Chapitre 6

Le clavier

Nous allons étudier ici l'implémentation du périphérique le plus utilisé, à savoir le clavier, avec une description fonctionnelle de l'aspect matériel mais surtout son étude du point de vue de la programmation système.

6.1 Principe du clavier

Comme le montre la figure 6.1, un clavier est essentiellement constitué d'une matrice de lignes et de de colonnes d'interrupteurs.

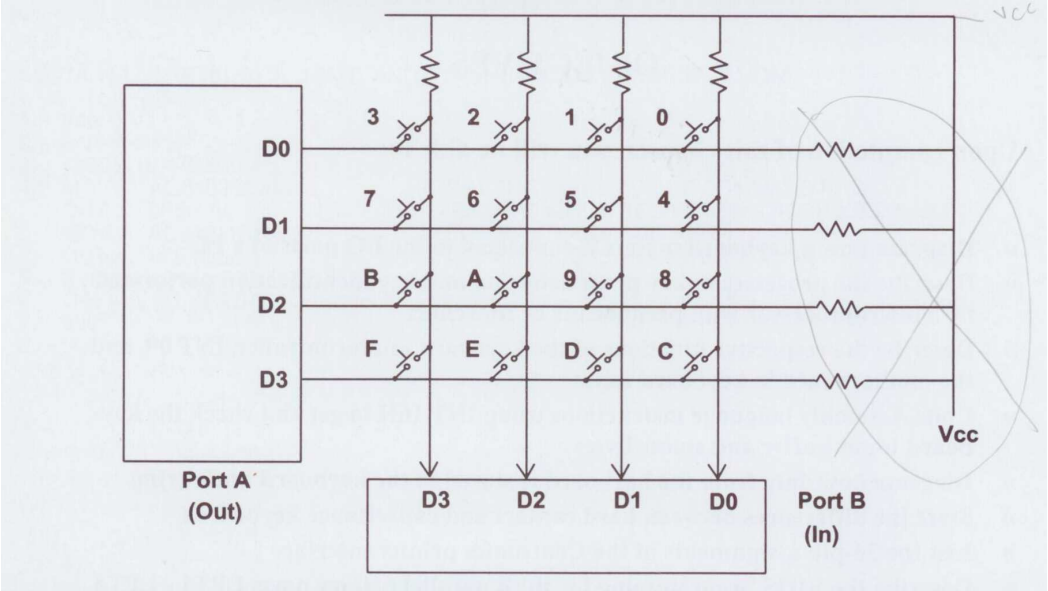


FIGURE 6.1 – Clavier simple

6.1.1 Clavier simple

6.1.1.1 Aspect matériel d'un clavier simple

Dans le cas d'un clavier simple, le microprocesseur accède directement aux lignes et aux colonnes à travers les ports d'entrées/sorties. Lorsqu'on appuie sur une touche, une connexion est établie entre une ligne et une colonne.

6.1.1.2 Aspect logiciel d'un clavier simple

Les problèmes de programmation liés au clavier.- Les problèmes liés à la programmation concernant un clavier sont au nombre de trois :

- **détection** : détecter qu'au moins une touche a été pressée ;
- **anti-rebond** (en anglais *debouncing*) : éliminer le phénomène de **rebond** à la fois lorsqu'on presse une touche et lorsqu'on cesse d'appuyer sur la touche ; en effet Le contact mécanique des touches d'un clavier dure plusieurs millisecondes ; il faut donc faire attention à ne pas prendre en compte plusieurs fois le fait qu'une touche est pressée ou relâchée ;
- **codage** : coder la touche par une valeur donnée.

Anti-rebond.- Le problème du rebond est réglé de façon logicielle en insérant un *délai* suffisamment long pour s'assurer que les contacts sont stabilisés. Ceci se fait à la fois lorque l'appui sur une touche est détecté mais également lorsqu'un relâchement est détecté.

Parcours et identification des touches pressées.- La figure 6.1 montre une matrice 4×4 reliée à deux ports. Les lignes sont reliées à un port sortant et les colonnes à un port entrant. Si on n'a appuyé sur aucune touche, lire le port entrant donnera 1 pour toutes les colonnes, puisqu'elles sont toutes reliées au niveau haut (V_{CC}). Si toutes les lignes sont reliées à la terre et qu'on appuie sur une touche, une des colonnes sera à 0 puisque la touche pressée permet au courant de s'écouler.

Le microprocesseur doit continuellement examiner le clavier pour détecter et identifier la touche qui est pressée. Nous allons voir comment on effectue ceci.

Mise à la terre des lignes et lecture des colonnes.- Pour détecter la touche pressée, le microprocesseur met à la terre toutes les lignes, en appliquant 0 au loquet (*latch*) de sortie, puis lit les colonnes. Si la donnée lue sur les colonnes est D3 – D0 = 1111, c'est qu'aucune touche n'est pressée. On continue ainsi jusqu'à ce qu'une touche soit pressée.

Si un des bits de colonnes est à zéro, c'est qu'une touche est pressée. Par exemple, si D3 – D0 = 1101, ceci signifie qu'une touche de la colonne D1 est pressée. Lorsque le microprocesseur détecte qu'une touche est pressée, il passe à l'identification de celle-ci. Pour cela, il met à la terre la seule ligne du haut et lit les colonnes. Si la donnée lue n'est constituée que de 1, c'est que la touche pressée ne se situe pas sur cette ligne et on passe à la ligne suivante. Ce processus continue jusqu'à ce que la ligne soit identifiée.

Après identification de la ligne, la tache suivante consiste à trouver à quelle colonne appartient la touche pressée.

Bien entendu ce processus en deux temps ne fonctionne que parce que la durée d'appui sur la touche est bien plus long que le cycle du microprocesseur.

Exemple.- À partir du dispositif de la figure 6.1, déterminer quelle est la touche pressée lorsque :

(a) D3 – D0 = 1110 pour la ligne, D3 – D0 = 1011 pour la colonne.
(b) D3 – D0 = 1101 pour la ligne, D3 – D0 = 0111 pour la colonne.

Réponse. (a) La ligne est D0 et la colonne est D2, donc la touche '2' est pressée.
(b) La ligne est D1 et la colonne est D3, donc la touche '7' est pressée.

6.1.1.3 Un programme

Le programme suivant permet la détection et l'identification des touches pressées, en supposant que les constantes `PORT_A` et `PORT_B` soient initialisées et que le sous-programme `DELAY` permette un délai (ici de 20 ms) :

```
;scan codes des touches (a placer dans le segment des donnees)
KCOD_0   DB 0,1,2,3              ; codes des touches de la premiere ligne
KCOD_1   DB 4,5,6,7              ; codes des touches de la seconde ligne
KCOD_2   DB 8,9,0AH,0BH          ; codes des touches de la troisieme ligne
KCOD_3   DB OCH,ODH,OEH,OFH      ; codes des touches de la quatrieme ligne

; A placer dans le segment de code
         PUSH  BX                ;sauvegarder BX
         SUB   AL,AL             ;AL = 0
         OUT   PORT_A,AL         ;pour mettre a la terre les lignes
K1:      IN    AL,PORT_B         ;lire les colonnes
         AND   AL,00001111B      ;masquer les bits non utilises (D7-D4)
         CMP   AL,00001111B      ;toutes les touches sont-elles relachees ?
         JNE   K1                ;sinon recommencer
         CALL  DELAY             ;attendre 20 ms
K2:      IN    AL,PORT_B         ;lire les colonnes
         AND   AL,00001111B      ;masquer D7-D4
         CMP   AL,00001111B      ;une touche est-elle pressee ?
         JE K2                   ;s'il n'y en a aucune
         CALL  DELAY             ;attendre 20 ms pour l'anti-rebond
  ;apres l'anti-rebond regarder si elle est encore pressee
         IN    AL,PORT_B         ;lire les colonnes
         AND   AL,00001111B      ;masquer D7-D4
         CMP   AL,00001111B      ;une touche est-elle pressee ?
         JE K2                   ;si c'etait un bruit ou un effleurement
                                 ;poursuivre l'interrogation
  ;identification en mettant a la terre une ligne a la fois
         MOV   AL,11111110B      ;ligne 0 (D0 = 0)
         OUT   PORT_A,AL         ;mise a la terre
         IN    AL,PORT_B         ;lire les colonnes
         AND   AL,00001111B      ;masquer D7-D4
         CMP   AL,00001111B      ;reperer la colonne
         JE    RO_1              ;si aucune mettre la ligne 1 a la terre
         MOV   BX,OFFSET KCOD_0  ;BX = debut de la table des touches
                                 ;de la colonne 0
         JMP   FIND_IT           ;identifier la touche
RO_1:    MOV   AL,11111101B      ;ligne 1 (D1 = 0)
         OUT   PORT_A,AL         ;mise a la terre
         IN    AL,PORT_B         ;lire les colonnes
         AND   AL,00001111B      ;masquer D7-D4
         CMP   AL,00001111B      ;reperer la colonne
         JE    RO_2              ;si aucune mettre la ligne 2 a la terre
         MOV   BX,OFFSET KCOD_1  ;BX = debut de la table des touches
                                 ;de la colonne 1
         JMP   FIND_IT           ;identifier la touche
```

```
RO_2:    MOV   AL,11111011B       ;ligne 2 (D2 = 0)
         OUT   PORT_A,AL          ;mise a la terre
         IN    AL,PORT_B          ;lire les colonnes
         AND   AL,00001111B       ;masquer D7-D4
         CMP   AL,00001111B       ;reperer la colonne
         JE    RO_3               ;si aucune mettre la ligne 3 a la terre
         MOV   BX,OFFSET KCOD_2   ;BX = debut de la table des touches
                                  ;de la colonne 2
         JMP   FIND_IT            ;identifier la touche
RO_3:    MOV   AL,11110111B       ;ligne 3 (D3 = 0)
         OUT   PORT_A,AL          ;mise a la terre
         IN    AL,PORT_B          ;lire les colonnes
         AND   AL,00001111B       ;masquer D7-D4
         CMP   AL,00001111B       ;reperer la colonne
         JE    K2                 ;si aucune alors fausse entree, recommencer
         MOV   BX,OFFSET KCOD_3   ;BX = debut de la table des touches
                                  ;de la colonne 3
;Une touche a ete detectee et la ligne identifiee.
;Trouver maintenant laquelle
FIND_IT: RCR   AL,1               ;rotation de l'entree de colonne pour
                                  ;chercher 0
         JNC   MATCH              ;si zero, aller au code
         INC   BX                 ;sinon, pointer vers le code suivant
         JMP   FIND_IT            ;et continuer a chercher
;Obtenir le code de la touche pressee et le renvoyer
MATCH:   MOV   AL,[BX]            ;obtenir le code pointe par BX
         POP   BX                 ;revenir avec AL=code de la touche pressee
         RET
;Delai (voir ailleurs)
```

Commentaires.- 1^o) Pour être sûr que la touche précédente a été relâchée, des 0 sont envoyés à toutes les lignes et les colonnes sont lues et vérifiées jusqu'à ce que toutes les colonnes soient à un niveau haut.

Lorsque la touche est relâchée, le programme attend un peu (20 ms) avant de passer à l'étape suivante, c'est-à-dire attendre qu'une touche soit pressée.

- 2^o) Pour voir si une touche est pressée, les colonnes sont examinées encore et encore jusqu'à ce que l'une d'entre elles ait un niveau bas. Lorsqu'une touche pressée est détectée, on attend un petit peu pour être sûr qu'il ne s'agit pas d'un simple effleurement de la part de l'utilisateur et on recommence. Si après ce délai de 20 ms, une touche est encore pressée, on passe à l'étape suivante de détection de la ligne à laquelle appartient cette touche.

- 3^o) Pour détecter à quelle ligne appartient la touche pressée, on met à la terre une ligne à la fois (et non plus toutes en même temps). Si on trouve que toutes les colonnes sont à un niveau haut, ceci signifie que la touche pressée n'appartient pas à cette ligne. On passe donc à la ligne suivante et on continue ainsi jusqu'à ce qu'on trouve la ligne à laquelle appartient la touche pressée.

Il se peut qu'aucune ligne ne soit trouvée, par exemple si l'appui sur la touche n'est pas suffisamment longue. Dans ce cas, on revient à l'interrogation.

Lorsqu'on trouve la ligne à laquelle appartient la touche pressée, on initialise l'adresse du début de la table contenant les scan codes de la ligne en question.

- 4^o) Pour identifier la touche pressée, on effectue une rotation des bits de la colonne, un bit à la fois, dans l'indicateur de report CF et on regarde si ce dernier est à niveau bas. Si c'est le cas, on récupère le scan code de la touche pressée dans la table. Sinon on incrémente l'index de la table.

- 5^o) Ce programme convient dans la situation où on suppose qu'une seule touche peut être pressée à la fois. Ceci est le cas du clavier d'une petite calculette, par exemple. Dans le cas d'une calculette plus perfectionnée ou d'un ordinateur, plusieurs touches peuvent être légalement pressées en même temps, par exemple la touche 'Maj' et la touche 'a' pour indiquer 'A'.

Le principe de la détection est le même, sauf qu'on peut avoir plusieurs réponses au lieu d'une seule.

6.1.2 Circuit intégré de détection

Le microprocesseur est fortement sollicité dans la détection et l'identification de la touche pressée. Il existe des circuits intégrés spécialisés, tels que le MM74C923 de *National Semiconductor* qui permet l'examen du clavier et le décodage, sans être un microprocesseur généraliste.

6.2 Aspect matériel

Notre but est de faire de la programmation système et non de construire ou interfacer un clavier. Nous allons donc dire l'essentiel pour ce but de l'aspect matériel, en passant très rapidement sur ce qui n'a pas d'incidence sur sa programmation.

D'un point de vue externe, le clavier du PC est contitué essentiellement d'une *planche* munie de nombreuses *touches*, dans certains cas de LED, et d'un *câble* terminé par une broche DIN.

6.2.1 Disposition des touches

Le PC a été livré successivement avec trois claviers, maintenant appelés **clavier PC/XT**, puisqu'il a accompagné le PC d'origine et le PC/XT, le **clavier AT**, puisqu'il a accompagné le PC/AT et le **clavier étendu**, dit aussi *clavier MF II* pour *Multi-Function 2*. Les fonctions des touches ont une compatibilité ascendante en ce sens que l'on retrouve toujours une touche donnée sur le clavier plus récent, bien que pas nécessairement au même endroit, avec des touches supplémentaires à chaque fois.

Le nombre de touches est le même, pour un clavier donné, pour tous les pays mais le caractère gravé sur chacune d'elles dépend de celui-ci. Nous allons décrire le clavier américain, dit **disposition qwerty** (en référence aux premières lettres de la première ligne des lettres) et non le clavier français, dit **disposition azerty**.

Le clavier PC/XT comporte 83 touches :

FIGURE 6.2 – Clavier PC/XT

que l'on peut schématiser de la façon suivante :

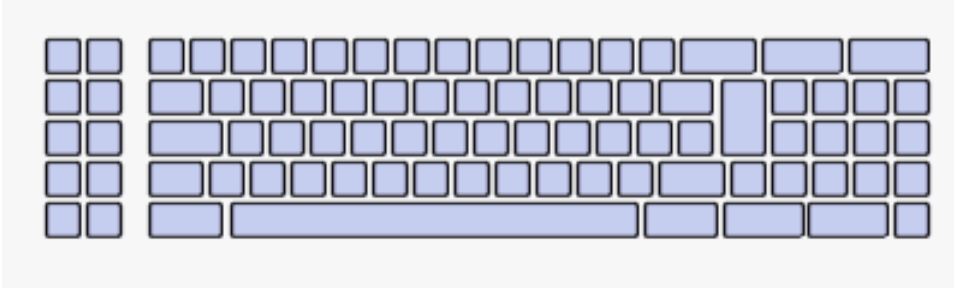


FIGURE 6.3 – Disposition des touches du clavier PC/XT

Il y a trois groupes principaux de touches : à gauche dix touches dites de fonction, étiquetées de F1 à F10 ; au milieu les touches d'une machine à écrire ordinaire, en particulier les lettres et les chiffres ainsi que certains caractères spéciaux tels que le changement de mode, des symboles spéciaux, le verrouillage majuscules et chiffres, le retour arrière et la recopie d'écran ; à droite un pavé numérique de 15 touches qui selon la position de la touche 'NumLock' constituent un clavier de type calculatrice ou gère le curseur.

Du point de vue ergonomique, ce clavier avait des défauts importants : la touche « entrée » et les deux touches « Majuscule » étaient beaucoup trop petites.

6.2.2 Clavier intelligent

Le clavier de l'IBM PC est un clavier « intelligent » dans la mesure où il embarque un micro-contrôleur, constitué d'un microprocesseur, de RAM et d'EPROM, et de plusieurs ports d'entrées-sorties, tout cela sur un même circuit intégré.

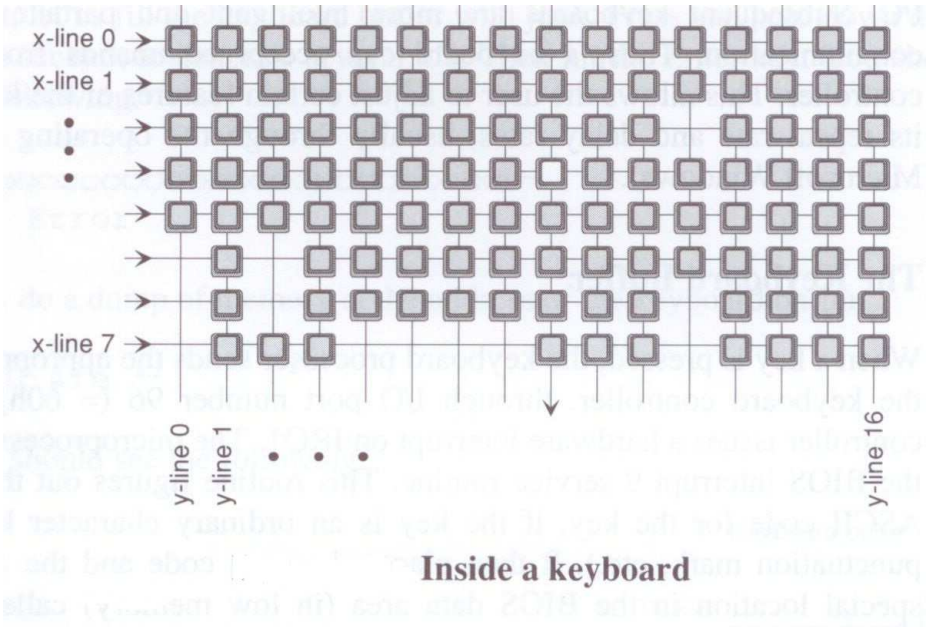


FIGURE 6.4 – Matrice d'un clavier

Le micro-contrôleur utilisé à l'origine est un 8042 d'*Intel*. Le programme stocké dans l'EPROM a pour principales fonctions :

— la réalisation d'un diagnostic de vérification lors de la mise sous tension,

— l'examen continuel du clavier et l'identification des touches appuyées et relâchées,

— la gestion d'une zone tampon (*buffer*) permettant de mémoriser jusqu'à vingt touches de clavier,

— la gestion de la communication bidirectionnelle en série avec la carte mère lors du transfert de chaque code de recherche.

6.2.3 Câble

Pour permettre au clavier d'être détaché de l'unité centrale, il est relié à la carte mère grâce à un câble.

Pour éviter la déperdition des données, le transfert des données entre le clavier (plus exactement le micro-contrôleur du clavier) et l'unité centrale (plus exactement le bus des données de la carte mère) s'effectue en série.

Le câble se termine par le **connecteur de clavier**, qui est un connecteur DIN à cinq broches.

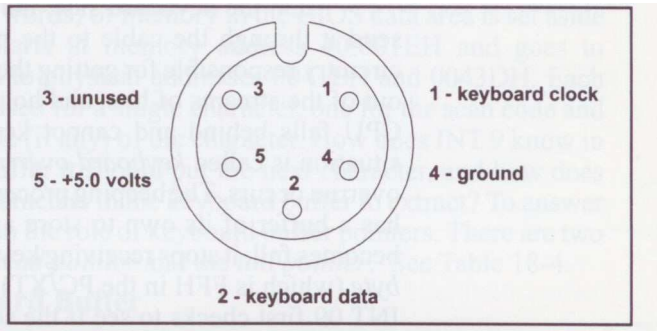


FIGURE 6.5 – Câble du clavier du PC

La figure 6.5 donne la numérotation des broches et leurs fonctions :

- Le signal d'**horloge du clavier** (broche 1) permet de décider du moment de l'échange des données entre le clavier et l'interface du clavier. Le transfert se fait donc de façon synchrone, contrairement à ce qui fait pour la liaison série avec l'UART 8250.
- Le signal de **données du clavier** (broche 2) permet l'échange des données entre le clavier et l'interface du clavier, en série, la structure **SDU** (pour *Service Data unit*) occupant 11 bits, répartis de la façon suivante :
  - un bit de début (**STRT** pour *START*, toujours égal à 0),
  - les huit bits du code de recherche, de **DB0** à **DB7** correspondent aux bits de 0 à 7 de l'octet envoyé,
  - un bit **PAR** de parité impaire,
  - un bit **STOP** de fin (toujours égal à 1).

Le programmeur du PC n'a pas vraiment à se préoccuper de cette liaison série car on utilise sur la carte mère du PC/XT un registre à décalage série-parallèle, le 74LS322, pour recevoir les données. Il récupère le code de recherche transmis à partir de la trame reçue et le place sur le port A du 8255, situé au port d'entrée-sortie 60h.

6.3 Aspect logiciel

6.3.1 Les codes des touches

6.3.1.1 Principe

Notion.- Nous avons vu deux dispositions de clavier : azerty et qwerty. Vous avez peut-être déjà vu des claviers pour d'autres langues que le français et l'anglais, pour le russe par exemple.

Comment faire face à cette diversité ?

Chaque clavier PC/XT, par exemple, a 83 touches, toujours disposées de la même façon (mais marquées de caractères différents), numérotées de 1 à 83. Ce numéro est appelé son **code clavier** (*scan code* en anglais, soit code de recherche, sous-entendu dans une table).

On utilise ensuite une *table de données* des valeurs désirées pour les touches, le code clavier étant utilisé comme index. Ceci permet de changer les valeurs (voir par exemple clavier *azerty* contre clavier *qwerty*).

Codes d'appui et de relâchement.- Le micro-contrôleur du clavier envoie le code clavier d'une touche lorsqu'on appuie sur celle-ci (on parle alors de **code d'appui**, *make-code* en anglais), mais également une donnée lorsqu'on relâche celle-ci (on parle alors de **code de relâchement**, *break-code* en anglais).

Le code de relâchement d'une touche est simplement le code d'appui de celle-ci avec un 1 pour le septième bit, c'est-à-dire qu'on lui a ajouté 128.

Le code de relâchement est utilisé, par exemple, pour savoir si on a appuyé sur la touche 'Majuscule' et qu'on ne l'a pas relâchée. Ceci permet de savoir, de façon logicielle, si on veut un 'c' ou un 'C'.

Nombre maximum de touches.- Remarquons que puisque le micro-contrôleur de clavier envoie un octet à l'unité centrale et que chaque touche doit avoir un code lorsqu'on appuie dessus et un lorsqu'on la relâche, on est limité à 128 touches pour le clavier. Ceci est suffisant pour le moment, en utilisant les combinaisons de touches pour obtenir plus de caractères.

Différence entre majuscule et minuscule.- Le code clavier d'une lettre minuscule est le même que celui de la majuscule correspondante. Nous verrons que c'est la routine de service de l'interruption 9h du BIOS qui fait la différence en vérifiant qu'on a appuyé sur la touche majuscule et qu'on ne l'a pas relâchée.

Répétition des touches.- Une fonction de répétition est implémentée sur le micro-contrôleur du clavier, qui répète le transfert du code d'appui de la touche sur laquelle on appuie jusqu'à ce qu'on n'en ait plus besoin. Cela évite, par exemple, d'appuyer 80 fois sur la touche '-' pour remplir une ligne.

Sur le PC/XT, le taux de répétition est fixé à 10 caractères par seconde. C'est le micro-contrôleur du clavier qui s'occupe de déterminer cette répétition, le programmeur du PC proprement dit n'a pas à s'en occuper.

6.3.1.2 Numérotation des touches

La figure 6.6 montre les codes clavier des touches du clavier PC/XT. Les touches sont tout simplement énumérées continûment.

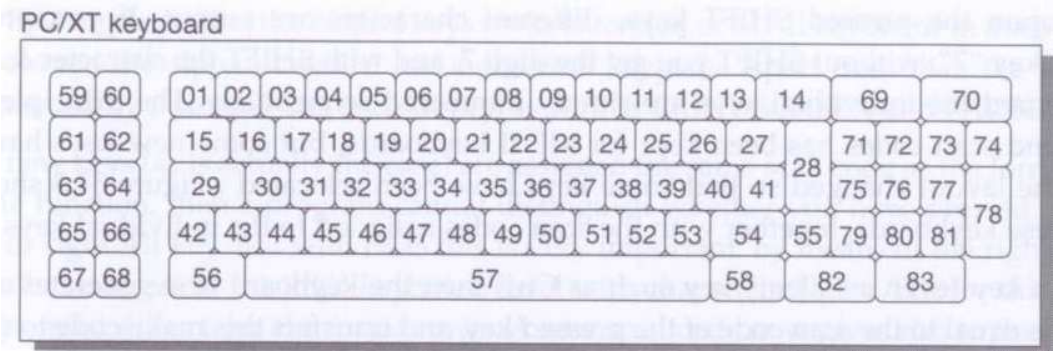


FIGURE 6.6 – Numérotation des touches du clavier PC/XT

6.3.1.3 Caractères associés aux touches

Le tableau suivant donne les caractères du clavier américain associées aux numéros des touches [IBM-83], sans et avec la touche majuscule enfoncée, et enfin avec 'Ctrl', ce que l'on peut vérifier pour les deux premiers cas grâce aux figures 6.2 et 6.6 :

Déc.	Hex	Base	Majuscule	Ctrl
1	01	Esc	Esc	Esc
2	02	1	!	-1
3	03	2	@	Nul (000)
4	04	3	#	-1
5	05	4	$	-1
6	06	5	%	-1
7	07	6	^	RS (030)
8	08	7	&	-1
9	09	8	*	-1
10	0A	9	(	-1
11	0B	0	)	-1
12	0C	-	_	US (031)
13	0D	=	+	-1
14	0E	Backspace (008)	Backspace (008)	Del (127)
15	0F	tab (009)	tab inverse	-1
16	10	q	Q	DC1 (017)
17	11	w	W	ETB (023)
18	12	e	E	ENQ (005)
19	13	r	R	DC2 (018)
20	14	t	T	DC4 (020)
21	15	y	Y	EM (025)
22	16	u	U	NAK (021)
23	17	i	I	HT (009)
24	18	o	O	SI (015)
25	19	p	P	DLE (016)
26	1A	[	{	Esc (027)
27	1B	]	}	GS (029)
28	1C	enter	enter	LF (010)
29	1D	Pas de code ASCII : touche de décalage ctrl		
30	1E	a	A	SOH (001)
31	1F	s	S	DC3 (019)
32	20	d	D	EOT (004)
33	21	f	F	ACK (006)
34	22	g	G	BEL (007)
35	23	h	H	BS (008)
36	24	j	J	LF (010)
37	25	k	K	VT (011)
38	26	l	L	FF (012)
39	27	;	:	-1
40	28	'	"	-1
41	29	`	~	-1

Déc.	Hex	Base	Majuscule	Ctrl
42	2A	Pas de code ASCII : touche de décalage LeftShift		
43	2B	\	|	FS (028)
44	2C	z	Z	SUB (026)
45	2D	x	X	CAN (024)
46	2E	c	C	ETX (003)
47	2F	v	V	SYN (022)
48	30	b	B	STX (002)
49	31	n	N	SO (014)
50	32	m	M	CR (013)
51	33	,	<	-1
52	34	.	>	-1
53	35	/	?	-1
54	36	Pas de code ASCII : touche de décalage RightShift		
55	37	*	PrtSc	
56	38	Pas de code ASCII : touche de décalage Alt		
57	39	Spacebar	Spacebar	Spacebar
58	3A	Pas de code ASCII : touche de décalage CapsLock		
59	3B	Pas de code ASCII : touche F1		
60	3C	Pas de code ASCII : touche F2		
61	3D	Pas de code ASCII : touche F3		
62	3E	Pas de code ASCII : touche F4		
63	3F	Pas de code ASCII : touche F5		
64	40	Pas de code ASCII : touche F6		
65	41	Pas de code ASCII : touche F7		
66	42	Pas de code ASCII : touche F8		
67	43	Pas de code ASCII : touche F9		
68	44	Pas de code ASCII : touche F10		
69	45	Pas de code ASCII : touche de décalage NumLock		
70	46	Pas de code ASCII : touche de décalage ScrollLock		

Pour les dernières touches, cela dépend si on a appuyé ou non sur ‘NumLock’ plutôt que sur ‘Majuscule’ :

Déc.	Hex	NumLock	Base	Ctrl
71	47	7	Home	Clear Screen
72	48	8	UpArrow	Top of Text and Home
73	49	9	PgUp	-1
74	4A	- (gray)		Reverse Word
75	4B	4	LeftArrow	-1
76	4C	5	-1	-1
77	4D	6	RightArrow	Advance Word
78	4E	+ (gray)	+	-1
79	4F	1	End	Erase to EOL
80	50	2	DownArrow	-1
81	51	3	PgDn	Erase to EOL
82	52	0	Ins	-1
83	53	.	Del	

6.3.2 Zone de communication BIOS du clavier

Nous avons vu au chapitre 3 la notion de zone de communication du BIOS et la partie concernant le clavier, comprise entre les adresses `40:17h` et `40:97h`.

6.3.2.1 Octets d'état du clavier

La zone de communication du BIOS contient deux octets d'état du clavier (*Keyboard shift status byte*).

Le premier octet, nommé KB_FLAG, se trouve à l'emplacement `40:17h` :

```
| d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 |
```

les bits ayant les significations suivantes (lignes 97 à 104 du listing du BIOS) :

— d0 vaut 1 lorsque 'RightShift' a été pressé et n'a pas été relâché ;
— d1 vaut 1 lorsque 'LeftShift' a été pressé et n'a pas été relâché ;
— d2 vaut 1 lorsque 'Ctrl' a été pressé et n'a pas été relâché ;
— d3 vaut 1 lorsque 'Alt' a été pressé et n'a pas été relâché ;
— d4 vaut 1 lorsque 'ScrollLock' a été pressé un nombre impair de fois ;
— d5 vaut 1 lorsque 'NumLock' a été pressé un nombre impair de fois ;
— d6 vaut 1 lorsque 'CapsLock' a été pressé un nombre impair de fois ;
— d7 vaut 1 lorsque 'Insert' a été pressé un nombre impair de fois.

Le second octet, nommé KB_FLAG_1, se trouve à l'emplacement `40:18h` :

```
| d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 |
```

les bits ayant les significations suivantes (lignes 108 à 112 du listing du BIOS) :

— d3 vaut 1 lorsque 'Pause' (combinaison de Ctrl et de Numlock) est activé ;
— d4 vaut 1 lorsque 'ScrollLock' a été pressé et n'a pas été relâché ;
— d5 vaut 1 lorsque 'NumLock' a été pressé et n'a pas été relâché ;
— d6 vaut 1 lorsque 'CapsLock' a été pressé et n'a pas été relâché ;
— d7 vaut 1 lorsque 'Insert' a été pressé et n'a pas été relâché.

6.3.2.2 Le tampon du clavier

Le BIOS écrit les caractères correspondants aux touches sur lesquelles on a appuyé dans un tampon temporaire, appelé **tampon du clavier** (*keyboard buffer*, nommé KB_BUFFER), de 32 octets. Il commence à l'adresse `40:1Eh` et se termine donc à l'adresse `40:3Dh`.

Structure du tampon du clavier.- Chaque caractère est stocké dans le tampon sous la forme de deux octets : l'octet de poids fort est le code clavier et celui de poids faible le code ASCII correspondant sur le clavier qwerty américain. Le tampon peut donc stocker temporairement 16 caractères.

Seuls les caractères ASCII sont placés dans le tampon. Les touches spéciales servent seulement à déterminer ceux-ci, de même que les codes de relâchement.

Code clavier étendu.- La plupart des touches ou combinaison de touches donnent lieu à un caractère ASCII. Dans ce cas les deux octets placés dans le tampon sont, comme nous venons de le dire, le code clavier et le code ASCII. Cependant ce n'est pas toujours possible : par exemple si on a appuyé sur la touche de fonction F1, il n'existe pas de code ASCII correspondant. Dans ce cas on place 0 dans l'octet de poids faible (qui devrait contenir le code ASCII) et ce qu'on appelle un **code clavier étendu** dans l'octet de poids fort. Cela ne permet pas de mettre 0 pour le code ASCII 0, qui se retrouvera comme code clavier étendu :

code clavier étendu	Obtenu par
3	caractère nul
15	Shift + Tab
16	Alt + Q
17	Alt + W
18	Alt + E
19	Alt + R
20	Alt + T
21	Alt + Y
22	Alt + U
23	Alt + I
24	Alt + 0
25	Alt + P
30	Alt + A
31	Alt + S
32	Alt + D
33	Alt + F
34	Alt + G
35	Alt + H
36	Alt + J
37	Alt + K
38	Alt + L
44	Alt + Z
45	Alt + X
46	Alt + C
47	Alt + V
48	Alt + B
49	Alt + N

50	Alt + M
59	F1
60	F2
61	F3
62	F4
63	F5
64	F6
65	F7
66	F8
67	F9
68	F10
71	Home
72	Cursor Up ↑
73	Page Up ⇑
75	Cursor Left ←
77	Cursor Right →
79	End
80	Cursor Down ↓
81	Page Down ⇓
82	Ins (Insert)
83	Del (Delete)
84	Shift + F1
85	Shift + F2
86	Shift + F3
87	Shift + F4
88	Shift + F5
89	Shift + F6
90	Shift + F7
91	Shift + F8
92	Shift + F9
93	Shift + F10
94	Ctrl + F1
95	Ctrl + F2
96	Ctrl + F3
97	Ctrl + F4
98	Ctrl + F5
99	Ctrl + F6
100	Ctrl + F7
101	Ctrl + F8
102	Ctrl + F9
103	Ctrl + F10
104	Alt + F1
105	Alt + F2
106	Alt + F3
107	Alt + F4
108	Alt + F5
109	Alt + F6
110	Alt + F7
111	Alt + F8

112	Alt + F9
113	Alt + F10
114	Ctrl + PrtSc (Start/Stop Echo to Printer)
115	Ctrl + Cursor Left (Reverse Word)
116	Ctrl + Cursor Right (Advance Word)
117	Ctrl + End [Erase to End of Line (EOL)]
118	Ctrl + PgDn [Erase to End of Screen (EOS)]
119	Ctrl + Home (Clear Screen and Home)
120	Alt + 1
121	Alt + 2
122	Alt + 3
123	Alt + 4
124	Alt + 5
125	Alt + 6
127	Alt + 7
128	Alt + 9
129	Alt + 0
130	Alt + '-'
132	Alt + '='
133	Ctrl + PgUp (Top 25 Mines of Text and Home Cursor)

Gestion du tampon du clavier.- Le tampon du clavier est organisé comme un tampon circulaire géré grâce à deux index. Les valeurs de ces index sont stockées dans la zone de données du BIOS aux adresses `40:1Ah` et `40:1Ch`, d'un mot chacun. Le premier, l'**index d'écriture** (*head pointer* KB_HEAD), indique la position libre suivante dans le tampon du clavier, là où le prochain caractère sera stocké. Le second index, l'**index de lecture** (*tail pointer* KB_TAIL), indique la position du prochain caractère à lire dans le tampon, c'est-à-dire le premier caractère à fournir à un programme. En raison du caractère circulaire du tampon, il se peut que la valeur de l'index de lecture soit supérieure à celle de l'index d'écriture.

Dans le cas où l'index de lecture est strictement inférieur à l'index d'écriture, les mots pointés par une valeur située entre l'index de lecture strictement inférieure à celle de l'index d'écriture sont des caractères provenant du clavier.

Dans le cas où l'index de lecture est strictement plus grand que l'index d'écriture, les mots compris entre l'index de lecture et la fin du tampon (`40:3Dh`) et ceux compris entre le début du tampon (`40:1Eh`) et l'index d'écriture sont des caractères provenant du clavier.

Le tampon est vide si les index de lecture et d'écriture coïncident.

Le tampon est plein si l'index d'écriture vient juste avant l'index de lecture. Ceci a pour conséquence que, bien que le tampon ait une capacité de 32 octets, il peut seulement stocker 15 caractères de deux octets chacun.

L'index d'écriture est mis à jour par la routine de service de l'interruption 09h alors que l'index de lecture est mis à jour par celle de l'interruption 16h.

Lorsqu'on lit un caractère dans le tampon grâce à l'interruption INT 16h du BIOS, le pointeur d'écriture est mis à jour. Le caractère est ainsi retiré logiquement du tampon, bien que le code ASCII et le code de recherche soient encore présents physiquement, tant qu'ils n'ont pas été remplacés par une écriture.

Dépassement de capacité du tampon du clavier.- Lorsque le tampon est plein et qu'on appuie à nouveau sur une touche du clavier, on se trouve dans la situation d'un dépassement de capacité du tampon du clavier (*keyboard overrun*).

Le BIOS émet alors un bip sonore. Si aucune application ne récupère de caractères dans le tampon, le système est bloqué.

La circuiterie du clavier possède son propre tampon pour stocker un maximum de 20 touches. Lorsque ce tampon est plein, il ne prend plus en compte aucune touche et envoie un octet spécial, appelé **octet de dépassement de capacité** (*overrun byte*) à la carte mère (qui est FFh pour le PC/XT et 00h pour le PC/AT).

Après avoir récupéré le code de recherche, la routine de service de INT 09h verifie s'il s'agit de l'octet de dépassement de capacité. Si c'est le cas, elle émet un bip. Sinon elle teste s'il s'agit d'une touche majuscule, etc.

Exemple.- Si l'on tape au début 'abc<enter>', `INT 09h` stocke les caractères de la façon suivante :

- 'a' dans le tampon à l'adresse `41Eh` et son scan code `1Eh` à l'adresse `41Fh` ;
- 'b' à l'adresse `420h` et son scan code `30h` à l'adresse `421h` ;
- 'c' à l'adresse `422h` et son scan code `2Eh` à l'adresse `423h` ;
- '<Enter>' à l'adresse `424h` et son scan code `E0h` à l'adresse `425h`.

La valeur de l'index d'écriture est alors `426h` :

```
a     1Eh   b     30h   c     2Eh   Ent   E0h   ...
41E   41F   420   421   422   423   424   425   426
```

Lorsque le programme a fait appel quatre fois à `INT 16h`, tous les caractères sont lus et l'origine est aussi à l'adresse `426h`, donc le tampon est vide.

Supposons maintenant que l'utilisateur tape 'fghijklmnopqrs<Enter>'. `INT 09h` stocke les caractères en commençant avec comme 'index d'écriture l'adresse `426h` :

```
 r   s  Ent  *   f   g   h   i   j   k   l   m   n   o   p   q
41E 420 422 424 426 428 42A 42C 42E 430 432 434 436 438 43A 43C
```

Líndex d'écriture a alors comme valeur l'adresse `424h`, immédiatement devant l'origine d'adresse `426h`.

6.3.2.3 Lecture des codes de recherche

Nous sommes maintenant en mesure de récupérer les caractères du clavier, ou plus exactement leurs codes de recherche.

Principe.- Pour lire un code de recherche, on scrute l'arrivée d'un 1 pour le bit OUTB du registre de statut du contrôleur de clavier. Lorsqu'il apparaît, on peut récupérer la valeur du code de recherche dans le tampon de sortie du contrôleur de clavier.

Dans la configuration de l'IBM-PC, ceci est effectué par la routine de service associée à l'interruption IRQ1, soit INT 09h, comme nous l'avons vu. Elle est implémentée dans le BIOS ; si on veut la court-circuiter, il faut commencer par l'annihiler.

Un programme.- Le programme suivant affiche les codes de recherche des touches sur lesquelles on appuie jusqu'à ce qu'on appuie sur la touche d'échappement (de code de recherche 1). Nous écrivons ce programme en langage C pour nous faciliter la vie en ce qui concerne l'affichage. Pour désactiver IRQ1, il suffit de masquer le bit 1 du registre IMR du PIC 8259A (port d'adresse 21h).

```
#include <conio.h>
#include <stdio.h>
void main()
     {
     int statut;        /* statut du controleur de clavier */
     int scancode;      /* valeur du code de recherche recupere */

     outp(0x21, 0x02);  /* blocage de IRQ1 */
     do
        {
        do              /* attente d'un caractere disponible */
           {            /* dans le tampon de sortie */
           statut = inp(0x64);
           }
        while ((statut & 0x01) != 0x01);
        scancode = inp(0x60);  /* lecture du code */
                               /* dans le tampon de sortie */
        printf("\t%d", scancode); /* et affichage */
        }
     while (scancode != 0x01); /* arret si ESC */
     outp(0x21, 0x00);         /* remettre IRQ1 en service */
     }
```

On voit bien que lorsqu'on appuie sur une touche, cela donne lieu à au moins deux codes de recherche : le code d'appui et le code de relâchement correspondant. Il peut même y en avoir plus pour certaines touches spéciales.

6.3.3 Le contrôleur de clavier

Le schéma suivant montre l'aspect logiciel du contrôleur de clavier :

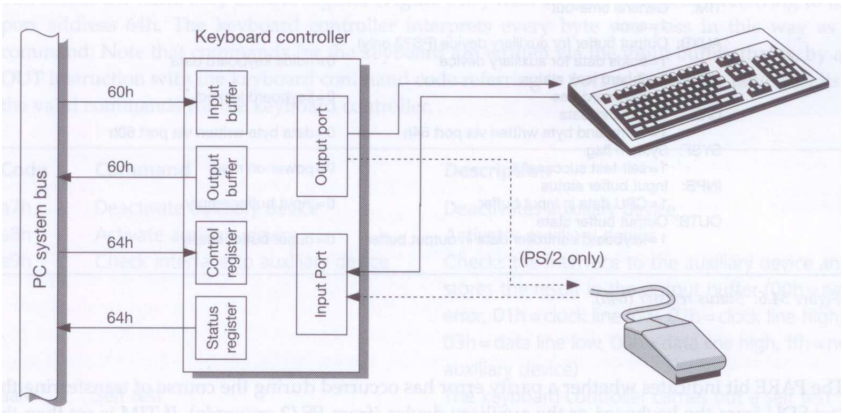


FIGURE 6.7 – Contrôleur du clavier

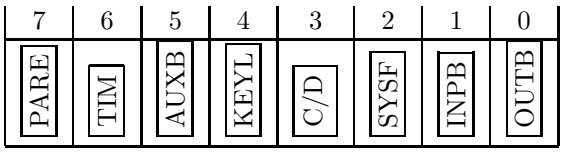
On utilise les deux ports d'adresses 60h et 64h pour accéder aux quatre registres du contrôleur de clavier, à savoir le **tampon d'entrée**, le **tampon de sortie**, le **registre de contrôle** et le **registre de statut**. Il suffit de deux ports puisque chacun de ces registres est seulement accessible soit en écriture (W), soit en lecture (R), comme l'indique le tableau suivant :

Port	Registre	Mode d'accès
60h	Tampon de sortie	R
60h	Tampon d'entrée	W
64h	Registre de contrôle	W
64h	Registre de statut	R

Le contrôleur de clavier du PX/XT est seulement capable de transférer les codes de recherche *via* le port d'adresse 60h et de produire une interruption matérielle.

6.3.4 Le registre de statut

En utilisant le registre de statut, on peut déterminer l'état du contrôleur de clavier. Ce registre peut être lu uniquement, grâce à une instruction IN à l'adresse 64h. La structure de ce registre est montrée sur la figure suivante :



— le bit 7 `PARE` (pour *PARity Error*) indique si une erreur de parité est intervenue lors du dernier transfert du clavier (ou du périphérique auxiliaire, à partir du PS/2).
  — 1 : dernier octet avec erreur de parité ;
  — 0 : dernier octet sans erreur de parité.

— le bit 6 `TIM` (pour *TIMing*) est à 1 lorsque le clavier (ou la souris) n'ont pas répondu à une requête dans la période de durée prédéfinie. Dans ce cas, on doit reformuler la requête en utilisant la commande `Resend` que nous verrons plus loin.
  — 1 : erreur ;
  — 0 : pas d'erreur.

— le bit 5 `AUXB` (pour *AUXiliary B*, c'est-à-dire la souris) dit si un octet de données de la souris est disponible dans le tampon de sortie (uniquement à partir du PS/2).
  — 1 : données pour le périphérique auxiliaire ;
  — 0 : pas de données.

— le bit 4 `KEYL` (pour *KEYboard LOck*) donne le statut de verrouillage du clavier.
  — 1 : clavier libre ;
  — 0 : clavier vérouillé.

— le bit 3 `C/D` (pour *Command/Data*) spécifie si le dernier octet écrit est un octet de commande, transféré par le microprocesseur *via* le port d'adresse 64h, ou un octet de données, écrit par le microprocesseur *via* le port d'adresse 60h.
  — 1 : octet de commande écrit via le port 64h ;
  — 0 : octet de données écrit via le port 60h.

— le bit 2 `SYSF` (pour *SYStem Flag*, soit indicateur système) spécifie des indications sur le système.
  — 1 : auto-test satisfaisant ;
  — 0 : réinitialisation.

— le bit 1 `INPB` (pour *INPut keyBoard*) spécifie le statut du tampon d'entrée, à savoir si un caractère est encore présent dans le tampon d'entrée ou si le microprocesseur peut en envoyer un autre.
  — 1 : données de la CPU dans le tampon d'entrée ;
  — 0 : tampon d'entrée vide.

— le bit 0 `OUTB` (pour *OUTput keyBoard*) donne l'état du tampon de sortie. S'il est à 1, un octet de données provenant du clavier est disponible dans le tampon de sortie :
  — 1 : données du contrôleur de clavier dans le tampon de sortie ;
  — 0 : tampon de sortie vide.

Lorsqu'un octet de données est présent, provenant soit du clavier (`OUTB` à 1), soit de la souris (`AUXB` à 1), le bit `AUXB` ou `OUTB` est mis automatiquement à 0 lorsque le microprocesseur lit cet octet. Avant de lire le tampon de sortie du contrôleur, en utilisant une instruction IN, il faut toujours vérifier, *via* `OUTB` et `AUXB`, que le contrôleur a bien transféré un octet dans le tampon d'entrée. Ceci peut en effet prendre du temps. Le contrôleur de clavier ne peut pas accepter d'autre caractère *via* son tampon d'entrée tant que le microprocesseur n'a pas récupéré le dernier caractère lu, situé dans le tampon de sortie.

6.4 Commentaire du BIOS : ISR de l'interruption 16h

Nous avons déjà décrit ce que fait l'interruption 16h, description d'ailleurs rappelée dans l'en-tête de la routine de service. Cette routine est assez simple comme le montre le code :

```
                1705
                1706    ;---- INT 16 ------------------------------------------------------
                1707    ; KEYBOARD I/O                                                    :
                1708    ;       THESE ROUTINES PROVIDE KEYBOARD SUPPORT                   :
                1709    ; INPUT                                                           :
                1710    ;       (AH)=0 READ THE NEXT ASCII CHARACTER STRUCK FROM THE KEYBOARD :
                1711    ;              RETURN THE RESULT IN (AL), SCAN CODE IN (AH)       :
                1712    ;       (AH)=1 SET THE Z FLAG TO INDICATE IF AN ASCII CHARACTER IS :
                1713    ;              AVAILABLE TO BE READ.                              :
                1714    ;              (ZF)=1 -- NO CODE AVAILABLE                        :
                1715    ;              (ZF)=0 -- CODE IS AVAILABLE                        :
                1716    ;              IF ZF = 0, THE NEXT CHARACTER IN THE BUFFER TO BE READ :
                1717    ;              IS IN AX, AND THE ENTRY REMAINS IN THE BUFFER      :
                1718    ;       (AH=2) RETURN THE CURRENT SHIFT STATUS IN AL REGISTER     :
                1719    ;              THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE :
                1720    ;              THE EQUATES FOR KB_FLAG                            :
                1721    ; OUTPUT                                                          :
                1722    ;       AS NOTED ABOVE, ONLY AND FLAGS CHANGED                    :
                1723    ;       ALL REGISTERS PRESERVED                                   :
                1724    ;-------------------------------------------------------------------:
                1725            ASSUME  CS:CODE,DS:DATA
E82E            1726            ORG     0E82EH
E82E            1727    KEYBOARD_IO     PROC    FAR
E82E FB         1728            STI                             ; INTERRUPTS BACK ON
E82F 1E         1729            PUSH    DS                      ; SAVE CURRENT DS
E830 53         1730            PUSH    BX                      ; SAVE BX TEMPORARILY
E831 E82512     1731            CALL    DDS
E834 0AE4       1732            OR      AH,AH                   ; AH=0
E836 740A       1733            JZ      K1                      ; ASCII_READ
EA38 FECC       1734            DEC     AH                      ; AH=1
EA3A 741E       1735            JZ      K2                      ; ASCII_STATUS
EA3C FECC       1736            DEC     AH                      ; AH=2
EA3E 742B       1737            JZ      K3                      ; SHIFT_STATUS
E840 EB2C       1738            JMP     SHORT   INTIO_END       ; EXIT
                1739
                1740    ;----- READ THE KEY TO FIGURE OUT WHAT TO DO
                1741
E842            1742    K1:                                     ; ASCII READ
E842 FB         1743            STI                             ; INTERRUPTS BACK ON DURING LOOP
E843 90         1744            NOP                             ; ALLOW AN INTERRUPT TO OCCUR
E844 FA         1745            CLI                             ; INTERRUPTS KACK OFF
E845 8B1E1A00   1746            MOV     BX,BUFFER_HEAD          ; GET POINTER TO HEAD OF BUFFER
E849 3B1E1C00   1747            CMP     BX,BUFFER_TAIL          ; TEST END OF BUFFER
E84D 74F3       1748            JZ      K1                      ; LOOP UNTIL SOMETHING IN BUFFER
E84F 8B07       1749            MOV     AX,[BX]                 ; GET SCAN CODE AND ASCII CODE
E851 E81D00     1750            CALL    K4                      ; MOVE POINTER TO NEXT POSITION
E854 891E1A00   1751            MOV     BUFFER_HEAD,BX          ; STORE VALUE IN VARIABLE
E858 EB14       1752            JMP     SHORT INTIO_END         ; RETURN
                1753
                1754    ;----- ASCII STATUS
                1755
E85A            1756    K2:
E85A FA         1757            CLI                             ; INTERRUPTS OFF
E85B 8B1E1A00   1758            MOV     BX,BUFFER_HEAD          ; GET HEAD POINTER
E85F 3B1E1C00   1759            CMP     BX,BUFFER_TAIL          ; IF EQUAL (Z=1) THEN NOTHING HERE
E863 8B07       1760            MOV     AX,[BX]
E865 FB         1761            STI                             ; INTERRUPTS BACK ON
E866 5B         1762            POP     BX                      ; RECOVER REGISTER
E867 1F         1763            POP     DS                      ; RECOVER SEGMENT
E868 CA0200     1764            RET     2                       ; THROW AWAY FLAGS
                1765
                1766    ;----- SHIFT SATUS
                1767
E86B            1768    K3:
E86B A01700     1769            MOV     AL,KB_FLAG              ; GET THE SHIFT STATUS FLAGS
E86E            1770    INTIO_END:
E86E 5B         1771            POP     BX                      ; RECOVER REGISTER
E86F 1F         1772            POP     DS                      ; RECOVER REGISTERS
```

```
E870 CF          1773          IRET                        ; RETURN TO CALLER
                 1774    KEYBOARD_IO    ENDP
```

Commentaires.- 1^o) Le code de cette routine est placé à l'adresse E82Eh de la mémoire vive (ligne 1726), adresse donnée pour l'interruption 16h dans le tableau des vecteurs des interruptions par le BIOS (voir par ailleurs), comme on peut le vérifier.

- 2^o) Comme toute routine de service d'interruption, elle est considérée comme un sous-programme lointain (ligne 1727), pour permettre de passer outre la limitation des 64 KiO. Les interruptions de priorité plus élevée prendront le pas (ligne 1728). Enfin le segment des données utilisé est celui de la zone de communication du BIOS (lignes 1729 à 1731, qui renvoient au sous-programme de la ligne 5488, déjà étudié).

- 3^o) Selon la valeur de AH, on renvoie à la partie concernée (lignes 1732 à 1737). Si AH est différent de 0, 1 et 2, on renvoie à la ligne 1773 qui termine le sous-programme de façon propre.

- 4^o) Dans le cas AH = 0, c'est-à-dire pour la lecture d'un caractère dans le tampon, on permet à une interruption de priorité plus élevée de prendre la main pendant un très court instant, puis on interdit toute autre interruption masquable (lignes 1743 à 1745). Si le tampon du clavier est vide (lignes 1746 à 1748), on attend qu'il y ait au moins un caractère dans celui-ci (instruction bloquante). Sinon, on place le premier caractère du tampon dans AX, comme désiré (ligne 1749), on avance l'index de lecture (lignes 1750 et 1751 ; la procédure K4 a déjà été étudiée) et on termine proprement la routine de service.

- 5^o) Dans le cas AH = 1, c'est-à-dire pour regarder si un caractère est présent, on interdit toute autre interruption masquable (lignes 1757). On place l'index de lecture dans le registre BX (ligne 1758) et on le compare à l'index d'écriture (ligne 1759), ce qui donne la valeur voulue de l'indicateur ZF. Dans les deux cas, on place le caractère pointé par l'index de lecture dans AX, même si cela n'a d'intérêt que lorsque le tampon est non vide (ligne 1760). On ne termine pas la routine de la façon habituelle (lignes 1761 à 1764), puisqu'on veut prendre en compte l'éventuel changement du registre des indicateurs.

- 6^o) Dans le cas AH = 1, c'est-à-dire pour obtenir le masque de statut du clavier, il suffit de placer celui-ci dans AX (ligne 1769).

- 7^o) La terminaison propre de la routine de service (lignes 1770 à 1774) consiste à restaurer les valeurs de BX et de DS et de renvoyer à l'appelant.

6.5 Commentaire du BIOS : ISR de l'interruption 9

Lorsqu'on appuie sur une touche ou lorsqu'on en relâche une, le contrôleur de clavier déclenche une interruption matérielle IRQ1, soit INT 9, sur le microprocesseur. Les concepteurs de l'IBM-PC traitent cette interruption au niveau du BIOS. La routine de cette interruption est chargée de récupérer le code de recherche et de le placer dans le *tampon du clavier*, situé dans l'aire des données du BIOS.

6.5.1 Principe

Le clavier communique avec la carte mère à travers l'interruption matérielle IRQ1 du 8259, qui est configurée comme INT 09. La routine de service de INT 09 est implémentée dans le BIOS. Le micro-contrôleur du clavier examine en permanence la matrice du clavier. Lorsqu'une touche est pressée, elle est identifiée et son code de recherche d'appui est envoyé par liaison série à la carte mère à travers le câble du clavier. La circuiterie de la carte mère reçoit les bits un par un comme trame de bits et en retire un octet (le code de recherche) à l'aide d'un registre à décalage qu'elle présente au port A du 8255 à l'adresse d'entrée-sortie 60h. Le micro-contrôleur déclenche alors une interruption dont la routine de service effectue les actions suivantes :

- 1. Le code de recherche est lu sur le port 60h.
- 2. Ce code de recherche est testé pour voir s'il appartient à l'une des touches majuscule (RightShift et LeftShift), Alt, Ctrl,...
  - a) Si c'est le cas, le bit approprié des octets de statut du clavier (aux emplacements mémoire `40:17h` et `40:18h`) est positionné. Le code de recherche n'est, dans ce cas, pas écrit dans le tampon du clavier.
  - b) Sinon, la routine recherche le code ASCII de la touche. S'il en existe un, celui-ci ainsi que le code de recherche sont écrits dans le tampon du clavier. Sinon elle écrit 00h (à la place du code ASCII) et le code de recherche dans le tampon du clavier.
- 3. La routine engendre un EOI pour retirer le masque de IRQ1, de façon à pouvoir accepter un nouvel appel à IRQ1 et termine par un IRET.

Nous venons de voir ce qui se passe dans le cas d'un code d'appui. Lorsqu'une touche est relâchée, le micro-contrôleur du clavier envoie de la même façon un code de relâchement. L'étape 2 est alors différente :

- 2'. La routine regarde s'il y a une différence de 80h entre le dernier code recherche et celui-ci. Si c'est le cas, ceci signifie que la touche a été appuyée et relâchée, ce qui est le comportement normal, et ce second code de recherche est ignoré. Si la touche est tenue appuyée durant plus de 0,5 secondes, elle est interprétée comme une nouvelle touche et écrite une nouvelle fois dans le tampon du clavier (comportement appelé *typematic* par IBM.

6.5.2 Mise dans le tampon

Lorsqu'on appuie sur une touche, si le tampon interne du clavier n'est pas plein, le micro-contrôleur du clavier place le code de cette touche sur le port A du 8255, d'adresse 60h, et lève une interruption IRQ1 sur le 8259, correspondant à l'interruption matérielle 9h. La routine de service de cette interruption a pour tâche de déterminer si cela conduit, soit à un changement de l'un des octets d'état du clavier, soit à un code ASCII (en fonction des octets d'état du clavier) et, si c'est le cas, de placer le code de la touche ainsi que le code ASCII correspondant dans le tampon du clavier de la mémoire vive, soit à une combinaison de touche ne correspondant pas à un code ASCII et, si c'est le cas, de placer le code étendu de la touche ainsi que 0 dans le tampon du clavier. Si c'est une combinaison de touches non permise, on ne fait rien.

Une très grande partie de la routine de service consiste à déterminer dans lequel de ces quatre cas on se trouve. Sautons cette partie pour l'instant et étudions la mise dans le tampon.

Le début du code du BIOS, en langage d'assemblage, consiste à redonner des noms aux divers ports du 8255, associés au clavier :

```
0060              30    KBD_IN       EQU     60H         ; KEYBOARD DATA IN ADDR PORT
0002              31    KBDINT       EQU     02          ; KEYBOARD INTR MASK
0060              32    KB_DATA      EQU     60H         ; KEYBOARD SCAN CODE PORT
0061              33    KB_CTL       EQU     61H         ; CONTROL BITS FOR KEYBOARD SENSE DATA
                  34
```

Curieusement, alors que le début de l'interruption 16h est clairement indiquée :

```
                  1705
                  1706    ;---- INT 16 -----------------------------------------
                  1707    ; KEYBOARD I/O                                       :
                  1708    ;      THESE ROUTINES PROVIDE KEYBOARD SUPPORT        :
```

il n'en est pas de même pour l'interruption 9h, qui commence à la ligne 1846, après la fin de l'interruption 16h et des tableaux de caractères qui suivent :

```
                  1845
                  1846    ;----- KEYBOARD INTERRUPT ROUTINE
                  1847
E987              1848            ORG     0E987H
E987              1849    KB_INT  PROC    FAR
E987 FB           1850            STI                     ; ALLOW FURTHER INTERRUPTS
E988 50           1851            PUSH    AX
E989 53           1852            PUSH    BX
E98A 51           1853            PUSH    CX
E98B 52           1854            PUSH    DX
E98C 56           1855            PUSH    SI
E98D 57           1856            PUSH    DI
E98E 1E           1857            PUSH    DS
E98F 06           1858            PUSH    ES
E990 FC           1859            CLD                     ; FORWARD DIRECTION
E991 E8C510       1860            CALL    DDS
E994 E460         1861            IN      AL,KB_DATA      ; READ IN THE CHARACTER
E996 50           1862            PUSH    AX              ; SAVE IT
E997 E461         1863            IN      AL,KB_CTL       ; GET THE CONTROL PORT
E999 8AE0         1864            MOV     AH,AL           ; SAVE VALUE
E99B 0C80         1865            OR      AL,80H          ; RESET BIT FOR KEYBOARD
E99D E661         1866            OUT     KB_CTL,AL
E99F 86E0         1867            XCHG    AH,AL           ; GET BACK ORIGINAL CONTROL
E9A1 E661         1868            OUT     KB_CTL,AL       ; KB HAS BEEN RESET
E9A3 58           1869            POP     AX              ; RECOVER SCAN CODE
E9A4 8AE0         1870            MOV     AH,AL           ; SAVE SCAN CODE IN AH ALSO
                  1871
                  1872    ;----- TEST FOR OVERRUN SCAN CODE FOR KEYBOARD
                  1873
```

Commentaires.- 1^o) Le code de cette routine est placé à l'adresse E967h de la mémoire vive (ligne 1848), adresse donnée comme début de la routine de service de l'interruption 9h dans le tableau des vecteurs des interruptions par le BIOS (voir chapitre 10).

- 2^o) Comme toute routine de service d'interruption, elle est considérée comme un sous-programme lointain (ligne 1849), pour permettre de passer outre la limitation des 64 KiO. Les interruptions de priorité plus élevée prendront le pas (ligne 1850). Tous les registres qui seront utilisés dans cette routine sont sauvegardés sur la pile (lignes 1851 à 1858). Lors des recherches dans les tables de caractères, on partira de la fin (ligne 1859). Enfin le segment des données utilisé dans cette routine de service est celui de la zone de communication du BIOS (ligne 1860, qui renvoie à la ligne 5488), commençant à l'adresse 400h :

```
                5487
FA59            5488   DDS     PROC    NEAR
FA59 50         5489           PUSH    AX              ; SAVE AX
FA5A B84000     5490           MOV     AX,DATA
FA5D 8ED8       5491           MOV     DS,AX           ; SET SEGMENT
FA5F 58         5492           POP     AX              ; RESTORE AX
FA60 C3         5493           RET
                5494   DDS     ENDP
                5495
```

- 3^o) Le code de la touche est lu là où il a été placé par le micro-contrôleur du clavier, c'est-à-dire sur le port A du 8255 (ligne 1861), et est placé dans le registre AL.

On remarquera qu'on utilise une deuxième étiquette en langage symbolique pour ce port, qui se veut plus explicite.

- 4^o) Les lignes 1862 à 1869 permettent de libérer l'interruption clavier, par l'intermédiaire du bit 7 du port 61h. Cela indique que le caractère a bien été reçu.

- 5^o) Le code de la touche, qui se trouve maintenant dans AL, est également placé dans AH (ligne 1870). Cela permet de conserver ce code et de triturer ce qui a été obtenu pour le transformer en code ASCII correspondant ou en 00h (dans le cas d'un code étendu).

Le code ASCII (ou 00h dans le cas d'un code étendu) est obtenu à partir du contenu de AH, c'est-à-dire du code de la touche, et placé dans AL, d'une façon que nous détaillerons ensuite. Ainsi AX finit par contenir ce qu'il faut placer dans le tampon du clavier. On place donc son contenu dans le tampon :

```
                2214
                2215   ;----- PUT CHARACTER INTO BUFFER
                2216
EBD5            2217   K57:                            ; BUFFER-FILL
EBD5 3CFF       2218           CMP     AL,-1            ; IS THIS AN IGNORE CHAR
EBD7 741F       2219           JE      K59              ; YES, DO NOTHING WITH IT
EBD9 80FCFF     2220           CMP     AH,-1            ; LOOK FOR -1 PSEUDO SCAN
EBDC 741A       2221           JE      K59              ; NEAR_INTERRUPT_RETURN
                2222
                2223   ;----- HANDLE THE CAPS LOCK PROBLEM
                2224
EBDE            2225   K58:                            ; BUFFER-FILL-NOTEST
EBDE F606170040 2226           TEST    KB_FLAG,CAPS_STATE ; ARE WE IN CAPS LOCK STATE
EBE3 7420       2227           JZ      K61              ; SKIP IF NOT
                2228
                2229   ;----- IN CAPS LOCK STATE
                2230
EBE5 F606170003 2231           TEST    KB_FLAG,LEFT_SHIFT+RIGH_SHIFT ; TEST FOR SHIFT STATE
EBEA 74DF       2232           JZ      K60              ; IF NOT SHIFT, CONVERT LOWER TO UPPER
                2233
                2234   ;----- CONVERT ANY UPPER CASE TO LOWER CASE
                2235
EBEC 3C41       2236           CMP     AL,'A'           ; FIND OUT IF ALPHABETIC
EBEE 7215       2237           JB      K61              ; NOT_CAPS_STATE
EBF0 3C5A       2238           CMP     AL,'Z'
EBF2 7711       2239           JA      K61              ; NOT_CAPS_STATE
EBF4 D420       2240           ADD     AL,'a'-'A'       ; CONVERT TO LOWER CASE
EBF6 EB0D       2241           JMP     SHORT K61        ; NOT_CAPS_STATE
EBF8            2242   K59:                            ; HEAR-INTERRUPT-RETURN
EBF8 E95EFE     2243           JMP     K26              ; INTERRUPT_RETURN
```

```
                2244
                2245    ;----- CONVERT ANY LOWER CASE TO UPPER CASE
                2246
EBFB            2247    K60:                            ; LOWER-TO-UPPER
EBFB 3C61       2248            CMP     AL,'a'          ; FIND OUT IF ALPHABETIC
EBFD 7206       2249            JB      K61             ; NOT_CAPS_STATE
EBFF 3C7A       2250            CMP     AL,'z'
EC01 7702       2251            JA      K61             ; NOT_CAPS_STATE
EC03 2C20       2252            SUB     AL,'a'-'A'      ; CONVERT TO UPPER CASE
EC05            2253    K61:                            ; NOT_CAPS_STATE
EC05 8B1E1C00   2254            MOV     BX,BUFFER_TAIL  ; GET THE END POINTER TO THE BUFFER
EC09 8BF3       2255            MOV     SI,BX           ; SAVE THE VALUE
EC0B E863FC     2256            CALL    K4              ; ADVANCE THE TAIL
EC0E 3B1E1A00   2257            CMP     BX,BUFFER_HEAD  ; HAS THE BUFFER WRAPPED AROUND
EC12 7413       2258            JE      K62             ; BUFFER_FULL_BEEP
EC14 89D4       2259            MOV     [SI],AX         ; STORE THE VALUE
EC16 891E1C00   2260            MOV     BUFFER_TAIL,BX  ; MOVE THE POINTER UP
EC1A E93CFE     2261            JMP     K26             ; INTERRUPT_RETURN
                2262
```

Commentaires.- 6^o) S'il s'agit d'une combinaison de touches non prévue (lignes 2218 à 2221), on indique ce fait en mettant -1 dans AL ou dans AH. Dans ce cas, on ignore tout simplement le caractère, on va donc directement à la fin de la routine de service (K59 qui, ligne 2243, renvoie à K26) que nous étudierons ci-dessous.

- 7^o) Si l'état 'CapsLock' est actif (lignes 2226 et 2227) ou si l'une des touches majuscules est appuyée (lignes 2231 et 2232), on convertit les lettres majuscules en minuscule (lignes 2236 à 2241) et les lettres minuscules en majuscule (lignes 2248 à 2252).

- 8^o) Après ces transformations éventuelles, on place l'index d'écriture dans le registre BX, on le sauvegarde dans le registre SI pour pouvoir placer le contenu de AX dans le tampon et on appelle le sous-programme K4 pour mettre à jour cet index (lignes 2254 à 2256).

- 9^o) Le sous-programme K4 incrémente deux fois BX (puisque l'index d'écriture occupe deux octets). Si on est arrivé à la fin de la zone du tampon, on passe au début de celle-ci puisqu'il s'agit d'un tampon circulaire.

```
                1775
                1776    ;----- INCREMENT A BUFFER POINTER
                1777
E871            1778    K4      PROC    NEAR
E871 43         1779            INC     BX              ; MOVE TO NEXT WORD IN LIST
E872 43         1780            INC     BX
E873 3B1E8200   1781            CMP     BX,BUFFER_END   ; AT END OF BUFFER?
E877 7504       1782            JNE     K5              ; NO. CONTINUE
E879 8B1E8000   1783            MOV     BX,BUFFER_START ; YES, RESET TO BUFFER BEGINNING
E87D            1784    K5:
E87D C3         1785            RET
                1786    K4      ENDP
                1787
```

- 10^o) On regarde ensuite si le tampon du clavier est plein. S'il en est ainsi, on émet un bip en faisant appel à K62 (lignes 2257 et 2258), étudié dans la section suivante.

- 11^o) Sinon on place le contenu de AX à l'emplacement voulu du tampon du clavier (ligne 2259), on met à jour l'index d'écriture (ligne 2260) et on termine proprement la routine de service de l'interruption par appel à K26 (ligne 2261), commençant à la ligne 1972 :

```
EA59              1972    K26:                            ; INTERRUPT_RETURN
EA59 FA           1973            CLI                     ; TURN OFF INTERRUPT COMMAND
EA5A B020         1974            MOV     AL,EOI          ; END OF INTERRUPT COMMAND
EA5C E620         1975            OUT     020H,AL         ; SEND COMMAND TO INT CONTROL PORT
EA5E              1976    K27:                            ; INTERRUPT-RETURN-NO-EOI
EA5E 07           1977            POP     ES
EA5F 1F           1978            POP     DS
EA60 5F           1979            POP     DI
EA61 5E           1980            POP     SI
EA62 5A           1981            POP     DX
EA63 59           1982            POP     CX
EA64 5B           1983            POP     BX
EA65 58           1984            POP     AX              ; RESTORE STATE
EA66 CF           1985            IRET                    ; RETURN, INTERRUPTS BACK ON
                  1986                                    ;  WITH FLAG CHANGE
```

La terminaison propre de la routine de service (lignes 1972 à 1986), consiste :

— à ne pas permettre pour un court instant à une interruption (masquable), même de priorité plus haute, de prendre la main,

— à envoyer une commande de fin d'interruption au port 20h, premier port du contrôleur d'interruptions programmable 8259A de l'IBM-PC.

Remarquons que ce premier port du PIC 8259A apparaît comme un nombre magique alors que le symbole INTA00, défini à la ligne 19, aurait pu être utilisé.

La commande EOI est définie à la ligne 21 en tant que 20h, c'est-à-dire que dans le mot OCW2 :

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	R	SL	EOI	0	0	L2	L1	L0

R et SL sont nuls et EOI est égal à 1, on a donc un EOI non spécifique, c'est-à-dire que le bit IS est mis à 1 (dans le mode *fully nested* du PC), permettant la prise en compte d'une interruption de niveau inférieur,

— et enfin à restaurer les valeurs des registres aynt été sauvegardées telles qu'elles étaient avant l'entrée dans la routine de service et à sortir de celle-ci.

6.5.3 Cas où le tampon est plein

Lorsque le tampon interne du clavier est plein, celui-ci envoie FFh au lieu d'un code de touche matérielle. Le rôle de la routine de service consiste alors à émettre un bip. C'est le premier cas qu'elle prend en compte.

```
                1871
                1872    ;----- TEST FOR OVERRUN SCAN CODE FOR KEYBOARD
                1873
E9A6 3CFF       1874            CMP     AL,0FFh         ; IS THIS AN OVERRUN CHAR
E9A8 7503       1875            JNZ     K16             ; NO, TEST FOR SHIFT KEY
E9AA E97A02     1876            JMP     K62             ; BUFFER_FULL_BEEP
                1877
```

La partie de la routine qui fait biper commence, quant à elle, à la ligne 2275 :

```
                2274
                2275    ;-----BUFFER IS FULL, SOUND THE BEEPER
                2276
EC27            2277    K62:                            ; BUFFER-FULL-BEEP
EC27 B020       2278            MOV     AL,EOI          ; END OF INTERRUPT COMMAND
EC29 E620       2279            OUT     20H,AL          ; SEND COMMAND TO INT CONTROL PORT
EC2B B88000     2280            MOV     BX,080H         ; NUMBER OF CYCLES FOR 1/12 SECOND TONE
EC2E E461       2281            IN      AL,KB_CTL       ; GET CONTROL INFORMATION
EC30 50         2282            PUSH    AX              ; SAVE
EC31            2283    K65:                            ; BEEP-CYCLE
EC31 24FC       2284            AND     AL,0FCH         ; TURN OFF TIMER GATE AND SPEAKER DATA
EC33 E661       2285            OUT     KB_CTL,AL       ; OUTPUT THE CONTROL
EC35 B94800     2286            MOV     CX,48H          ; HALF CYCLE TIME FOR TONE
EC38            2287    K66:
EC38 E2FE       2288            LOOP    K66             ; SPEAKER OFF
EC3A 0C02       2289            OR      AL,2            ; TURN ON SPEAKER BIT
EC3C E661       2290            OUT     KB_CTL,AL       ; OUTPUT THE CONTROL
EC3E B94800     2291            MOV     CX,48H          ; SET UP COUNT
EC41            2292    K67:
EC41 E2F2       2293            LOOP    K67             ; ANOTHER HALF CYCLE
EC43 4B         2294            DEC     BX              ; TOTAL TIME COUNT
EC44 75EB       2295            JNZ     K65             ; DO ANOTHER CYCLE
EC46 58         2296            POP     AX              ; RECOVER CONTROL
EC47 E661       2297            OUT     KB_CTL,AL       ; OUTPUT THE CONTROL
EC49 E912FE     2298            JMP     K27
                2299
```

6.5.4 Cas d'une touche de décalage

Nous venons de voir que la routine de service regarde d'abord si le « code de recherche » envoyé est FFh, qui n'est pas un code de touche proprement dit mais un code indiquant que le tampon interne du clavier est plein. Lorsqu'il ne s'agit pas de ce pseudo-code de recherche, la routine commence par regarder s'il s'agit du code correspondant à une **touche de décalage** (*shift key*), c'est-à-dire d'insertion, de blocage des majuscules, de verrouillage numérique, de défilement, de touche alternative, de contrôle, de majuscule gauche ou droite. Dans ce cas-là, aucune donnée n'est placée dans le tampon du clavier mais un bit est changé dans un ou deux octets d'état du clavier, conformément à la touche et aux octets d'état du clavier.

Une table des codes clavier des touches de décalage et des masques correspondant est définie à partir de la ligne 1788 :

```
                1787
                1788    ;----- TABLE OF SHIFT KEYS AND MASK VALUES
                1789
E87E            1790    K6      LABEL   BYTE
E87E 52         1791            DB      INS_KEY          ; INSERT KEY
E87F 3A         1792            DB      CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
E880 45
E881 46
E882 38
E883 1D
E884 2A         1793            DB      LEFT_KEY,RIGHT_KEY
E885 36
  0008          1794    K6L     EQU     $-K6
                1795
                1796    ;----- SHIFT_MASK_TABLE
                1797
E886            1798    K7      LABEL   BYTE
E886 80         1799            DB      INS_SHIFT        ; INSERT MODE SHIFT
E887 40         1800            DB      CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
E888 20
E889 10
E88A 08
E88B 04
E88C 02         1801            DB      LEFT_SHIFT,RIGHT_SHIFT
E88D 01
                1802
```

Commentaires.- 1^o) Les lignes 1788 à 1795 sont occupées par la table des codes des touches de décalage et par la longueur de celle-ci.

Rappelons que symboles pour les codes des touches de décalage (NUM_KEY, SCROLL_KEY, ALT_KEY, CTL_KEY, CAPS_KEY, LEFT_KEY, RIGHT_KEY, INS_KEY) ont été définis aux lignes 122 à 130 de la zone de communication du BIOS.

- 2^o) Les lignes 1796 à 1802 sont occupées par la table des masques correspondants à ces touches tels qu'ils apparaissent dans les octets de statut du clavier.

Rappelons que les masques INS_SHIFT, CAPS_SHIFT, NUM_SHIFT, SCROLL_SHIFT ainsi que ALT_SHIFT, CTL_SHIFT, LEFT_SHIFT et RIGHT_SHIFT ont été définis aux lignes 101 à 111 de la zone de communication du BIOS.

Nous avons vu que lorsqu'il ne s'agit pas du pseudo-code de tampon plein, la routine de service renvoie à K16, qui commence ligne 1878 :

```
                1877
                1878    ;----- TEST FOR SHIFT KEYS
                1879
E9AD            1880    K16:                            ; TEST_SHIFT
E9AD 247F       1881            AND     AL,07FH         ; TURN OFF THE BREAK BIT
E9AF 0E         1882            PUSH    CS
E9B0 07         1883            POP     ES              ; ESTABLISH ADDRESS OF SHIFT TABLE
E9B1 BF7EE8     1884            MOV     DI,OFFSET K6    ; SHIFT KEY TABLE
E9B4 B9D800     1885            MOV     CX,K61          ; LENGTH
E9B7 F2         1886            REPNE   SCASB           ; LOOK THROUGH THE TABLE FOR A MATCH
E9B8 AE
E9B9 8AC4       1887            MOV     AL,AH           ; RECOVER SCAN CODE
E9BB 7403       1888            JE      K17             ; JUMP IF MATCH FOUND
E9BD E98500     1889            JMP     K25             ; IF NO MATCH, THEN SHIFT NOT FOUND
                1890
                1891    ;----- SHIFT KEY FOUND
                1892
E9C0 81EF7FE8   1893    K17:    SUB     DI,OFFSET K6+1  ; ADJUST PTR TO SCAN CODE MTCH
E9C4 2E8AA586E8 1894            MOV     AH,CS:K7[DI]    ; GET MASK INTO AH
E9C9 A880       1895            TEST    AL,80H          ; TEST FOR BREAK KEY
E9CB 7403       1896            JMP     K23             ; BREAK_SHIFT_FOUND
                1897
                1898    ;----- SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE
                1899
E9CD 80DF10     1900            CMP     AH,SCROLL_SHIFT
E9D0 7307       1901            JAE     K18             ; IF SCROLL SHIFT OR ABOVE, TOGGLE KEY
                1902
                1903    ;----- PLAIN SHIFT KEY, SET SHIFT ON
                1904
E9D2 D8261700   1905            OR      KB_FLAG,AH      ; TURN ON SHIFT BIT
E9D6 E98000     1906            JMP     K26             ; INTERRUPT_RETURN
                1907
                1908    ;----- TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
                1909
E9D9            1910    K18:                            ; SHIFT-TOGGLE
E9D9 F606170004 1911            TEST    KB_FLAG, CTL_SHIFT ; CHECK CTL SHIFT STATE
E9DE 7565       1912            JNZ     K25             ; JUMP IF CTL STATE
E9E0 3C52       1913            CMP     AL,INS_KEY      ; CHECK FOR INSERT KEY
E9E2 7522       1914            JNZ     K22             : JUMP IF NOT INSERT KEY
E9E4 F606170008 1915            TEST    KB_FLAG, ALT_SHIFT ; CHECK FOR ALTERNATE SHIFT
E9E9 755A       1916            JNZ     K25             ; JUMP IF ALTERNATE SHIFT
E9EB F606170020 1917    K19:    TEST    KB_FLAG, NUM_STATE ; CHECK FOR BASE STATE
E9F0 750D       1918            JNZ     K21             ; JUMP IF NUM LOCK IS ON
E9F2 F606170003 1919            TEST    KB_FLAG, LEFT_SHIFT+RIGHT_SHIFT
E9F7 740D       1920            JZ      K22             ; JUMP IF BASE STATE
                1921
E9F9            1922    K20:                            ; NUMERIC ZERO, NOT INSERT KEY
E9F9 B83052     1923            MOV     AX, 5230H       ; PUT OUT AN ASCII ZERO
E9FC E90601     1924            JMP     K57             ; BUFFER_FILL
E9FF            1925    K21:                            ; MIGHT NUMERIC, NOT INSERT
E9FF F606170003 1926            TEST    KB_FLAG, LEFT_SHIFT+RIGHT_SHIFT
EA04 74F3       1927            JZ      K20             ; JUMP NUMERIC, NOT INSERT
                1928
EA06            1929    K22:                            ; SHIFT TOGGLE KEY IT; PROCESS IT
EA06 84261800   1930            TEST    AH,KB_FLAG_1    ; IS KEY ALREADY DEPRESSED
EA0A 7540       1931            JNZ     K26             ; JUMP IF KEY ALREADY DEPRESSED
EA0C 08261800   1932            OR      KB_FLAG_1,AH    ; INDICATE THAT THE KEY IS DEPRESSED
EA10 30261700   1933            XOR     KB_FLAG,AH      ; TOGGLE THE SHIFT STATE
EA14 3C52       1934            CMP     AL,INS_KEY      ; TEST FOR 1ST MAKE OF INSERT KEY
EA16 7541       1935            JNE     K26             ; JUMP IF NOT INSERT KEY
EA18 B80052     1936            MOV     AX,INS_KEY*256  ; SET SCAN CODE INTO AH, 0 INTO AL
EA1B E98701     1937            JMP     K57             ; PUT INTO OUTPUT BUFFER
                1938
                1939    ;----- BREAK SHIFT FOUND
                1940
EA1E            1941    K23:                            ; BREAK-SHIFT-FOUND
EA1E 80FC10     1942            CMP     AH,SCROLL_SHIFT ; IS THIS A TOGGLE KEY
EA21 731A       1943            JAE     K24             ; YES, HANDLE BREAK TOGGLE
EA23 F6D4       1944            NOT     AH              ; INVERT MASK
EA25 20261700   1945            AND     KB_FLAG,AH      ; TURN OFF SHIFT BIT
EA29 3CB8       1946            CMP     AL,ALT_KEY+80H  ; IS THIS ALTERNATE SHIFT RELEASE
```

```
EA2B 752C          1947            JNE     K26             ; INTERRUPT_RETURN
                   1948
                   1949    ;----- ALTERNATE SHIFT KEY RELEASE, GET THE VALUE INTO BUFFER
                   1950
EA2D A01900        1951            MOV     AL,ALT_INPUT
EA30 B400          1952            MOV     AH,0            ; SCAN CODE OF 0
EA32 88261900      1953            MOV     ALT_INPUT,AH    ; ZERO OUT THE FIELD
EA36 3C00          1954            CMP     AL,0            ; HAS THE INPUT=0
EA38 741F          1955            JE      K26             ; INTERRUPT_RETURN
EA3A E9A101        1956            JMP     K58             ; IT WASN'T, SO PUT IN BUFFER
EA3D               1957    K24:                            ; BREAK-TOGGLE
EA3D F604          1958            NOT     AH              ; INVERT MASK
EA3F 20261800      1959            AND     KB_FLAG_1,AH    ; INDICATE NO LONGER DEPRESSED
EA43 EB14          1960            JMP     SHORT K26       ; INTERRUPT_RETURN
                   1961
```

Commentaires.- 3^o) On positionne, dans AL, à 0 le bit éventuel de code de relâchement (ligne 1881), car il ne nous intéresse pas dans une première étape : on se contente de regarder s'il s'agit du code de l'une des touches de décalage ; on pourra toujours retrouver le code originel grâce à AH. On fait ensuite pointer le segment supplémentaire sur la zone de communication du BIOS (lignes 1882 et 1883), on place dans DI le début de la table des codes de recherche des touches de décalage (ligne 1884) et dans CX la longueur de celle-ci (ligne 1885). On recherche si le code reçu apparaît dans cette table (ligne 1886). Si ce n'est pas le cas, on passe à un autre type de codes de recherche (ligne 1889), ce que nous étudierons dans la section suivante. Si c'est le cas (lignes 1887 et 1888), on replace le code de recherche (d'appui ou de relâchement) dans AL et on va agir en fonction de la nature de la touche de décalage.

- 4^o) On place son numéro d'emplacement dans DI (ligne 1893), ce qui permet d'en placer le masque dans AH (ligne 1894). On regarde s'il s'agit d'un code de recherche d'appui ou de relâchement, puisque le traitement est différent suivant le cas (lignes 1895 et 1896).

- 5^o) Il faut distinguer deux types de touches de décalage : les **touches de verrouillage** ('CapsLock', 'NumLock' et 'ScrollLock') conduisent à un état (à deux positions) lorsqu'on appuie dessus et cet état change lorsqu'on appuie à nouveau dessus ; pour les autres touches de décalage, l'état actif dure entre le moment où on a appuyé sur la touche et le moment où on la relâche. Le relâchement a donc une grande importance pour le deuxième type de touches (c'est d'ailleurs pour ces seules touches qu'il a un intérêt) alors qu'il n'en a aucun pour le premier.

S'il ne s'agit pas d'une touche de verrouillage (lignes 1900 et 1901), il suffit de placer le masque de la touche dans le premier octet de statut (ligne 1905) et de terminer proprement la routine de service (ligne 1906).

- 6^o) S'il s'agit d'une touche de verrouillage, si on a appuyé sur CTL auparavant (ligne 1911), on passe au cas des combinaisons de touches, qui sera étudié dans la section suivante (ligne 1912).

- 7^o) S'il s'agit de la touche d'insertion (lignes 1913 et 1914), on étudie les différents cas possibles :

— Si la touche 'Alt' est enfoncée, il ne s'agit pas d'une combinaison valide, on poursuit donc la recherche, telle que nous la verrons dans la section suivante (lignes 1915 et 1916).

— Si le verrouillage numérique est positionné (lignes 1917 et 1918) sans qu'on ait appuyé sur la touche majuscule gauche ou droite (lignes 1926 et 1927) ou qu'il n'est pas positionné mais qu'on a appuyé sur la touche majuscule gauche ou droite (lignes 1919 et 1920), c'est qu'il s'agit de '0', que l'on place donc dans le tampon du clavier (lignes 1922 à 1924).

- 8^o) S'il s'agit d'une touche de verrouillage et qu'on avait déjà appuyé dessus, on termine proprement la routine de service (lignes 1930 et 1031). Sinon on indique qu'on a appuyé sur cette touche dans le deuxième octet de statut du clavier (ligne 1932) et on bascule l'état de cette touche dans le premier octet de statut du clavier (ligne 1933). S'il ne s'agit pas de la première fois que l'on appuie sur la touche 'Insert', on place le code de recherche dans AH, 0 dans AL et on place le tout dans le tampon (lignes 1934 à 1937).

- 9^o) Dans le cas d'un code de relâchement, s'il ne s'agit pas d'une touche de verrouillage (lignes 1942 et 1943), on inverse le masque (ligne 1944) de façon à prendre en compte le basculement dans le premier octet de statut du clavier (ligne 1945).

- 10^o) S'il ne s'agit pas du code de relâchement de la touche 'Alt', on termine proprement la routine de service (lignes 1946 et 1947).

- 11^o) Rappelons que `ALT_INPUT` est un emplacement d'un octet de la zone de communication du BIOS, défini à la ligne 112, servant à former un nombre de 1 à 255, représentant un code ASCII. Celui-ci est obtenu en appuyant sur la touche 'Alt' et sur les touches de chiffres de '0' à '9' du clavier numérique de façon à spécifier ce nombre en numération décimale. Lorsqu'on relâche le touche 'Alt', le code ASCII désiré se trouve donc (nous verrons comment plus loin) dans `ALT_INPUT` ; on le place donc dans AL (ligne 1951). On réinitialise `ALT_INPUT` à 0 pour la prochaine fois (lignes 1952 et 1953). Si le code ASCII est 0 (ligne 1954), ce qu'il n'est pas permis d'obtenir de cette façon, on termine proprement la routine de service (ligne 1955). Sinon on place le caractère (avec code de recherche égal à 0) dans le tampon (ligne 1956).

- 12^o) Dans le cas où on relâche une touche de verrouillage, on indique qu'elle n'est plus enfoncée dans le second octet de statut du clavier (lignes 1958 et 1959) et on termine proprement la routine de service (ligne 1960).

6.5.5 Cas de la touche de pause

Après avoir recherché si le code recherche envoyé est le pseudo-code de recherche FFh ou le code de recherche d'une touche de basculement, la routine de service se débarasse du cas des codes de relâchement, qui n'ont d'intérêt que pour certaines touches de décalage, puis passe au cas de l'appui sur 'Pause', c'est-à-dire de l'appui simultané sur la touche 'Ctrl' et sur la touche 'NumLock'.

```
                    1962    ;----- TEST FOR HOLD STATE
                    1963
EA45                1964    K25:                                    ; NO-SHIFT-FOUND
EA45 3C80           1965            CMP     AL,80H                  ; TEST FOR BREAK KEY
EA47 7310           1966            JAE     K26                     ; NOTHING FOR BREAK CHARS FROM HERE ON
EA49 F606180008     1967            TEST    KB_FLAG_1,HOLD_STATE    ; ARE WE IN HOLD STATE
EA4E 7417           1968            JZ      K28                     ; BRANCH AROUND TEST IF NOT
EA50 3C45           1969            CMP     AL,NUM_KEY
EA52 7405           1970            JE      K26                     ; CAN'T END HOLD ON NUM_LOCK
EA54 B0261800F7     1971            AND     KB_FLAG_1,NOT HOLD_STATE ; TURN OFF THE HOLD STATE BIT
```

Commentaires.- 1^o) S'il s'agit d'un code de relâchement (ligne 1965), on passe directement à la terminaison propre de la routine de service (ligne 1966), déjà étudiée.

- 2^o) Si on a appuyé sur 'Pause', c'est-à-dire sur 'Ctrl' (ligne 1967) et sur 'Numlock' (ligne 1969), on bascule éventuellement le fait d'avoir appuyé sur la touche 'Ctrl' (ligne 1971) et on termine proprement la routine de service (lignes 1970 et à partir de la ligne 1972, déjà étudiées). Sinon on continuera notre recherche (ligne 1968).

6.5.6 Cas des combinaisons de touches avec 'Alt'

Après le pseudo-code de recherche FFh, les codes des touches de basculement, les codes de relâchement et l'appui sur 'Pause', on passe au cas où l'on a appuyé sur la touche 'Alt'.

L'appui sur 'Alt' a trois significations :

— L'appui simultané sur les touches 'Ctrl', 'Alt' et 'Del', spécifie que l'on veut un redémarrage du système.

— L'appui sur 'Alt' et, sans relâcher cette touche, sur jusqu'à trois touches de chiffres du clavier numérique (touches de codes 71 à 73, 75 à 77 et 79 à 82) spécifie un code ASCII compris entre 0 et 255.

— L'appui simultané sur 'Alt' et sur l'une des touches 2 à 13, 16 à 25, 30 à 38, 44 à 50 et 59 à 68 donne lieu à un code étendu, c'est-à-dire à 0 pour AL et un code qui n'est pas nécessairement celui de la touche pour AH. En particulier l'appui sur 'Alt' et sur l'une des dix touches de fonction donne les codes étendus 104 à 113, que l'on retrouve dans la table suivante, comme on peut le vérifier d'après ce que nous avons dit des codes étendus :

```
                    1834    ;----- ALT TABLE SCAN
E95F                1835    K13     LABEL   BYTE
E95F 68             1836                    DB      104,105,106,107,108
E960 69
E961 6A
E962 6B
E963 6C
E964 6D             1837                    DB      109,110,111,112,113
E965 6E
E966 6F
E967 70
E968 71
```

Nous avons vu que si nous ne sommes pas dans le cas de la touche pause, on est renvoyé à K28, qui commence à la ligne 1987 :

```
                    1987
                    1988    ;----- NOT IN HOLD STATE, TEST FOR SPECIAL CHARS
                    1989
EA67                1990    K28:                            ; NO-HOLD-STATE
EA67 F606170008     1991            TEST    KB_FLAG,ALT_SHIFT ; ARE WE IN ALTERNATE SHIFT ALSO
EA6C 7503           1992            JNZ     K29             ; JUMP IF ALTERNATE SHIFT
EA6E E99100         1993            JMP     K38             ; JUMP IF NOT ALTERNATE
                    1994
                    1995    ;----- TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)
                    1996
EA71                1997    K29:                            ; TEST-RESET
EA71 F606170004     1998            TEST    KB_FLAG,CTL_SHIFT ; ARE WE IN CONTROL SHIFT ALSO
EA76 7433           1999            JZ      K31             ; NO_RESET
EA78 3C53           2000            CMP     AL,DEL_KEY      ; SHIFT STATE IS THERE, TEST KEY
EA7A 752F           2001            JNE     K31             ; NO_RESET
                    2002
                    2003    ;----- CTL-ALT-DEL HAS BEEN FOUND, DO I/O CLEANUP
                    2004
EA7C C706720034 12  2005            MOV     RESET_FLAG, 1234H ; SET FLAG FOR RESET FUNCTION
EA82 EA5BE000F0     2006            JMP     RESET           ; JUMP TO POWER ON DIAGNOSTICS
                    2007
                    2008    ;----- ALT-INPUT-TABLE
EA87                2009    K30     LABEL   BYTE
EA87 52             2010            DB      82,79,80,81,75,76,77
EA88 4F
EA89 50
EA8A 51
EA8B 4B
EA8C 4C
EA8D 4D
EA8E 47             2011            DB      71,72,73         ; 10 NUMBERS ON KEYPAD
EA8F 48
EA90 49
                    2012    ;----- SUPER-SHIFT-TABLE
EA91 10             2013            DB      16,17,18,19,20,21,22,23 ; A-Z TYPEWRITER CHARS
```

```
EA92 11
EA93 12
EA94 13
EA95 14
EA96 15
EA97 16
EA98 15
EA99 18          2014          DB       24,25,30,31,32,33,34,35
EA9A 19
EA9B 1E
EA9C 1F
EA9D 20
EA9E 21
EA9F 22
EAA0 23
EAA1 24          2015          DB       36,37,38,44,45,46,47,48
EAA2 25
EAA3 26
EAA4 2C
EAA5 2D
EAA6 2E
EAA7 2F
EAA8 30
EAA9 31          2016          DB       49,50
EAAA 32
                 2017
                 2018    ;----- IN ALTERNATE SHIFT, RESET NOT FOUND
                 2019
EAAB             2020    K31:                              ; NO-RESET
EAAB 3C39        2021          CMP      AL,57              ; TEST FOR SPACE KEY
EAAD 7505        2022          JNE      K32                ; NOT THERE
EAAF B020        2023          MOV      AL,' '             ; SET SPACE CHAR
EAB1 E92101      2024          JMP      K57                ; BUFFER_FILL
                 2025
                 2026    ;----- LOOK FOR KEY PAD ENTRY
                 2027
EAB4             2028    K32:                              ; ALT_KEY-PAD
EAB4 BF87EA      2029          MOV      DI,OFFSET K30      ; ALT-INPUT-TABLE
EAB7 B90A00      2030          MOV      CX,10              ; LOOK FOR ENTRY USING KEYPAD
EABA F2          2031          REPNE    SCASB              ; LOOK FOR MATCH
EABB AE
EABC 7512        2032          JNE      K33                ; NO_ALT_KEYPAD
EABE 81EF88EA    2033          SUB      DI,OFFSET K30+1    ; DI NOW HAS ENTRY VALUE
EAC2 A01900      2034          MOV      AL,ALT_INPUT       ; GET THE CURRENT BYTE
EAC5 B40A        2035          MOV      AH,10              ; MULTIPLY BY 10
EAC7 F6E4        2036          MUL      AH
EAC9 03C7        2037          ADD      AX,DI              ; ADD IN THE LATEST ENTRY
EACB A2190D      2038          MOV      ALT_INPUT,AL       ; STORE IT AWAY
EACE EB89        2039          JMP      K26                ; THROW AWAY THAT KEYSTROKE
                 2040
                 2041    ;----- LOOK FOR SUPERSHIFT ENTRY
                 2042
EAD0             2043    K33:                              ; NO-ALT-KEYPAD
EAD0 C606190000  2044          MOV      ALT_INPUT,0        ; ZERO ANY PREVIOUS ENTRY INTO INPUT
EAD5 B91A00      2045          MOV      CX,26              ; DI,ES ALREADY POINTING
EAD8 F2          2046          REPNE    SCASB              ; LOOK FOR MATCH IN THE BUFFER
EAD9 AE
EADA 7505        2047          JNE      K34                ; NOT FOUND, FUNCTION KEY OR OTHER
EADC 8000        2048          MOV      AL,0               ; ASCII CODE OF ZERO
EADE E9F400      2049          JMP      K57                ; PUT IT IN THE BUFFER
                 2050
                 2051    ;----- LOOK FOR TOP ROW OF ALTERNATE SHIFT
                 2052
EAE1             2053    K34:                              ; ALT-TOP-ROW
EAE1 3C02        2054          CMP      AL,2               ; KEY WITH '1' ON IT
EAE3 7503        2055          JB       K35                ; NOT ONE OF INTERESTING KEYS
EAE5 3C0E        2056          CMP      AL,14              ; IS IT IN THE REGION
EAE7 7508        2057          JAE      K35                ; ALT-FUNCTION
EAE9 80C476      2058          ADD      AH,118             ; CONVERT PSEUDO SCAN CODE TO RANGE
EAEC B000        2059          MOV      AL,0               ; INDICATE AS SUCH
EAEE E9E400      2060          JMP      K57                ; BUFFER_FILL
                 2061
                 2062    ;----- TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
                 2063
```

```
EAF1                2064    K35:                            ; ALT-FUNCTION
EAF1 3C3B           2065            CMP     AL,59           ; TEST FOR IN TABLE
EAF3 7303           2066            JAE     K37             ; ALT-CONTINUE
EAF5                2067    K36:                            ; CLOSE-RETURN
EAF5 E961FF         2068            JMP     K26             ; IGNORE THE KEY
EAF8                2069    K37:                            ; ALT-CONTINUE
EAF8 3C47           2070            CMP     AL,71           ; IN KEYPAD REGION
EAFA 73F9           2071            JAE     K36             ; IF SO, IGNORE
EAFC BB5FE9         2072            MOV     BX,OFFSET K13   ; ALT SHIFT PSEUDO SCAN TABLE
EAFF E91B01         2073            JMP     K63             ; TRANSLATE THAT
                    2074
```

Commentaires.- 1^o) Si la touche 'Alt' est pressée (lignes 1991–1992), nous sommes dans le cas des combinaisons de touches faisant intervenir 'Alt', que nous allons maintenant étudier. Sinon on poursuit notre recherche (ligne 1993).

- 2^o) On commence par le cas de l'appui simultané sur les touches 'Ctrl', 'Alt' et 'Del', spécifiant un redémarrage. On regarde si on a appuyé sur la touche 'Ctl' (lignes 1998 et 1999) et sur la touche 'Del' (lignes 2000 et 2001). S'il en est ainsi, signifiant que l'on veut un redémarrage à chaud du PC, on place donc '1234' à l'emplacement voulu de la zone de communication du BIOS (ligne 2005) et on revient à la partie du BIOS consacrée au démarrage (ligne 2006), que nous étudierons plus tard.

- 3^o) Rappelons que les chiffres de '0' à '9' du clavier numérique ont pour codes de touche respectifs 82, 79, 80, 81, 75, 76, 77, 71, 72 et 73. C'est ce que rappelle la table des lignes 2008 à 2011.

- 4^o) Rappelons que l'appui sur 'Alt' et sur l'une des touches de lettres 'A' à 'Z' donne lieu à un code étendu compris entre 16 et 25, 30 et 38, 44 et 50. Il s'agit tout simplement du code de la touche correspondante. Ces codes sont rappelés sur la table des lignes 2012 à 2017.

- 5^o) Nous avons vu que lorsque la touche 'Alt' est pressée, sans que ce soit Ctl-Alt-Del, on est renvoyé en K31, qui commence ligne 2020.

S'il s'agit de la barre d'espacement (touche de code 57), on place le code ASCII correspondant (32 ou 20h) dans AL. Ainsi AH contient le code de recherche et AL le code ASCII. On peut donc aller à l'étiquette `BUFFER_FILL` pour placer ceux-ci dans le tampon du clavier (lignes 2021 à 2024).

- 6^o) Si la touche appuyée se trouve sur la partie clavier numérique (lignes 2029 à 2032), on place la valeur de la touche dans le registre DI (ligne 2033), en jouant sur le fait que pour un chiffre x sa valeur est x - '0'. On place la valeur conservée dans `ALT_INPUT` dans AL (ligne 2034), que l'on multiplie par 10 (lignes 2035 et 2036), puis on lui ajoute la valeur du chiffre placée dans DI (ligne 2037), on stocke le tout dans `ALT_INPUT` (ligne 2038), et on termine proprement la routine de service (ligne 2039).

Autrement dit on a utilisé la méthode de Hörner pour placer un nombre (et pas seulement un chiffre) dans `ALT_INPUT`.

- 7^o) Si on ne se trouve pas sur le pavé numérique, on réinitialise `ALT_INPUT` à 0 pour une prochaine fois (ligne 2044) puis on recherche si la touche appuyée est celle d'une lettre de 'A' à 'Z' (lignes 2045 à 2047). Si on a appuyé sur une telle touche, on place 0 dans AL pour indiquer qu'il s'agit d'un code étendu, le code de la touche se trouvant dans AH correspond au code étendu, il suffit donc de placer le contenu de AX dans le tampon (lignes 2048 et 2049).

- 8^o) S'il s'agit des touches de code 2 à 13 (lignes 2054 à 2057), c'est-à-dire de la première ligne du clavier (les chiffres '1' à '0', '-', '=' et PgUp), cela donne lieu à un code étendu de 120 à 133, c'est-à-dire le code de la touche plus 118. On translate donc le code de la touche de 118 (ligne 2058), on place 0 dans AL et le tout dans le tampon du clavier.

- 9^o) S'il s'agit d'un code de recherche compris entre 59 et 71, on l'ignore, c'est-à-dire qu'on termine proprement la routine de service (lignes 2065 à 2071).

- 10^o) Il reste le cas de l'appui simultané sur 'Alt' et d'une touche de fonction. On transforme le code de la touche en code étendu (lignes 2072 et 2073) grâce à la table définie lignes 1836 et 1837. Pour cela on saute à la partie du BIOS commençant ligne 2263 :

```
                2263    ;----- TRANSLATE SCAN FOR PSEUDO SCAN CODES
                2264
EC1D            2265    K63:                            ; TRANSLATE-SCAN
EC1D 2C38       2266            SUB     AL,59           ; CONVERT ORIGIN TO FUNCTION KEYS
EC1F            2267    K64:                            ; TRANSLATE-SCAN-ORGO
EC1F 2ED7       2268            XLAT    CS:K9           ; CTL TABLE SCAN
EC21 8AE0       2269            MOV     AH,AL           ; PUT VALUE INTO AH
EC23 B000       2270            MOV     AL,0            ; ZERO ASCII CODE
EC25 EBAE       2271            JMP     K57             ; PUT IT INTO THE BUFFER
                2272
                2273    KB_INT  ENDP
                2274
```

La table K12 est définie à partir de la ligne 1830 :

```
                1830    ;----- UC TABLE SCAN
E955            1831    K12     LABEL   BYTE
E955 54         1832                    DB      84,85,86,87,88,89,90
E956 55
E957 56
E958 57
E959 58
E95A 59
E95B 5A
E95C 5B         1833                    DB      91,92,93
E95D 5C
E95E 5D
```

6.5.7 Cas des combinaisons de touches avec 'Ctrl'

Après le pseudo-code FFh, les codes des touches de basculement, les codes de relâchement, l'appui sur 'Pause' et les combinaisons de touches faisant intervenir 'Alt', on passe aux combinaisons de touches faisant intervenir 'Ctrl'.

L'appui simultané sur la touche 'Ctrl' et sur l'une des touches de numéro 3, 7, 12, 14, 16–28, 30–38, 43–50, 55, 59–71, 73, 75, 77, 79 et 81 spécifie le code ASCII défini dans le tableau vu à propos des caractères associés aux touches, repris dans le table suivante, comme on pourra le vérifier :

```
                1813    ;----- CTL TABLE SCAN
E8C8            1814    K9      LABEL   BYTE
E8C8 5E         1815                    DB      94,95,96,97,98,99,100,101
E8C9 5F
E8CA 60
E8CB 61
E8CC 62
E8CD 63
E8CE 64
E8CF 65
E8D0 66         1816                    DB      102,103,-1,-1,119,-1,132,-1
E8D1 67
E8D2 FF
E8D3 FF
E8D4 77
E8D5 FF
E8D6 84
E8D7 FF
E8D8 73         1817                    DB      115,-1,116,-1,117,-1,118,-1
E8D9 FF
E8DA 74
E8DB FF
E8DC 75
E8DD FF
E8DE 76
E8DF FF
E8E0 FF         1818                    DB      -1
```

Les codes étendus 94 à 103 correspondent à l'appui simultané sur 'Ctrl' et l'une des touches de fonction (de codes 59 à 68), pas de code (donc -1) pour les touches 69 et 70 (c'est-à-dire 'NumLock' et 'ScrollLock'), puis aux touches du pavé numérique (de codes 71 à 83).

Nous avons vu que si nous ne sommes pas dans le cas d'une combinaison de touches avec 'Alt', on est renvoyé en K38, qui commence ligne 2075 :

```
                2075    ;----- NOT IN ALTERNATE SHIFT
                2076
EB02            2077    K38:                                    ; NOT-ALT-SHIFT
EB02 F606170004 2078            TEST    KB_FLAG,CTL_SHIFT       ; ARE WE IN CONTROL SHIFT
EB07 7458       2079            JZ      K44                     ; NOT-CTL-SHIFT
                2080
                2081    ;----- CONTROL SHIFT, TEST SPECIAL CHARACTERS
                2082    ;----- TEST FOR BREAK AND PAUSE KEYS
                2083
EB09 3C46       2084            CMP     AL,SCROLL_KEY           ; TEST FOR BREAK
EB0B 7518       2085            JNE     K39                     ; NO-BREAK
EB0D 8B1E8000   2086            MOV     BX,BUFFER_START         ; RESET BUFFER TO EMPTY
EB11 891E1A00   2087            MOV     BUFFER_HEAD,BX
EB15 891E1A00   2088            MOV     BUFFER_TAIL,BX
EB19 C606710080 2089            MOV     BIOS_BREAK,80H          ; TURN ON BIOS_BREAK BIT
EB1E CD1B       2090            INT     1BH                     ; BREAK INTERRUPT VECTOR
EB20 2BC0       2091            SUB     AX,AX                   ; PUT OUT DUMMY CHARACTER
EB22 E98088     2092            JMP     K57                     ; BUFFER_FILL
EB25            2093    K39:                                    ; NO-BREAK
EB25 3C45       2094            CMP     AL,NUM_KEY              ; LOOK FOR PAUSE KEY
EB27 7521       2095            JNE     K41                     ; NO-PAUSE
EB29 80DE180008 2096            OR      KB_FLAG_1,HOLD_STATE    ; TURN ON THE HOLD FLAG
EB2E B020       2097            MOV     AL,EOI                  ; END OF INTERRUPT TO CONTROL PORT
EB30 E620       2098            OUT     020H,AL                 ; ALLOW FURTHER KEYSTROKE INTS
```

```
                    2099
                    2100    ;----- DURING PAUSE INTERVAL, TURN CRT BACK ON
                    2101
EB32 803E490007     2102            CMP     CRT_MODE,7          ; IS THIS BLACK AND WHITE CARD
EB37 7407           2103            JE      K40                 ; YES, NOTHING TO DO
EB39 BAD803         2104            MOV     DX,D308H            ; PORT FOR COLOR CARD
EB3C A06500         2105            MOV     AL,CRT_MODE_SET     ; GET THE VALUE OF THE CURRENT MODE
EB3F EE             2106            OUT     DX,AL               ; SET THE CRT MODE, SO THAT CRT IS ON
EB40                2107    K40:                                ; PAUSE-LOOP
EB40 F606180008     2108            TEST    KB_FLAG_1,HOLD_STATE
EB45 75F9           2109            JNZ     K40                 ; LOOP UNTIL FLAG TURNED OFF
EB47 E914FF         2110            JMP     K27                 ; INTERRUPT_RETURN_NO_EOI
EB4A                2111    K41:                                ; NO-PAUSE
                    2112
                    2113    ;----- TEST SPECIAL CASE KEY 55
                    2114
EB4A 3C37           2115            CMP     AL,55
EB4C 7506           2116            JNE     K42                 ; NOT-KEY-55
EB4E B80072         2117            MOV     AX,114*256          ; START/STOP PRINTING SWITCH
EB51 E98100         2118            JMP     K57                 ; BUFFER_FILL
                    2119
                    2120    ;----- SET UP TO TRANSLATE CONTROL SHIFT
                    2121
EB54                2122    K42:                                ; NOT-KEY-55
EB54 BB8EE8         2123            MOV     BX,OFFSET K8        ; SET UP TO TRANSLATE CTL
EB57 3C3B           2124            CMP     AL,59               ; IS IT IN TABLE
                    2125                                        ; CTL-TABLE-TRANSLATE
EB59 7276           2126            JB      K56                 ; YES, GO TRANSLATE CHAR
EB5B                2127    K43:                                ; CTL-TABLE-TRANSLATE
EB5B BBC8E8         2128            MOV     BX,OFFSET K9        ; CTL TABLE SCAN
EB5E E9BC00         2129            JMP     K63                 ; TRANSLATE_SCAN
                    2130
```

Commentaires.- 1^o) Si la touche 'Ctrl' est pressée (lignes 2078), nous sommes dans le cas des combinaisons de touches faisant intervenir 'Ctrl' (sauf 'Ctrl-Alt-DEl' déjà traité), que nous allons maintenant étudier. Sinon on poursuit notre recherche (ligne 2079).

- 2^o) Si on est en train d'appuyer sur la touche 'ScrollLock', c'est qu'on veut passer à la fonction 'Break'. C'est le premier cas qui est traité (lignes 2084 et 2085). Pour cela on vide le tampon du clavier (lignes 2086 à 2088), on fait appel à l'interruption 1Bh (interruption du BIOS chargée de la fonction 'Break' ; ligne 2090), on place 0 dans AX (ligne 2091), correspondant au code ASCII de numéro 0, que l'on place dans le tampon du clavier (ligne 2092).

- 3^o) Si on est en train d'appuyer sur la touche 'NumLock', c'est qu'on veut une 'Pause' (lignes 2094 et 2095). Dans ce cas, on l'indique dans le second octet d'état du clavier (ligne 2096). On permet à une autre interruption de prendre la main (lignes 2097 et 2098). On gèle l'écran (lignes 2102 à 2106) et on attend jusqu'à ce qu'on appuie sur une autre touche (lignes 2107 à 2110).

- 4^o) Si on est en train d'appuyer sur la touche 'PrtSc' (de code 55 ; lignes 2115 et 2116), on place le code spécial de début/fin d'impression dans le tampon du clavier (lignes 2117 et 2118).

- 5^o) On place (ligne 2123) dans BX le début de la table des codes ASCII correspondants aux codes des touches dans le cas où on a appuyé sur la touche 'Ctrl' pour les 58 premiers codes de touches (ce que l'on peut vérifier avec le tableau des caractères donné antérieurement) :

```
                1802
                1803    ;----- SCAN CODE TABLES
                1804
E88E 1B         1805    K8              DB      27,-1,0,-1,-1,-1,30,-1
E88F FF
E890 00
E891 FF
E892 FF
E893 FF
E894 1E
E895 FF
E896 FF         1806                    DB      -1,-1,-1,31,-1,127,-1,17
E897 FF
E898 FF
E899 1F
E89A FF
E89B 7F
E89C FF
E89D 11
E89E 17         1807                    DB      23,5,18,20,25,21,9,15
E89F 05
E8A0 12
E8A1 14
E8A2 19
E8A3 15
E8A4 09
E8A5 0F
E8A6 10         1808                    DB      16,27,29,10,-1,1,19
E8A7 1B
E8A8 1D
E8A9 0A
E8AA FF
E8AB 01
E8AC 13
E8AD 04         1809                    DB      4,6,7,8,10,11,12,-1,-1
E8AE 06
E8AF 07
E8B0 08
E8B1 0A
E8B2 0B
E8B3 0C
E8B4 FF
E8B5 FF
E8B6 FF         1810                    DB      -1,-1,28,26,24,3,22,2
E8B7 FF
E8B8 1C
E8B9 1A
E8BA 18
E8BB 03
E8BC 16
E8BD 02
E8BE 0E         1811                    DB       14,13,-1,-1,-1,-1,-1,-1
E8BF 0D
E8C0 FF
E8C1 FF
E8C2 FF
E8C3 FF
E8C4 FF
E8C5 FF
E8C6 20         1812                    DB       ' ',-1
E8C7 FF
```

Si le code de touche est inférieur à 58, on utilise cette table (ligne 2124) pour obtenir le code ASCII puis on place le caractère correspondant dans le tampon du clavier (ligne ligne 2128) : on prend comme origine la table suivante car on décrit les tables à l'envers. On va donc en K63 (ligne 2129), partie déjà étudiée.

- 6^o) Si le code de la touche est supérieur à 59, on utilise la table K9 pour obtenir le code étendu et on place le résultat obtenu dans le tampon du clavier.

On est renvoyé en K56 (ligne 2126), qui commence à la ligne 2209 :

```
                2209    ;----- TRANSLATE THE CHARACTER
                2210
EBD1            2211    K56:                            ; TRANSLATE-CHAR
EBD1 FEC8       2212            DEC     AL              ; CONVERT ORIGIN
EBD3 2ED7       2213            XLAT    CS:K11          ; CONVERT THE SCAN
                2214
```

qui passe à K57, c'est-à-dire la mise dans le tampon, lorsqu'on a terminé.

6.5.8 Cas de l'appui sur une touche avec ou sans majuscule

Après le pseudo-code FFh, les codes des touches de basculement, les codes de relâchement, l'appui sur 'Pause', les combinaisons de touches faisant intervenir 'Alt', les combinaisons de touches faisant intervenir 'Ctrl', il ne reste plus que l'appui sur une seule touche ou sur une touche et l'une des touches 'Majuscule', avec la possibilité que l'état 'CapsLock' ou 'NumLock' soit actif.

Nous avons vu que si nous ne sommes pas dans le cas d'une combinaison de touches avec 'Ctrl', nous sommes renvoyés à K44, qui commence à la ligne 2131 :

```
                2130
                2131    ;----- NOT IN CONTROL SHIFT
                2132
EB61            2133    K44:                            ; NOT-CTL-SHIFT
EB61 3C47       2134            CMP     AL,71           ; TEST FOR KEYPAD REGION
EB63 732C       2135            JAE     K46             ; HANDLE KEYPAD REGION
EB65 F606170003 2136            TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
EB6A 745A       2137            JZ      K54             ; TEST FOR SHIFT STATE
                2138
                2139    ;----- UPPER CASE, HANDLE SPECIAL CASES
                2140
EB6C 3C0F       2141            CMP     AL,15           ; BACK TAB KEY
EB6E 7505       2142            JNE     K45             ; NOT-BACK-TAB
EB70 B8000F     2143            MOV     AX,15*256       ; SET PSEUDO SCAN CODE
EB73 EB60       2144            JMP     SHORT K57       ; BUFFER_FILL
EB75            2145    K45:                            ; NOT-BACK-TAB
EB75 3C37       2146            CMP     AL,55           ; PRINT SCREEN KEY
EB77 7509       2147            JNE     K46             ; NOT-PRINT-SCREEN
                2148
                2149    ;----- ISSUE INTERRUPT TO INDICATE PRINT SCREEN FUNCTION
                2150
EB79 B020       2151            MOV     AL,EOI          ; END OF CURRENT INTERRUPT
EB7B E620       2152            OUT     020H,AL         ;  SO FURTHER THINGS CAN HAPPEN
EB7D CD05       2153            INT     5H              ; ISSUE PRINT SCREEN INTERRUPT
EB7F E9DCFE     2154            JMP     K27             ; GO BACK WITHOUT EOI OCCURING
EB82            2155    K46:                            ; NOT-PRINT-SCREEN
EB82 3C3B       2156            CMP     AL,59           ; FUNCTION KEYS
EB84 7206       2157            JB      K47             ; NOT-UPPER-FUNCTION
EB86 BB55E9     2158            MOV     BX,OFFSET K12   ; UPPER CASE PSEUDO SCAN CODES
EB89 E9910D     2159            JMP     K63             ; TRANSLATE_SCAN
EB8C            2160    K47:                            ; NOT-UPPER-FUNCTION
EB8C BB18E9     2161            MOV     BX,OFFSET K11   ; POINT TO UPPER CASE TABLE
EB8F EB40       2162            JMP     SHORT K56       ; OK, TRANSLATE THE CHAR
                2163
                2164    ;----- KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
                2165
EB91            2166    K48:                            ; KEYPAD-REGION
EB91 F606170020 2167            TEST    KB_FLAG,NUM_STATE ; ARE WE IN NUM_LOCK
EB96 7520       2168            JNZ     K52             ; TEST FOR SURE
EB98 F606170003 2169            TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE
EB9D 7520       2170            JNZ     K53             ; IF SHIFTED, REALLY NUM STATE
                2171
                2172    ;----- BASE CASE FOR KEYPAD
                2173
EB9F            2174    K49:                            ; BASE-CASE
EB9F 3C4A       2175            CMP     AL,74           ; SPECIAL CASE FOR A COUPLE OF KEYS
EBA1 740B       2176            JE      K50             ; MINUS
EBA3 3C4E       2177            CMP     AL,76
EBA5 740C       2178            JE      K51
EBA7 2C47       2179            SUB     AL,71           ; CONVERT ORIGIN
EBA9 BB76E9     2180            MOV     BX,OFFSET K15   ; BASE CASE TABLE
EBAC EB71       2181            JMP     SHORT K64       ; CONVERT TO PSEUDO SCAN
EBAE            2182    K50:
EBAE B82D4A     2183            MOV     AX,74*256+'-'   ; MINUS
EBB1 EB22       2184            JMP     SHORT K57       ; BUFFER_FILL
EBB3            2185    K51:
EBB3 B82B4E     2186            MOV     AX,78*256+'+'   ; PLUS
EBB6 EB1D       2187            JMP     SHORT K57       ; BUFFER_FILL
                2188
                2189    ;----- MIGHT BE NUM LOCK, TEST SHIFT STATUS
                2190
```

```
EBB8                2191    K52:                            ; ALMOST-NUM-STATE
EBB8 F606170003     2192            TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
EBBD 75ED           2193            JNZ     K49             ; SHIFTED TEMP OUT OF NUM STATE
EBBF                2194    K53:                            ; REALLY_NUM_STATE
EBBF 2C46           2195            SUB     AL,70           ; CONVERT ORIGIN
EBC1 BB69E9         2196            MOV     BX,OFFSET K14   ; NUM STATE TABLE
EBC4 EB0B           2197            JMP     SHORT K56       ; TRANSLATE_CHAR
                    2198
                    2199    ;----- PLAIN OLD LOWER CASE
                    2200
EBC6                2201    K54:                            ; NOT-SHIFT
EBC6 3C3B           2202            CMP     AL,59           ; TEST FOR FUNCTION KEYS
EBC8 7204           2203            JB      K55             ; NOT-LOWER-FUNCTION
EBCA B000           2204            MOV     AL,0            ; SCAN CODE IN AH ALREADY
EBCC EB07           2205            JMP     SHORT K57       ; BUFFER_FILL
EBCE                2206    K55:                            ; NOT-LOWER-FUNCTION
EBCE BBE1E8         2207            MOV     BX,OFFSET K10   ; LC TABLE
                    2208
```

Commentaires.- 1^o) Si la touche a un numéro inférieur à 70 (lignes 2134 et 2135), elle ne se trouve pas sur le clavier numérique (on n'a donc pas à se préoccuper de l'état 'NumLock').

- 2^o) Si l'une des touches 'Majuscule' est appuyée (lignes 2136 et 2137), on est dans le cas des majuscules.

- 3^o) On commence par les cas particuliers. Si la touche appuyée est la barre d'espacement (lignes 2141 et 2142), on place le code de cette touche et le code ASCII correspondant dans AX (ligne 2143) et on place AX dans le tampon du clavier (ligne 2144).

- 3^o) Si la touche appuyée est la touche 'PrScr' (lignes 2146 et 2147), on met fin à l'interruption actuelle (lignes 2151 et 2152), on fait appel à l'interruption d'impression (ligne 2153) et on termine correctement la routine de service (ligne 2154), sauf la fin d'interruption déjà effectuée.

- 4^o) Si le code de la touche appuyée est inférieur aux codes des touches de fonction (lignes 2156 et 2157), on se positionne à la fin de la table K11 (ligne 2158, c'est-à-dire le décalage de la table K12 car on décrit les tables à l'envers) des codes ASCII correspondant aux codes lorsque la majuscule est activée :

```
                    1825    ;----- UC TABLE
E91B                1826    K11     LABEL   BYTE
E91B 1B             1827                    DB      27,'!@#&',37,05EH,'&*()_+',06H,0
E91C 21402324
E920 25
E921 5E
E922 262A28295F2B
E928 06
E929 00
E92A 515745255455955  1828                  DB      'QWERTYUIOP{}',0DH,-1,'ASDFGHJKL:"'
     494F507B7D
E936 0D
E937 FF
E938 4153444647484A
     4B4C3A22
E943 7E             1829                    DB       07EH,-1,'|ZXCVBNM<>?',-1,0,-1,' ',-1
E944 FF
E945 7C5A584356424E
     4D3C3E3F
E950 FF
E951 00
E952 FF
E953 2D
E954 FF
```

On traduit le code clavier en code ASCII et on place le contenu de AX dans le tampon (ligne 2159 qui renvoie à K63, partie déjà étudiée).

- 5^o) Si aucune des touches majuscule n'est pressée, nous sommes dans le cas des minuscules. On se positionne à la fin de la table K10 (ligne 2161, c'est-à-dire le décalage de la table K11 car on décrit les tables à l'envers) des codes ASCII correspondant aux codes lorsque la majuscule n'est pas activée :

```
                    1819    ;----- LC TABLE
E8E1                1820    K10     LABEL   BYTE
E8E1 1B             1821                    DB      01BH,'1234567890-=',08H,09H
E8E2 31323334353637
     3839302D3D
E8EE 08
E8EF 09
E8F0 71776572747975 1822                    DB     'qwertyuiop[]',0DH,-1,'asdfghjkl;',027H
     696F705B5D
E8FC 0D
E8FD FF
E8FE 6173646667686A
     6B6C3B
E908 27
E909 60             1823                    DB      60H,-1,5CH,'zxcvbnm,./',-1,'*',-1,' '
E90A FF
E90B 7A786376626E60
     2C2E2F
E916 FF
E917 2A
E918 FF
E919 20
E91A FF             1824                    DB      -1
```

On traduit le code clavier en code ASCII et on place le contenu de AX dans le tampon (ligne 2162 qui renvoie à K56, partie déjà étudiée).

- 6^o) On teste ensuite si la touche appuyée appartient au clavier numérique, ce qui nous a renvoyé en K48, c'est-à-dire à la ligne 2166. On distingue (lignes 2167 à 2170) deux cas suivant que 'NumLock' est actif ou que l'on a appuyé sur une touche majuscule (dans ce cas on a le clavier numérique proprement dit) ou non (dans ce cas on a le clavier de changement de position du curseur).

- 7^o) Dans le cas où il s'agit du clavier numérique proprement dit, on commence par traiter les deux cas particuliers du '-' (lignes 2175 et 2175), pour lequel on place le code clavier et le code ASCII dans le tampon du clavier (lignes 2183 et 2184) et du '+' (lignes 2176 à 2178), pour lequel on fait la même chose, mais évidemment avec des valeurs différentes (lignes 2185 à 2187).

Sinon on soutrait 71 (ligne 2179) pour que le premier code soit 0, on cherche le code ASCII grâce à la table K15 (lignes 2180 et 2181, renvoyant à une partie déjà étudiée). La table K15 est définie à partir de la ligne 1841 :

```
                    1841    ;----- BASE CASE TABLE
E976                1842    K15     LABEL   BYTE
E976 43             1843                    DB      71,72,73,-1,75,-1,77
E977 48
E978 49
E979 FF
E97A 4B
E97B FF
E97C 4D
E97D FF             1844                    DB      -1,79,80,81,82,83
E97E 4F
E9FF 5D
E97F 51
E980 52
E981 53
                    1845
```

- 8^o) Dans le cas où l'état 'NumLock' n'est pas actif mais que l'une des touches majuscules est enfoncée, on est ramené au cas précédent (lignes 2191 à 2192).

- 9^o) Dans le cas du clavier numérique, il suffit de rechercher le code ASCII grâce à la table K14 (lignes 2195 à 2197) et de placer ce qu'il faut dans le tampon du clavier.

La table K14 est définie à partir de la ligne 1838 :

```
                    1838    ;----- NUM STATE TABLE
E969                1839    K14     LABEL   BYTE
E969 3738392D343536 1827                    DB      '789-456+1230.'
     2B313233302E
```

- 10^o) Dans le cas des touches de fonction (lignes 2201 à 2203), il suffit de placer 0 dans AL (ligne 2204) pour indiquer qu'il s'agit d'un code étendu et le code de la touche dans AH, puisqu'il s'agit du code étendu, ce qui est déjà fait, puis de placer AX dans le tampon du clavier (ligne 2205).

- 11^o) Le dernier cas à traiter est celui des touches minuscules. On détermine le code ASCII grâce à la table K10 et on place AX dans le tampon du clavier (lignes 2206 et 2207, les lignes suivantes consistant à placer le tampon dans le clavier).

6.6 Évolution du clavier

6.6.1 Disposition des touches

Clavier PC/AT.- Il s'agit d'un clavier amélioré de 84 touches (figure 6.8), dont la disposition des touches est schématisée sur la figure 6.9.

FIGURE 6.8 – Clavier PC/AT

La touche « Entrée » est correctement dimensionnée de façon à ce que l'opérateur ne risque plus de taper à côté lors d'une frappe soutenue. De même les deux touches « Majuscule » ont la bonne taille.

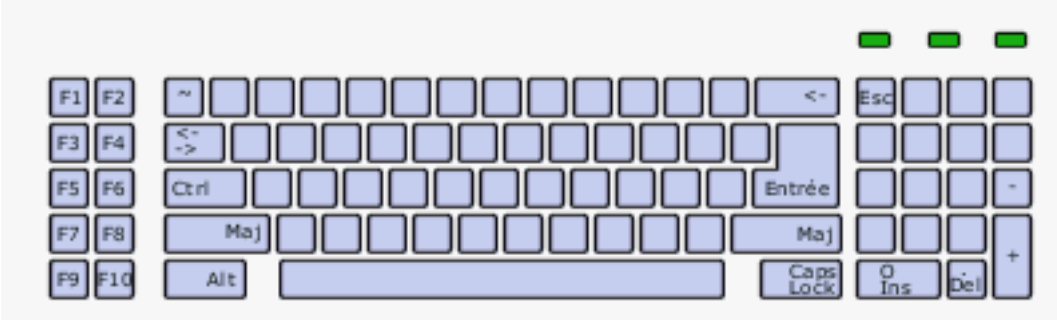


FIGURE 6.9 – Disposition des touches du clavier PC/AT

Mais comme les dimensions de l'ensemble n'ont pas varié, d'autres touches ont dû réduire leur encombrement : c'est le cas des touches de verrouillage numérique ('NumLock') et d'arrêt de défilement ('ScrollLock') qui ne sont pas d'un usage intensif et qui n'ont pas de raison d'être plus grande que les touches « Majuscule » ; le même sort frappe la touche '+' située à droite du pavé numérique.

Une nouvelle touche apparaît : la touche 'Sys', plus tard rebaptisée 'SysReq'. Elle devait constituer une sorte de touche de fonction spécialisée dans les appels au système ou aux programmes résidents, mais elle n'a jamais été exploitée par les concepteurs de logiciels.

Les touches de blocage possèdent maintenant un indicateur LED indiquant leur état.

Clavier étendu.- IBM décida de rompre avec le design de son clavier classique en introduisant le clavier étendu (*advanced keyboard* ou *enhanced keyboard*), conçu à l'origine pour le PC/AT.

Le clavier étendu (figure 6.10) présente des caractéristiques remarquables qui lui ont assuré une percée rapide :

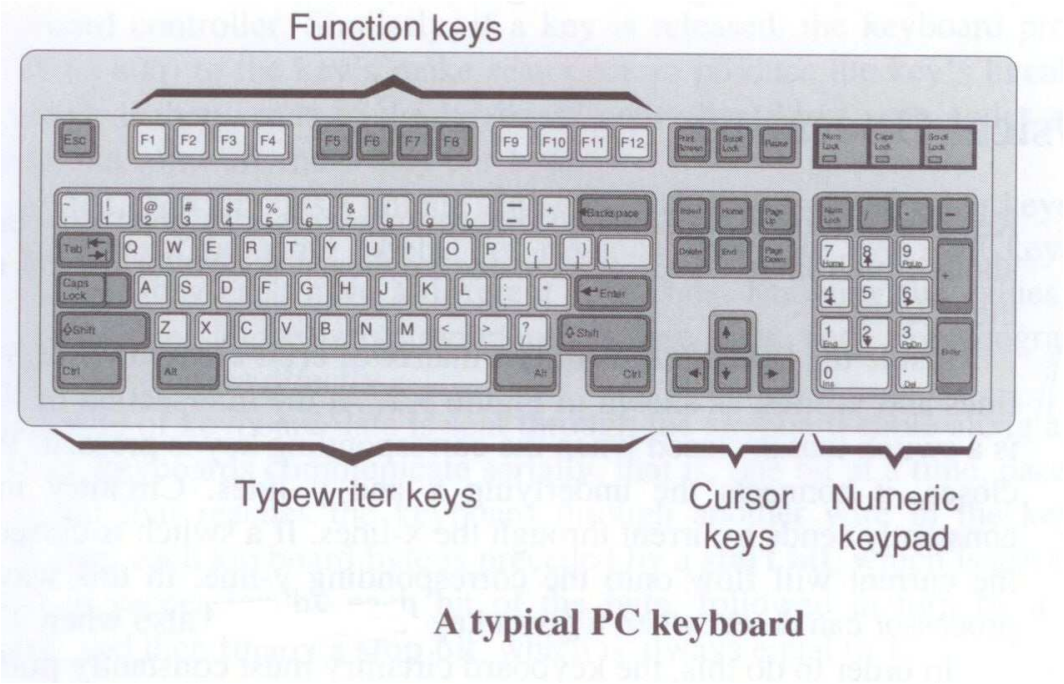


FIGURE 6.10 – Les quatre parties d'un clavier MF II

— un bloc de touches de direction a été rajouté pour décharger le pavé numérique ;

— les touches de fonction ont été déplacées dans la rangée supérieure, comme c'est le cas pour les claviers des grands systèmes IBM ;

— des touches de fonction supplémentaires F11 et F12 ont été ajoutées, également dans un but de compatibilité avec les grands systèmes ;

— la touche de commande 'Alt' a été placée dans dans la rangée de la barre d'espacement pour être plus accessible ; par ailleurs une touche 'Alt Gr' (pour *ALTernative GRey*) a été créée pour simuler la frappe simultanée des touches 'Alt' et 'Ctrl' ;

— trois diodes électroluminescentes affichent l'état du verrouillage numérique, du verrouillage des majuscules et de l'arrêt de défilement.

Le clavier étendu existe en deux versions différentes : la version américaine possède 101 touches tandis que la version européenne en comporte 102, soit une de plus. Cette touche supplémentaire a été disposée à droite de la touche <Majuscule gauche> qui a subi de ce fait une amputation. Elle n'apporte pas beaucoup d'avantages aux utilisateurs européens car les symboles qu'elle porte sont accessibles par une autre touche alors que la réduction de la touche <Majuscule gauche> restaure l'inconfort du clavier PC/XT tant décrié à l'époque.

6.6.2 Câble

Le signal de **réinitialisation du clavier** (broche 3) permet une réinitialisation du clavier (sic) avec certaines interfaces.

6.6.3 Programmation du clavier

Sur les IBM PC/AT et PS, le 74LS322 est remplacé par un 8042. Ceci permet de programmer le clavier. Ainsi deux micro-contrôleurs 8042, l'un sur le clavier et l'autre sur la carte mère, sont responsables de la communication bidirectionnelle entre l'unité centrale et le clavier.

Sur les claviers AT et étendu, on peut ainsi programmer le taux de répétition des touches avec des valeurs comprises entre 2 et 30 caractères par seconde.

6.6.3.1 Numérotation des touches

Cas du clavier AT.- La figure 6.11 montre les codes clavier des touches du clavier PC/AT.

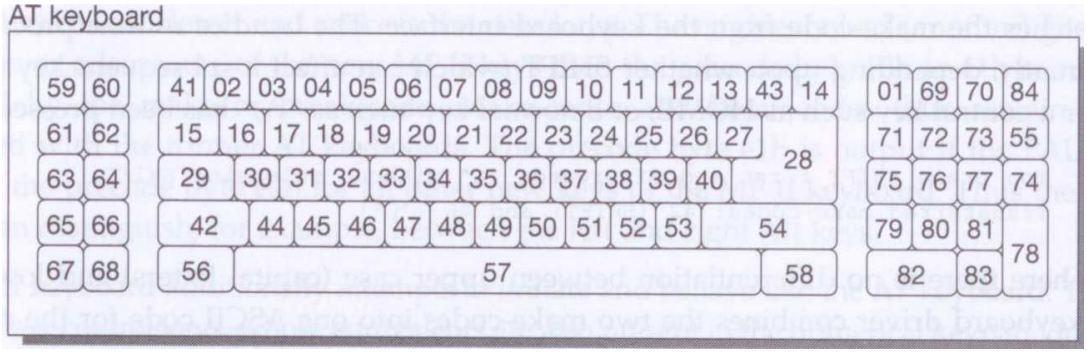


FIGURE 6.11 – Codes de recherche du clavier PC/AT

Les touches correspondantes ont le même code de recherche que sur le PC/XT, bien qu'elles ne soient pas placées au même endroit. La nouvelle touche 'Sys Req' se voit tout naturellement attribuer le code de recherche 84.

Cas du clavier étendu.- La figure 6.12 montre les codes de recherche associés aux touches du clavier PC/MF II à 102 touches.

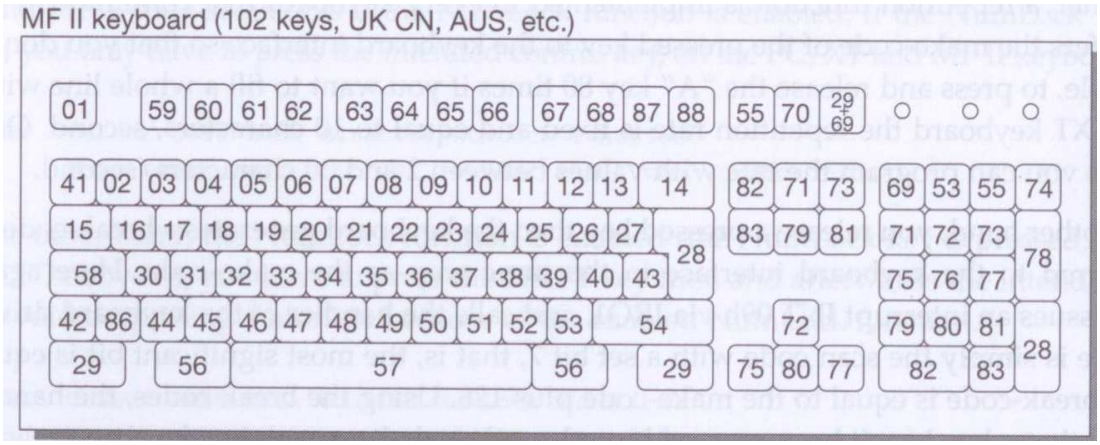


FIGURE 6.12 – Codes de recherche du clavier PC/MF II

Pour des raisons de compatibilité, le code de recherche des touches correspondantes est le même. Ceci ne permet pas *a priori* de distinguer les touches qui ont été dédoublées l'une de l'autre : elles ont le même code de recherche et donc un programme ne peut pas distinguer si on a appuyé, par exemple, sur la touche 'Alt' gauche ou droite. Les concepteurs du clavier étendu ont résolu ce problème de la façon suivante : lorsqu'on appuie ou qu'on relâche une touche dédoublée, un précode est d'abord envoyé, suivi par le code d'appui ou le code de relâchement. Il s'agit de E0h pour la touche 'PAUSE' et de E1h pour toutes les autres nouvelles touches.

Le tableau suivant donne les codes de recherche de combinaisons de touches [IBM-83] :

Hex	Keys	Hex	Keys	Hex	Keys	Hex	Keys
54	Shift F1	60	Ctrl F3	6C	Alt F5	78	Alt 1
55	Shift F2	61	Ctrl F4	6D	Alt F6	79	Alt 2
56	Shift F3	62	Ctrl F5	6E	Alt F7	7A	Alt 3
57	Shift F4	63	Ctrl F6	6F	Alt F8	7B	Alt 4
58	Shift F5	64	Ctrl F7	70	Alt F9	7C	Alt 5
59	Shift F6	65	Ctrl F8	71	Alt F10	7D	Alt 6
5A	Shift F7	66	Ctrl F9	72	Ctrl PrtSc	7E	Alt 7
5B	Shift F8	67	Ctrl F10	73	Ctrl LeftArrow	7F	Alt 8
5C	Shift F9	68	Alt F1	74	Ctrl RightArrow	80	Alt 9
5D	Shift F10	69	Alt F2	75	Ctrl End	81	Alt 10
5E	Ctrl F1	6A	Alt F3	76	Ctrl PgDn		
5F	Ctrl F2	6B	Alt F4	77	Ctrl Home		

Et enfin le tableau suivant donne les codes de recherche supplémentaires du clavier étendu [IBM-83] :

Hex	Keys	Hex	Keys	Hex	Keys	Hex	Keys
85	F11	8E	Ctrl -	97	Alt Home	A0	Alt DownArrow
86	F12	8F	Ctrl 5	98	Alt UpArrow	A1	Alt PgDn
87	Shift F11	90	Ctrl +	99	Alt PgUp	A2	Alt Insert
88	Shift F12	91	Ctrl DownArrow	9A		A3	Alt Delete
89	Ctrl F11	92	Ctrl Insert	9B	Alt LeftArrow	A4	Alt /
8A	Ctrl F12	93	Ctrl Delete	9C		A5	Alt Tab
8B	Alt F11	94	Ctrl Tab	9D	Alt RightArrow	A6	Alt Enter
8C	Alt F12	95	Ctrl /	9E			
8D	Ctrl UpArrow	96	Ctrl *	9F	Alt End		