qui régit d'habitude le UART du PC. On trouve dans le commerce plusieurs sortes de convertisseurs RS-485. Pour un microcontrôleur, on peut brancher un l'émetteur/récepteur ou trancepteur (*tranceiver*) RS-485 au port sériel. La plupart des réseaux mettent en œuvre un signal supplémentaire pour le contrôle du trancepteur. Côté PC, on peut alors utiliser le signal RTS.

Le pourquoi de l'aptitude des réseaux RS-485 à opérer à longue distance, c'est la faculté des récepteurs de réagir à la différence de tension entre les conducteurs, la ligne étant alimentée symétriquement. La grande majorité des parasites qui atteignent le câble agissent pareillement sur les deux fils et n'influencent donc pas la tension différentielle. Au contraire, lors d'une transmission en mode commun, comme sous RS-232, le fil de retour est relié à la masse.

L'émetteur doit envoyer une différence de tension d'au moins 1,5 V pour tenir compte de l'atténuation et s'affranchir du bruit résiduel. À l'endroit des nœuds, il convient de garder le câblage aussi court que possible et de n'utiliser que des paires torsadées et blindées pour éviter d'induire du bruit.

Dans une telle configuration, il y a lieu de s'entendre sur les niveaux logiques. Les feuillets de caractéristiques des puces d'interface appellent « ligne A » celle qui reste en phase avec le signal d'origine et « ligne B » celle qui a subi l'inversion. Quand la ligne A porte une tension plus positive d'au moins 200 mV que la ligne B, la sortie du récepteur est au niveau haut et réciproquement. Pour toute différence de potentiel inférieure à 200 mV, l'état de sortie n'est pas défini.

La **figure 1** présente un exemple de réseau RS-485. À l'entrée du réseau, nous voyons trois résistances, deux de 470 Ω et une de 120 Ω. Elles fixent les potentiels de la ligne aussi longtemps qu'aucun émetteur n'est actif.

En début et en fin de réseau, il faut placer une résistance de bouclage pour atténuer les réflexions sur la ligne. Sa valeur, en RS-485, est comprise entre 100 et 150 Ω. Pour un débit de 115 en installer à d'autres endroits ne ferait que surcharger les composants de transmission.

La norme RS-485 prescrit en outre une résistance de 100 re, ces résistances limiteront le courant.

Le convertisseur de RS-232 à 485

Très simple, le montage qui assure la conversion des signaux RS-232 vers le RS-485, comme l'illustre la **figure 2**. L'adaptation des niveaux issus du port sériel à ceux

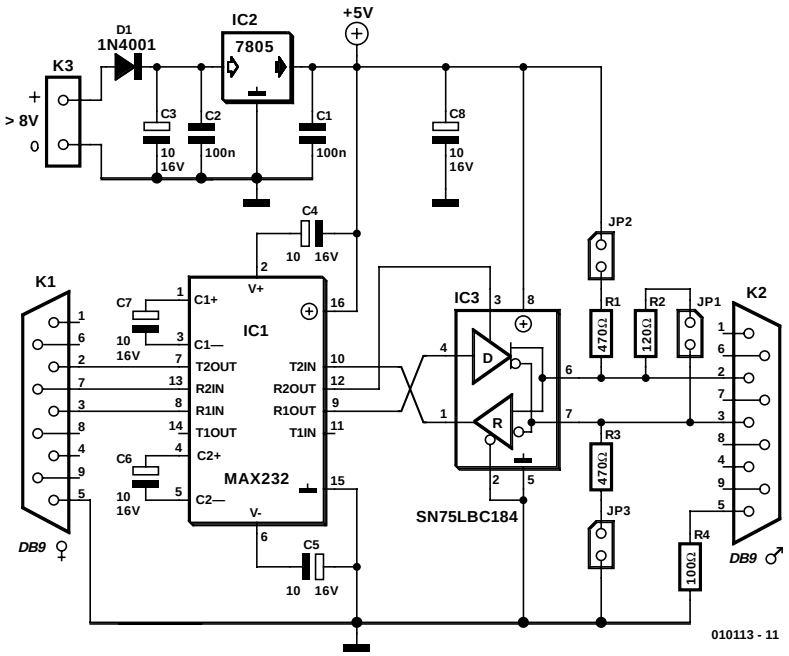


Figure 2. Ne brille-t-il pas par sa simplicité, ce convertisseur RS-232/RS-485 basé sur un UART SN75LBC184 ?

du SN75LBC184, nous la confions à IC1, un MAX232. C'est le convertisseur continu-continu intégré dans le MAX232 qui requiert la présence des condensateurs électrolytiques C4 à C7. La spécialité du trancepteur (*tranceiver*) SN75LBC184, IC3, c'est l'émission et la réception de données. On peut le remplacer par un 75176, qui n'est pas protégé contre la décharge électrique disruptive (ESD).

Avec la broche 2 de cette puce à la masse, la réception des données qui transitent sur le bus est permanente. La broche 3 est reliée, par l'intermédiaire du MAX232, à la broche 7 (RTS) du port sériel. Nous allons utiliser ce signal –c'est le logiciel qui s'en chargera– pour activer et désactiver l'émetteur. Le cavalier 1 permet de brancher ou non la résistance de terminaison en ce point. Les résistances R1 et R3, par le truchement des cavaliers 2 et 3, branchent le plus et le moins de l'alimentation de manière à forcer l'état de la ligne quand aucun pilote n'est actif. Puisque le convertisseur se place en début de ligne, nous utiliserons les trois cavaliers.

Comme source d'alimentation, on peut prendre un adaptateur secteur classique de 9 V, sans tenir compte d'aucun critère particulier : la puce IC3, un 7805, s'occupe de la stabilisation à 5 V.

Le schéma du poste

La **figure 3** vous fournit le schéma complet d'un des postes. Devinez qui se coltine presque tout le boulot ? C'est le microcontrôleur IC2, un AT90S8515 de Atmel, muni du logiciel apte à assurer le bon fonctionnement du poste. Sa fréquence d'horloge est fixée par le quartz X1 à 3,6864 MHz, une valeur choisie de façon à délivrer une synchronisation exacte pour le générateur de taux de transmission du UART IC4, en accord avec le PC. IC4 vient s'insérer entre les lignes RXD, TXD du contrôleur et les connecteurs sub-D K3 et K4 destinés aux raccordements du bus.

Pour consulter l'adresse sélectionnée, il faut un registre à décalage, IC1, dont les cellules sont influencées par les interrupteurs DIP S1-1 à S1-6. Tous ouverts, l'adresse du poste sera 63, tous fermés, zéro ; on peut donc bien brancher 64 postes au même bus. Deux d'entre eux ne peuvent évidemment pas porter la même identité, sous peine de les voir tenter en vain de répondre simultanément au chef et brouiller la communication sur le bus.

Les interrupteurs S1-7 et S1-8 sont libres et donc disponibles pour des extensions éventuelles, à condition de modifier en conséquence le logiciel du microcontrôleur.

La LED D9 indique si le AT90S8515 est prêt à recevoir des données. Aussi longtemps que le poste est actif, que ce soit en émission ou en réception, la LED D12 est allumée. Les LED D10 et D11 n'ont pas encore de fonction, elles vous attendent pour des applications ultérieures.

Les broches 32 à 39 du microcontrôleur fonctionnent comme sorties et pilotent les LED D1 à D8. Elles sont également reliées au connecteur K2, qui peut servir à brancher une interface de sortie. Les entrées sont les broches 21 à 28, elles sont flanquées de résistances de rappel au niveau haut, le réseau R6, et accessibles par le connecteur K2 à l'intention d'une interface d'entrée.

Le circuit de mise à zéro, nous avons préféré le constituer d'un TL7705, IC3. Le condensateur électrolytique C1 en constitue l'élément de temporisation. À noter que c'est le signal de la sortie inversée de IC3 qui est appliqué au microcontrôleur.

Il nous faut penser au branchement du panneau LCD de 2 x 20 caractères. Ce sera le rôle du connecteur K5. Le réglage de contraste se fera par le potentiomètre P1. PD1 et PD2 sont les lignes de commande de l'affichage (RS et E). Comme la broche 5 de l'écran (R/W) va à la masse, il ne sera utilisé que dans le sens de l'écriture vers les cristaux liquides.

Côté alimentation, un adaptateur secteur simple de 9 V, tout comme pour le convertisseur de la figure 2, fera l'affaire. Il se raccorde à K6.

Les platines

Les **figures 4 et 5** vous dévoilent le tracé des pistes et la disposition des composants du convertisseur et du poste secondaire. La première platine est particulièrement compacte et, avec aussi peu de composants, la construction ne devrait pas vous prendre plus d'une heure de travail.

Il en va autrement de la platine du poste, que devront garnir davantage de puces. Il est préférable d'y consacrer le temps voulu, mais pour qui sait souder, le labeur n'est pas ardu.

Les embases et connecteurs sont autant que possible disposés en bordure de la platine, ce qui simplifiera leur branchement. Seul K5, pour l'écran LCD, fait exception.

En raison de la faible consommation de courant, les régulateurs de 5 V des deux platines ne réclament pas de radiateur.

Comment prendre le bus

La **figure 6** nous indique la manière de raccorder au bus trois postes, à titre d'exemple, et où installer les résistances de bouclage.
Nous le savons déjà, l'adresse du poste s'inscrit à l'aide de six interrupteurs DIP de S1. Dans l'exemple de la **figure 7**, c'est l'adresse 12 qui a été sélectionnée (décomposé en binaire, 12 = 4 + 8). Les cavaliers JP1 et JP2 dans le circuit du poste sont destinés à polariser le bus en amont (à son début, leur fermeture se traduit par la mise en circuit d'une résistance de 470 Ω, dans le premier cas vers la masse dans le second vers le +5 V). Comme le poste 12, dans la figure 4, se situe en fin de bus, c'est à son cavalier JP3 qu'il revient de brancher la résistance de bouclage.
Quelques instants après avoir branché l'alimentation (un adaptateur secteur de 9 V), la LED D9 s'allume. Elle nous dit que le CI est alors en mesure de recevoir des données et d'en envoyer. Si un écran LCD est raccordé au poste, le texte suivant va s'y afficher :

```
DCI-BUS Addr : 12
Waiting for data. U
```

Le nombre 12 correspond à l'adresse de poste inscrite, le U en bout de seconde ligne identifie la version du logiciel chargé dans le microcontrôleur.

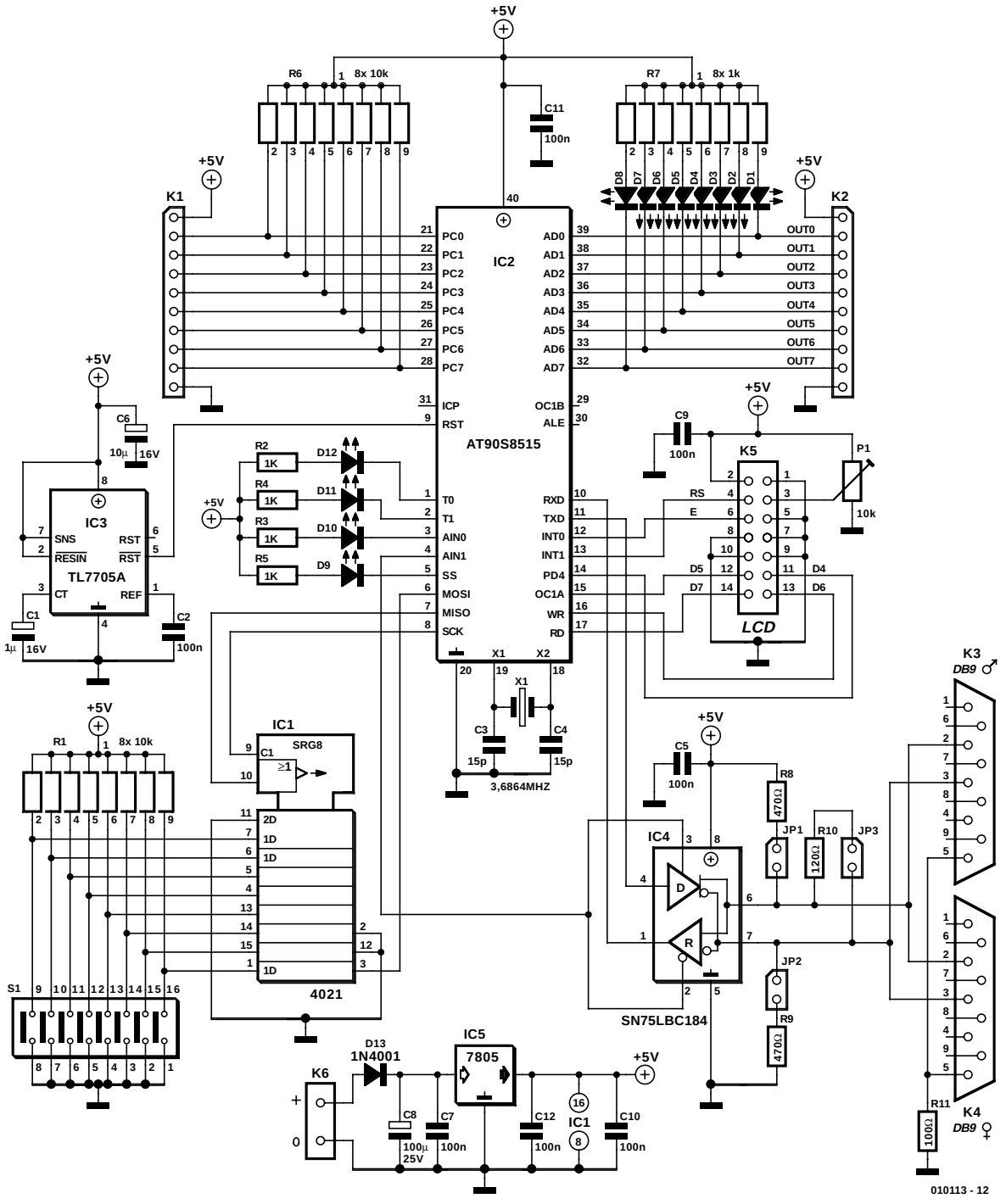


Figure 3. Le schéma complet du poste secondaire. Le microcontrôleur AT90S8515 y occupe le rôle essentiel.

Liste des composants du convertisseur

Résistances :
R1,R3 = 470 Ω
R2 = 120 Ω
R4 = 100 Ω

Condensateurs :
C1,C2 = 100 nF
C3 à C8 = 10 µF/50 V radial

Semi-conducteurs :
D1 = 1N4001
IC1 = MAX232 (Maxim)
IC2 = 7805
IC3 = SN75LBC184 (ou 75176)

Divers :
JP1 à JP3 = embase autosécable mâle à 2 contacts + cavalier
K1 = embase sub-D femelle encartable en équerre à 9 contacts
K2 = embase sub-D mâle encartable en équerre à 9 contacts

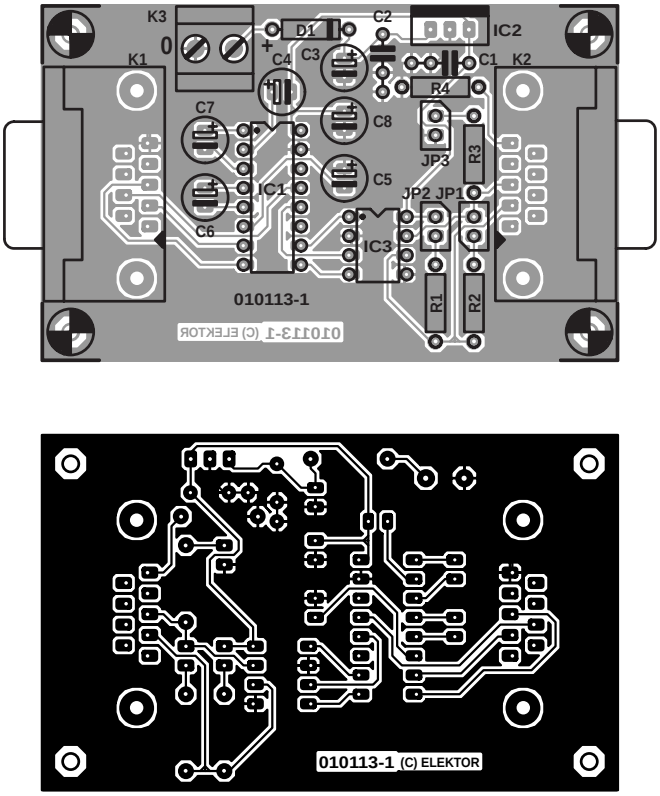


Figure 4. La platine du convertisseur de RS-232 vers RS-485.

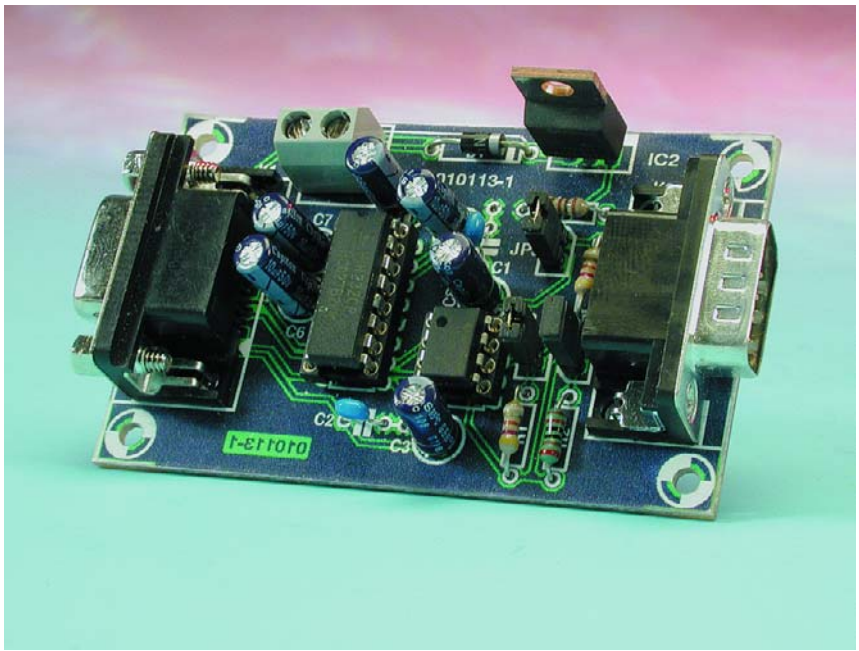
Liste des composants du poste

Résistances :
R1,R6 = réseau de 8 résistances de 10 kΩ
R2 à R5 = 1 kΩ
R7 = réseau de 8 résistances de 1 kΩ
R8,R9 = 470 Ω
R10 = 120 Ω
R11 = 100 Ω
P1 = ajustable 10 kΩ

Condensateurs :
C1 = 1 µF/16 V radial
C2,C5,C7,C9 à C12 = 100 nF
C3,C4 = 15 pF
C6 = 10 µF/16 V
C8 = 100 µF/25 V

Semi-conducteurs :
D1 à.D12 = LED rouge rectangulaire à haut rendement
D13 = 1N4001
IC1 = 4021
IC2 = AT90S8515-8PC (programmé **EPS 010113-41**)
IC3 = TL7705-ACP
IC4 = SN75LBC184 (ou75176)
IC5 = 7805

Divers :
JP1 à JP3 = embase autosécable mâle à 2 contacts + cavalier
K1, K2 = embase autosécable à 1 rangée de 10 contacts
K3 = embase sub-D mâle encartable en équerre à 9 contacts
K4 = embase sub-D femelle encartable en équerre à 9 contacts
K5 = embase autosécable à 2 rangées de 7 contacts
K6 = bornier à 2 contacts au pas de 5 mm
S1 = octuple interrupteur DIP
X1 =quartz 3,686 4 MHz
module LCD à 2 lignes de 20 caractères



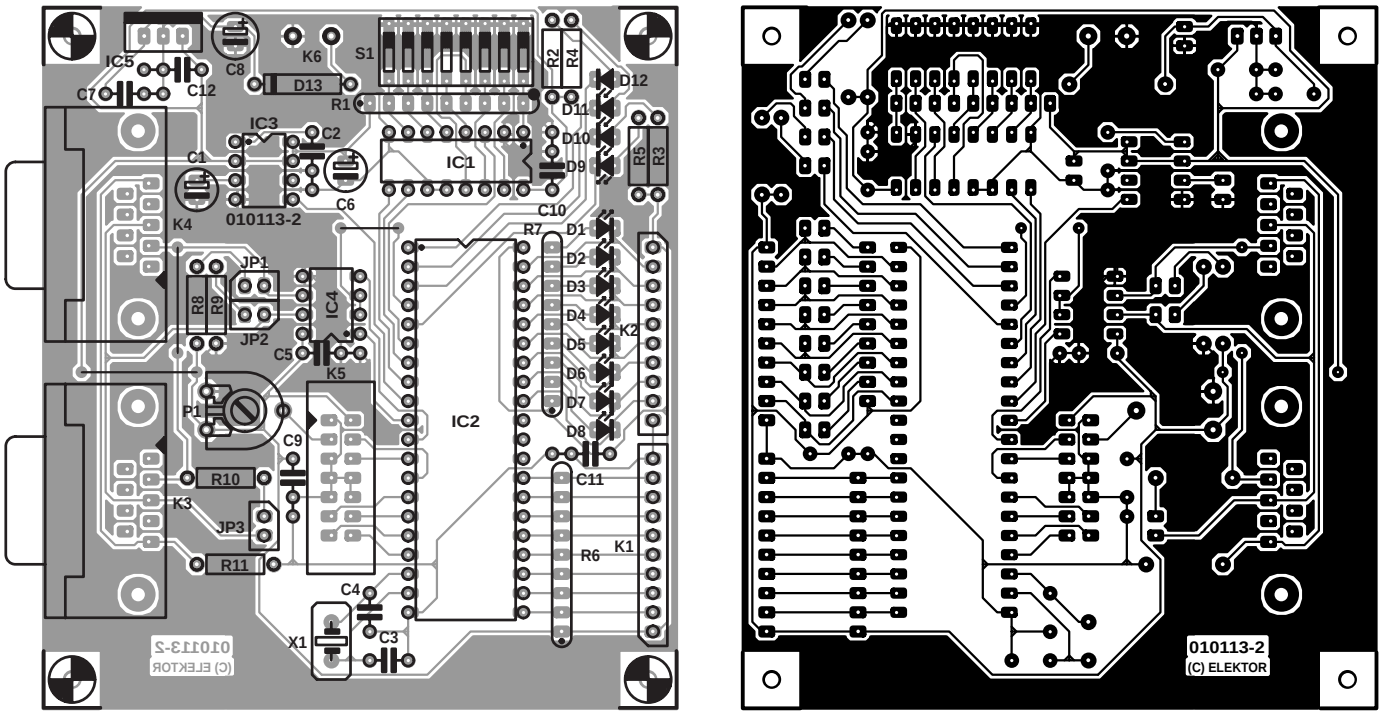
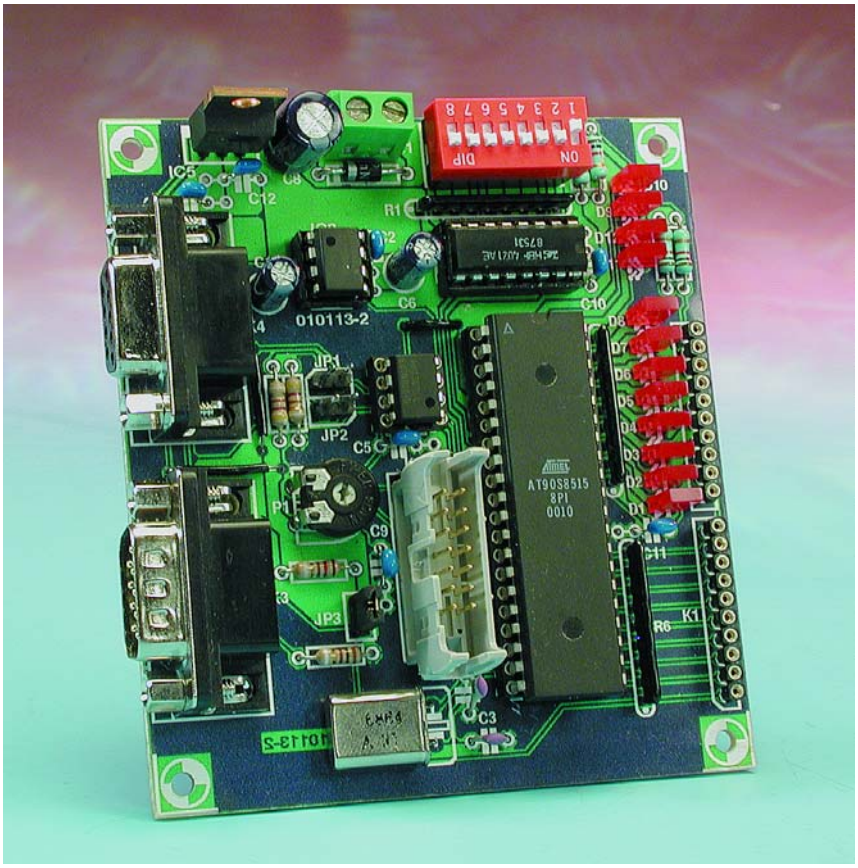


Figure 5. Et la platine du poste secondaire. À part K5, tous les connecteurs sont accessibles par le bord du circuit imprimé.

Du logiciel pour le PC

Après l'installation du matériel et le lancement du programme, il faut sélectionner le bon port. Cliquons sur « Select port » dans le menu « Ports » et choisissons celui qui est mis en service. Toujours dans le menu « Ports », cliquons alors sur « Open port » pour faire apparaître une fenêtre de dialogue qui indique que l'ouverture du port a réussi. Si le port choisi n'existait pas, on recevrait un avis selon lequel le système n'arrive pas à trouver le fichier demandé. Cliquer ensuite sur « Run » dans le menu « Bus » suffit à mettre le logiciel en mode « RUN » (voir **figure 8**).



À côté de chaque bouton « SetStation » se trouve un dispositif crescendo-decrescendo qui permet d'atteindre le numéro du poste à joindre. Cliquez ensuite sur le bouton « SetStation » et le poste est en ligne. À gauche du bouton s'inscrit alors le numéro du poste choisi. Dès maintenant, le texte présent dans les deux boîtes va s'afficher sur l'écran du poste sélectionné et ses sorties seront commutées dans l'état prévu. De plus, l'état des entrées va s'afficher. Le logiciel permet de commander quatre postes à la fois.

Par action sur les flèches du bouton « adjust bus », on peut adapter la périodicité dans la zone critique entre 0 et 20 ms. Une valeur de 3 à 6 ms convient bien à un Pentium 400 ou 475. Cette option revêt probablement une plus grande utilité sur les ordinateurs récents qui sont encore plus rapides. Il faut savoir, en effet, qu'un logiciel qui opère sous Windows ne travaille jamais en temps réel, ce qui peut causer des erreurs sur le bus. La démonstration dans le diagramme de la **figure 9**. Si le signal RTS du port sériel n'est pas au niveau bas avant la réponse d'un poste, ces infor-

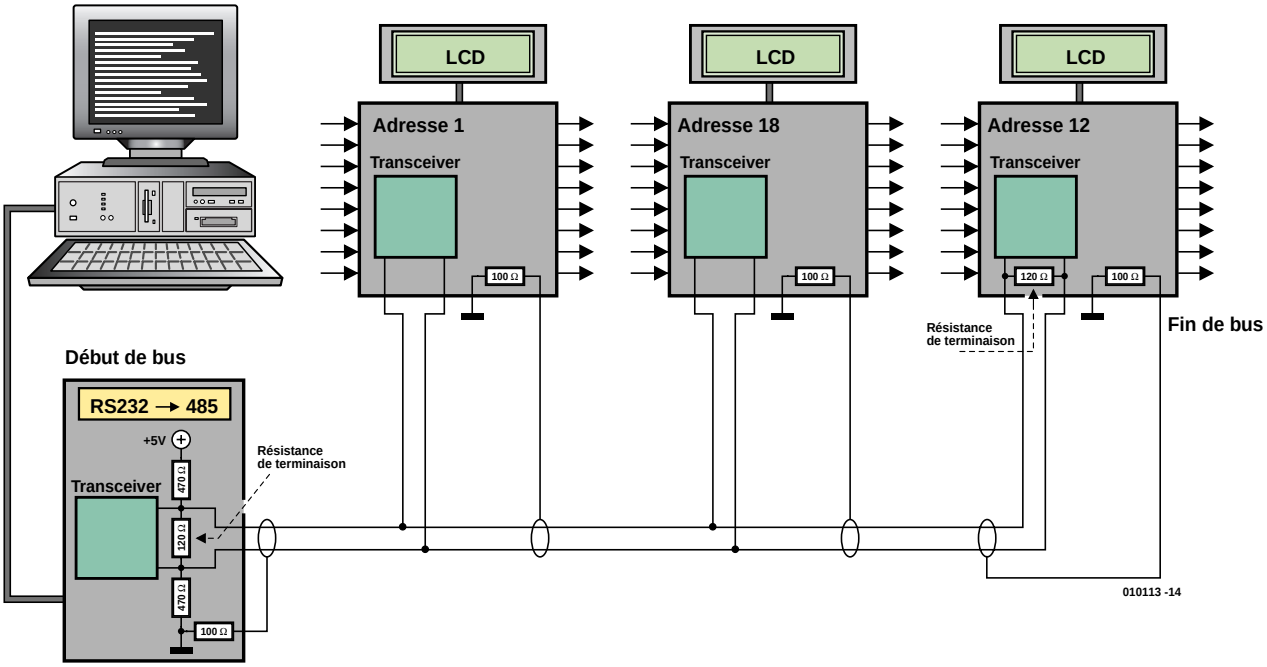


Figure 6. Exemple pratique d'un système de bus à trois postes.

mations sont perdues. C'est ce qui arrive par exemple quand le système d'exploitation, Windows en l'occurrence, estime nécessaire de consacrer en totalité le temps de calcul du processeur à une autre tâche, comme de lancer un nouveau logiciel, parcourir un autre CD-ROM, passer à un écran de veille, opérer un contrôle anti-virus, etc. En pareil cas, il se peut que le signal RTS ne devienne pas bas à temps.

Si nous voulons l'éviter, il faut prévoir dans le logiciel une sécurité qui vérifie l'exactitude de la lecture de l'état des entrées. Voici comment on peut procéder.

0. On place sur tous les postes le signal « send enable » au niveau bas.
1. Côté PC, on abaisse aussi le signal RTS.

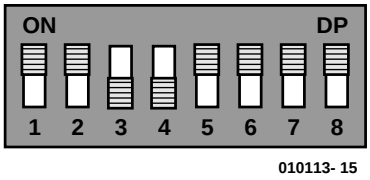


Figure 7. C'est l'adresse de poste 12 qui s'inscrit ici sur les interrupteurs DIP S1.

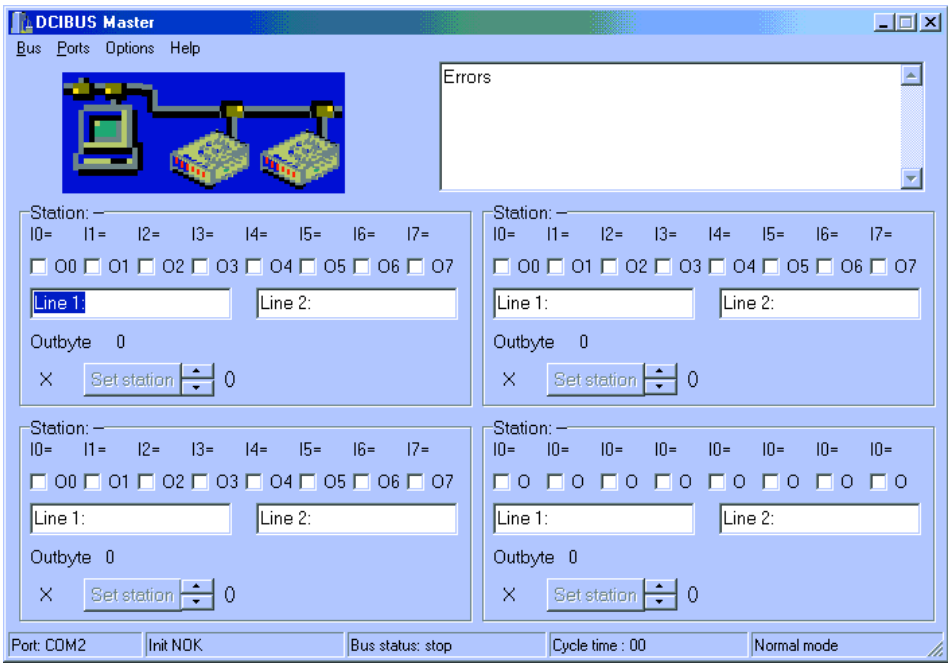


Figure 8. Copie d'écran du programme en mode RUN.

2. Le PC envoie sur le bus deux fois le code ASCII « 10 ».
3. Le PC lance sur le bus l'adresse du poste que nous voulons appeler.
4. Le PC transmet l'état des sorties.
5. Côté PC, RTS est mis au niveau haut.
6. Le poste appelé positionne en haut le signal « send enable ».
7. Le poste transmet au bus son adresse.
8. Le poste communique l'état de ses entrées.
9. Le poste répète son adresse.
10. Le poste positionne en bas son signal « send enable ».
11. Le PC vérifie que le premier octet reçu est égal à l'adresse qu'il a envoyée.
12. Le PC stocke le deuxième octet reçu (valeur des entrées).
13. Le PC vérifie la concordance des

deux adresses transmises.

14. Si les contrôles 11 et 13 sont probants, les états d'entrée sont mis à jour. Sinon, il faut recommencer toute la procédure jusqu'à ce qu'il n'y ait plus d'erreur.

De cette manière, nous sommes certains que les entrées seront toujours recopiées avec exactitude. Chaque erreur détectée entraîne l'affichage d'un message reprenant le numéro du poste dont les données sont incorrectes. Si nous essayons, par logiciel, d'activer un poste qui n'est pas connecté au bus, nous obtiendrons le même avertissement.
Si nous échangions le convertisseur simple de RS-232 vers RS535 contre une version à microcontrôleur équipé d'un tampon de données, les soucis de lecture s'envoleraient. Ce sera, à n'en pas douter, l'objet d'un futur projet.
Encore quelques remarques générales à propos du logiciel.
Sur la disquette de logiciel qui accompagne ce projet (EPS010113-11), mais que vous pouvez aussi bien télécharger de notre site www.Elektor.fr, vous trouverez, outre le logiciel pour le PC, le code source rédigé en C++ (Builder 4). Le listage HEX pour le microcontrôleur s'y trouve aussi, de quoi l'adapter à son gré et selon ses exigences propres.
Ne pas oublier, cependant, si l'on veut modifier le logiciel, qu'il faut à un poste secondaire une petite seconde pour rafraîchir les données de l'écran. Prévoir donc une pause équivalente, dans le logiciel du PC, avant de relancer le poste en question.
Et puis, bien sûr, le code ASCII « 10 » est réservé, il ne peut pas figurer dans le texte à transférer à l'écran LCD, puisque c'est lui qui sert à activer le poste.

Pilotage par logiciel externe

Le programme **Busmstr.exe** contient une interface de service COM (Component Object Model) très simple. Elle permet de lancer le programme Busmstr.exe et de piloter le matériel par l'intermédiaire de logiciels tels que Excel ou Visual Basic.
Prenons un exemple :
Lançons d'abord le programme Busmstr.exe et fermons-le.
Lançons à présent Excel, ouvrons le fichier « Client.xls » et faisons démarrer l'éditeur du Visual Basic <Alt+F11>.
Dans le menu « Image », cliquons sur « Explorateur d'objets » (F2).
À l'aide de la souris, allons vers la fenêtre « Explorateur d'objets » pour y cliquer du bouton droit.
Il faut maintenant cliquer sur « Références… » dans le menu déroulant.
Choisissons « Parcourir » dans la fenêtre de dialogue « Références… » et sélectionnons, comme type de fichier, Fichiers exécutables (« *.exe, *.dll »).
Dans le répertoire qui contient Busmstr.exe, sélectionnons-le et cliquons sur « Ouvrir ».
Du coup, on trouve dans la fenêtre « Classes » un objet du nom de « TBusServer ». En cliquant dessus, on fait apparaître dans la fenêtre les membres de l'objet. Il s'agit des fonctions « About », « InputGet », « LCDTxt » et « OutputSet ».
Retournons dans Excel et cliquons sur le bouton « INFO ».
Le programme Busmstr.exe démarre. Choisissons le bon port sériel, ouvrons-le et activons le bus.
Il faut à présent cliquer sur « Server mode » dans le menu « Options ».
Régler ensuite un poste sur l'adresse 3 et l'activer à l'aide du bouton « SetStation ».
De retour dans Excel, fermer la fenêtre de dialogue qui contient les informations sur le programme Busmstr.exe.
Cliquer sur le bouton « Refresh ». Le nombre dans la case C4 donne l'état des entrées du poste 3. Le texte dans la case C8 apparaît sur l'écran du poste 3 et le nombre que nous inscrivons dans la case C6 reflète les états que vont prendre les sorties du poste 3. Le texte dans la case C8 ne peut pas dépasser 40 caractères.
Passons en « Mode objet ».
Cliquons du bouton droit de la souris sur le bouton « Refresh » et choisissons « Afficher les codes programme ».
Nous pouvons dès lors travailler sur les codes et les étudier en vue de créer d'autres automatismes dans le fichier Excel.

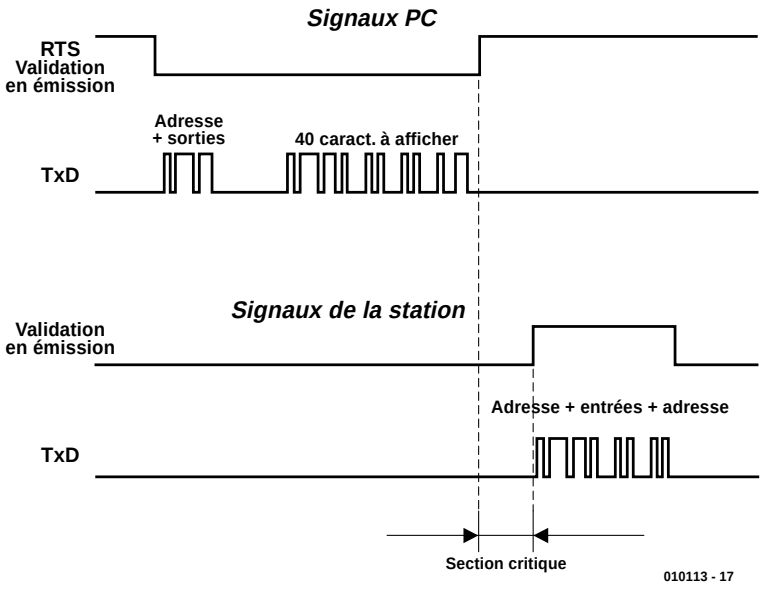


Figure 9. Il faut que le signal RTS soit bas au moment où le poste transmet ses données, sinon elles seront perdues.

Remarques

1. Si le programme **Busmstr.exe** a été lancé par Excel, il sera refermé chaque fois que l'on repasse en mode objet.
2. Ne jamais fermer le programme serveur (Busmstr.exe) quand le programme client (Excel dans ce cas-ci) est encore en activité !

(010113)

NdlR : Ne perdez pas votre temps à essayer de trouver de référence sur un bus DCI. Il doit son nom à celui de son auteur : De Coninck Ivo, d'où le DCI.
De plus amples informations sont à votre disposition sur le site Web de l'auteur : http://home.planetinternet.be/~dc1 1cd/index.htm
Vos remarques et suggestions recevront un accueil favorable à l'adresse de courriel : ivo.deconinck@planetinternet.be