extensions MSX (5)

programmateur d'EPROM

(2ème partie)

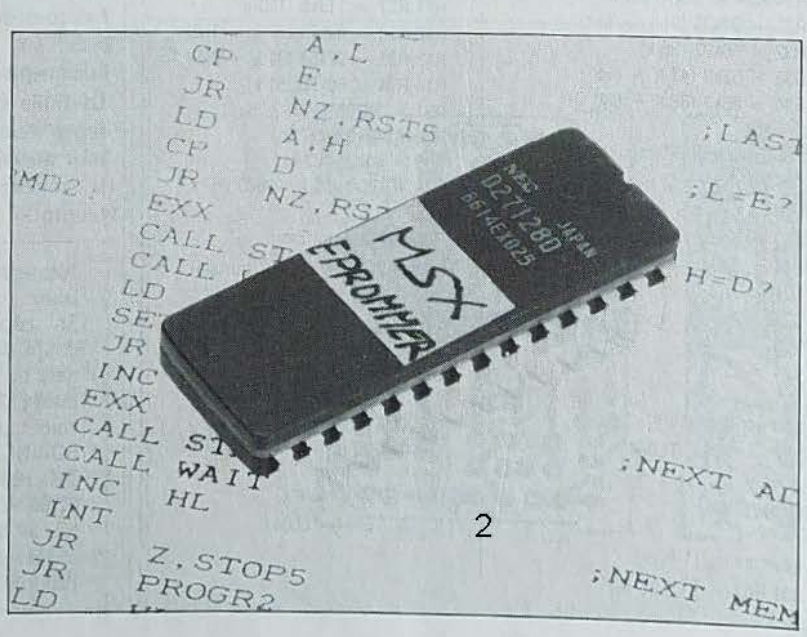
Pour fonctionner, notre programmateur ne pouvait se passer de logiciel. Celui-ci a pris la forme d'une EPROM remplie de code machine pour Z80 dont l'exécution se traduit par l'affichage d'un menu déroulant très complet associé à plusieurs pages d'informations; il comporte en outre des sous-programmes de test, d'affichage de messages d'informations relatifs à l'état de l'EPROM, et, cela va de soi, de (nombreux) messages d'erreurs extrêmement utiles lorsque les choses ''tournent au vinaigre''.

Après avoir consacré l'article du mois dernier au matériel du programmateur d'EPROM pour MSX, nous allons voir aujourd'hui comment l'ordinateur s'y prend pour tenir tout ce petit monde bien en main.

Au cours de ces dix dernières années, la croissance de la capacité des EPROM a suivi une courbe exponentielle. Si en 1980, on acceptait d'attendre une minute avant que ne soit terminée la programmation d'une 2708, deux minutes dans le cas de celle d'une 2716 (1 an et demi plus tard!), et quatre pour celle d'une 2732, il ne saurait être question en 1987 de perdre une demi-heure pour une 27256, voire une heure pour une 27512, deux types d'EPROM devenus presque communs. Il fallait donc trouver une nouvelle technique de programmation; ce qui fut fait; on en baptisa l'algorithme "**intel***ligent programming*", que nous avons évoqué dans l'article du mois dernier. Mais l'intelligent programming, qu'est-ce que c'est?

L'algorithme de programmation intelligente

A l'époque des balbutiements de la micro-informatique, on considérait que pour programmer une EPROM il fallait 50 ms par octet. A cette époque-là, personne ne voyait d'inconvénient à attendre 60 secondes pour la programmation d'une 2708. Et puis, 120 secondes, ce n'est ni plus ni moins que le temps nécessaire pour apprécier qui un café, qui un rafraîchissement, aussi les esprits ne s'échauffèrent-ils pas trop lors de l'arrivée de la 2716. Cependant lorsque vint la 2732, certains trouvèrent le temps bien long... et d'autres se rebiffèrent. 4 minutes de programmation, cela commençait à bien faire; en effet, si l'on poursuivait à ce rythme d'escargot, on ne serait bientôt (5 ans plus tard) plus capable de programmer en tout et pour tout qu'une demi-douzaine d'EPROM (de 128 Koctets) par tour de cadran d'horloge.



2

Cela ne pouvait pas durer, il fallait faire quelque chose. Intel, Fujitsu, National Semiconductor et d'autres fabricants d'EPROM imaginèrent différentes versions d'un algorithme de programmation intelligent grâce auquel on pourrait accélérer le processus de chargement des données. Comme le suggère son appellation, cette méthode repose sur l'emploi d'un microprocesseur, au lieu de compteurs/temporisateurs à période fixe pour la génération des impulsions de programmation. L'ordinogramme du **tableau 4** montre que le principe de l'algorithme de programmation intelligente réside dans le passage de V_{cc} de +5 V à +6 V et surtout dans la longueur **variable** du cycle de programmation. La boucle de programmation et de vérification n'est quittée que lorsque l'une des deux conditions suivantes est remplie: soit l'octet est programmé convenablement, soit il ne l'est pas encore correctement après un cycle de 25 impulsions. Comme il suffit souvent d'un nombre relativement faible d'impulsions pour obtenir la programmation stable d'un octet à une adresse donnée, la valeur de la variable *x* est faible. Après ce nombre variable d'impulsions préliminaires, une impulsion supplémentaire de 3 *x* ms garantit une programmation définitive de la donnée dans l'EPROM. Voyons à l'aide d'un exemple comment fonctionne cet algorithme:

Supposons qu'il faille 9 impulsions pour obtenir la programmation (préliminaire) correcte d'un octet dans l'EPROM. La durée totale du cyle de programmation définitive sera donc de:

(9 × 1) + (3 × 9) soit 36 ms.

La **figure 8** montre que dans certains cas, le cycle de programmation peut s'allonger considérablement. En fait, la programmation intelligente n'est

pas nécessairement plus rapide qu'une programmation normale (50 ms), qu'une programmation rapide en mode 1 (20 ms) ou qu'une programmation rapide en mode 2 (10 ms), sachant que dans le pire des cas, la durée d'un cycle peut atteindre 25 + (3 × 25) soit 100 ms. En pratique, vous aurez vite fait de découvrir qu'une EPROM neuve de technologie récente se contente bien souvent de la durée minimale de 4 ms pour être programmée correctement et cela de manière fiable. Il suffit ainsi de 2 minutes environ pour programmer une 27128 (32 K × 8).

Les techniques de programmation intelligente préconisées par Intel (*intejigent programming* , sic) et Fujitsu (Quick Pro™) ne diffèrent que très peu par la durée de l'impulsion de programmation, le nombre d'itérations effectuées avant que l'EPROM ne soit rejetée comme défectueuse et le facteur de multiplication des impulsions. L'algorithme de National Semiconductor repose sur des impulsions de 0,5 ms, un nombre maximal d'itérations de 20, l'absence de facteur de multiplication et un niveau de tension de programmation V_{pp} de 13 V au lieu des 12,5 V habituels. Le programmateur d'EPROM pour ordinateur MSX que nous vous proposons ne respecte pas au pied de la lettre l'algorithme de National Semiconductor, ce qui ne l'empêche pas cependant de donner d'excellents résultats avec les EPROM de ce fabricant.

Comme on pouvait s'y attendre, la chronologie des cycles de programmation repose sur des interruptions; elle est commandée conjointement par l'unité centrale (CPU) de l'ordinateur et le compteur/temporisateur (CTC) de la **cartouche timer + interface d'E/S**. Le programme de commande s'arrange pour que le temporisateur T2 du CTC fournisse le nombre d'impulsions nécessaires à la programmation correcte d'un octet à l'adresse prévue en EPROM. Les itérations et le facteur de multiplication respectent les données de l'ordinogramme du tableau 4. Des essais intensifs et exhaustifs ont prouvé que l'algorithme adopté donne des résultats satisfaisants avec l'immense majorité des EPROM aptes à une programmation intelligente.

Bien que cela n'apparaisse pas explicitement sur l'ordinogramme, le logiciel de commande et le CTC attendent, avant de permettre l'écriture, que la donnée à programmer dans l'EPROM (les signaux présents sur les lignes de données en fait) et ceux des lignes d'adresses correspondantes soient stables. C'est la raison pour laquelle le temporisateur T0 du CTC génère des retards de 4 µs, déjà mentionnés le mois dernier.

Description du programme

Un ordinateur répondant aux normes MSX peut posséder jusqu'à 4 connecteurs primaires numérotés 0, 1, 2 et 3; la capacité de mémoire de chacun d'entre eux est de 64 Koctets subdivisés en 4 pages de 16 Koctets. Il est possible de procéder à l'extension d'un connecteur en le dotant de 4 sous-connecteurs numérotés X—0, X—1, X—2 et X—3. Théoriquement, on disposerait ainsi de 16 connecteurs identifiés par les numéros 0—0 à 3—3 inclu.

Le Z80(A) est un microprocesseur à 8 bits; il est donc en mesure d'adresser une zone de mémoire de 64 Koctets, soit 4 pages de 16 Koctets (qui peuvent être réparties librement entre les connecteurs (étendus ou pas). Le système peut fonctionner avec par exemple la page 0 du connecteur 0, la page 1 du connecteur 2 et les pages 2 et 3 du connecteur 3—2. Dans ces conditions, les adresses absolues sont: page 0 = 0000—3FFF; page 1 = 4000—7FFF; page 2 = 8000—BFFF; page 3 = C000—FFFF.

A l'aide d'instructions système particulières, dans le détail desquelles nous n'entrerons pas ici, il est possible de changer de page, ou de mettre l'une ou l'autre d'entre elles en ou hors-fonction. En règle générale, la page 0 est réservée au BIOS (*Basic Input/Output System*) MSX; la page 3 est réservée à la pile (*stack*), à la zone brouillon (*scratch*), à la table des variables et au tampon du clavier (*keyboard buffer*) entre autres. Lors de la mise sous tension, un ordinateur MSX examine toujours les pages 1 et 2 de ses différents connecteurs pour essayer d'y détecter la présence de programmes résidents en (E)PROM, ceux-ci étant immédiatement exécutés s'il découvre un code d'identification spécifique dans les 16 premiers emplacements de mémoire. Sinon, c'est la ROM BASIC de la page 1 qui est validée, et l'ordinateur démarre (*boot*) conformément aux instructions de celle-ci. Pour vous éviter de devoir taper des Kilos de données à la main, nous avons mis le logiciel de commande du programmateur d'EPROM dans... (quoi de plus approprié)... une EPROM du type 27128 (16 Koctets), placée dans le support pour EPROM de la **cartouche universelle** décrite dans le numéro 92 d'Elektor (février 1986, page 56 et suivantes). Ceci fait, il reste à interconnecter les différents sous-ensembles évoqués le mois dernier pour obtenir un système fonctionnel, nous y reviendrons dans l'un des prochains paragraphes.

Tableau 4. Programmation d'EPROM interactive (Fujitsu).

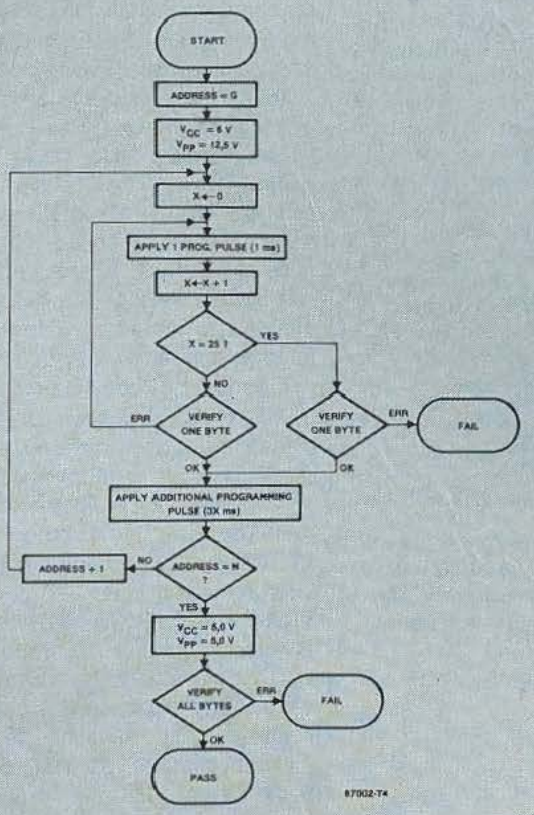


Tableau 4. Le but de la programmation intelligente est d'appliquer le nombre minimum d'impulsions strictement nécessaire pour obtenir la programmation correcte d'un octet.

Dès la mise sous tension de l'ordinateur, le logiciel de gestion du programme est lancé à partir de la page 2. Après exécution des sous-programmes d'initialisation indispensables, le programme se charge de découvrir quel connecteur possède de la RAM en page 1 et 2, pour l'utiliser comme zone de données pour l'EPROM (taille maximale 32 Koctets, 4000—BFFF). Il se recopie ensuite partiellement dans la zone de RAM la plus élevée possible en page 3, c'est-à-dire qu'il s'insère entre la pile et les blocs du brouillon et de la table des variables. Ces préparatifs terminés, le logiciel rend le contrôle à la procédure de lancement exécutée normalement par l'ordinateur, ce qui signifie que dans la plupart des cas on retourne dans l'interpréteur BASIC.

On peut alors lancer l'exécution du logiciel de l'EPROM en entrant **CALL EPROMx**, instruction dans laquelle **x** correspond au numéro de la cartouche convenable 0, 1, 2 ou 3. Une fois appelé, le programme sélecte automatiquement le(s) connecteur(s) convenable(s) pour le tampon de RAM et à l'aide des sous-programmes de la page 3, il revient à son point de départ. L'utilisateur ne remarque rien de ces allées et

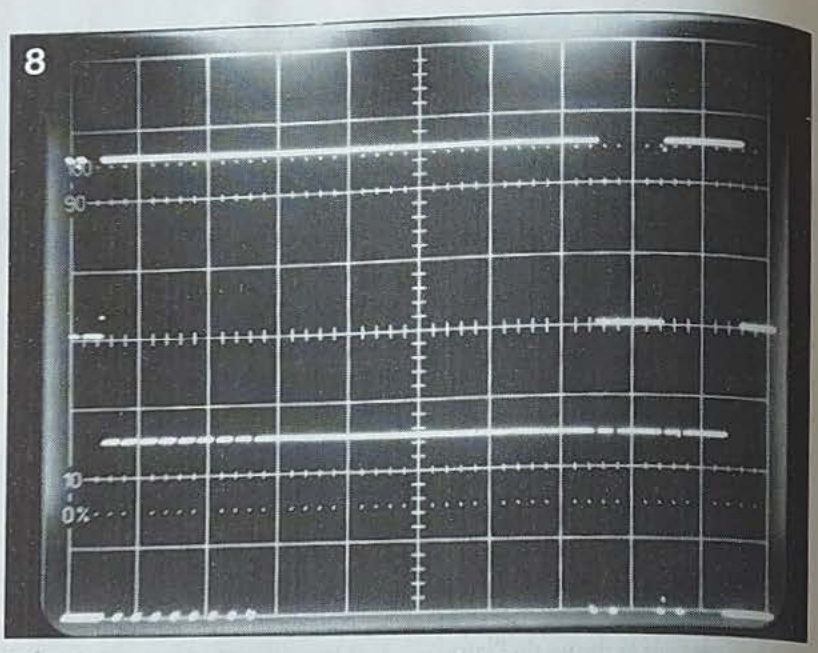


Figure 8. Cet oscillogramme prouve qu'il peut se faire qu'une adresse nécessite un nombre d'impulsions de programmation plus important avant d'être programmée convenablement. Courbe du haut: ligne d'adresse A0, courbe du bas (8 × 1) + 24 ms pour le premier octet, (1 × 1) + 3 ms pour le second et le troisième.

venues entre la RAM et les sous-programmes résidant en EPROM. Le logiciel est conçu pour fonctionner avec tout ordinateur MSX disposant d'un minimum de 64 Koctets de RAM. Le logiciel utilise intensément l'adressage vectorisé et toutes les entrées/sorties du clavier et de l'écran passent par le BIOS en page 0.

Pour s'assurer que certaines données en provenance de (ou destinées à) l'EPROM ne risquent pas, lors de leur écriture, de détruire la pile du système, ou, avec des conséquences tout aussi catastrophiques, une portion du programme de commande en RAM, il vous est recommandé de vérifier qu'il reste suffisamment de place pour vos données en entrant, après avoir exécuté une première fois l'instruction **CALL EPROM x** et être retourné en BASIC, l'instruction suivante:

PRINT HEX$(FRE(0)+&H8000).

L'adresse affichée en réponse à l'exécution de cette instruction devrait être supérieure à celle de l'emplacement mémoire le plus élevé nécessaire, sachant, comme nous l'avons indiqué précédemment, qu'une partie de la mémoire disponible sert à la table des variables et à la pile; des adresses hautes où elle se trouve à l'origine, cette zone croît vers les adresses plus basses. Le possesseur d'un ordinateur MSX doté d'un lecteur de disquettes devra limiter légèrement l'espace disponible pour DISKBASIC en maintenant enfoncée la touche CONTROL lors de la mise sous tension du système, technique qui sert, comme vous le savez, à signaler au système qu'il n'existe qu'un seul lecteur de disquette virtuel. De manière similaire, une action sur la touche SHIFT au démarrage du système met le lecteur de disquette hors-fonction.

Figure 9. Recopie d'écran de l'écran d'entrée des commandes (les choix sont aléatoires et uniquement donnés à titre d'exemple).

9

```
      ===== ELEKTOR MSX EPROMMER =====

EPROM TYPE:                          27128

PROGRAMMING VOLTAGE:                 12.5V

EPROM BEGIN:                         &H0000
EPROM END:                           &H3FFF

MEMORY BEGIN:                        &H4000
MEMORY END:                          &H7FFF

BLANK CHECK:                         YES
PROGRAM:                             YES
VERIFY:                              YES
READ AND RUN CHECK SUM:              NO
DISPLAY MEMORY:                      NO

PROGRAM MODE:                        INTELL

CHECK SUM:                           &H0000
ADDRESS COUNTER                      &H3FFF

RESULT: PROGRAMMED
```

87002-11

Liste de commandes disponibles

Grâce à la technique de menu déroulant utilisée, le logiciel est extrêmement simple à utiliser. Il n'en est pas moins fortement recommandé de regarder de près, une fois au moins, ses différentes fonctions, commandes et possibilités. Vous n'aurez plus ensuite la moindre question concernant son mode d'emploi.

Après avoir frappé l'instruction **CALL EPROMx**, on voit s'afficher une page de titre. Une action sur une **touche quelconque** permet ensuite d'aller aux **pages informatives** qui donnent les caractéristiques des différentes EPROM et les informations nécessaires pour leur programmation. Une action sur la **touche de curseur** adéquate permet de **feuilleter les différentes pages du menu**. Une action sur la **touche "espace"** (*SPACE*) permet à tout instant de **retourner** à l'écran comportant les **informations de programmation**.

Voici une liste des touches utilisées en mode entrée de commande:

Les **touches** ↑ et ↓ permettent de passer d'une ligne à l'autre du menu pour déterminer l'action désirée.

La touche **H** appelle les pages informatives.

La touche **P** entraîne l'envoi du contenu actuel de l'écran (*screen dump*) vers la sortie imprimante; pensez à vérifier que l'imprimante est connectée correctement à l'ordinateur et qu'elle est alimentée en papier, sinon vous verrez apparaître un message d'erreur du genre *NO PRINTER*.

La touche **T** démarre l'exécution d'un programme de test qui, par des interruptions générées par le CTC, valide successivement toutes les fonctions du programme (les LED correspondantes s'illuminent brièvement l'une après l'autre), tandis que la LED **PGM** clignote. Assurez-vous de **l'absence** du cavalier de court-circuit J1 et ne **procédez jamais à ce test en présence d'une EPROM dans le support à force d'insertion nulle (FIN)**.

La **barre "espace"** permet de choisir l'option désirée (elle assure une sorte de fonction de bascule entre les deux possibilités offertes).

La touche **S** démarre l'exécution de l'ensemble des commandes définies au cours de la procédure de sélection précédente.

La touche **I** permet d'initialiser le programmateur de manière à mettre un programme **BASIC** en EPROM; la programmation effectuée, on placera cette EPROM dans le support de la cartouche universelle pour obtenir l'exécution dudit programme lors de la mise sous tension de l'appareil. Pour de plus amples informations à ce sujet, voir l'article consacré à la cartouche universelle (février 1986, page 56 et suivantes).

Comme le montre la **figure 9**, il faut définir un certain nombre de paramètres pour que le programmateur fasse très exactement ce que vous attendez de lui. Passons-les successivement en revue:

EPROM TYPE (TYPE D'EPROM) ET **PROGRAMMING VOLTAGE** (TENSION DE PROGRAMMATION):

Consultez le tableau 1 ou la page informative correspondante; utilisez la touche "ESPACE" pour choisir le type d'EPROM adéquat. Notez au passage que les adresses de DEBUT et de FIN de domaine (**EPROM BEGIN & END**) varient en fonction de la capacité de l'EPROM concernée. Il est **possible** de ne programmer qu'une **partie** d'une EPROM en définissant en conséquence le domaine d'adresses. Les valeurs données au cours de l'éxécution du

programme doivent impérativement être hexadécimales, car le logiciel n'est pas conçu pour travailler dans une base différente. Si vous tentez de définir un domaine d'adresses impossible, ou que les données de DEBUT et de FIN de domaine (**EPROM BEGIN & END**) ne sont pas compatibles avec les données de DEBUT et de FIN de domaine de MEMOIRE (**MEMORY BEGIN & END**), vous verrez apparaître un message d'erreur sur l'écran.
Rien de plus parlant qu'un exemple. Supposons que vous vouliez programmer **la première moitié** d'une EPROM du type 2764 (8 Koctets). Dans ce cas, **EPROM BEGIN = 0000**; **EPROM END = 0FFF**; **MEMORY BEGIN = 4000**; **MEMORY END = 4FFF**.
La routine de VERIFICATION DE VIRGINITE (**BLANK CHECK**) d'une EPROM ne devrait pas vous poser de problème. Cette routine vérifie, en utilisant les informations de DEBUT et de FIN d'EPROM (**EPROM BEGIN & END**) que vous lui donnez, la présence à tous les emplacements concernés de l'EPROM de la valeur FF qui indique qu'il est possible d'y programmer un octet.
Le sous-programme de PROGRAMMATION (**PROGRAM**) n'exige que très peu d'explications. Il se base également sur les informations de DEBUT et de FIN d'EPROM (**EPROM BEGIN & END**) et de DEBUT et de FIN de domaine de MEMOIRE (**MEMORY BEGIN & END**) données en cours de procédure.
La routine de VERIFICATION DE LA PROGRAMMATION (**VERIFY**) s'assure que les contenus de l'EPROM et du tampon de RAM sont identiques; cette procédure utilise bien évidemment les informations de DEBUT et de FIN d'EPROM (**EPROM BEGIN & END**) et celles de DEBUT et de FIN de domaine de MEMOIRE (**MEMORY BEGIN & END**) pour savoir quels sont les domaines d'adresses à comparer.
Le sous-programme de LECTURE ET EXECUTION DE LA SOMME DE VERIFICATION (**READ AND RUN CHECKSUM**) charge les données de l'EPROM dans le tampon de la RAM et additionne les valeurs de tous les octets pour fournir la somme de vérification (*checksum*) correspondante codée sur 16 bits.
La routine d'AFFICHAGE DU CONTENU DE LA MEMOIRE (**DISPLAY MEMORY**) offre la possibilité à l'utilisateur de charger le contenu de l'EPROM dans la RAM de l'ordinateur en vue d'un examen (formats hexadécimal et ASCII, 8 octets par ligne, précédés par l'adresse). Il n'est **pas possible de modifier** les octets visualisés sur l'écran.
Le sous-programme de MODE DE PROGRAMMATION (**PROGRAM MODE**) permet tout simplement la sélection du mode de programmation désiré: **normal**, **fast—1** (rapide—1), **fast—2** (rapide—2) ou **intelligent**, en fonction du type d'EPROM à programmer. Consultez le tableau 1 ou la page informative adéquate.
Le COMPTEUR D'ADRESSES (**ADDRESS COUNTER**) présent dans la partie inférieure droite de l'écran est un compteur sur 16 bits qui indique en permanence quel est à cet instant l'emplacement mémoire lu ou dans lequel a lieu une opération d'écriture.

La ligne de RESULTAT (**RESULT**) présente dans le bas de l'écran sert à la visualisation de messages suffisamment explicites (une action sur la touche **H** permet de revenir aux pages informatives):
ERREUR D'ADRESSE (ADDRESS ERROR) est un message passe-partout indiquant qu'il faut retaper les données de DEBUT et de FIN d'EPROM (**EPROM BEGIN & END**) et de DEBUT et de FIN de domaine de MEMOIRE (**MEMORY BEGIN & END**) avant d'actionner à nouveau la touche **S**.
VIERGE (BLANK) signale que le domaine d'adresses examiné ne contient que des octets ayant la valeur FF. Le domaine d'EPROM correspondant n'est pas recopié en RAM.
PAS VIERGE (NOT BLANK) indique tout simplement qu'un octet (au moins) du domaine d'EPROM spécifié possède une valeur différente de FF. Le compteur d'adresses (ADDRESS COUNTER) indique la première adresse concernée; l'exécution du programme est stoppée.
LECTURE TERMINEE (READING COMPLETED) n'exige pas d'explications supplémentaires. Le contenu de l'EPROM est disponible pour examen par le sous-programme de VISUALISATION DE LA MEMOIRE (**DISPLAY MEMORY**). Pour pouvoir modifier les données en mémoire, il vous faudra recourir au BASIC ou à un logiciel utilitaire permettant ce genre de manipulations.
VERIFIE (VERIFIED) signifie que lors de son exécution, le sous-programme de vérification n'a pas découvert d'erreur.
ERREUR DE VERIFICATION (VERIFY ERROR) indique qu'il existe une différence (au moins) entre le contenu de l'EPROM et celui de la RAM. Le compteur d'adresses (ADDRESS COUNTER) donne l'adresse du premier emplacement de mémoire concerné; l'exécution du programme est arrêtée.
REPROGRAMMABLE (REPROGRAMMABLE) signale qu'en cours de procédure de vérification une erreur a été découverte, mais que l'octet en question est reprogrammable; ceci signifie que certains bits de cet octet qui devaient être à 0 sont encore à 1: il est possible de les programmer en dépit du problème rencontré. L'inverse, c'est-à-dire le passage d'un niveau logique bas vers un niveau logique haut ne peut s'effectuer que par exposition de l'EPROM à une dose adéquate de rayons ultraviolets.
NON PROGRAMMABLE (NOT PROGRAMMABLE) signale qu'il est impossible de charger correctement dans l'EPROM la donnée présente à l'adresse indiquée par le compteur d'adresses, même après avoir appliqué 25 impulsions de programmation (voir tableau 4, ne concerne pas la programmation intelligente).
EXECUTION ARRETEE (EXECUTION STOPPED) s'affiche sur la ligne des messages en réponse à une action sur le bouton-poussoir de remise à zéro matérielle (Reset) du programmateur d'EPROM.
ERREUR D'INTERFACE D'E/S (DEVICE I/O ERROR) indique que l'ordinateur ne reçoit pas d'interruption en provenance de la cartouche, qui n'est peut-être pas positionnée à la bonne adresse.
PAS D'IMPRIMANTE (NO PRINTER) n'exige pas d'explication supplémentaire: ce message dit très exactement ce qu'il veut dire.
SUCCESSION DE COMMANDES ILLEGALE (ILLEGAL COMMAND ORDER) vous demande de procéder à nouveau au choix OUI/NON (YES/NO) d'une ou de plusieurs des commandes. Notez qu'il est licite de choisir simultanément une réponse OUI (YES) pour BLANK CHECK,

Tableau 5. Données de commande du port C.

EPROM	READ	VERIFY	WRITE
2716	0B	08 + V_{pp}	28 + V_{pp}
2732	0F	0C + V_{pp}	68 + V_{pp}
2764	0B	08 + V_{pp}	28 + V_{pp}
27128	0B	08 + V_{pp}	28 + V_{pp}
27256	0B	48 + V_{pp}	68 + V_{pp}
27512	0F	0C + V_{pp}	68 + V_{pp}
2516	0B	0B	68 + V_{pp}
2532	0B	0B	68 + V_{pp}
2564	0B	0B	48 + V_{pp}

Toutes les valeurs sont hexadécimales.
V_{pp} = 5 V: 3. V_{pp} = 12,5 V: 2.
V_{pp} = 21 V: 1 V_{pp} = 25 V: 0.
Le bit de poids fort du port C est programmé en entrée.

Tableau 5. Configuration de la donnée de commande à appliquer au port C.

Ce programmateur d'EPROM a été conçu exclusivement pour les ordinateurs répondant aux normes MSX. Nous n'envisageons donc pas sa modification pour le connecter à un micro-ordinateur ne répondant pas aux normes MSX. Vous ne nous en voudrez pas trop si nous ne vous donnons pas non plus d'informations dans ce but.

PROGRAMM et VERIFY; le logiciel exécute ces diverses procédures dans l'ordre correct sans nécessiter d'action intermédiaire sur la touche **S** une fois l'exécution de la procédure lancée.

Au risque de nous répéter, nous insistons sur la nécessité de bien réfléchir à son geste avant d'actionner la touche **S**, et, ce faisant, de lancer l'exécution partielle ou complète du logiciel de programmation. En cas de message d'erreur, pas de panique: examinez attentivement l'écran de commande pour tâcher d'y découvrir l'erreur et essayer d'en comprendre la nature. Après une brève période d'accoutumance, vous constaterez combien l'utilisation de ce programmateur d'EPROM est agréable, grâce à la présence des pages informatives en particulier, auxquelles on accède instantanément par action sur la touche **H**.

Si vous ne savez pas quelle procédure adopter pour programmer une EPROM n'apparaissant pas dans le tableau 1 (1ère partie), commencez tout simplement par choisir **la tension de programmation la plus faible** (12,5 V) pour voir si le contenu de l'EPROM change; de cette manière, vous ne courrez pas le risque de détruire le circuit intégré, à condition de ne pas choisir le mode de programmation intelligent, car dans ce cas, la tension d'alimentation V_{cc} passe de 5 à 6 V au cours du cycle de programmation.

En guise de conclusion, quelques trucs tout simples.

Si pour un type d'EPROM donné, il est préconisé d'utiliser le mode normal (50 ms), on peut, pour raccourcir la durée de programmation, essayer le mode *fast—1* ou *fast—2*. Si vous désirez vous rappeler du choix des commandes correspondant à un certain type d'EPROM, il n'est pas bête d'effectuer une recopie d'écran sur imprimante (*screen dump*) de manière à disposer d'une trace matérielle de la somme de vérification et des autres informations importantes.

Rappelez-vous que pour programmer une EPROM du type 27512 (64 Koctets) il vous faudra effectuer deux passes de 32 Koctets. Actionnez la touche CONTROL—STOP pour revenir en MSX BASIC, faites ensuite CALL EPROMx pour lancer une nouvelle exécution du programme.

Utilisez un assembleur ou un utilitaire générateur de langage machine pour mettre des octets dans le tampon de RAM (en vue d'un transfert ultérieur vers une EPROM), en vous assurant que les données en question ne sont pas écrasées par l'utilisation de la pile ou du tampon par un autre programme exécuté parallèlement au logiciel du programmateur d'EPROM.

Rappelez-vous que l'exécution d'un programme BASIC utilisant les instructions PLAY exige la réinitialisation de l'ordinateur et donc celle du logiciel du programmateur d'EPROM, car ce dernier programme place sa table de sauts et son tableau de variables dans la zone d'attente (*voice queue*). En d'autres termes, n'utilisez pas le logiciel du programmateur d'EPROM tant que vous n'êtes pas certain qu'il ne traîne pas d'autre(s) programme(s) (ne serait-ce que sous la forme de restes) à un endroit quelconque de la mémoire. Le meilleur moyen d'éviter tout problème est de réinitialiser l'ordinateur (bouton Reset) une fois l'ensemble du programmateur d'EPROM en place.

Pour finir, vous trouverez dans le **tableau 5** les mots de commande pour les différents types d'EPROM. Ces mots de 7 bits sont spécifiques des différents types d'EPROM concernés et seront extrêmement utiles à tous ceux d'entre vous qui envisagent d'écrire leur propre version du logiciel de commande du programmateur d'EPROM.

Mise en oeuvre

Commencez par mettre en place les cavaliers de court-circuit **B**, **D**, **E** et **I** sur le circuit du programmateur d'EPROM et implantez l'EPROM (contenant le bon programme, EPROM disponible aux sources habituelles) **dans le support 28 broches de la cartouche universelle**. Enfichez cette cartouche dans le connecteur pour cartouche de l'ordinateur. Enfichez ensuite la cartouche timer + interface d'E/S dans un (ou le) deuxième connecteur de l'ordinateur (ou si votre système n'en comporte qu'un, enfichez-la dans le connecteur que comporte la cartouche universelle, cf les photos d'illustration).

Il vous reste à connecter le programmateur d'EPROM à la cartouche timer + interface d'E/S par l'intermédiaire du câble plat à 50 conducteurs pour vous trouver en présence d'un système fonctionnel (voir **figure 10**).

Veuillez noter qu'il n'est pas possible d'utiliser simultanément la cartouche timer + interface d'E/S et la carte de bus multi-connecteur (décrite en mars 1986).

Attendez avant d'implanter une EPROM dans le support à FIN! Commencez par mettre le système sous tension et appelez le programme d'initialisation. Après apparition de la page de titre et de copyright, passez à l'écran de commande et procédez à l'exécution de la routine de test présente dans l'EPROM-programme avant de commencer à travailler sur une EPROM de quelque type qu'elle soit. Si toutes les LED présentes sur la face supérieure du programmateur d'EPROM clignotent dans le bon ordre, on peut supposer, sans trop grand risque de se tromper, que le matériel et le logiciel fonctionnent correctement; vous pouvez maintenant essayer le système et l'utiliser ensuite sur un type d'EPROM quelconque. ◀

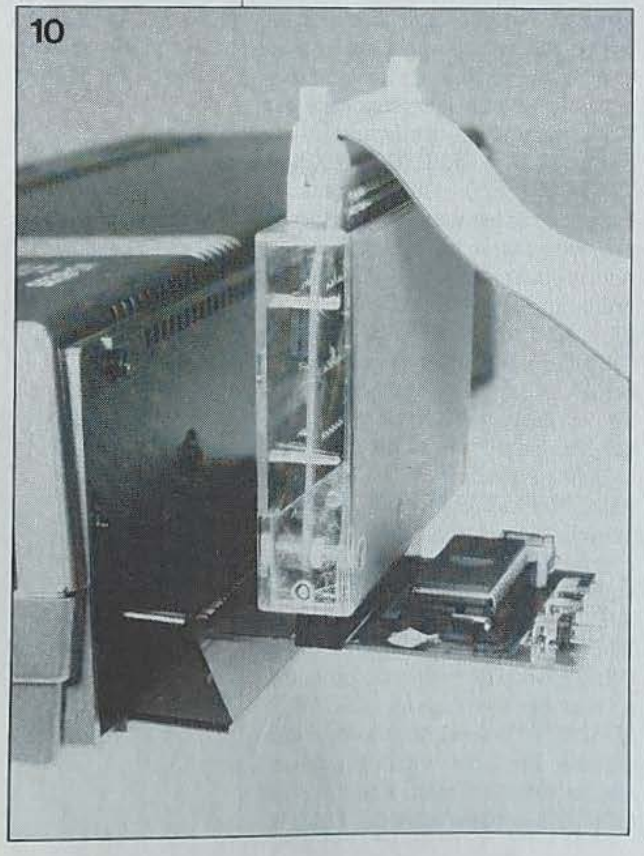


Figure 10. Un seul connecteur peut recevoir à la fois la cartouche universelle et la cartouche timer + interface d'E/S.

Bibliographie: Nous en voici à la seconde partie du 5ème article consacré aux extensions pour ordinateur MSX. Il se pourrait que vous désiriez savoir où trouver les articles précédents:

février 1986: modification du bus I/O universel pour utilisation avec un ordinateur MSX.

février 1986: une cartouche universelle

mars 1986: carte de bus multi-connecteur

septembre et octobre 1986: microscope (à utiliser avec le circuit suivant)

janvier 1987: cartouche + interface d'E/S

mars et avril 1987: programmateur d'EPROM