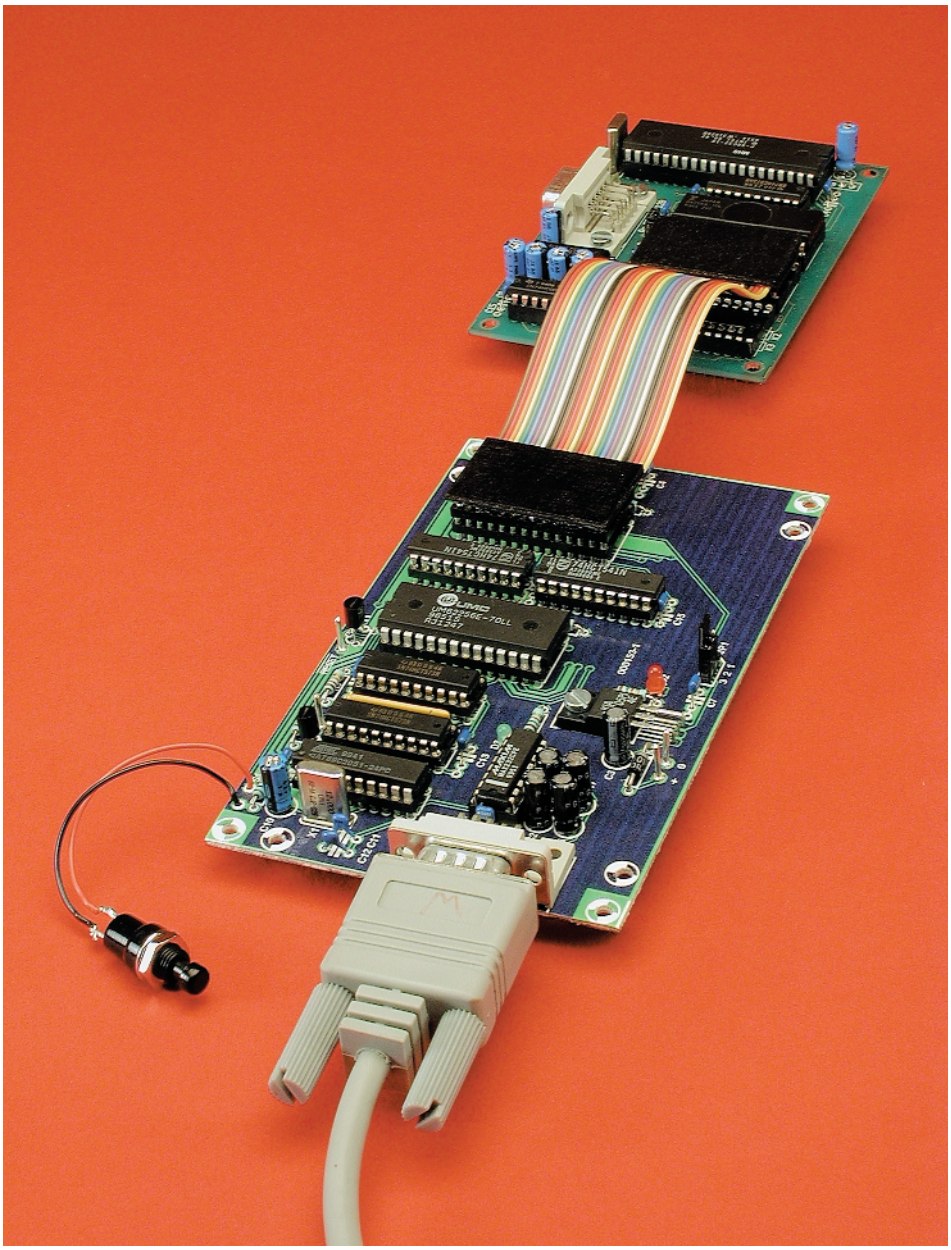
Simulateur d'EPROM

Pour la 27C256

Projet : Benoît Legrand & David Mautaulon



Cela fait déjà près de 2 lustres que nous avons eu l'occasion de vous proposer notre dernier simulateur ou émulateur d'EPROM. Cette nouvelle version répondant aux exigences actuelles permettra aux amateurs et même aux professionnels de mettre au point des systèmes à base de microprocesseurs comportant une EPROM. Nous avons opté pour la 27256 sachant que c'est le type d'EPROM le plus répandu.

Caractéristiques techniques

- Émule l'EPROM la plus courante, la 27256
- Universel par l'utilisation du port série
- Se contente d'un programme classique (Hyperterminal) pour assurer le transfert des données entre le PC et le simulateur d'EPROM
- Reconnaît le format de fichier le plus courant, Intel Hex
- Pourrait être adaptée aux 2764 et 27128 si l'on prend quelques mesures au niveau des broches 26 et 27 et éventuellement à la 27512 par quelques modifications matérielles et logicielles

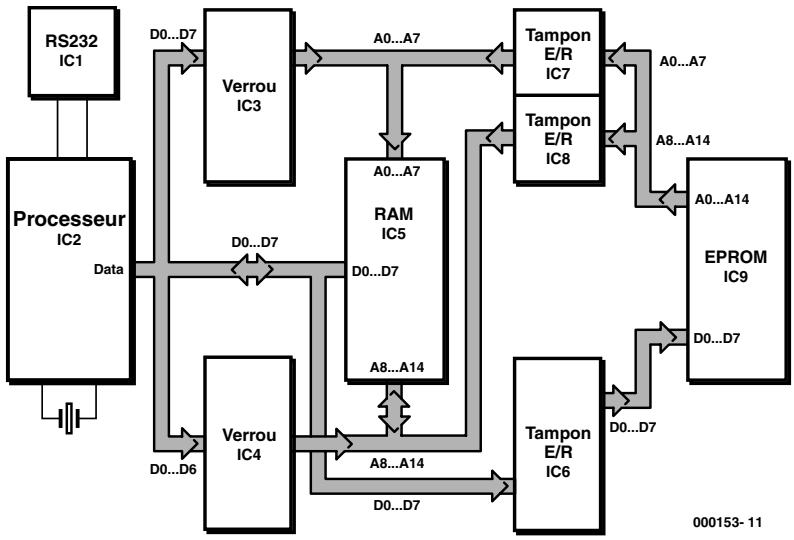


Figure 1. Le synoptique de notre simulateur d'EPROM se distingue par la présence d'un microcontrôleur.

La première question que l'on se pose à la lecture du titre est : à quoi peut bien servir un simulateur d'EPROM.
Il s'agit en fait d'un outil de développement destiné à faciliter la réalisation personnelle de montages comportant une EPROM. Les vraies EPROM ont l'inconvénient de requérir, avant toute nouvelle reprogrammation, un effacement aux rayons ultraviolets (UV). Toute modification du programme, aussi minime soit-elle, implique une procédure d'effacement suivie d'une reprogrammation. C'est là que, deus ex machina, entre en jeu notre simulateur d'EPROM. Fini les longues séances d'effacement et de reprogrammation. On pourra ainsi se limiter à une seule et unique programmation lorsque le programme sera, enfin, parfaitement au point.
Il faut reconnaître qu'en cette époque de composants à mémoire de programme Flash reprogrammable en site (d'où l'acronyme de ISP pour *In Site Programming*) et de composants dotés de capacités de mémoire de plus en plus importantes, on est en droit de se poser la question de l'utilité d'un tel outil.
Il n'en reste pas moins que nombre de réalisations font appel à des microcontrôleurs requérant une EPROM externe, qu'il s'agisse des familles du 8051, du 68HC11, du 80C5XX et de bien d'autres microcontrôleurs. Et c'est bien là justement le domaine d'action de prédilection de ce simulateur d'EPROM universel.

Le synoptique de principe

Le synoptique représenté en **figure 1** est classique pour ce type d'applications.
Le principe sur lequel repose un simulateur d'EPROM est de remplacer la mémoire morte (EPROM ou ROM) par de la mémoire vive (RAM) à double accès. Celle-ci est entourée de 2 verrous et épaulée par un microcontrôleur.
Comme dans tout simulateur ou émulateur d'EPROM qui se respecte nous trouvons au cœur du montage la RAM (IC5) chargée de prendre la place de l'EPROM (IC9) d'un montage–cible. Cette RAM est épaulée par quelques verrous (IC3 et IC4) en amont et, en aval, quelques tampons (IC6 à IC8). Nous verrons les fonctions de chacun de ces composants dans le paragraphe qui suit.
Ce qui distingue ce synoptique de ce que vous avez pu rencontrer jusqu'à présent est la présence d'un microcontrôleur, IC2. Ce composant, chargé du pilotage des verrous (*latches*) et la RAM se charge en outre de la gestion de la réception du code et de son traitement, est un microcontrôleur de la famille 8051, un 89AT2051 de l'écurie Atmel. La raison du choix de ce type de microcontrôleur est la présence d'un port série intégré et d'une EPROM interne dans laquelle prend place le programme de réception du code et de gestion du simulateur.
Nous verrons dans le paragraphe consacré au schéma comment ce synoptique se traduit en une électronique compacte.
Prenons les choses dans l'ordre.

Le schéma

Le schéma du simulateur d'EPROM représenté en **figure 2** reprend, dans les grandes lignes, la disposition adoptée pour le synoptique de la figure 1.
IC1, un MAX232, assure l'opération classique de conversion de tension d'un niveau RS-232 (±10 V) vers un niveau logique (+5 V/0 V) et ce dans les 2 sens. Ceci permet au microcontrôleur de communiquer avec d'autres périphériques par le biais de l'interface RS-232. La plupart des PC peuvent également s'accommoder de niveaux logiques (+5 V/0 V) mais cela implique de procéder à une inversion des signaux RxD et TxD !
IC2, un AT89C2051, constitue le coeur de ce montage. Il pilote les verrous (*latch*) IC3 et IC4 ainsi que les tampons IC6 à IC8. Ces circuits intégrés lui permettent d'attaquer la RAM, IC5; de par leur présence, ils font également en sorte que, soit le processeur, soit la circuiterie externe, mais jamais les deux

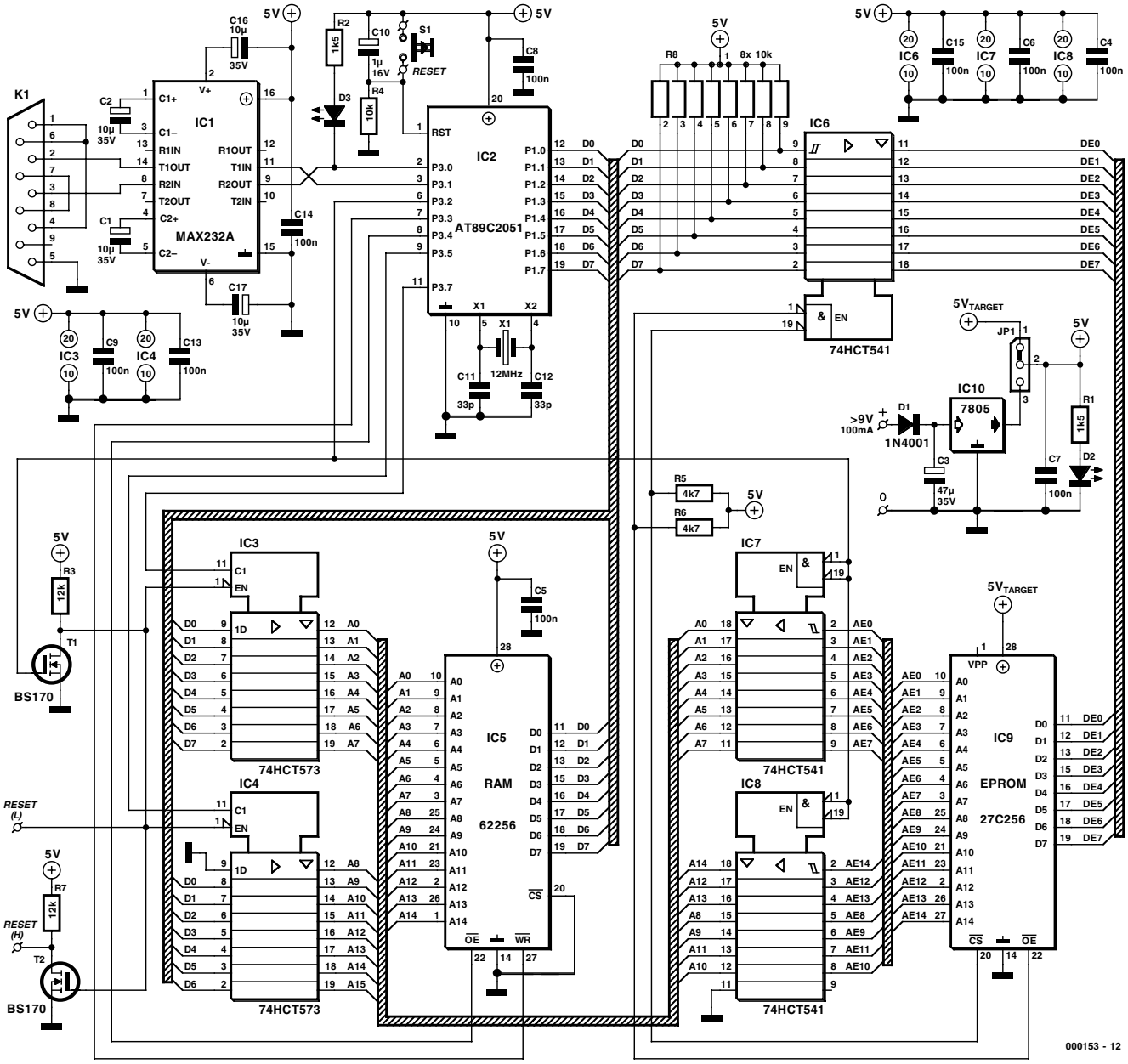


Figure 2. L'électronique du simulateur d'EPROM a été réduite à sa plus simple expression ce qui permet de réaliser un montage compact au professionnalisme indiscutable.

simultanément, puisse accéder à la RAM. Il devient impossible ainsi que 2 périphériques aient accès, au même moment, à la RAM.
IC5, la RAM, remplit également une fonction primordiale sachant que c'est elle qui simulera l'EPROM. Dans la présente application cette RAM est mise à contribution par 2 périphériques. L'électronique du simulateur d'EPROM basée sur le processeur se chargera de permettre l'écriture en RAM du programme à tester, le montage externe, la cible en fait, accédera à cette RAM pour y trouver le programme à exécuter.
Les verrous IC3 et IC4 s'occupent des lignes d'adresses de la RAM. Sur les instructions du processeur, ces circuits intégrés transfèrent les données présentes sur le bus de données interne, D0 à D7. Chacune des entrées de validation (*Latch Enable*) de ces 2 circuits est pilotée par une ligne d'entrée/sortie du processeur différente. Cette approche permet au microcontrôleur de piloter le bus d'adresses de la RAM. Une fois le programme chargé en RAM, ces 2 verrous passent à l'état de haute impédance de façon à éviter toute interférence au niveau de la RAM lorsque le système se trouve en mode de simulation.
Les tampons IC6 à IC8 assurent la « connexion » des lignes d'adresses et de données de la RAM avec la circuiterie externe.

L'alimentation

L'alimentation du simulateur d'EPROM peut se faire de 2 façons. La première, que nous qualifierons de standard, consiste à alimenter le montage par le biais d'un adaptateur secteur.
L'alimentation du montage est alors

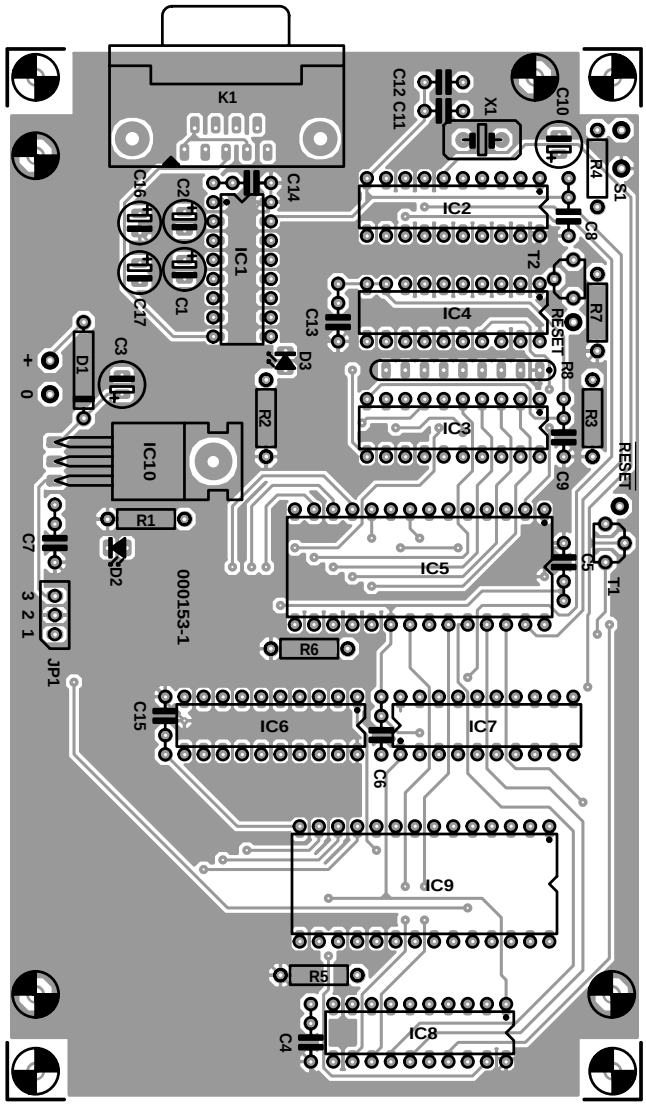
réalisée à partir d'un 7805, IC10, un régulateur de tension tripode classique. Ce composant fournit le +5 V requis par l'électronique. Comme ce composant peut fournir, refroidi correctement, jusqu'à 1,5 A et que notre montage ne consomme que quelque 100 mA nous sommes à l'abri de toute mauvaise surprise. La diode D1 permet de protéger le montage en cas d'inversion de la polarité des lignes d'alimentation. La LED D2 s'allume pour signaler la présence de la tension d'alimentation.
La seconde option, consiste à mettre à contribution l'alimentation du montage-cible vu que dans la majorité des cas ce dernier dispose également d'une tension de +5 V régulée. Si l'on opte pour cette seconde solution, on pourra s'offrir le luxe de n'implanter ni IC10, le régulateur 5 V tripode ni la diode D1 ni le condensateur C3.
Le cavalier JP1 pris à proximité du régulateur de tension permet de choisir le mode d'alimentation à utiliser : alimentation externe ou alimentation par le montage-cible.

Liaison série

Il faudra réaliser entre le PC et le microcontrôleur une interface capable de mettre en forme les signaux émis et reçus par chacun de ces communicants.
Cette interface a été réalisée à partir d'un classique MAX232 et de 4 condensateurs. Nous aurions pu opter pour une version CMS de ce composant pour loger cette interface dans la prise DB25 qui se connecte au PC. Mais en vue de simplifier la réalisation nous avons choisi de mettre la totalité des composants sur la platine, ce qui permet l'utilisation d'un câble sériel d'usine.
Une remarque en ce qui concerne la liaison RS-232. Celle-ci se fera par le biais d'un câble RS-232 standard (ne pas utiliser de câble du type modem-zéro !!!).
Il est temps maintenant de nous intéresser au…

…Fonctionnement

Le processus de travail du simulateur d'EPROM peut se subdiviser en 2 étapes : le chargement de la RAM et la simulation du programme du montage-cible. Pour la première fonction le PC envoie, par l'intermédiaire de son port série, le code hexadécimal destiné à être « stocké » dans la pseudo-



Liste des composants

Résistances :
R1,R2 = 1kΩ5
R3,R7 = 12 kΩ
R4 = 10 kΩ
R5,R6 = 4kΩ7
R8 = réseau de 8 résistances de 10 kΩ

Condensateurs :
C1,C2,C16,C17 = 10 µF/35 V
C3 =47 µF/35 V
C4 à C9,C13 à C15 =100 nF
C10 =1 µF/16 V
C11,C12 =33 pF

Semi-conducteurs :
D1 = 1N4001
D2,D3 = LED à haut rendement
T1,T2 = BS170
IC1 = MAX232 (Maxim)
IC2 = AT89C2051 (Atmel - programmé **EPS000153-41**)
IC3,IC4 = 74HCT573
IC5 = 62256 (RAM)
IC6 à IC8 = 74HCT541
IC9 = D.U.T. (*Device Under Test*)
IC10 = 7805

Divers :
K1 =embase Sub-D à 9 contacts femelle encartable
PC1 à PC4= picot
JP1 = embase autosécable à 3 contacts + cavalier
S1 = bouton-poussoir à contact travail
X1 =quartz 12 MHz

Figure 3a. La sérigraphie de l'implantation des composants de la platine double face à trous métallisés.

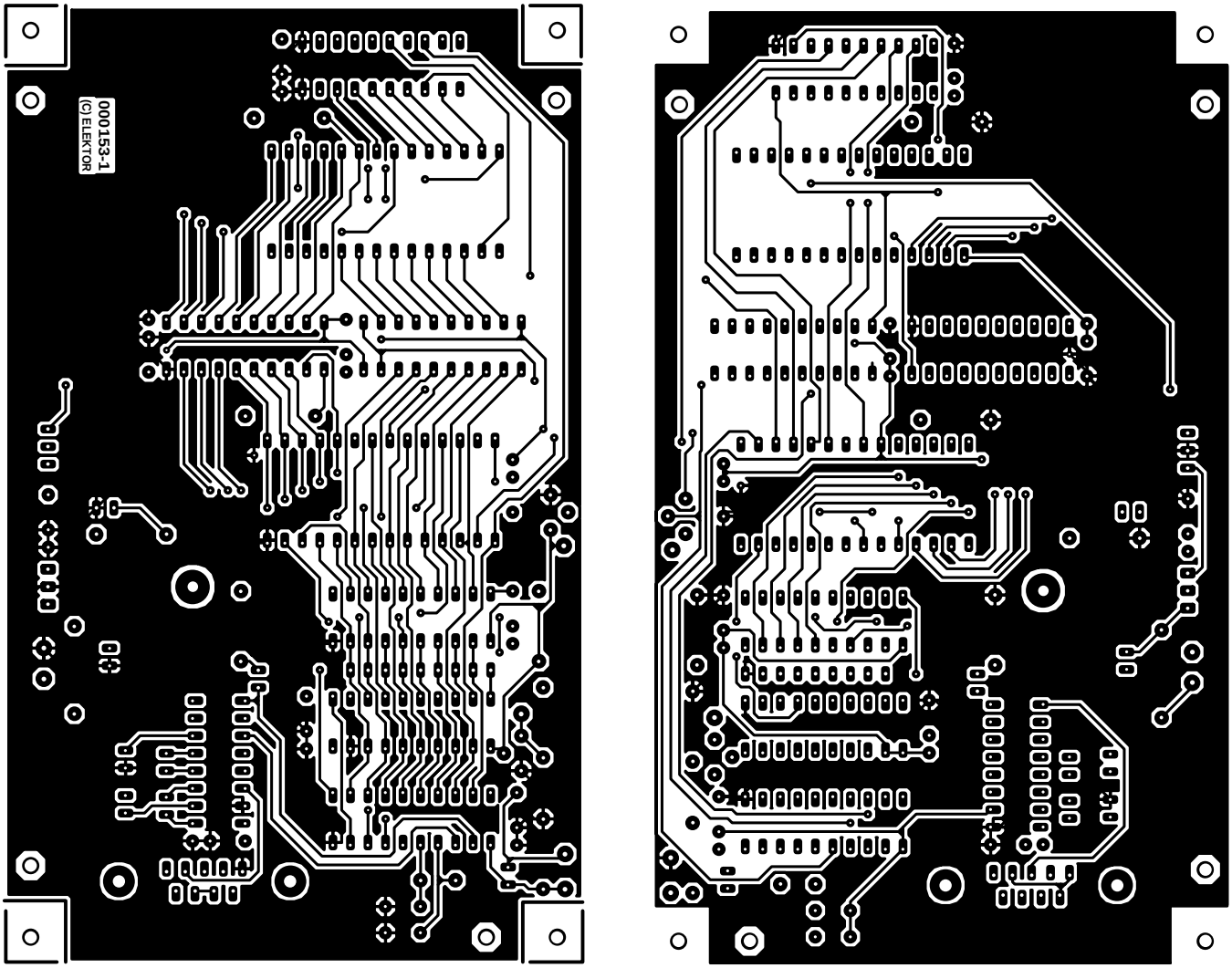


Figure 3. La sérigraphie du dessin des pistes de la platine double face à trous métallisés dessinée à l'intention de cette réalisation.

EPROM. Pour la seconde, il faut, pour que le montage-cible prenne le relais, faire passer le bit P3.2 du microcontrôleur à l'état bas pour activer les drivers 3 états IC7 et IC8. Les transistors MOS du type BS170, T1 et T2, mettent le montage-cible en Reset.

Voyons-en le détail.

Dès la mise sous tension du simulateur d'EPROM le processeur fait en sorte que l'électronique soit prête à permettre l'écriture de données vers la RAM. Pour cela, il force au niveau haut la quasi-totalité des lignes de son port P3, exception faite des lignes de port P3.7 et P3.5.

Il faut, pour que le microcontrôleur prenne le contrôle de la RAM que la ligne de port P3.2 soit mise à l'état haut.

Ainsi, pour charger la RAM, on va présenter sur le port P1 du microcontrôleur l'adresse haute de la première donnée puis on ouvre et on ferme le driver 3 états IC4 par l'intermédiaire du bit P3.5 du microcontrôleur afin de bloquer cette adresse.

On réitère l'opération pour l'adresse basse en utilisant le bit P3.7 pour commander le driver 3 états IC3.

P3.2 a pour mission de mettre les sorties des tampons IC6 à IC8 à l'état de haute impédance de manière à éviter toute influence de l'électronique externe sur le fonctionnement de la RAM pendant le processus d'écriture de données dans la RAM.

Ce même signal subit également une inversion introduite par la paire R3/T1. Le signal inversé ainsi obtenu sert à activer IC3 et IC4 de manière à leur permettre de piloter les lignes d'adresses de la RAM. Le signal de P3.2 est en outre mis à la disposition du circuit-cible par le biais d'une paire de picots, RESET(L) et RESET(H). On pourra utiliser ce signal en vue de mettre l'électronique de la cible en état d'initialisation (*Reset*) tant que la RAM ne contient pas encore le programme dans sa totalité.

Ceci fait, c'est-à-dire une fois que le programme complet a été transféré dans la RAM, le processeur de manifeste par le biais du port RS-232 (série) au travers d'un message. La liaison avec le port RS-232 se fait de façon classique, à savoir par le biais d'un circuit de commande MAX232.

Il faut que le PC transmette le fichier en format Intel Hex vers le port RS-232. Le processeur se charge, à la réception de ces données (notons que la LED D3 s'allume au passage de chacun des caractères transmis du PC vers le simulateur), de faire en sorte que chacune d'entre elles soit écrite à l'adresse en RAM correcte.

Pour ce faire, IC2 place les lignes d'adresses A0 à A7 sur le port P1, ce qui se traduit par l'apparition d'une impulsion sur P3.7. Le verrou IC3 prend en compte ce mot. Le même processus se répète en ce qui concerne les lignes d'adresses A8 à A13. Les données qu'elles véhiculent sont stockées dans le verrou IC4 à l'apparition d'une impulsion sur la ligne de port P3.5. Pour finir, la donnée proprement dite est placée sur le port P1, opération à laquelle succède l'application d'une impulsion négative sur la ligne P3.3. Cette dernière pilote l'entrée d'écriture ($\overline{WR}$) de la RAM. Lors de l'apparition de cette fameuse impulsion négative, la RAM transfère la donnée présente sur le port P1 à l'adresse prévue.
Ensuite on place la donnée sur le port P1 et on positionne la RAM en écriture ($\overline{WR}$ mis à l'état bas) par le bit P3.3 afin d'écrire la donnée à l'adresse choisie.
On réutilise la même procédure pour la transmission de toutes les données du code.
Une fois la totalité du fichier Intel Hex reçue, le processeur fait passer le circuit en mode de simulation, c'est-à-dire qu'il met la RAM en mode lecture ; pour ce faire, il force au niveau bas la ligne P3.3 et sélectionne la validation de sortie $\overline{OE}$ (Output Enable) par la ligne P3.4.
En forçant la ligne P3.3 au niveau bas, il active les sorties des circuits tampons, IC7 et IC8, tout en mettant simultanément, par le biais de leurs entrées $\overline{EN}$ respectives, à haute impédance les sorties des verrous.
En complément de ces 2 fonctions importantes, le signal de P3.3 est également chargé de rendre inactives les 2 sorties de remise à zéro (RAZ) du simulateur. Si l'on relie l'une de ces 2 sorties de RAZ au montage-cible, cette électronique externe se verra automatiquement remise à zéro pendant l'opération de chargement du fichier en RAM. Dès que la RAM est chargée, le montage-cible se réinitialisera automatiquement (une caractéristique on ne peut plus pratique surtout lorsque le montage-cible ne dispose pas de son propre bouton de réinitialisation (Reset).
La ligne P3.4 attaque la RAM de manière à ce que cette dernière mette en permanence des données sur le bus de données interne.
Pour éviter que, dans ces conditions, la RAM ne fournisse en permanence des données sur le bus de données externe, les 2 entrées de validation du tampon IC6 sont reliées aux lignes $\overline{OE}$ (*Output Enable*) et $\overline{CS}$ (*Chip Select*) de l'électronique externe. Cette approche garantit le transfert de données de la RAM vers le bus de données externe que lorsque le montage-cible adresse le simulateur d'EPROM.
Si l'on se trouve dans l'obligation d'écrire de nouvelles données dans la RAM, le programme précédent n'étant pas encore opérationnel à 100%, il suffit (!!!) d'appuyer sur le bouton de remise à zéro et de redémarrer le processus à son début...

La réalisation

Comme le montre la photo en début d'article il aurait été possible de réaliser une platine encore plus compacte, surtout si nous avions intégré le MAX232 dans la coquille du câble sériel. Mais l'approche que nous avons adoptée permet une réalisation mise à la portée de tous nos lecteurs.
La mise en place des composants sur la platine double face à trous métallisés dont on retrouve en **figure 3** le dessin des pistes et la sérigraphie de l'implantation des composants n'appelle que peu de remarques particulières. De par sa caractéristique de double face elle ne comporte pas de pont de câblage (il ne maquerait plus que cela !).
Comme d'accoutumée on commencera par l'implantation des composants de petite taille, résistances, condensateurs, transistors. Attention lors de leur positionnement à ce qu'ils n'interférent pas avec les (supports de) circuits intégrés. On veillera au positionnement correct du réseau de résistances R8 (point commun orienté vers le bord de la platine). On pourra, en ce qui concerne les supports, se limiter à 3 supports (de bonne qualité) seulement, un pour le processeur, un second pour la RAM et un dernier pour le connecteur à câble en nappe allant vers l'EPROM à simuler. Attention à la polarité de IC7 qui est orienté à 180° par rapport aux autres circuits intégrés (compte non tenu d'IC1 et d'IC10).
Il est recommandé, avant de mettre en place le processeur et la RAM, commencer par s'assurer de la présence, aux points prévus, de la tension d'alimentation. Par son allumage la LED D2 signale la présence de la tension d'alimentation.

Le logiciel

Le programme stocké dans le microcontrôleur a été écrit de telle façon à ce que l'on puisse utiliser Hyperterminal de Windows pour communiquer avec le simulateur d'EPROM.
Le chargement du code se fera à partir d'une liaison série à une vitesse de transfert de 4 800 bauds du PC vers le simulateur.
Ce simulateur reconnaît le format de fichier le plus répandu à savoir celui baptisé Hexadécimal Intel (Intel Hex).
On pourra utiliser, pour la communication entre le PC et le simulateur d'EPROM, le programme Hyperterminal (qui fait partie des outils standard de Windows). Le transfert du fichier Intel Hex doit avoir lieu par le biais d'une transmission ASCII standard (on n'utilisera pas, partant, un autre protocole tel que Z-modem, ou autre Kermit).
Le paramétrage de transmission est partant 4800 bauds/8 bits/N/1.
Sous Hyperteminal on optera pour la fonction *Send Textfile*.
Pour ceux d'entre vous auxquels cette évocation de Z-modem dit encore quelque chose, la transmission sous DOS pourra se faire par le biais de l'instruction
`COPY INTEL.HEX COM1:`
et, sous Linux, ce système d'exploitation ne cessant de voir le nombre de ses adeptes croître, on fera :
`Cat INTEL.HEX \dev\xxx`
(xxx représentant dans cette instruction le port auquel le simulateur d'EPROM est connecté).

(000153)

Littérature

– Émulateur d'EPROM II,
B.C. Zschocke & N. Breidohr,
Elektor 173, novembre 1992
page 52 et suivantes

– simEPROM,
B.C. Zschocke,
Elektor 137, novembre 1989
page 26 et suivantes