P.R. Boldt

eprogrammateur

lampe à bronzer + eprogrammateur = éprommes-frites

Dans la gamme des accessoires de micro-programmation proposés par Elektor, il manquait une pièce de choix! Dans le numéro de décembre 1981, nous avons décrit un premier programmateur d'EPROM dont l'usage était limité par sa simplicité. Nous en arrivons maintenant au programmateur universel que nous vous avions annoncé; "universel", c'est à dire conçu pour les micro-ordinateurs domestiques tels que le SC/MP ou le Junior Computer, et adaptable à la plupart des autres systèmes. Les dimensions du circuit sont plus que raisonnables, puisqu'il tient sur une carte au format européen. Il accepte aussi bien des EPROM du type 2716 que 2732, qu'il permet de relire pour en vérifier le contenu.

Publier à intervalle aussi rapproché deux programmateurs d'EPROM relève apparemment de la gageure. Nous n'avons pourtant pas hésité à le faire, en raison de la très forte demande dont nous avons eu de nombreux échos (continuez de nous faire savoir ce que vous souhaitez voir publié!).

Le circuit a été doté d'un connecteur universel, permettant l'adaptation à tout système à microprocesseur, bien que nous le destinions en priorité à nos lecteurs équipés d'un SC/MP ou d'un Junior Computer.

Avant d'aborder le programmateur proprement dit, il nous faut parler de la programmation. Dans le magazine de décembre 1981, à propos du programmateur pour 2650, nous avons déjà expliqué comment les choses se passaient. Nous ne reprenons ici que les aspects essentiels. On applique une tension de programmation de 25 V à la broche 21 d'une 2716 (broche 20 d'une 2732). Il suffit ensuite d'une impulsion de programmation de 50 ms appliquée à l'entrée $\overline{CE}$ pour programmer les données présentes à l'adresse de travail. Le système à microprocesseur pour lequel on désire programmer des EPROM, va donc être mis à contribution pour la programmation elle-même. Ce qui ne va d'ailleurs pas sans quelques petites difficultés avec le Junior Computer; il n'est pas possible, en effet, de "caler" le 6502 dans un cycle d'écriture pendant les 50 ms requises pour la programmation d'une donnée dans l'EPROM. Ceci nous contraint à adopter un circuit "tampon verrouillable", qui gèle les données pendant le temps qu'il faut. Ce problème ne se pose pas avec le SC/MP auquel on adaptera le programmateur après avoir modifié la position de quelques straps.

On notera par ailleurs que le circuit a été conçu de sorte que les "manipulations" du processus de programmation sont effectuées par des fonctions matérielles.

Le circuit

Renversons la vapeur à présent, et examinons le programmateur avant de détailler la programmation. Le circuit complet est donné par la figure 1. Nous avons déjà évoqué la présence de tampons, destinés à conserver les données et les adresses; il s'agit de IC1, IC2, IC3 et IC4. Selon les type d'EPROM à programmer, il y aura 11 (pour les 2716) ou 12 (pour les 2732) bits d'adresses dans les tampons correspondants. La douzième ligne d'adresse est connectée à l'EPROM à l'aide de S1 (choix entre 2716/2732).

Les lignes de données D0 . . . D7 sont reliées à deux verrous parallèles (IC1 et IC2). On remarquera que ces derniers sont montes tête-bêche, puisque les sorties de l'un attaquent les entrées de l'autre, et inversement. Le but de ce petit jeu est que l'on puisse écrire dans l'EPROM (pour la programmer, bien sûr) et y lire, pour vérifier son contenu.

Venons-en au décodage d'adresse; celui-ci est réalisé par un comparateur à quatre bits, IC5. D'un côté, ce circuit est relié aux lignes d'adresse A12 . . . A15, tandis que l'autre, il reçoit les signaux appliqués par les interrupteurs S3 . . . S6. Un interrupteur fermé délivre un niveau logique bas; aux lignes A12 . . . A15 correspondent respectivement les interrupteurs S3 . . . S6. Cette combinaison permet de couvrir 4 Koctets, ce qui peut paraître beaucoup à qui n'est pas familier des 2732 . . . Pour les 2716, dont la capacité est deux fois moins grande, cela signifie que l'on pourra les adresser soit à partir de X000, soit à partir de X800.

IC6 est un multivibrateur astable (avec start/stop) dont la constante de temps délivre une durée d'impulsion de 10 ms. La sortie de ce circuit est reliée via un inverseur au registre à décalage IC7, tandis que l'entrée d'initialisation (reset) d'IC6 est reliée à la sortie Q de FF2. L'entrée horloge de ce dernier (ainsi que celle de FF1) reçoit, via N6, des impulsions qui proviennent du connecteur reliant le programmateur au système à microprocesseur (broche 31a). Les deux bascules reçoivent donc un signal d'horloge chaque fois que le processeur émet le signal d'écriture. Au moment où l'espace mémoire à décoder est identifié, les lignes A et B d'IC5 sont au même niveau logique, et le signal "A=B" (niveau logique haut) apparaît sur la broche 6. Cette impulsion apparaît immédiatement après (une fois que les bascules on reçu un signal d'écriture du processeur en mode programmation d'EPROM) à la sortie Q de FF2, de sorte que le multivibrateur IC6 est déclenché. FF1 bascule en même temps que FF2, permettant ainsi l'application de la tension de programmation de 25 V sur la broche convenable de l'EPROM. Cette tension est fournie par un petit circuit auxiliaire construit autour de T1, T2 et T3, avec les composants associés au grand complet: un condensateur de filtrage précédé par un pont redresseur. Il ne reste plus qu'à appliquer le secondaire d'un transfo de 24 V. C'est donc la

1

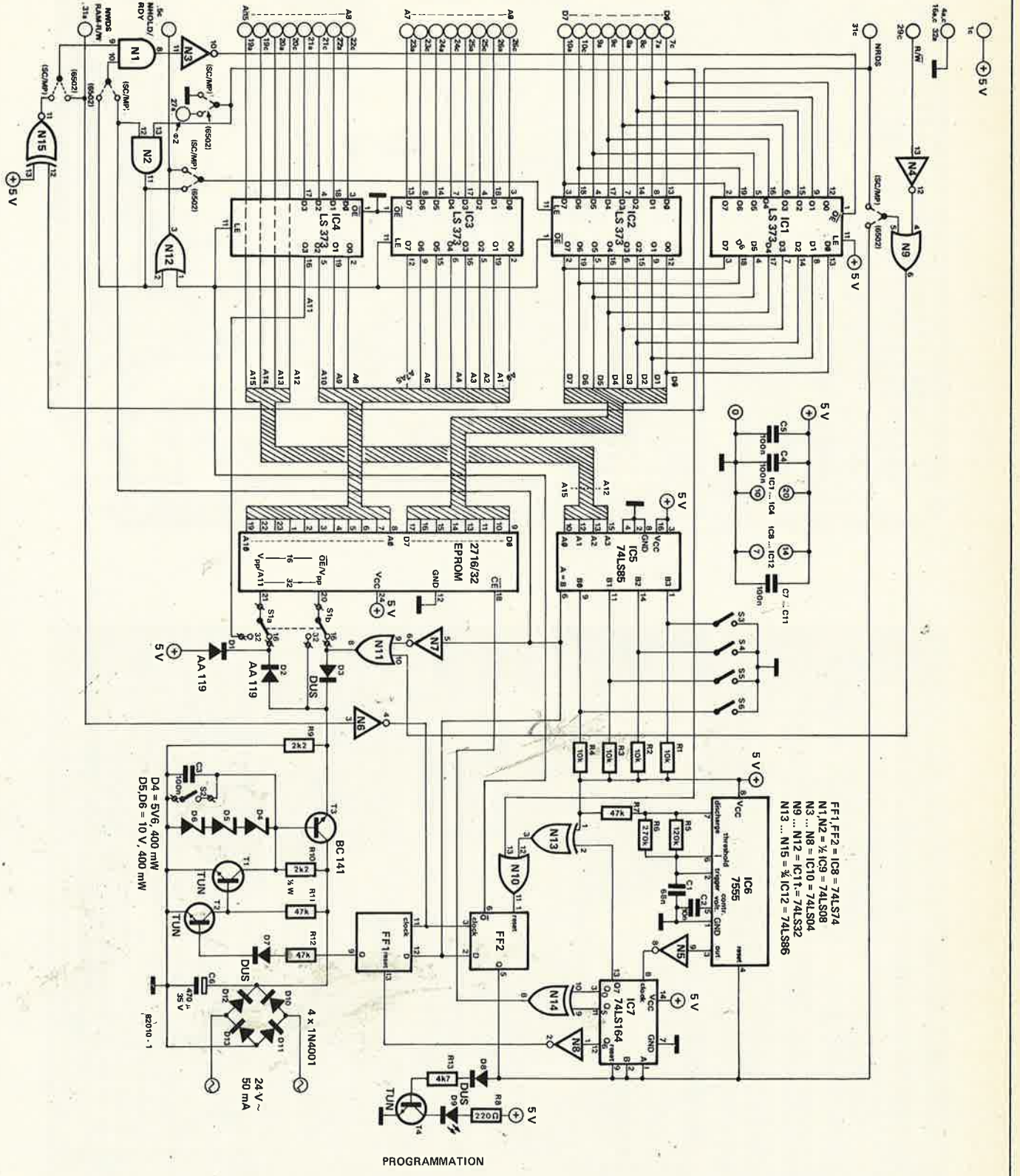

Figure 1. Le circuit complet du programmateur d'EPROM. Le pavé central n'est rien d'autre que l'EPROM à programmer, avec au-dessus le circuit de décodage d'adresses. A gauche, on trouve les tampons d'adresses et de données, et à droite la logique de contrôle du déroulement d'une séquence de programmation. En bas à droite, le circuit d'alimentation pour la tension de programmation de 25 V.

2

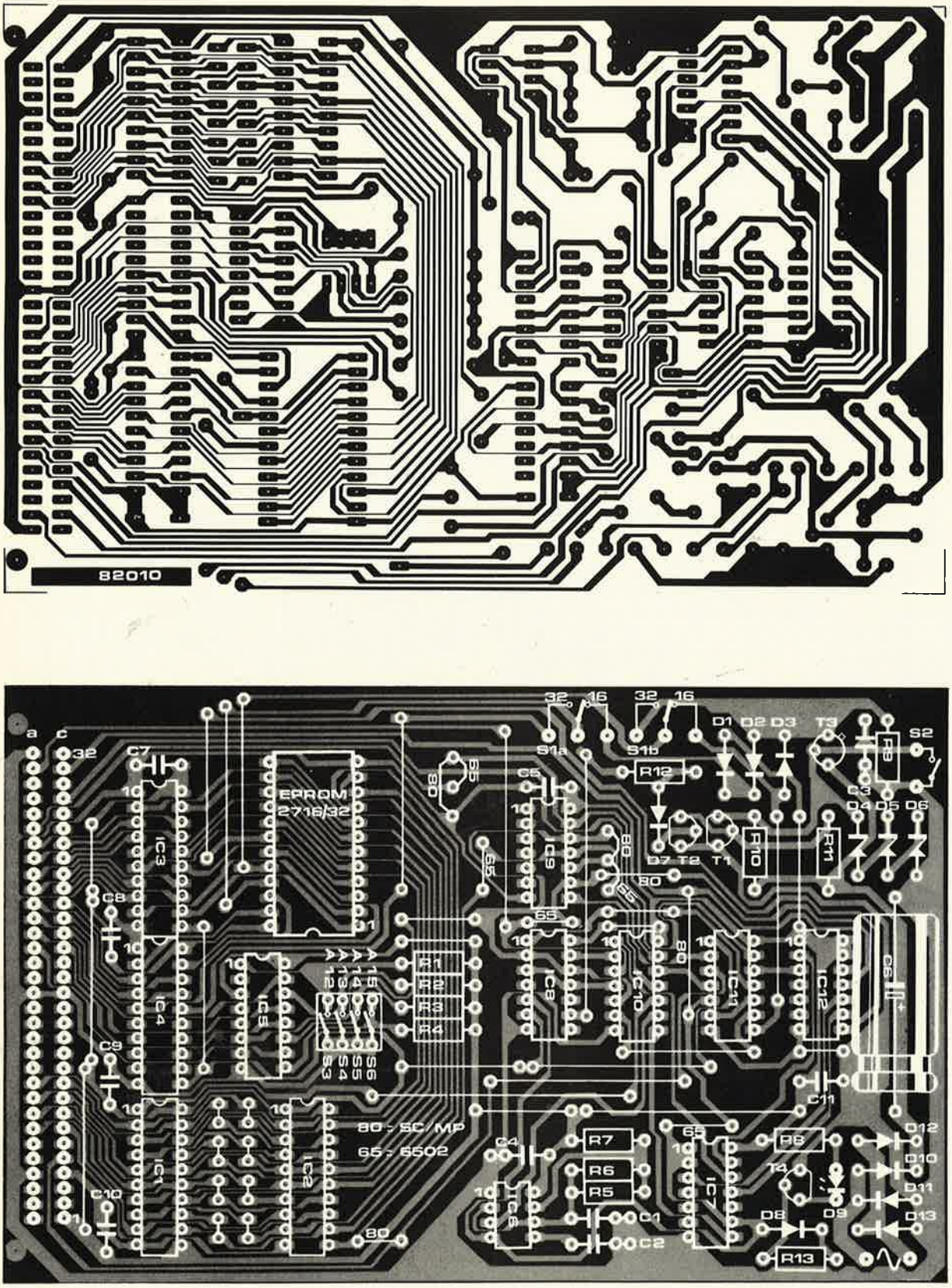

Figure 2. Le dessin du circuit imprimé, avec sérigraphie pour l'implantation des composants du programmateur d'EPROM.

Liste des composants

Résistances:
R1 . . . R4 = 10 k
R5,R6 = 220 k
R7,R11,R12 = 47 k
R8 = 220 Ω
R9 = 2k2
R10 = 2k2/½W
R13 = 4k7

Condensateurs:
C1 = 68 n
C2 = 10 n
C3,C4,C5,C7 . . . C11 = 100 n
C6 = 470 µ/35 V

Semiconducteurs:
D1,D2 = AA 119
D3,D7,D8 = DUS
D4 = diode zener 5V6/400 mW
D5,D6 = diode zener 10 V/400 mW
D9 = LED
D10 . . . D13 = 1N4001
T1,T2,T4 = TUN
T3 = BC 141
IC1 . . . IC4 = 74LS373
IC5 = 74LS85
IC6 = 7555
IC7 = 74LS164
IC8 = 74LS74
IC9 = 74LS08
IC10 = 74LS04
IC11 = 74LS32
IC12 = 74LS86

Divers:
S1 = inverseur bipolaire
S2 = interrupteur
S3 . . . S6 = 4 interrupteurs miniature en boîtier DIL
Support pour circuit intégré (avec levier de blocage) 24 broches
Connecteur 64 broches (90°) selon DIN 41612

sortie de FF1 qui commande l'application de la tension de programmation . . . si tant est que S2 est dans la bonne position. Lorsque celui-ci est fermé, le signal de commande provenant de la sortie de FF1 via les transistors T2 et T1 sur la base de T3, est court-circuité.

Mais revenons maintenant au mode "programmation"; nous avions laissé IC6 au moment où il était déclenché par le basculement de FF2. Au rythme des impulsions fournies par le temporisateur, IC7 est chargé de "1" (niveau logique haut). 10 ms après la mise en service de la tension de programmation de 25 V, la sortie Q0 d'IC7 passe au niveau logique haut à son tour, de même que la broche $\overline{CE}$ de l'EPROM à programmer. La configuration EXOR (N14) entre les lignes Q0 et Q5 d'IC7 maintient le niveau logique haut sur la broche $\overline{CE}$ de l'EPROM à programmer pendant 50 ms. Après 10 ms, c'est la sortie Q6 qui passe au niveau haut, de sorte que FF1 est initialisé, et la tension de programmation supprimée.

Pendant tout ce temps, il faut considérer que les données et les adresses sont verrouillées dans les tampons adéquats.

On remarquera encore la présence de l'indicateur lumineux à LED autour de T4, celui-ci fonctionne lorsque la programmation est en cours.

Tout ce processus n'a pas lieu pendant une opération de lecture pour la bonne et simple raison que le processeur ne délivre pas de signal d'écriture à ce moment là! Les sorties d'IC2 passent en état haute impédance, tandis qu'IC1 est mis en fonction. Le bus de données du système à microprocesseur est donc en "prise directe" sur l'EPROM.

L'inverseur S1 permet de déterminer le type d'EPROM à programmer ou à lire. Sur le circuit, il est en position "2716"; et la tension de programmation est appliquée à la broche 21 de l'EPROM, tandis que l'entrée $\overline{OE}$ est reliée à N11. Dans l'autre position de S1 (2732), la ligne d'adresse A11 est reliée à la broche 21, et la tension de programmation est appliquée à la broche 20.

Les quelques portes logiques non décrites jusqu'ici (N1 . . . N4, N9, N11, N12 et N15) sont nécessaires à l'adaptation du programmateur aux différents types de signaux de contrôle et de commande, selon le processeur hôte.

La réalisation

La figure 2 propose le dessin d'un circuit imprimé pour le programmateur d'EPROM. Tous ceux qui ont déjà derrière eux la réalisation d'un ou plusieurs systèmes à microprocesseurs (couronnée de succès, bien sûr!) n'éprouveront aucune difficulté avec cette carte au format européen. Nous recommandons le choix d'un support d'EPROM de très bonne qualité, dans lequel il fuffit de poser le circuit sans forcer; le verrouillage mécanique et électrique n'ayant lieu qu'ensuite, grâce à un petit levier, on s'évitera ainsi toutes les difficultés que posent les supports ordinaires, en raison du grand nombre de broches des 2716 ou 2732.

Les straps sont à implanter selon le type de processeur utilisé, comme indiqué sur le schéma et la sérigraphie. Hormis la tension de programmation fournie par un transformateur de 24 V, tous les autres signaux passent par le connecteur universel à 64 broches.

SC/MP + programmateur

Les conditions d'utilisation du programmateur avec le SC/MP sont très confortables. On commence par définir le décodage d'adresse de la carte de programmation, puis on la connecte au bus du système, après avoir fermé S2 et inséré une EPROM vierge sur le support. On peut alors ouvrir S2 et appliquer la tension de programmation. On veillera à refermer S2 dès la fin de la programmation.

Pour le SC/MP, il n'est pas nécessaire de disposer d'un logiciel de programmation particulier, ELBUG fait tout à fait l'affaire. Si l'on désire faire de la programmation à des adresses isolées, il suffit d'introduire MO . . . YYYY, où YYYY est l'adresse à laquelle on veut programmer.

Si l'EPROM est vierge, on voit alors apparaître FF sur l'affichage. Si la donnée à programmer est par exemple Ø8, il suffit d'introduire Ø8, et le tour est joué.

Lorsque l'on désire programmer des blocs de données plus importants, il faudra que ceux-ci soient introduits en RAM d'abord, puis transférés de là en EPROM (BL . . . SSSS, EEEE, BBBB, où S = adresse de départ, E = adresse de fin et B = adresse de destination du bloc).

La lecture se fait comme toute opération de lecture ordinaire.

Junior Computer + programmateur

Cette fois il nous faut un logiciel de programmation spécifique au Junior Computer. Celui-ci vous est fourni par le tableau 1, et commence à l'adresse Ø2ØØ pour finir à Ø277. Une fois que ce programme est en mémoire vive, le programmateur d'EPROM peut être mis en place sur le bus du Junior Computer.

Le programme que nous venons d'évoquer pourra être stocké sur cassette, ou, plus logiquement, dans une EPROM! Dans ce dernier cas, il faudra veiller à modifier les adresses absolues de saut. Il reste de la place dans l'EPROM contenant Tape Monitor; de ØC8Ø à ØFFF, il n'y a pour l'instant que des emplacements vierges. C'est à dire que de Ø48Ø à Ø7FF, dans l'EPROM de TM considérée hors le décodage d'adresse du Junior Computer, nous ne trouvons que le caractère de remplissage FF. Il suffit donc de reloger le programme en ØC8Ø, (après avoir modifié les adresses absolues de saut (toutes les instructions comportant trois octects, et dont le troisième est Ø2)). Examinons à présent les trois routines PROGRAM, DUPLICATE et VERIFY.

La routine de programmation

L'octet de poids fort, et celui de poids faible de l'adresse à laquelle doit commencer la programmation sont placés en MOVH et MOVL (ØØØ4, ØØØ5). On peut alors lancer le programme du tableau 1 (Ø2ØØ). Sur l'affichage apparaîssent alors l'adresse et la donnée correspondant à la première adresse de programmation (spécifiée par l'utilisateur dans MOVL/MOVH). Si l'EPROM est vierge à cet endroit-là, les deux afficheurs de droite indiquent FF. Supposons que l'on veuille introduire le code opératoire A9, on commence par introduire, via le clavier, un A (l'affichage ne réagit pas), puis un 9. Aussitôt l'affichage s'éteint (pendant 70 ms) puis se rallume, indiquant toujours la même adresse, mais avec la donnée programmée.

Pour faire apparaître l'adresse suivante, il suffit d'actionner la touche +, et pour y introduire une nouvelle donnée, il suffit de réitérer la procédure que nous venons de décrire.

Pour une lecture seule de l'EPROM, il suffit de rester (ou de retourner) dans le

```
0010: 0200                    ORG   $0200
0020:
0030:                 DATE :  10-7-'81
0040:
0050:
0060:                 PAGE ZERO DATA BUFFERS :
0070:
0080: 0200            SAL     *     $0000  DATA BLOCK START ADDRESS
0090: 0200            SAH     *     $0001
0100: 0200            EAL     *     $0002  DATA BLOCK END ADDRESS + 1
0110: 0200            EAH     *     $0003
0120: 0200            MOVL    *     $0004  EPROM PROGRAM START ADDRESS
0130: 0200            MOVH    *     $0005
0140:
0150: 0200            INH     *     $00F9  DISPLAY BUFFER  ( DATA )
0160: 0200            POINTL  *     $00FA     "       "    ( ADDRESS L )
0170: 0200            POINTH  *     $00FB     "       "    ( ADDRESS H )
0180:
0190:                 EXTERNAL SUBROUTINES :
0200:
0210: 0200            GETBYT  *     $1D6F
0220: 0200            SCAND   *     $1D88
0230:
0240:                 JUNIOR MONITOR START :
0250:
0260: 0200            RESET   *     $1C1D
0270:
0280:
0290:                 PROGRAM START ADDRESS     : $0200  ( PROG )
0300:                 DUPLICATE START ADDRESS   : $0222  ( DUPL )
0310:                 VERIFY START ADDRESS      : $0233  ( VERIFY )
0320:
0330:
0340:                 ****************
0350:                 EPROM-PROGRAMMER
0360:                 ****************
0370:
0380: 0200 20 55 02   PROG    JSR   TRF     TRANSFER MOVL(H) TO POINTL(H)
0390: 0203 A0 00      PRGR    LDYIM $00     CLEAR Y-REGISTER
0400: 0205 B1 FA              LDAIY POINTL  GET DATA SPECIFIED BY POINTL(H) AND
0410: 0207 85 F9              STAZ  INH     STORE THIS IN DISPLAY BUFFER INH
0420: 0209 20 6F 1D           JSR   GETBYT  READ TWO HEXKEYS AND STORE THEIR VALUE IN THE
0430:                                       ACCUMULATOR. RETURN WITH N=1 IF
0440:                                       ONLY HEXKEYS WHERE DEPRESSED.
0450:                                       IF A COMMAND KEY WAS DEPRESSED,
0460:                                       RETURN WITH N=0
0470: 020C 10 07              BPL   PR      COMMAND KEY DEPRESSED?
0480: 020E A0 00              LDYIM $00     CLEAR Y-REGISTER
0490: 0210 91 FA              STAIY POINTL  PROGRAM THE CONTENTS OF THE ACCUMULATOR IN
0500:                                       THE EPROM MEMORY LOCATION
0510:                                       SPECIFIED BY POINTL(H)
0520: 0212 4C 03 02           JMP   PRGR
0530: 0215 C9 12      PR      CMPIM $12
0540: 0217 D0 35              BNE   PRGR    +KEY?
0550: 0219 E6 FA              INCZ  POINTL  INCREMENT ADDRESS BY ONE
0560: 021B D0 E6              BNE   PRGR
0570: 021D E6 FB              INCZ  POINTH
0580: 021F 4C 03 02           JMP   PRGR
ID=02
0010: 0222 20 55 02   DUPL    JSR   TRF     TRANSFER MOVL(H) TO POINTL(H)
0020: 0225 A0 00      DU      LDYIM $00
0030: 0227 B1 00              LDAIY SAL     GET DATA SPECIFIED BY SAL(H)
0040: 0229 91 FA              STAIY POINTL  PROGRAM THE CONTENTS OF THE ACCUMULATOR IN
0050:                                       THE EPROM MEMORY LOCATION
0060:                                       SPECIFIED BY POINTL(H)
0070: 022B 20 5E 02           JSR   INCMNT  INCREMENT SAL(H) AND POINTL(H) BY ONE
0080: 022E D0 F5              BNE   DU      NOT LAST ADDRESS
0090: 0230 4C 1D 1C           JMP   RESET   RETURN TO JUNIOR MONITOR
0100:
0110: 0233 20 55 02   VERIFY  JSR   TRF     TRANSFER MOVL(H) TO POINTL(H)
0120: 0236 A0 00      VER     LDYIM $00
0130: 0238 B1 FA              LDAIY POINTL  GET DATA SPECIFIED BY POINTL(H)
0140: 023A D1 00              CMPIY SAL     COMPARE THIS DATA WITH DATA SPECIFIED BY SAL(H)
```

```
0150: 023C F0 0F            BEQ   NEXT    DATA EQUAL?
0160: 023E 20 88 1D ANYKEY  JSR   SCAND   DISPLAY EPROM ADDRESS AND DATA
0170: 0241 D0 FB            BNE   ANYKEY  ANY KEY DEPRESSED?
0180: 0243 20 88 1D         JSR   SCAND   DISPLAY EPROM ADDRESS AND DATA
0190: 0246 D0 F6            BNE   ANYKEY  ANY KEY DEPRESSED?
0200: 0248 20 88 1D NKEY    JSR   SCAND   DISPLAY EPROM ADDRESS AND DATA
0210: 024B F0 FB            BEQ   NKEY    NO KEY DEPRESSED?
0220: 024D 20 5E 02 NEXT    JSR   INCMNT  INCREMENT SAL(H) AND POINTL(H) BY ONE
0230: 0250 D0 E4            BNE   VER     NOT LAST ADDRESS?
0240: 0252 4C 1D 1C         JMP   RESET   RETURN TO JUNIOR MONITOR
0250:
0260:                       ***********
0270:                       SUBROUTINES
0280:                       ***********
0290:
0300: 0255 A5 04      TRF   LDAZ  MOVL
0310: 0257 85 FA            STAZ  POINTL  TRANSFER MOVL TO POINTL
0320: 0259 A5 05            LDAZ  MOVH
0330: 025B 85 FB            STAZ  POINTH  TRANSFER MOVH TO POINTH
0340: 025D 60               RTS
0350:
0360: 025E 20 88 1D INCMNT  JSR   SCAND   DISPLAY FOR ABOUT 5MS POINTH, POINTL
0370:                                     AND INH ( = EPROM ADDRESS AND DATA
0380:                                     ON THIS ADDRESS )
0390: 0261 E6 00            INCZ  SAL     INCREMENT SAL(H) BY ONE
0400: 0263 D0 02            BNE   INCDA
0410: 0265 E6 01            INCZ  SAH
0420: 0267 E6 FA      INCDA INCZ  POINTL  INCREMENT POINTL(H) BY ONE
0430: 0269 D0 02            BNE   COMP
0440: 026B E6 FB            INCZ  POINTH
0450: 026D A5 01      COMP  LDAZ  SAH
0460: 026F C5 03            CMPZ  EAH     COMPARE EAH WITH SAH
0470: 0271 D0 04            BNE   RTRN    EAH NOT EQUAL SAH?
0480: 0273 A5 00            LDAZ  SAL
0490: 0275 C5 02            CMPZ  EAL     COMPARE EAL WITH SAL
0500: 0277 60         RTRN  RTS
```

Tableau 1. Listing du programme nécessaire à la programmation avec le Junior Computer.

moniteur, et d'actionner la touche + pour aller d'adresse en adresse.

Routine de duplication

Les lecteurs qui auraient craint, au vu de qui précède, que toutes les données à programmer doivent l'être une à une, peuvent se rassurer. La routine DUPLICATE permet de transférer les données par blocs entiers dans l'EPROM. Il faut commencer par spécifier l'adresse de début du bloc à transférer (SAL, SAH aux emplacements 0000, 0001), puis l'adresse de fin + 1 (attention! l'adresse de fin **plus une**) (EAL, EAH aux emplacements 0002, 0003); et enfin l'adresse à laquelle le premier octet doit être chargé dans l'EPROM (MOVL, MOVH aux emplacements 0004, 0005). On peut alors lancer le programme de duplication (0222); l'affichage s'éteint. Après chaque programmation d'octet, le Junior se manifeste brièvement en affichant l'adresse de la donnée qui vient d'être programmée, ainsi que cette donnée elle-même. Lorsqu'il est arrivé au bout du bloc à programmer, le Junior Computer affiche la dernière adresse, **plus une** . . .

La routine de vérification

Il s'agit à présent de comparer le contenu de l'EPROM au bloc de données qui vient d'être programmé. Une fois encore, il faut spécifier l'adresse de départ, de fin plus une et l'adresse de destination, comme pour la routine de duplication. Le programme peut être lancé en 0233. Dès qu'il détecte une erreur, le programme s'arrête, et affiche l'adresse à laquelle se trouve la donnée erronnée; précisons qu'il s'agit de l'adresse d'EPROM, avec sa donnée. Il suffit d'actionner une touche "logicielle" quelconque (hormis REST et ST) pour que la vérification reprenne son cours.

Arrivé au bout du bloc, le Junior Computer affiche comme précédemment l'adresse de fin, plus une. Nous sommes alors revenus dans le moniteur du Junior Computer.

Voilà toutes les informations qu'il nous paraissait intéressant de vous donner, afin de pouvoir utiliser au mieux notre programmateur d'EPROM universel. L'intérêt d'un tel circuit n'est plus à démontrer et nous espérons qu'il comblera tous vos espoirs. ◀

Pour compléter la panoplie du petit programmateur d'EPROM, il ne vous manque plus que la lampe "à bronzer" (rayons U.V.) sur carte au format européen . . .