Chapitre 3

Logiciel de commande

Principe

Le vocabulaire correspondant à l'application est stocké en mémoire ROM ou RAM selon l'organisation représentée par la figure 18. En général, plusieurs expressions, mots ou éléments de vocabulaire (phonèmes par exemple) sont stockés dans un espace-mémoire déterminé. Afin de faciliter l'accès à ce vocabulaire et pour rendre le programme de commande indépendant de celui-ci, on dispose en début de la zone mémoire de vocabulaire une table d'adresse (directory) indiquant l'adresse de début de chacun de ses éléments. Chacune de ces adresses est codée sur deux octets, poids fort en tête; ceci permet donc de trouver l'adresse de l'adresse d'une expression quelconque à partir de son numéro d'ordre N, par la relation:

$$AN = A0 + 2 \times N$$

où A0 = Adresse de début de table, AN = Adresse de l'adresse de l'expression N.

La table d'adresses se termine par la séquence "FF 00" ou "FF FF" selon le dispositif de codage utilisé.

Chaque expression débute elle même par un "en tête" de 4 octets, dont les deux premiers représentent le nombre total d'octets de l'expression (en-tête compris) et le quatrième le pitch initial. Le troisième octet représente le pitch en fin d'expression ou est à 00 selon le dispositif de codage ; il n'est en général pas exploité par le logiciel de synthèse. Les trames qui suivent étant elles-mêmes codées sur 4 octets, la longueur d'une expression est toujours un multiple de 4 octets.

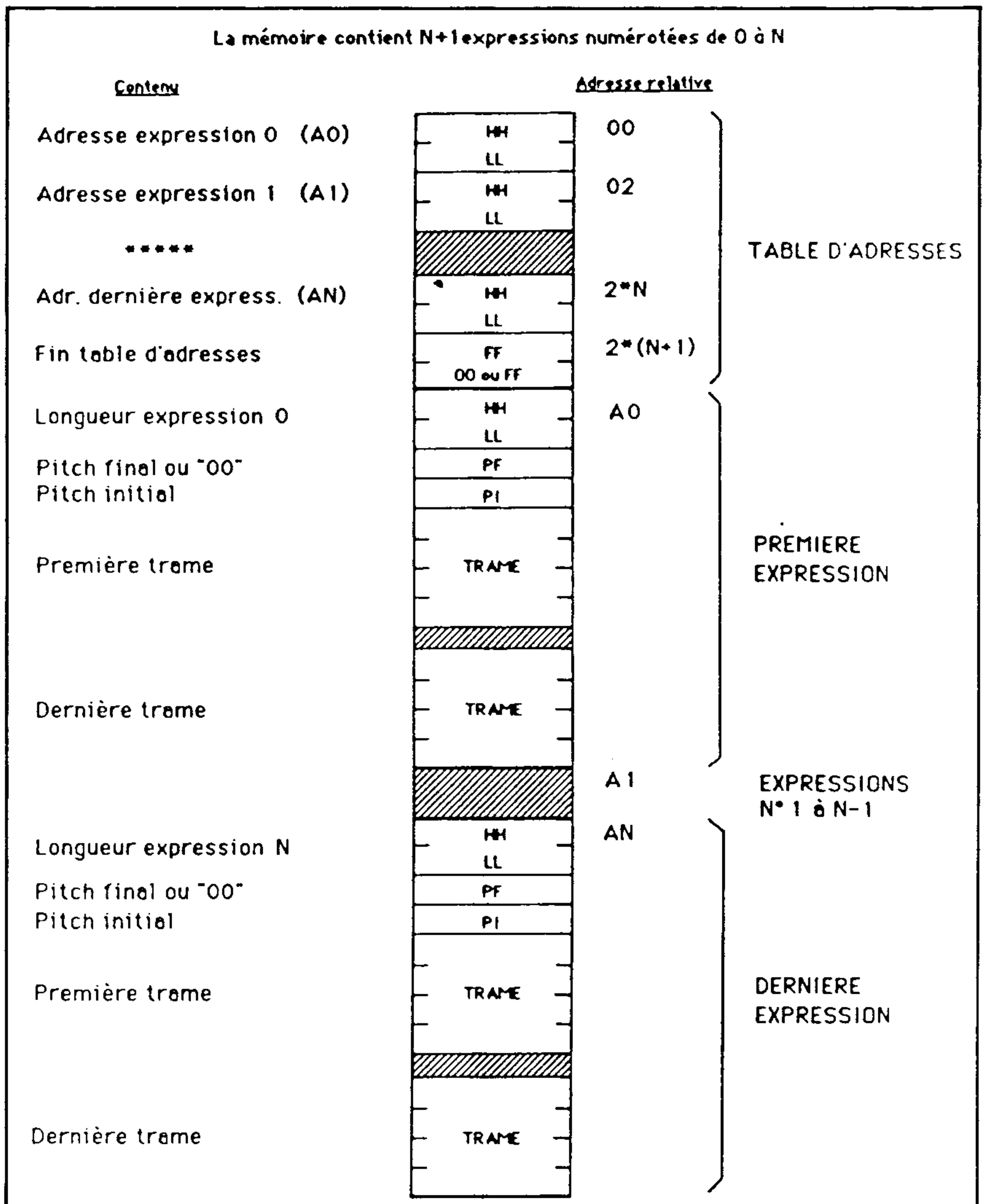

Figure 18 - Organisation de la mémoire de vocabulaire

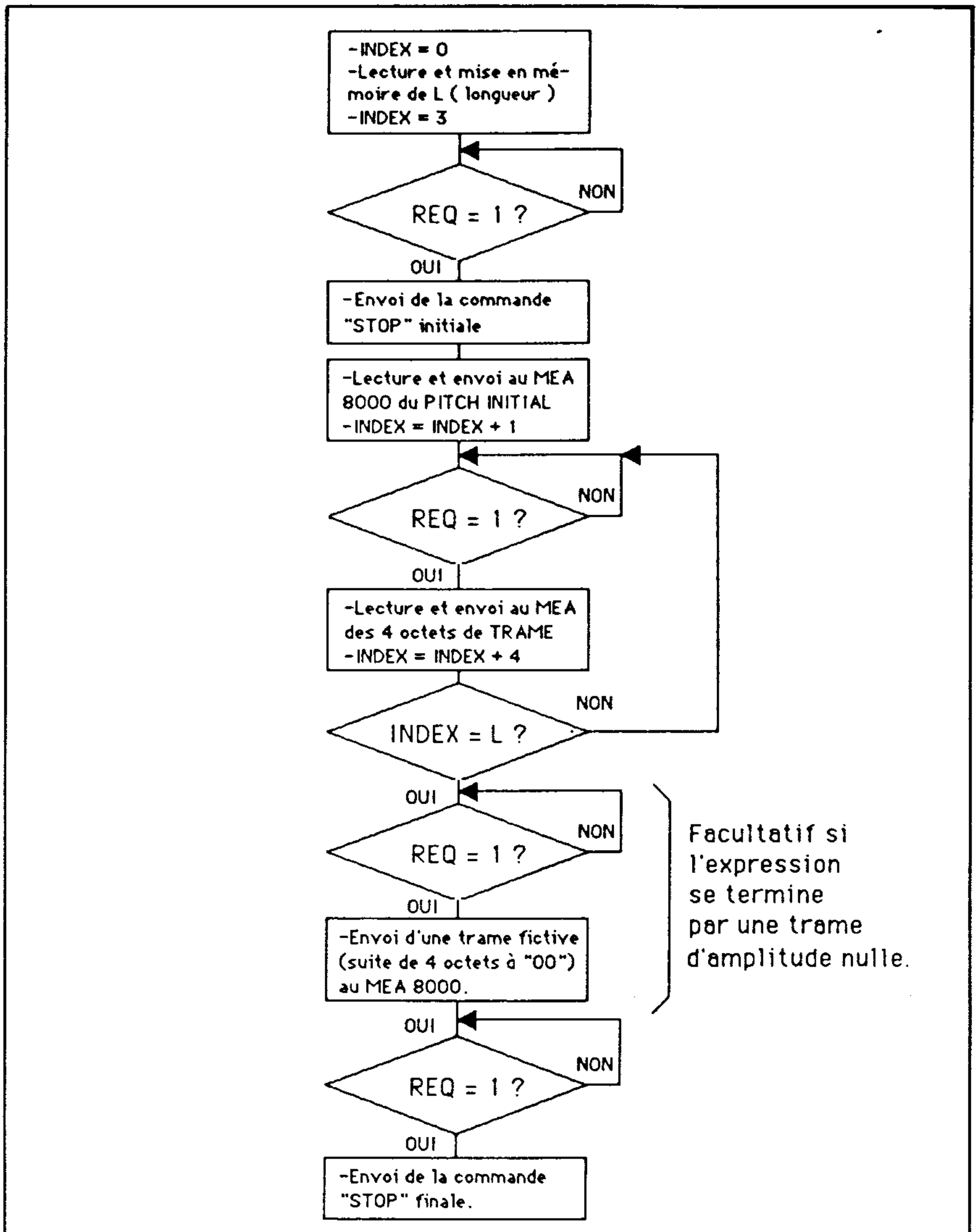

Figure 19 - Organigramme général de la routine de parole

Connaissant ce qu'il faut envoyer au synthétiseur et la façon dont les informations sont stockées dans la mémoire, il est nous maintenant possible d'en déduire l'organigramme général d'une routine permettant de faire prononcer au MEA 8000 une expression en mémoire.

Cette routine devra être réalisée en langage-machine, le BASIC n'étant pas assez rapide pour cette application. Son organigramme est représenté par la figure 19. L'adresse de début de l'expression à prononcer est supposée avoir été préalablement chargée à un endroit approprié pour l'adressage indirect.

Après avoir testé l'état du bit "REQ", on initialise le synthétiseur par l'envoi d'une commande STOP (A0 = 1) qui déterminera en outre la procédure d'arrêt suivie en cas d'interruption dans l'envoi des codes ("continu" ou "arret lent").

Il faut ensuite lire les deux octets de la longueur d'expression et les stocker dans un registre ou en mémoire. Il est commode, pour adresser les octets successifs de l'expression, d'utiliser un **index** qui pourra être un registre permettant l'adressage indexé. Cet index sera donc positionné à 0 au départ pour pointer la longueur d'expression; après lecture de la longueur, on le positionnera à 3 pour pointer l'octet de pitch initial.

C'est le premier octet de données vocales à envoyer au MEA 8000 (A0 à "0"), après quoi on incrémente l'index pour pointer l'octet suivant. On teste de nouveau le bit REQ avant l'envoi des trames vocales. Les quatre octets de la trame peuvent être envoyés en bloc sans test de REQ entre chacun d'eux lorsque l'on travaille par scrutation de D7, après quoi on incrémente l'index de 4.

On peut aussi envoyer chaque octet et incrémenter de 1 l'index, ce qui permet d'utiliser la même routine pour l'envoi du pitch initial et des octets de trame; c'est la solution que nous avons choisie.

Après chaque trame, il faut tester si l'index a atteint la valeur de la longueur de l'expression; si ce n'est pas le cas, on teste de nouveau REQ, on envoie la trame suivante, et ainsi de suite. Lorsque les deux valeurs sont égales, on est arrivé à la fin de l'expression et on peut envoyer une trame silencieuse (00 00 00 00) suivie, après test de REQ, de la commande STOP finale.

Cette trame "fictive" n'est pas indispensable si on a pris la précaution, lors du codage, de terminer chaque expression par une trame d'amplitude nulle. C'est le cas pour le vocabulaire fourni en annexe et les expressions créées par les logiciels de composition phonétique décrits qui permettent de se passer de trame fictive dans la routine de base.

On peut se rendre compte, sur l'organigramme de la figure 19, que certaines opérations sont effectuées à plusieurs endroits du programme, ce qui justifie leur réalisation au moyen d'un sous-programme:

— STOP: Envoi de la commande STOP
— TREQ: Test de l'état de REQ
— TRAM: Envoi d'un octet de données et incrémentation d'index.

Le programme principal définit le séquencement des opérations et assure directement la réalisation de celles ne figurant qu'une une seule fois (initialisation, lecture de la longueur, test de fin).

Application aux microprocesseurs 6800/6802

Bien que plus très employés dans l'industrie, ces deux microprocesseurs (6800 et 6802) sont toujours utilisés dans de nombreux lycées et collèges techniques, car très simples, ce qui est essentiel pour des étudiants qui découvrent les "micros". Pour en avoir fait l'expérience, la synthèse de parole est une application spectaculaire, donc très motivante en milieu scolaire. C'est pourquoi nous donnons les bases de réalisation autour de ces deux microprocesseurs.

Bien que l'utilisation en mode "périphérique" soit tout à fait possible avec l'entrée d'interruption $\overline{IRQ}$, nous nous bornerons à l'utilisation en mode "mémoires". Les "fanas de l'IRQ" pourront toujours s'inspirer de l'exemple donné avec le 6809 qui n'est, en fait, qu'un 6800 amélioré !

Le MEA 8000 est perçu, par le micro, comme deux emplacements mémoires qui doivent être consécutives, car un seul bit (A0) permet d'accéder aux registres internes.

A0=0	Registre de donnees
A0=1	Registre de commande

Figure 20

Schéma de base

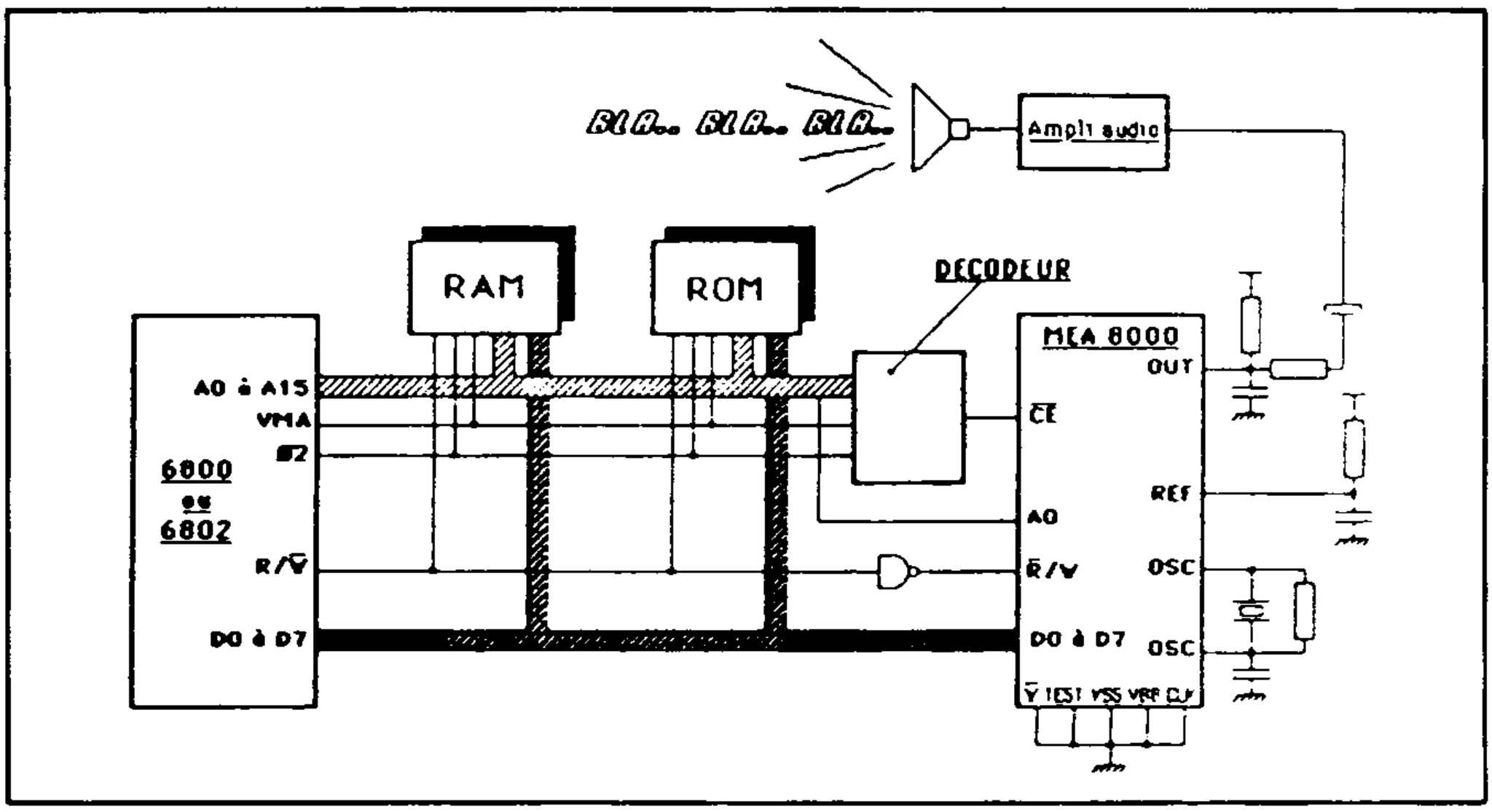

Figure 21

La validation du circuit ($\overline{CE}$) est générée par le circuit de décodage (décodeur) qui doit répondre à l'équation suivante : $\overline{CE} = X.\overline{VMA}.\Phi 2$ où X = Somme logique des adresses appropriées, $\overline{VMA}$ = Valid Memory Address (signal de validation de l'accès aux memoires), Φ 2 = Deuxième phase d'horloge.

$\overline{R}/W$ est issu de $R/\overline{W}$ après passage dans un inverseur. Les données (D0 à D7) sont reliées directement au bus DATA du micro.

Logiciel

Routine de prononciation d'une expression de longueur limitée à 255 octets

Adr.	Op.code	Etiq.	Mnémo		Commentaire
			ORG	$9F00	
		DONN	EQU	$E7FE	
		RCOM	EQU	$E7FF	
9F00	01	VAL	FCB	1	Rang dans la table
9F01	C6 01	DEBUT	LDA B	#1	Premier mot dans la table
9F03	F7 9F00		STA B	VAL	Charge VAL
9F06	BD 9F0A		JSR	PARL	S/PROG PARL
9F09	3F		SWI		
9F0A	B6 9F00	PARL	LDA A	VAL	Compare le rang de la table
9F0D	81 15		CMP A	#21T	21 mots dans la table
9F0F	22 26		BHI	FIN	Si plus grand : FIN
9F11	48		ASL A		2 octets par adresse
9F12	CE A000		LDX	#$A000	Début de la table
9F15	BD 9F3B		JSR	ADXA	Calcul la pos. de l'adr. relat.
9F18	A6 01		LDA A	1,X	Adresse relative
9F1A	CE A000		LDX	#$A000	Début de la table
9F1D	BD 9F3B		JSR	ADXA	Calcul l'adresse absolue
9F20	A6 01		LDA	1,X	Longueur du mot
9F22	C6 1B		LDA B	#$1B	Arrêt lent + REQ
9F24	F7 E7FF		STA B	RCOM	Registre de commande
9F27	08		INX		
9F28	08		INX		Saute les 3 octets suivants
9F29	08		INX		
9F2A	80 03		SUB A	#3	Moins 3 sur la longueur
9F2C	E6 00	RET	LDA B	0,X	Hauteur initiale ou données
9F2E	08		INX		Positionne X
9F2F	7D E7FF	AR3	TST	RCOM	Demande d'octet ?
9F32	2A FA		BPL	AR3	Non. On boucle sur AR3
9F34	F7 E7FE		STA B	DONN	Envoie l'octet demandé
9F37	4A		DEC A		Décompte les octets
9F38	26 F7		BNE	RET	Si non zéro, on continue
9F3A	39	FIN	RTS		

Sous-programme

```
9F3B 16       ADXA   TAB          Pour conserver A
9F3C 27 04           BEQ     AV1   Test si zéro ; Oui alors RTS
9F3E 08       AR1    INX          Plus un sur X
9F3F 5A              DEC B        Moins un sur B
9F40 26 FB           BNE     AR1   Si non égal a 0 ; Alors AR1
9F42 39       AV1    RTS
```

L'exemple logiciel que nous donnons a été réalisé a partir d'une application fonctionnant sur TO7. Les deux registres du MEA 8000 sont implantés aux adresses :

$E7FE pour le registre de données (DONN)
et $E7FF pour le registre de commande (RCOM)

La table d'adresses est positionnée à partir de l'adresse $A000. La constante VAL définit le rang (dans la table) du mot ou de l'expression que l'utilisateur souhaite entendre prononcer par son dispositif. Le programme "DEBUT" permet d'appeler le sous-programme "PARL" et détermine, en chargeant VAL, le mot qui sera traité par PARL.

PARL compare la valeur de VAL avec le nombre de mots codés disponibles en mémoire (21 dans notre exemple). Si VAL est supérieure, rien ne se passe et on retourne au programme principal. Le sous-programme "ADXA" additionne la valeur de l'index X et l'accumulateur A. Ceci afin de calculer l'adresse de la valeur qu'il convient d'ajouter au début de la table ($A000) pour pointer sur l'adresse du premier octet du mot choisi. Cette fonction est réalisée par un deuxième appel du sous-programme "ADXA".

Après avoir chargé l'accumulateur A avec la longueur du mot, le MEA 8000 est positionné en mode "ARRET LENT et $\overline{REQ}$ VALIDE", par l'envoi du code hexadécimal $1B dans le registre de commmande. Le pointeur X est incrémenté de trois et l'accumulateur A, spécifiant la longueur, est décrémenté d'autant. La hauteur initiale est chargée dans B et le pointeur incrémenté de 1. Le bit 7 du registre de commande est testé afin de vérifier s'il y a bien une demande d'octet de la part du MEA 8000.

Dans l'affirmative, on envoie la donnée. Dans le cas contraire, on boucle en AR3 en attendant l'accord d'envoyer du MEA 8000. Puis on teste la fin du mot, sinon on retourne en RET.

Application au microprocesseur 6809

Le 6809 est le microprocesseur employé sur les machines THOMSON (TO7, TO7/70, MO5, TO9) et TANDY (TRS/80). Il permet, suivant l'application souhaitée, de considérer le MEA 8000 soit comme deux emplacements mémoires consécutives, soit comme un périphérique qui utilise les interruptions, assez nombreuses sur le 6809 ($\overline{IRQ}$, $\overline{FIRQ}$, $\overline{NMI}$), pour interrompre le déroulement du programme en cours afin d'être servi en données vocales. La figure ci-dessous indique le schéma général de raccordement du MEA 8000 au 6809.

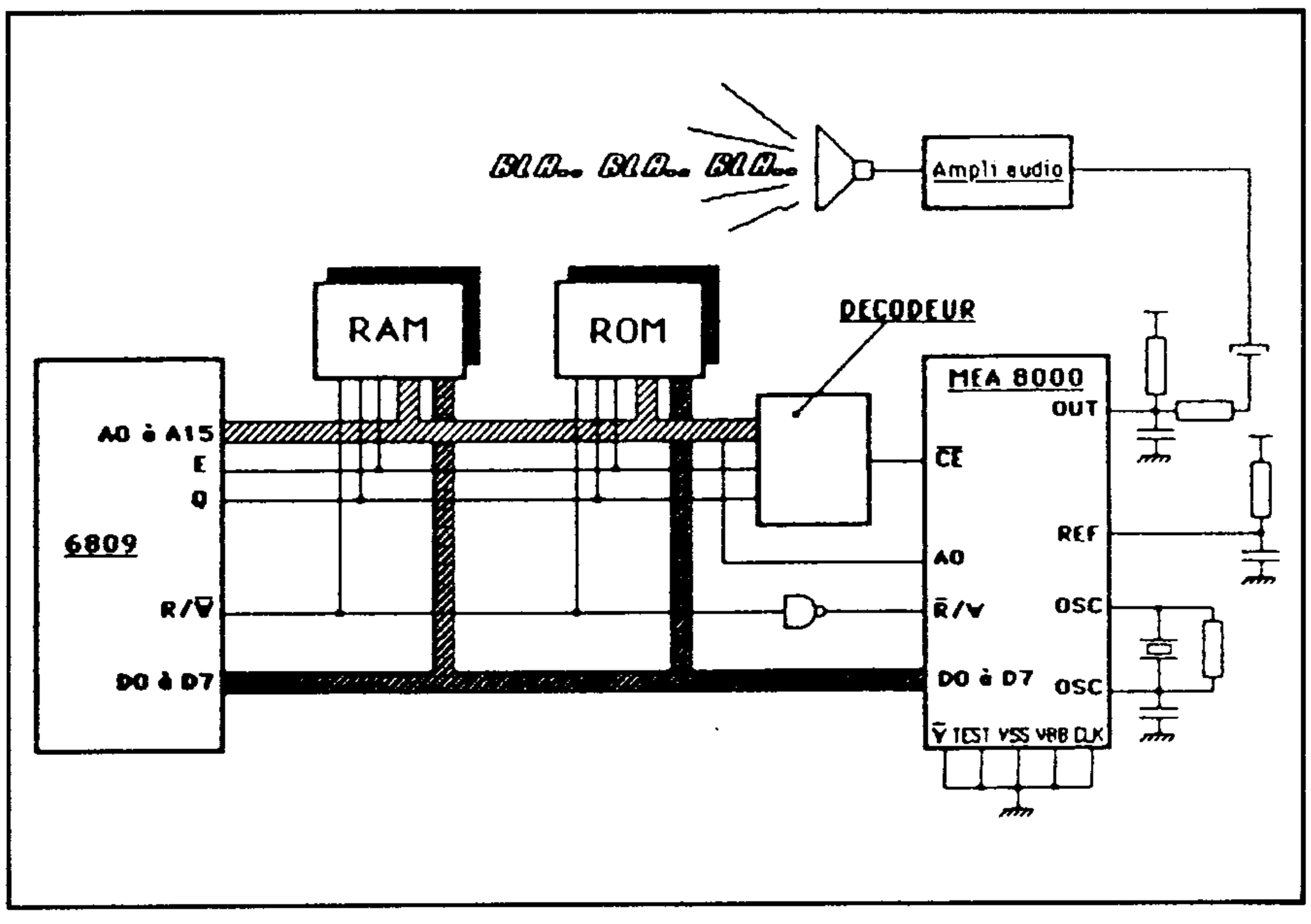

Figure 23

L'entrée A0 du MEA 8000 est connectée directement au 6809. $\overline{CE}$ est issu d'un décodage approprié des adresses A1 à A15, mais aussi de E qui est, avec Q, un signal qui détermine le cycle de mémorisation des données. $R/\overline{W}$ du 6809 est inversé par une porte NON avant d'être appliqué au MEA 8000. Le bus de données est raccordé directement aux bornes équivalentes du MEA 8000.

Au point de vue logiciel, nous donnerons comme exemple une application qui "tourne" sur TO7. Les adresses du MEA 8000 ont été réservées, en tenant compte des emplacements disponibles sur TO7, aux locations $E7FE et $E7FF. La table des données vocales, précédée du directory (table d'allocation d'adresses relatives) est implantée à partir de l'adresse $A000.

Le programme "PARL", implanté en $9FD3, commence par comparer le rang de l'expression proposée avec le nombre de mots disponibles en mémoire et calcule l'adresse du premier octet de l'expression à prononcer.

Le pointeur, chargé dans U, acquis, il convient d'initialiser le MEA 8000 dans le mode de fonctionnement souhaité (mode ARRET LENT et $\overline{\text{REQ}}$ VALIDE - code $1B - dans notre exemple). Le registre X est ensuite chargé avec la longueur du mot et ajusté en fonction de la valeur du pointeur (U) à cet instant.

Le premier octet correspondant à la hauteur initiale est chargé dans l'accumulateur B, puis le bit 7 du registre de commmande est testé afin de valider le chargement du registre de données. Le registre X est décrémenté a chaque octet transmis au synthétiseur. Un test après chaque décrémentation permet de détecter la fin du message. Comme ce programme est "emprunté" à l'application de base sur TO7, les listings (source et objet) sont visibles dans le paragraphe concernant cette machine (page 75).

Application aux microprocesseurs 6502/6510

Ces microprocesseurs étant utilisés dans de nombreuses machines très répandues (APPLE IIe et II+, COMMODORE 64, ORIC 1 et ATMOS...) ainsi que d'autres moins connues, les routines de base pourront trouver d'autres applications que les utilisations particulières décrites plus loin. Le 6502 est un processeur relativement ancien qui ne dispose pas d'instructions d'entrée-sortie spécialisées. Le synthétiseur sera donc vu par le microprocesseur comme une mémoire et occupera deux adresses consécutives (A0 = 1 et A0 = 0). La figure 24 indique le schéma général d'interfaçage:

— A0 est directement connectée au MEA 8000 (borne 11)
— A1 à A15 par un décodage approprié fournissent le signal $\overline{CE}$ (borne 12); ce même signal peut être éventuellement utilisé pour dévalider une mémoire occupant les mêmes adresses.
— R/$\overline{W}$ du 6502 est appliqué après inversion à $\overline{R}$/W du MEA 8000 (borne 22)
— D0 à D7 sont connectés directement aux bornes 10 à 3 du MEA 8000.

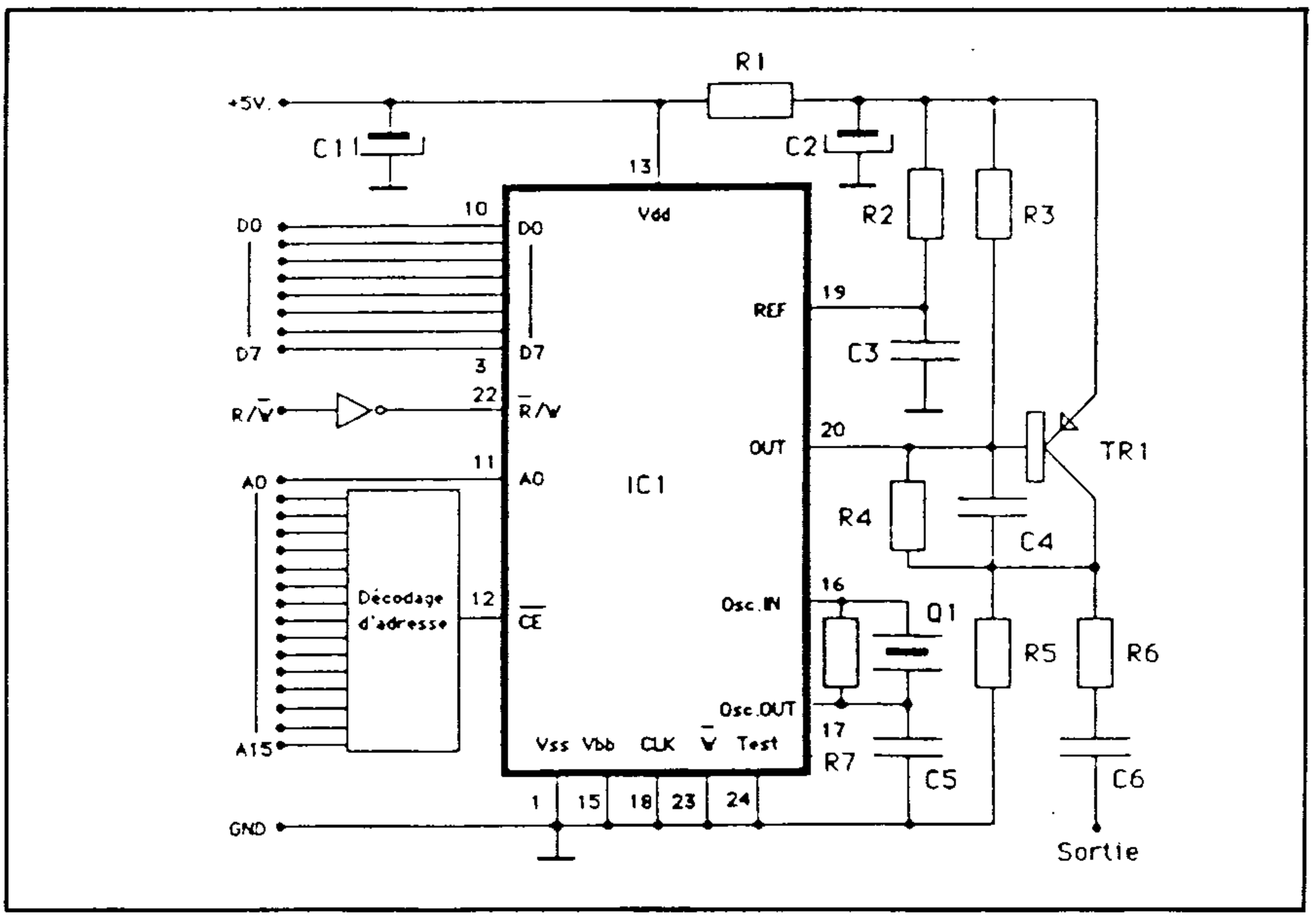

Figure 24 - Schéma général d'interfaçage avec le 6502

Routine de prononciation d'une expression de longueur < 256 o.

Pour le premier exemple, qui correspond à une application sur ORIC, les adresses hexadécimales décodées $03FE et $03FF ont été choisies pour le MEA 8000 et le programme est implanté à l'adresse $B000. Afin de pouvoir aisément adresser les octets successifs composant l'expression, nous utiliserons l'adressage indirect post-indéxé ; celui-ci n'est toutefois utilisable qu'en page zéro (adresses hexadécimales de $00 à $FF).

Dans ce mode d'adressage, le registre d'index est obligatoirement Y. On chargera donc au préalable l'adresse de début de l'expression à prononcer en page zéro, poids faible en tête à une adresse paire. Ici les adresses hexadécimales $64 et $65 sont utilisées.

Enfin, pour simplifier cette première application, nous ne lirons que l'octet de poids faible de la longueur d'expression, ce qui ne permettra de prononcer que des expressions de moins de 256 octets (c'est le cas de la plupart des mots isolés du vocabulaire fourni en annexe) ; cet octet sera stocké à l'adresse hexadécimale $B0FF et comparé au registre d'index Y après chaque trame pour détecter la fin de l'expression qui est atteinte lorsque ces deux valeurs sont égales. Le listing source commenté de ce programme est donné ci-dessous.

Adr. Op.code Etiq. Mnémo. Commentaire

Programme principal

```
B000  A0 01     DEBU: LDY $01     Index = 1 (longueur p. faible)
B002  B1 64           LDA (64),Y  Acc = longueur expression
B004  8D FF B0        STA B0FF    B0FF = long. expression
B007  A0 03           LDY $03     Index = 3 (pitch initial)
B009  20 2D B0        JSR TREQ    Test de REQ
B00C  20 35 B0        JSR STOP    Initialisation MEA 8000
B00F  20 3B B0        JSR TRAM    Envoi pitch initial + incr.Y
B012  20 2D B0  NEXT: JSR TREQ    Test de REQ
B015  20 3B B0        JSR TRAM    1e octet trame + incr.Y
B018  20 3B B0        JSR TRAM    2e octet trame + incr.Y
B01B  20 3B B0        JSR TRAM    3e octet trame + incr.Y
B01E  20 3B B0        JSR TRAM    4e octet trame + incr.Y
B021  CC FF B0        CPY B0FF    Index = Longueur? (Y = L?)
B024  30 EC           BMI NEXT    Si NON, tr. suivante (NEXT)
B026  20 2D B0        JSR TREQ    Si OUI, test de REQ
B029  20 35 B0        JSR STOP    Et envoi cde STOP finale
B02C  60              RTS         Fin de l'expression
```

Sous-programme de test du bit d'état (REQ)

```
B02D A9 80     TREQ: LDA $80    Acc = 80H (1000 0000)
B02F CD FF 03        CMP 03FF   Mot état = Acc? (REQ = 1?)
B032 D0 F9           BNE TREQ Si NON, recommencer
B034 60              RTS
```

Sous-programme d'envoi du mot de commande (STOP)

```
B035 A9 1A     STOP: LDA $1A    Arrêt lent + REQ sur D7
B037 8D FF 03        STA 03FF   Envoi commande (A0 = 1)
B03A 60              RTS
```

Sous-programme d'envoi d'un octet de données vocales (TRAM)

```
B03B B1 64     TRAM: LDA (64),Y Acc = octet à envoyer
B04D 8D FE 03        STA 03FE   Envoi donnée vocale (A0 = 0)
B040 C8              INY        Incrément. index (Y = Y + 1)
B041 60              RTS
```

Routine de prononciation d'une expression de longueur quelconque

Ce programme correspond à une application sur APPLE et utilise le même principe d'interfaçage, mais le décodage d'adresse est réduit à sa plus simple expression par connexion de la borne DEVICE SELECT de l'APPLE au $\overline{CE}$ du MEA 8000. De ce fait, les adresses décodées sont $C0C0 et $C0C1 si la carte est connectée sur le slot n° 4. Ici, le programme permet la prononciation d'une expression de longueur et d'adresse de début quelconques.

L'adresse de début sera préalablement chargée en page zéro aux adresses $FE (poids faible) et $FF (poids fort), de façon à permettre l'adressage indirect post-indéxé par Y. Cette indexation ne pouvant se faire que sur 256 octets, nous incrémenterons le contenu de l'adresse $FF chaque fois que Y passera par 0 ("retenue"), ainsi que le registre X qui servira à faire la comparaison sur le poids fort de la longueur pour détecter la fin de l'expression. Le programme débute à l'adresse $6EF3 (DEBU) par la lecture de la longueur de l'expression et sa sauvegarde aux adresses $6FFE (poids fort) et $6FFF (poids faible). On peut le commencer à l'adresse $6F00 (DEBU2) si ces opérations sont effectuées par un autre programme appelant celui-ci.

Les sous-programmes TREQ et STOP sont identiques à ceux du précédent, à l'exception des adresses, et seuls le programme principal et le sous-programme TRAM diffèrent en raison de l'utilisation des registres X et Y. Le listing source commenté est représenté à la page 50 .

```
Adr. Op.code    Etiq.    Mnémo.      Commentaire
Programme principal
6EF3 A0 00      DEBU:  LDY $00      Y = 0 (index p. fort longueur)
6EF5 B1 FE             LDA (FE),Y  Lecture poids fort longueur
6EF7 8D FE 6F          STA 6FFE    Sauvegarde en 6FFE
6EFA C8                INY         Y = 1 (index p. faible long.)
6EFB B1 FE             LDA (FE),Y  Lecture p. faible longueur
6EFD 8D FF 6F          STA 6FFF    Sauvegarde en 6FFF
6F00 A2 00      DEBU2: LDX $00     X = 0 (poids fort ad. relative)
6F02 A0 03             LDY $03     Y = 3 (pitch initial)
6F04 20 2D 6F          JSR TREQ    Test de REQ
6F07 20 35 6F          JSR STOP    Initialisation MEA 8000
6F0A 20 3B 6F          JSR TRAM    Envoi pitch initial + incr.Y
6F0D 20 2D 6F  NEXT:   JSR TREQ    Test de REQ
6F10 20 3B 6F          JSR TRAM    1e octet trame + incr.Y
6F13 20 3B 6F          JSR TRAM    2e octet trame + incr.Y
6F16 20 3B 6F          JSR TRAM    3e octet trame + incr.Y
6F19 20 3B 6F          JSR TRAM    4e octet trame + incr.Y
6F1C CC FF 6F          CPY 6FFF    Y = Poids faible longueur ?
6F1F D0 EC             BNE NEXT    Si NON, tr. suivante (NEXT)
6F21 EC FE 6F          CPX 6FFE    X = Poids fort longueur ?
6F24 30 E7             BMI NEXT    Si NON, tr. suivante (NEXT)
6F26 20 2D 6F          JSR TREQ    Si OUI, test de REQ
6F29 20 35 6F          JSR STOP    Et envoi cde STOP finale
6F2C 60                RTS         Fin de l'expression
Sous-programme de test du bit d'état (REQ)
6F2D A9 80      TREQ:  LDA $80     Acc = 80H (1000 0000)
6F2F CD C1 C0          CMP C0C1    Mot état = Acc? (REQ = 1?)
6F32 D0 F9             BNE TREQ    Si NON, recommencer
6F34 60                RTS
Sous-programme d'envoi du mot de commande (STOP)
6F35 A9 1A      STOP:  LDA $1A     Arrêt lent + REQ sur D7
6F37 8D C1 C0          STA C0C1    Envoi commande (A0 = 1)
6F3A 60                RTS
Sous-programme d'envoi d'un octet de données vocales (TRAM)
6F3B B1 FE      TRAM:  LDA (FE),Y  Acc = octet à envoyer
6F3B 8D C0 C0          STA C0C0    Envoi donnée vocale (A0 = 0)
6F40 C8                INY         Incrément. index (Y = Y + 1)
6F41 C0 00             CPY $00     Poids faible longueur = 0?
6F43 D0 03             BNE 6F48    Si NON, retour
6F45 E8                INX         Si OUI, incrémenter X ...
6F46 E6 FF             INC FF      Et le contenu de FF (p. fort)
6F48 60                RTS
```

Application au microprocesseur Z-80 (ou Z-80A)

Ces microprocesseurs, utilisés dans de nombreuses machines 8 bits récentes (AMSTRAD, MSX, PHILIPS VG 5000, SHARP...), sont dotés d'instructions et de signaux d'entrée-sortie spécialisés. Ceci permettra de considérer le synthétiseur comme un périphérique et non comme une mémoire, en utilisant le signal IORQ du Z-80.

Le synthétiseur occupera deux adresses consécutives (A0 = 0 et A0 = 1) parmi les 256 possibles par le décodage de A0-A7. La figure 25 représente le schéma général de connexion :

— A0 est directement reliée à la borne A0 (borne 11)
— A1 à A7 et $\overline{IORQ}$ inversé, fournissent par décodage $\overline{CE}$ (borne 12)
— $\overline{RD}$ commande $\overline{R}/W$ du MEA 8000 (borne 22) après compensation du temps de propagation par 2 inverseurs
— D0 à D7 sont directement reliées aux bornes 10 à 3 du MEA 8000.

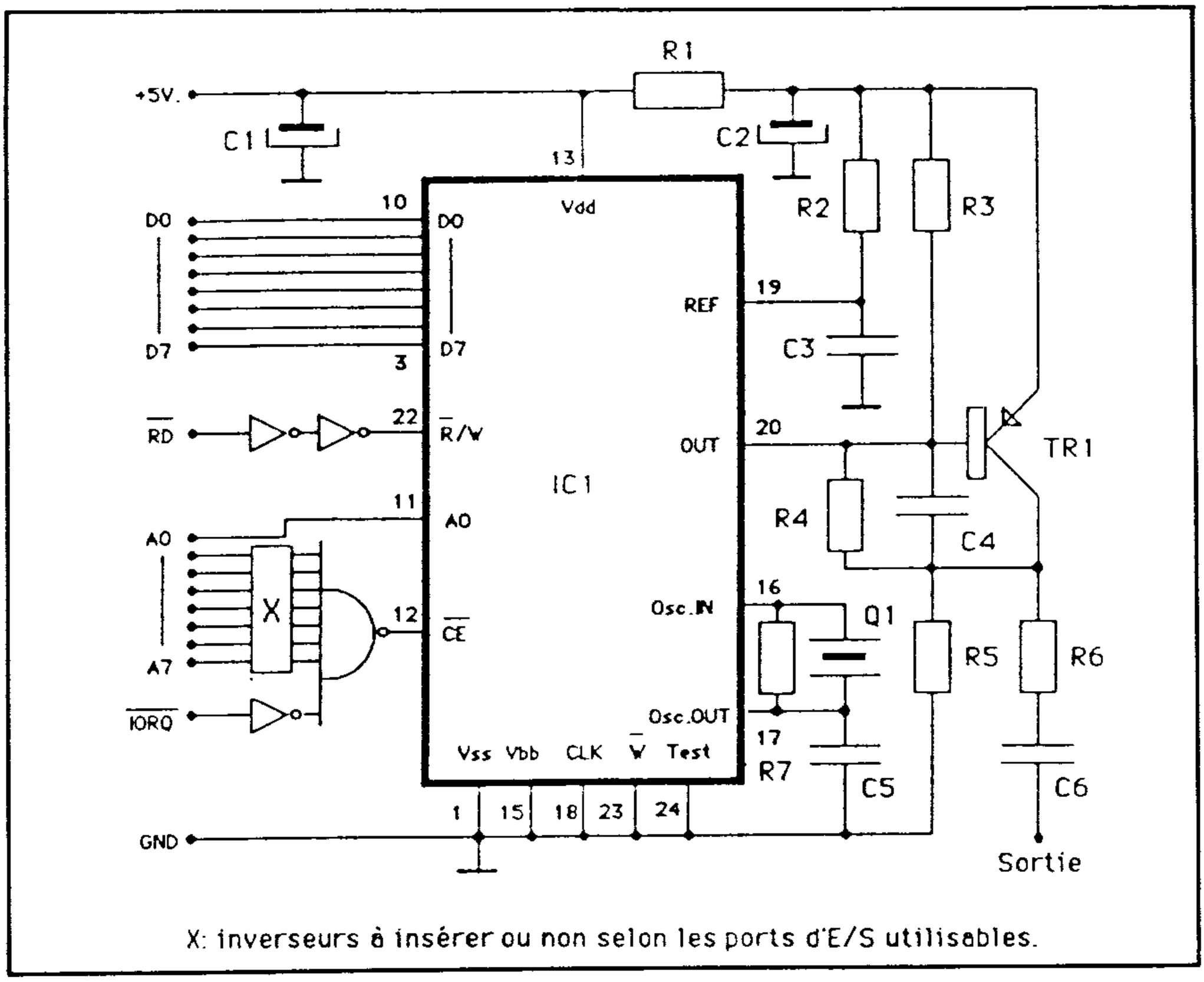

Figure 25 - Schéma général d'interfaçage avec le Z-80

Routine de prononciation d'une expression de longueur quelconque

L'exemple proposé ici correspond à une application sur SHARP MZ-700, et les adresses d'entrée-sortie choisies sont $DE (données) et $DF (commande). Le programme commence à l'adresse $A5B8 et l'adresse de début de l'expression à prononcer doit avoir été préalablement chargée aux adresses $A5FE (poids faible) et $A5FF (poids fort). Le programme commence par stocker l'adresse de début dans la paire de registres HL, qui sera utilisée pour pointer l'octet en cours par incrémentation. La paire de registres BC est ensuite chargée par la longueur de l'expression, et décrémentée à chaque octet; le test de sa valeur après chaque trame permettra de détecter la fin de l'expression lorsque BC deviendra nul. Pour ce programme, dont l'architecture est identique à celle des deux précédents, les mêmes noms d'étiquettes ont été utilisés. Son listing source commenté est visible ci-dessous.

```
Adr. Op.code     Etiq.     Mnémo.        Commentaire

Programme principal

A5B8 00          DEBU:NOP                E5 (PUSH HL) pour VG 5000
A5B9 2A FE A5            LD HL,(A5FE)Adresse expression dans HL
A5BC 46                  LD B,(HL)   Poids fort longueur dans B
A5BD 23                  INC HL      HL pointe p. faible longueur
A5BE 4E                  LD C,(HL)   Poids faible longueur dans C
A5BF 23                  INC HL
A5C0 23                  INC HL      HL pointe pitch initial
A5C1 0B                  DEC BC
A5C2 0B                  DEC BC
A5C3 0B                  DEC BC      BC = longueur restante
A5C4 CD EB A5            CALL TREQ   Test de REQ
A5C7 CD F2 A5            CALL STOP   Initialisation MEA 8000
A5CA CD F7 A5            CALL TRAM   Pitch initial + inc.HL + dec.BC
A5CD CD EB A5NEXT:       CALL TREQ   Test de REQ
A5D0 CD F7 A5            CALL TRAM   1e octet trame + i.HL + d.BC
A5D3 CD F7 A5            CALL TRAM   2e octet trame + i.HL + d.BC
A5D6 CD F7 A5            CALL TRAM   3e octet trame + i.HL + d.BC
A5D9 CD F7 A5            CALL TRAM   4e octet trame + i.HL + d.BC
A5DC AF                  XOR A       Acc = 0
A5DD B9                  CP C        P. faible longueur = 0 ?
A5DE 20 ED               JR NZ,NEXT  Si NON, tr. suivante (NEXT)
A5E0 B8                  CP B        P. fort longueur = 0 ?
A5E1 20 EA               JR NZ,NEXT  Si NON, tr. suivante (NEXT)
A5E3 CD EB A5            CALL TREQ   Si OUI, test de REQ
A5E6 CD F2 A5            CALL STOP   Et envoi cde STOP finale
A5E9 C9                  RET         Fin de l'expression
A5EA 00                  NOP         E1 (POP HL) pour VG 5000
```

Sous-programme de test du bit d'état (REQ)

```
A5EB DB DF    TREQ:IN A,DFH     Acc = Mot d'état
A5ED CB 7F         BIT 7,A      Test de REQ (D7 = 1 ?)
A5EF 28 FA         JR Z,TREQ    Si NON, recommencer
A5F1 C9            RET
```

Sous-programme d'envoi du mot de commande (STOP)

```
A5F2 3E 1A    STOP: LD A,1AH    Arret lent + REQ sur D7
A5F4 D3 DF          OUT DFH,A   Envoi commande (A0 = 1)
A5F6 C9             RET
```

Sous-programme d'envoi d'un octet de données vocales (TRAM)

```
A5F7 7E       TRAM:LD A,(HL)    Acc = octet à envoyer
A5F8 D3 DE          OUT DEH,A   Envoi donnée vocale (A0 = 0)
A5FA 23             INC HL      Incrément. adresse octet
A5FB 0B             DEC BC      Décrem. longueur restante
A5FC C9             RET
```