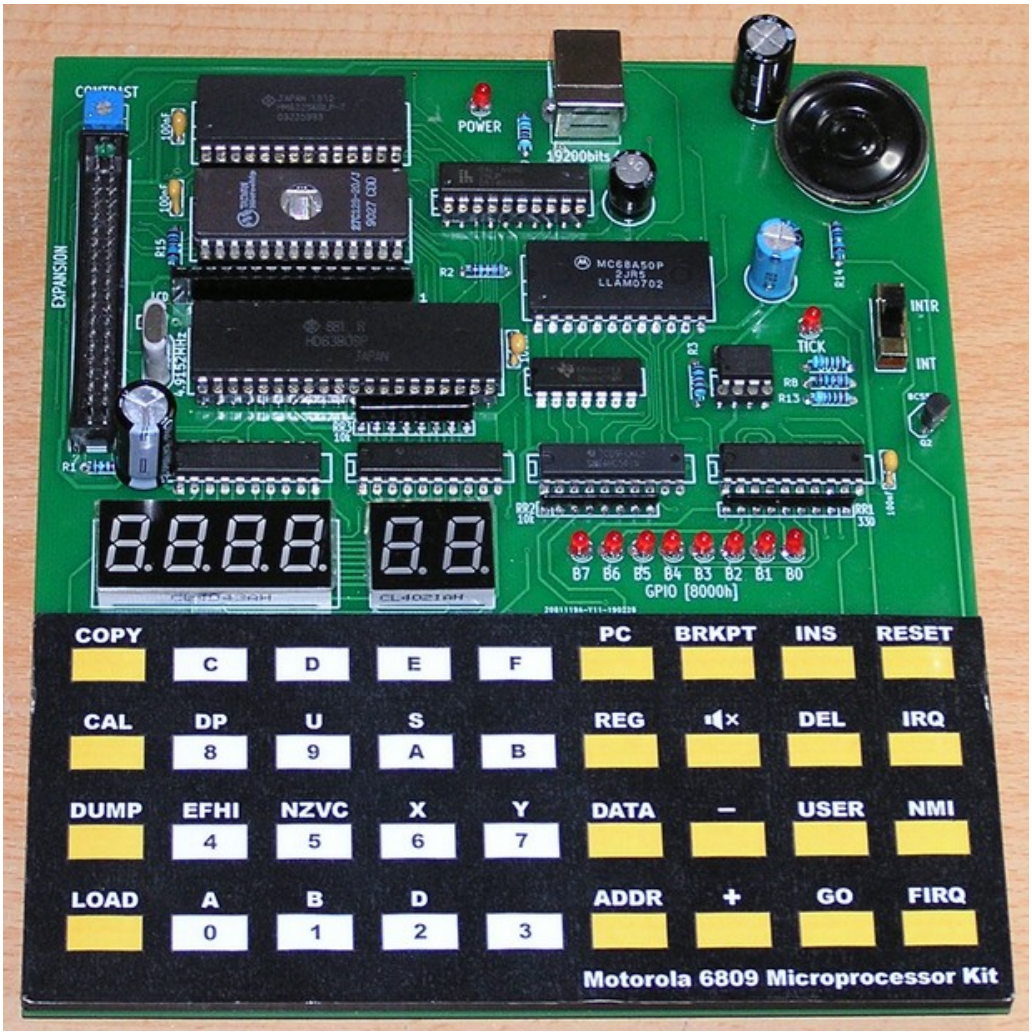
Livret de programmation 6809



Sommaire

1 Organisation mémoire

Le 6809 est un microprocesseur 8 bits pouvant adresser 64Ko de mémoire. Dans ce kit, cette mémoire est découpée en 4 zones :

- la mémoire programme (ROM) qui contient le moniteur
- la mémoire vive (RAM) qui contient les programmes utilisateurs et les données volatiles
- les entrées-sorties internes qui permettent de contrôler l'affichage et le clavier
- une zone mémoire libre permettant d'ajouter des cartes d'extensions

La ***Figure 1*** donne le mapping mémoire adopté par le système 6809.

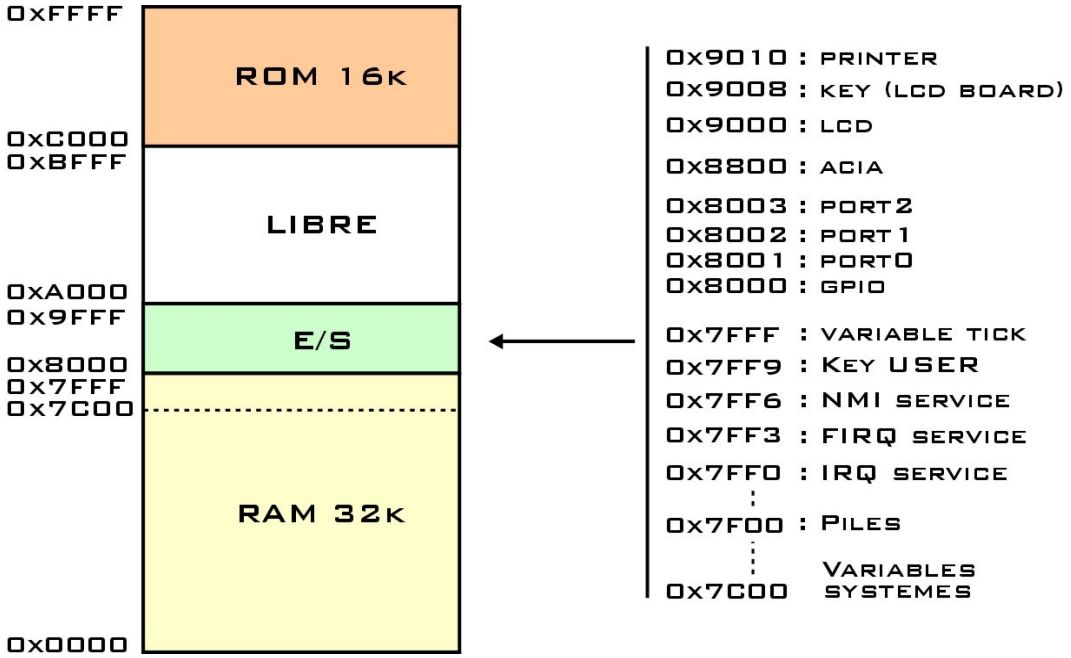


Figure 1: Mapping mémoire du kit 6809

Le 6809 dispose de 8 registres :

A : accumulateur 8bits
B : accumulateur 8bits
D : accumulateur 16 bits (A=OPS, B=OMS)
X : index 16bits
Y : index 16bits
DP : registre de page (8bits)
CC : registre d'état
U : pointeur de pile utilisateur
S : pointeur de pile système

2 Fonctions du clavier

Le clavier comporte 36 touches réparties comme le montre la ***Figure 2***.

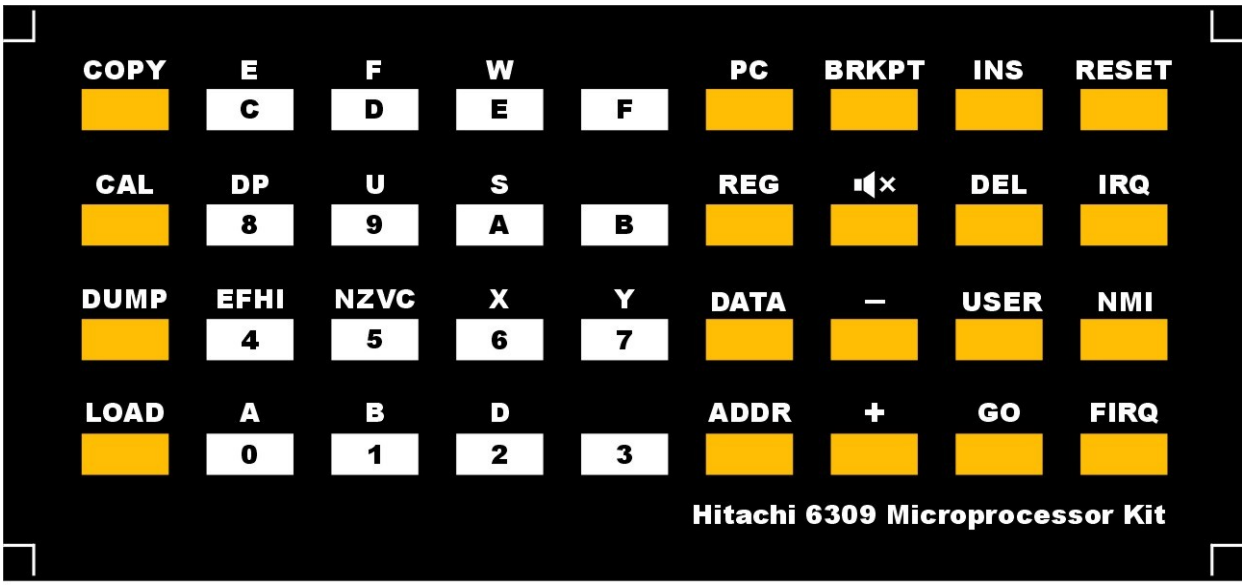


Figure 2: Clavier du kit 6809

- 0-F : clavier alphanumérique hexadécimal
- COPY : permet de copier une zone mémoire vers une autre (Start, End, Destination)
- CAL : calculatrice pour calculer les offsets
- DUMP : envoi un dump de 512 octets vers l'USB
- LOAD : lecture d'un programme via USB (format motorola S)
- PC : affiche la valeur actuelle de PC
- REG : permet de visualiser et modifier les registres du 6809
- DATA : active la saisie des données
- ADDR : active la saisie des adresses
- BRKPT : place/enlève un point d'arrêt
- SOUND : active/désactive le haut-parleur
- INS : insère un octet à l'adresse suivante et déplace un bloc de 512 octets
- DEL : supprime l'octet à l'adresse actuelle et déplace un bloc de 512 octets
- USER : touche utilisateur, provoque le branchement à l'adresse $7FF9
- GO : lance un programme
- RESET : reset du système
- IRQ : interruption IRQ
- NMI : interruption NMI
- FIRQ : interruption FIRQ

3 Fonctions disponibles pour l'utilisateur

La ROM du moniteur propose des fonctions de contrôle du matériel pour l'utilisateur final.

Fonction	Adresse	Paramètres(s)	Action
INITLCD	$F000	aucun	Initialise l'afficheur LCD
CLS	$F02D	aucun	Efface l'écran
CURSOR	$F039	A=état curseur	Active/désactive le curseur
LPUTCH	$F047	A=caractère	Affiche 1 caractère
LPUTS	$F051	X=début de la chaîne	Affiche 1 chaîne de car,
GOTOXY	$F062	A=X,B=Y	Place le curseur en X,Y
PRINTHEX	$F08C	A=nombre	Affiche 1 nombre hexa (2dig.)
PRINTHEX4	$F099	X=nombre	Affiche 1 nombre hexa (4 dig.)
PRINTINT	$F0A6	A=nombre	Affiche 1 nombre décimal (3 dig.)
DELAY2	$F0D6	aucun	Tempo 0,25s
DELAY5	$F0E3	aucun	Tempo 0,5s
TONE1K	$F0F5	X=durée	Produit un son @1kHz
TONE2K	$F110	X=durée	Produit un son @2kHz
CIN	$F127	A=caractère	Attend 1 car. envoyé par le terminal
COUT	$F132	A=caractère	Envoie 1 car. au terminal
NWLINE	$F141	aucun	Envoie CRLF au terminal
PSTRING	$F149	X=début de la chaîne	Envoie 1 chaîne de car. au terminal
OUT2X	$F168	A=nombre	Envoie 1 nb hexa (2dig.) au terminal
OUT4X	$F174	X=nombre	Envoie 1 nb hexa (4 dig.) au terminal
OUTINT	$F183	A=nombre	Envoie 1 nb décimal (3 dig.) au terminal
SCANKB	$F1B2	A=code de la touche	Lecture du clavier

Tableau 1: Fonctions utilisateur du kit

La ***Figure 3*** donne les codes des touches du clavier principal renvoyés par la routine SCANKB. Les touches grisées ne retournent aucun code car elles activent des commandes matérielles.

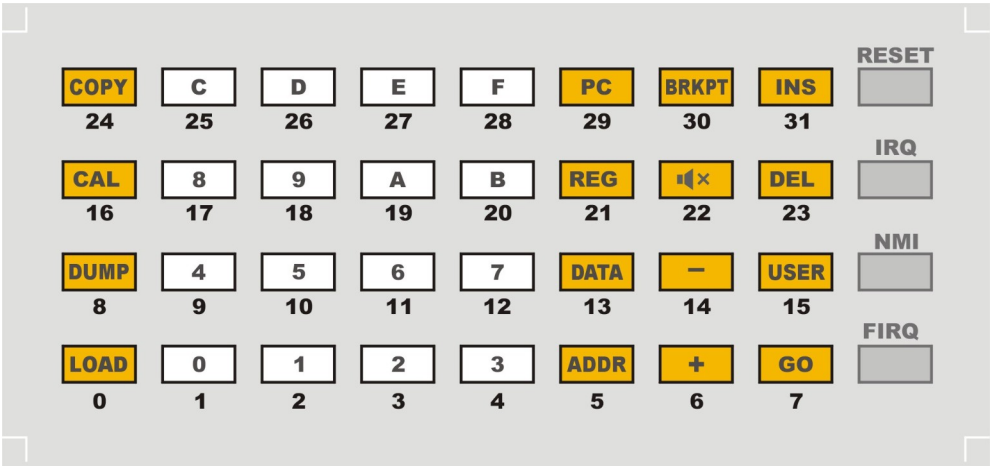


Figure 3: Codes des touches du clavier (fonction SCANKB)

4 Platines d'extension

Le kit d'initiation MPF1-6809 peut être connecté à différentes cartes d'extension permettant d'ajouter des entrées-sorties logiques, analogiques, sonores ou encore de la mémoire. Ces extensions occupent un espace mémoire situé entre $B000 et $BFFF.

4.1 Platine d'extension 1

L'extension 1 propose :

- (1) AY3-8910 : synthétiseur musical 3 voies + générateur de bruit (cadencé à 1MHz) + 2 ports de 8 bits soit 16 E/S. Les ports A et B sont connectés à 2 barrettes de connexion noires à gauche de la platine. La sortie est disponible sur le jack audio 3,5mm situé en haut de la platine (cavalier en position AY3).
- (2) ROM : 512Ko de mémoire à accès indirect.

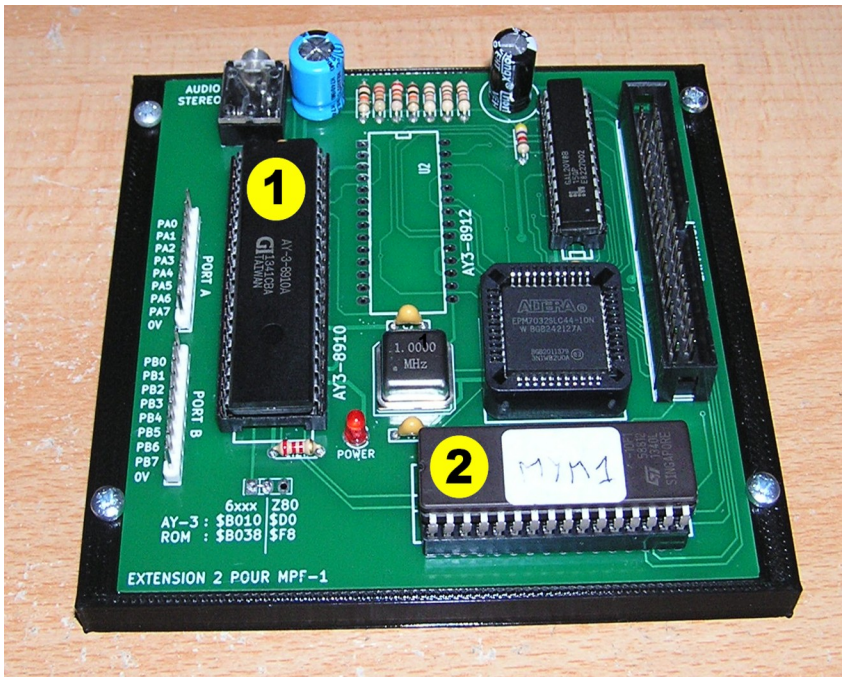


Figure 4: Platine d'extension 1

4.1.1 Implantation mémoire détaillée

La **Figure 5** donne le mapping mémoire détaillé.

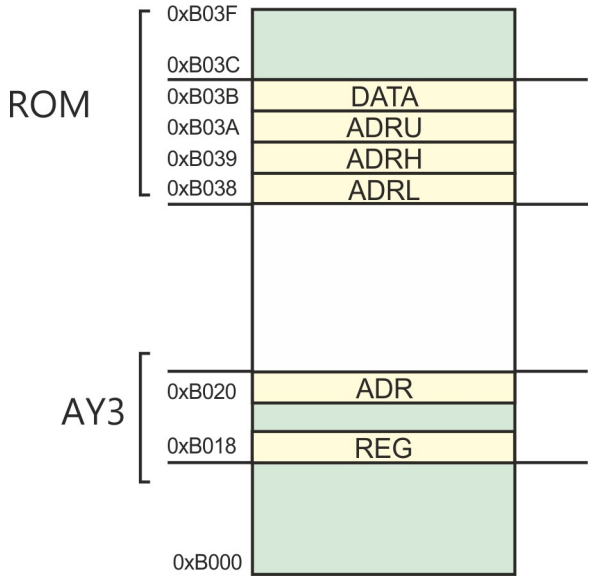


Figure 5: Implantation mémoire de l'extension 1

4.1.2 Circuit AY3-8910

Le circuit AY3-8910 est un synthétiseur sonore capable de restituer 3 voix (8 octaves) plus un générateur de bruit. Il possède en plus 2 ports d'E/S à usage général. Chaque port peut être configuré soit en entrée, soit en sortie.

Il occupe 2 cases mémoires ($B018-$B020) :

- ADDR ($B020)= adresse du registre visé
- REG ($B018)= valeur du registre

ADR	Fonction	D7	D6	D5	D4	D3	D2	D1	D0
$00	Générateur voie A	Fréq. (8 bits poids faibles)							
$01						Fréq. (4 bits poids forts)			
$02	Générateur voie B	Fréq. (8 bits poids faibles)							
$03						Fréq. (4 bits poids forts)			
$04	Générateur voie C	Fréq. (8 bits poids faibles)							
$05						Fréq. (4 bits poids forts)			
$06	Générateur de bruit					5 bits fréquence bruit			
$07	Réglage mixeur	In/Out		Bruit			Tonalité		
		IOB	IOA	C	B	A	C	B	A
$08	Volume Voie A					M	L3	L2	L1
$09	Volume Voie B					M	L3	L2	L1
$0A	Volume Voie C					M	L3	L2	L1
$0B	Fréquence enveloppe	8 bits poids faible							
$0C		8 bits poids forts							
$0D	Forme de l 'enveloppe					CONT	ATT	ALT	HOLD
$0E	IO port B	8 bits Port B							
$0F	IO port A	8 bits Port A							

Figure 6: Les registres du générateur sonore

4.1.3 Circuit ROM

La ROM de 512Ko est en accès indirect, c'est à dire que le système ne peut lire qu'une seule case mémoire à la fois.

La commande de cette mémoire occupe 4 cases mémoires ($B038-$B03B) :

- ADRU, ADRH, ADRL ($B038-$B03A)= adresse mémoire (0-524287 soit 512Ko). ADRU=poids fort, ADRL=poids faible.
- DATA ($B03B)= contenu de la case mémoire visée.

Remarque : Tous les registres d'adresse sont en écriture seule. Il n'est donc pas possible de lire leurs contenus.

La mémoire EPROM (MYM1) contient des musiques de l'AMSTRAD CPC au format MYM (les N° sont exprimés en Hexadécimal).

N°	Titre	N°	Titre	N°	Titre	N°	Titre
00	Le 5ème AXE	0C	Zombie	18	Marche à l'ombre	23	Shadow of the beast
01	Antiriad	0D	Druid	19	Passager du temps	24	Saboteur 2
02	Captain' Blood	0E	Exolon	1A	Passager du vent 1	25	H,A,T,E
03	Cauldron I	0F	Jet set willy	1B	Passager du vent 3	26	Deliverance
04	Cauldron II	10	Krakout	1C	Passager du vent 8	27	Amaurote
05	Equinox	11	Metro cross	1D	Target renegade	28	Glider rider
06	Gyroscope	12	Nemesis	1E	Robocop	29	Boulder dash
07	Ikari Warrior	13	Sapiens	1F	Hotshot	2A	Gunfright
08	James Bond	14	Prehistorik	20	Galivan	2B	M'enfin
09	Barbarians	15	The last V8	21	Crafton & Xunk	2C	Starfighter
0A	Benedict (démo)	16	Rambo	22	Mag max	2D	1943
0B	Deflektor	17	Prohition				

Tableau 2: N° des musiques de l'EPROM MYM1

4.1.4 Fonctions pour l'utilisateur

Pour simplifier la gestion des différents périphériques, la ROM du moniteur intègre un certain nombre de fonctions de contrôle des éléments de la platine d'extension 1.

INITSND	$F300	Aucun	Initialise le circuit AY3-8910
MUTE	$F317	Aucun	Coupe le son
VOLUME	$F329	A=volume (0..15)	Définit le volume pour les 3 voies
PLAY_A	$F33D	X=fréquence (0..4095)	Joue un son sur la voie A
PLAY_B	$F35B	X=fréquence (0..4095)	Joue un son sur la voie B
PLAY_C	$F361	X=fréquence (0..4095)	Joue un son sur la voie C
MODEIO	$F374	A : b7=IOB, b6=IOA 0=out, 1=in	Définit le mode des IOs (IN/OUT)
RDAY_A	$F396	A=valeur lue	Lecture PORT IOA
WRAY_A	$F39F	A=valeur à écrire	Écriture PORT IOA
RDAY_B	$F3AC	A=valeur lue	Lecture PORT IOB
WRAY_B	$F3B5	A=valeur à écrire	Écriture PORT IOA
ROMRD	$F3C2	X=addr, A=up addr	Lecture ROM indirecte [@AX] dans A
PLAYMYMR	$F405	B=N° musique	Lecture d'une musique MYM en ROM
PLAYSONGS	$F690	Aucun	Lecture de toutes les musiques de la ROM
PLAYSONGSN	$F691	B=N° 1ère musique	Lecture des musiques à partir du N° B

Tableau 3: Fonctions de contrôle des éléments de la platine d'extension 1

4.2 Platine d'extension 2

La platine d'extension 2 propose :

- (1) un PIA 6821 : 16 E/S. Les ports A et B sont connectés à 2 barrettes de connexion noires à gauche de la platine. 2 bargraphs, placés juste à côté, indiquent l'état des E/S (allumé=1, éteint=0). Les sorties CBx sont utilisées ainsi que PB7 pour réaliser un contrôleur I²C.
- (2) une EEPROM I²C (24C04 à 24C512)=mémoire non volatile
- (3) un ADC/DAC I²C (0-5V) 8 bits PCF8591
- (4) une NVRAM/RTC I²C PCF8583=horloge temps réel sauvegardée par batterie.

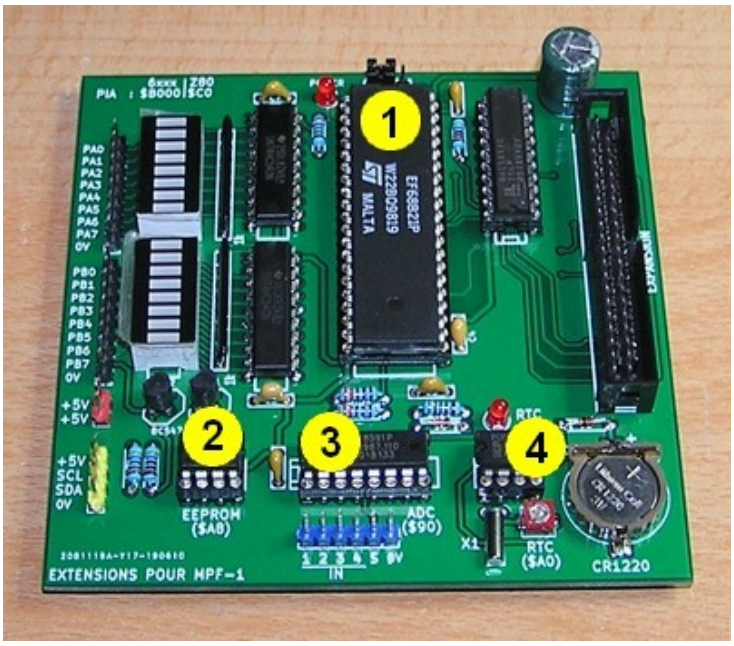


Figure 7: Platine d'extension 2

4.3 Implantation mémoire détaillée

La **Figure 8** donne le mapping mémoire détaillé de l'extension 2.

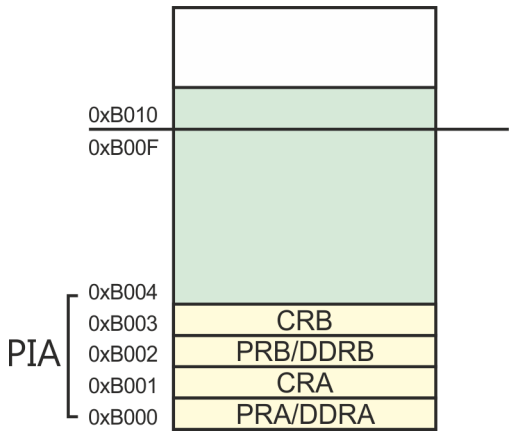


Figure 8: Implantation mémoire de l'extension 2

4.3.1 Circuit PIA (6821)

Le PIA est un circuit d'entrées sorties numériques. Il possède 2 ports 8 bits configurables bit à bit en entrée ou en sortie ainsi que 4 signaux de handshake pour l'échange de données avec d'autres périphériques.

Seuls les 2 ports d'E/S du 6821 sont disponibles sur les barrettes de connexion de la platine. Les signaux CAx et CBx ne sont pas disponibles. Les signaux CBx (+PB7) sont utilisés pour commander un contrôleur logiciel de bus I²C.

Le PIA dispose de 6 registres accessibles à 4 adresses mémoire.

PRA/PRB ($x0,$x2)= valeurs présentes sur les PORTA et PORTB

DDRA/DDRB ($x0,$x2)= registres de direction des ports A et B (bit à 0=entrée, bit à 1=sortie)

CRA/CRB ($x1,$x3)= registres de contrôle

Les registre PRx et DDRx partagent les mêmes adresses. C'est le bit 2 des registres de contrôles qui détermine lequel est accessible.

Adresses	Bit de contrôle		Registres
	CRA.b2	CRB.b2	
$B000	1	x	PRA
$B000	0	x	DDRA
$B001	x	x	CRA
$B002	x	1	PRB
$B002	x	0	DDRB
$B003	x	x	CRB

Tableau 4: Accès aux registres du PIA

Exemple : Pour définir le PORTA en sortie et le PORTB en entrée, il suffit de faire :

CRA=0 ; DDRA=$FF ; CRA=4 ;

CRB=0 ; DDRB=0 ; CRB=4 :

4.3.2 Fonctions pour l'utilisateur

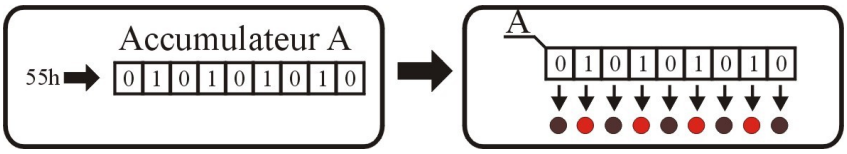
Pour simplifier la gestion des différents périphériques, la ROM du moniteur intègre un certain nombre de fonctions de contrôle des éléments de la platine d'extension 2.

INITI2C	$F700	Aucun	Initialise le bus I2C
START	$F756	Aucun	Condition START
STOP	$F75F	Aucun	Condition STOP
WRITEI2C	$F767	A=valeur	Émission un octet
READI2C	$F789	A=valeur	Réception d'un octet
RDACK	$F7A7	Aucun	Acquittement
RDNACK	$F7B2	Aucun	Non acquittement
WRT_RTC	$F7DF	X=adr	Écrit la zone depuis X (16o) dans RTC
RDT_RTC	$F7F9	X=adr	Lis la zone RTC (16o) vers X
WRM_RTC	$F825	X=adr	Écrit la zone depuis X dans RAM RTC
RDM_RTC	$F831	X=adr	Lis la zone RAM RTC vers X
WR_EE	$F845	A=val, X=adr	Écrit une valeur à l'adresse X
RD_EE	$F854	A=val, X=adr	Lis une valeur à l'adresse X
WR_EEP	$F86F	X=adr EE, Y=adr RAM	RAM @ Y dans EEPROM @ X (128o)
RD_EEP	$F87F	X=adr EE, Y=adr RAM	EEPROM @ X dans RAM @ Y (128o)

Tableau 5: Fonctions de contrôle des éléments de la platine d'extension 2

5 Exercices d'entraînement

5.1 Exercice 1 – Instructions de lecture et d'écriture



Addr	Hex code	Label	Instruction
0200			org $200
0200	86 55	main	lda #$55
0202	B7 80 00		sta $8000
0205	3F		swi
			end

L'accumulateur A est chargé avec la valeur $55. Puis il est écrit dans la case mémoire d'adresse $8000. Cette adresse correspond aux 8 leds GPIO placées à droite des afficheurs. L'instruction SWI permet de retourner au moniteur à la fin du programme.

Entrez ce programme à partir de l'adresse $200. Pour cela, appuyez sur la touche ADDR pour obtenir l'affichage de l'adresse en cours et la données contenue à cette adresse. Les 4 points décimaux de la partie adresse sont allumés pour indiquer que l'adresse peut être modifiée en appuyant sur les touches numériques.

Ajuster l'adresse si nécessaire puis appuyez sur la touche DATA pour passer en mode édition de données (les 2 points décimaux de la partie données sont allumés).

Entrez la 1ère valeur de la colonne *Hex code* puis appuyez sur la touche + pour passer à l'adresse suivante. En cas d'erreur de saisie, il suffit de saisir à nouveau la valeur pour écraser l'ancienne. Les touches + et – permettent de se déplacer dans la mémoire.

Une fois la saisie terminée, entrez à nouveau l'adresse $200 à l'aide de la touche ADDR puis appuyez sur la touche GO pour lancer votre programme.

Que se passe-t-il ?

Modifiez la valeur placée dans A de $55 à $AA. Comment faire ?

Relancez votre programme. Testez avec différentes valeurs. Que montre l'affichage de GPIO ?

5.2 Exercice 2 – Écrire des valeurs en mémoire

```
Addr   Hex code      Label   Instruction
0200                       org $200
0200   86 00        main    lda #$00
0202   1F 8B                tfr a,dp
0204   86 55                lda #$55
0206   B7 80 00              sta $8000
0209   97 40                sta $40
020B   B7 60 00              sta $6000
020E   3F                   swi
                            end
```

L'accumulateur A est chargé avec la valeur 0, puis cette valeur est transférée dans le registre de page DP. Ce registre permet de définir l'adresse haute (8 bits de poids fort) de la 'page 0'.

L'accumulateur A est ensuite chargé avec la valeur 0x55 puis cette valeur est écrite dans la mémoire aux adresses $8000, $40 et $6000.

Pour écrire dans la mémoire aux adresses $8000 et $6000, il est fait appel à un adressage absolu (extended) qui nécessite 3 octets (le code de l'instruction suivi de l'adresse sur 2 octets) : B7 80 00

Pour écrire à l'adresse $40, il est fait appel à l'adressage en 'page 0'. Ce mode d'adressage utilise le registre de page DP pour définir les 8 bits de poids fort de l'adresse si bien que l'instruction n'a besoin que de 2 octets (le code de l'instruction suivi des 8 bits de poids faible de l'adresse) : 97 40. En contrepartie, ce mode d'adressage ne peut adresser que 256 octets (une page).

Pour tester ce programme, il faut d'abord regarder les valeurs stockées à ces adresses avant exécution. Utilisez la touche ADDR pour entrer les adresses afin de lire leurs contenus.

Adresse	Contenu avant exécution	Contenu après exécution
$40		
$6000		
$8000		

Entrez le programme dans la mémoire puis exécutez le. Vérifiez ensuite le contenu des 3 cases mémoire. Conclusion ?

Modifiez le contenu de DP (par exemple $10) et lancez à nouveau votre programme. Regardez le contenu de la case mémoire $1040.

5.3 Exercice 3 – Écrire des valeurs en mémoire en utilisant un registre d'index

```
Addr    Hex code        Label   Instruction
0200                            org $200
0200    86 33           main    lda #$33
0202    C6 50                   ldb #$50
0204    8E 10 00                ldx #$1000      ;adresse de base
0207    A7 84                   sta ,x
0209    A7 01                   sta 1,x
020B    A7 02                   sta 2,x
020D    A7 03                   sta 3,x
020F    A7 1F                   sta -1,x
0211    A7 88 7F                sta $7F,x
0214    A7 89 10 00             sta $1000,x
0218    A7 85                   sta b,x
021A    3F                      swi
                                end
```

L'accumulateur A est chargé avec la valeur 0x33 tandis que l'accumulateur B est chargé avec la valeur $50. Puis la valeur de A est écrite en mémoire en utilisant l'adressage indexé associé au registre d'index X.

Entrez ce programme sans l'exécuter puis calculez les adresses de chaque case mémoire visée et regardez leurs contenus avant exécution.

Instruction	Adresse effective	Contenu avant	Contenu après
sta ,x	$1000		
sta 1,x			
sta 2,x			
sta 3,x			
sta $7f,x			
sta $1000,x			
sta b,x			

Lancez votre programme puis regardez le contenu de chaque case mémoire. Modifiez votre programme pour utiliser la valeur $44 dans l'accumulateur et utiliser le registre d'index Y au lieu du registre d'index X. Pour ajouter du code au milieu d'un programme, il suffit de se placer à une adresse N puis d'appuyer sur la touche INS pour libérer un octet situé à l'adresse N+1 tout en déplaçant le bloc des 512 octets suivants vers le haut.

5.4 Exercice 4 – Écrire des valeurs en mémoire en utilisant l'adressage indirect

```
Addr    Hex code        Label   Instruction
0200                            org $200
0200    A6 9F C0 00     main    lda [$C000]
0204    B7 80 00                sta $8000       ; GPIO
0207    3F                      swi
                                end
```

L'accumulateur A est chargé avec la valeur contenue dans la case mémoire dont l'adresse est stockée en $C000. Cette valeur est ensuite envoyée sur les leds du kit (GPIO)

Quelle est l'adresse de l'emplacement mémoire contenant la valeur placée dans A ?

Entrez ce programme puis exécuter le. Que se passe-t-il ?

Modifiez le contenu de l'emplacement mémoire précédent puis relancez votre programme.

5.5 Exercice 5 – Point d'arrêt et inspection des registres

Addr	Hex code	Label	Instruction
0200			org $200
0200	86 12	main	lda #$12
0202	C6 45		ldb #$45
0204	1E 89		exg a,b
0206	3F		swi
			end

L'accumulateur A est chargé avec la valeur $12 tandis que l'accumulateur B est chargé avec la valeur $45 puis leur contenu est échangé.

Entrez ce programme en mémoire et placez vous à l'adresse $0204 puis appuyez sur la touche BRKPT (Breakpoint=point d'arrêt). Retournez à l'adresse $200 et lancez votre programme.

Le programme s'est arrêté en $204 au niveau de l'instruction exg a,b qui n'a pas encore été exécutée.

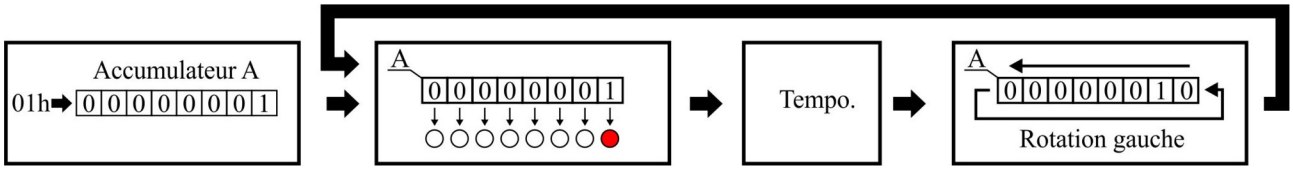
Appuyez sur la touche REG puis la touche 0 (registre A) afin de contrôler le contenu des registres A et B. Appuyez ensuite sur la touche ADDR pour revenir au mode normal puis terminez l'exécution du programme en appuyant sur GO. Vérifiez à présent le contenu de A et B.

Relancez votre programme depuis $200. Lorsqu'il est arrêté en $204, appuyez sur la touche REG puis 0 (registre A). Appuyez maintenant sur la touche DATA. Vous pouvez à présent modifier le contenu du registre, par exemple $36. Pour retourner au mode visualisation, il suffit d'appuyer à nouveau sur la touche DATA. Terminez l'exécution du programme puis vérifiez le nouveau contenu des registres.

Pour supprimer le point d'arrêt, il suffit de retourner à l'adresse à laquelle il est placé puis d'appuyer sur la touche BRKPT. Si vous avez oublié son emplacement, il suffit d'appuyer 2 fois sur la touche BRKPT.

Remarque : il n'est possible de définir qu'un seul point d'arrêt à la fois.

5.6 Exercice 6 – Temporisation utilisant un registre d'index



L'accumulateur A est chargé avec la valeur $01 puis cette valeur est écrite sur les leds (GPIO). Le programme appelle ensuite une temporisation et la valeur dans l'accumulateur subit une rotation.

Addr	Hex code	Label	Instruction
0200			org $200
0200	86 01	main	lda #$01
0202	B7 80 00		sta $8000
0205	8D **03**		bsr tempo
0207	49		rola
0208	20 **xx**		bra loop
020A	8E 30 00	tempo	ld x,#$3000
020D	30 1F	tmp1	leax -1,x
020F	26 **xx**		bne tmp1
0211	39		rts
			end

Pour pouvoir entrer ce programme en mémoire, il faut compléter les 2 valeurs manquantes. Elles correspondent aux offsets des instructions de branchement relatif. (BSR= appel à un sous-programme, BRA :branchement inconditionnel (goto), BNE : branchement si non nul).

La valeur se calcule à partir de la valeur de PC juste après l'instruction de branchement et l'adresse de destination : offset=destination-PC.

Pour faciliter ce calcul, la touche CAL propose une petite calculatrice destinée à cet usage.

Il faut d'abord appuyer sur CAL puis entrer l'adresse de destination ensuite appuyer sur la touche – puis entrer la valeur de PC juste après l'instruction et enfin appuyer sur GO pour obtenir le résultat.

Pour l'instruction BSR, l'offset vaut $20A - $207 = $03

Calculez les 2 autres offsets puis entrez ce programme en mémoire. Les instructions utilisées sont limitées à un branchement court, c'est à dire que la valeur de l'offset n'est codée que sur un octet. Le saut peut donc aller de -126 à +127. La calculatrice fournit une valeur sur 16 bits donc il ne faut prendre que l'octet de poids faible.

Lancez votre programme. Pour l'arrêter, il faut appuyer sur la touche RESET. Le programme n'est pas effacé durant cette opération.

Comment changer la vitesse de défilement ?

Comment changer le sens de défilement ?

5.7 Exercice 7 – Utilisation des interruptions (générateur d'impulsions 10ms)

```
Addr  Hex code     Label        Instruction              
0200                           org $200                 
0200  86 7E        main         lda #$7E                 
0202  B7 7F F0                 sta $7FF0                
0205  8E 30 00                 ldx #$3000               ; int_service
0208  BF 7F F1                 stx $7FF1                
020B  0F 00                    clr $00                  
020D  0F 01                    clr $01                  
020F  1C EF                    andcc #%11101111         Autorise IRQ
0211  20 FE                    bra $0211                Boucle infinie

3000                           org $3000                
3000  0C 00        int_service  inc $00                 
3002  96 00                    lda $00                  
3004  81 64                    cmpa #100                
3006  26 09                    bne skip                 
3008  0F 00                    clr $00                  
300A  0C 01                    inc $01                  
300C  96 01                    lda $01                  
300E  B7 80 00                 sta $8000                
3011  3B          skip         rti                      
                               end
```

La 1ère partie du programme détourne les interruptions vers le programme situé en $3000 puis initialise 2 cases mémoire ($00 et $01) à 0. Elles servent de variables à notre programme ($00 compte le temps tandis que $01 stocke la valeur envoyée aux leds).

La 2ème partie constitue le programme d'interruption. Ce programme est associé à un signal reçu sur la patte IRQ du microprocesseur. Le kit 6809 possède un générateur d'interruption qui envoie une impulsion (requête d'interruption) toutes les 10ms. Pour l'activer il suffit de placer l'interrupteur sur INT.

Dans ce programme, la variable d'adresse $00 est incrémentée jusqu'à atteindre la valeur 100 (100x10ms=1s). Lorsque cette valeur est atteinte, la variable d'adresse $01 est incrémentée puis sa valeur est envoyée sur les leds.

Comment peut-on augmenter la fréquence de comptage à 10Hz ?

5.8 Exercice 8 – Utilisation des interruptions (générateur d'impulsions 10ms)

```
Addr  Hex code   Label  Instruction          
0200              org $200
0200  0F 01      main   clr $01
0202  1C EF             andcc #%11101111
0204  B6 7F FF   loop   lda $7FFF           ;TICK
0207  81 64             cmpa #100
0209  26 F9             bne loop
020B  7F 7F FF          clr $7FFF
020E  96 01             lda $01
0210  8B 01             adda #$01
0212  19                daa
0213  97 01             sta $01
0215  B7 80 00          sta $8000          ;GPIO
0218  20 EA             bra loop
                        end
```

On peut simplifier le programme précédent en utilisant la variable système TICK. Après le RESET, l'interruption IRQ est automatiquement dirigée vers une routine de gestion de la variable temporelle TICK. Toutefois, l'interruption IRQ reste masquée.
Pour activer le comptage de cette variable, il suffit d'activer l'interruption IRQ pour qu'elle soit incrémentée à chaque appel IRQ (10ms). Le programme va ensuite lire cette variable et s'en servir comme base de temps.

Tapez ce programme puis exécutez le. Qu'est ce qui change par rapport au programme précédent ?

5.9 Exercice 9 – Affichage d'un message sur l'afficheur LCD

Pour cet exercice, il faut ajouter un afficheur LCD sur la platine. Ce programme permet d'afficher du texte sur la 1ère ligne de l'écran LCD. Pour cela, il fait appel aux fonctions du Tableau 1. La fonction INITLCD permet d'initialiser l'afficheur suivie de la fonction LPUTS pour afficher la chaîne de caractères placée à l'adresse contenue dans X (la chaîne doit se terminer par un 0).

```
Addr   Hex code          Label   Instruction
0200                           org $200
0200   BD F0 00         main    jsr $F000          ; INITLCD
0203   8E 03 00                 ldx #text1
0206   BD F0 51                 jsr $F051          ; LPUTS
0209   3F                       swi

0300   48 65 6C 6C      text1   fcc ‘‘Hello world’’
       6F 20 77 6F
       72 6C 64
030B   00                       fcb 0
                                end
```

Testez ce programme puis ajoutez du texte dans la chaîne de caractère (n'oubliez pas le 0 à la fin). Relancez votre programme. Que s'est-il passé ?

En fait la 2ème ligne commence à la position 64, il faut donc déplacer le curseur pour y accéder. C'est le rôle de la fonction GOTOXY qui a été ajoutée dans la suite du programme (ci-dessous).

```
Addr   Hex code          Label   Instruction
0209   CC 03 01                 ldd #0301          ; A=3, B=1
020C   BD F0 62                 jsr $F062          ; GOTOXY
020F   8E 03 0C                 ldx #text2
0212   BD F0 51                 jsr $F051          ; LPUTS
0215   3F                       swi

0300   48 65 6C 6C 6F  text1   fcc ‘‘Hello world’’
       20 77 6F 72 6C
       64
030B   00                       fcb 0
030C   49 20 61 6D 20  text2   fcc ‘‘I am 6809 CPU’’
       36 38 30 39 20
       43 50 55
0319   00                       fcb 0
                                end
```

Complétez votre programme puis testez le à nouveau. Résultat ?

5.10 Exercice 10 – Chenillard 16 voies

Cet exercice utilise le PIA de l'extension 2. Le but est de créer un chenillard faisant défiler une led allumée sur les 16 disponibles dans les sens haut vers bas.

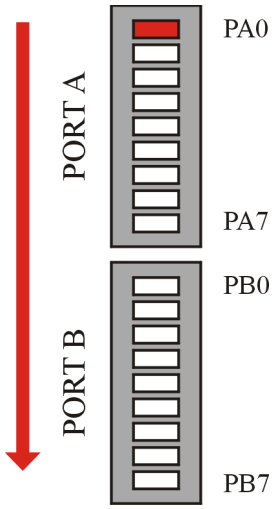


Figure 9: Chenillard 16 voies

La première étape consiste à définir les 2 ports A et B en sortie. Pour cela, il faut initialiser DDRA et DDRB à $FF. Ensuite, il faut programmer la séquence de défilement.

```
Addr    Hex code    Label    Instruction
0200                         org $200
0200    86 00       main     lda #$00
0202    B7 B0 01             sta $B001          ; Accès DDRx
0205    B7 B0 03             sta $B003
0208    86 FF                lda #$FF
020A    B7 B0 00             sta $B000
020D    B7 B0 02             sta $B002
0210    86 04                lda #$04
0212    B7 B0 01             sta $B001          ; Accès PORTx
0215    B7 B0 03             sta $B003
0218                         ...
```

5.11 Exercice 11 – Orgue électronique

Cet exercice nécessite l'extension 1, Il utilise le synthétiseur AY3-8910 pour transformer le clavier du kit en instrument de musique permettant de jouer des notes simples du DO^2 au SI^5.

	2	3	4	5
DO	131	262	523	1046
	$01DE			
RE	147	294	587	1175
	$01A9			
MI	165	330	659	1318
	$017B			
FA	175	349	698	1397
	$0166			
SOL	196	392	784	1568
	$013F			
LA	220	440	880	1760
	$011C			
SI	247	494	988	1975
	$00FD			

← Octave
← Fréq. (Hz)
← Valeur TP

Tableau 6: Fréquences des notes

La fréquence de sortie du circuit synthétiseur est obtenue par la formule $F_S=F_{CLK}/16.TP$ où FCLK est la fréquence d'horloge (1MHz) et TP la valeur de commande sur 12 bits. Complétez ce tableau avec les valeurs TP au format hexadécimal (ce sera plus facile pour la suite).

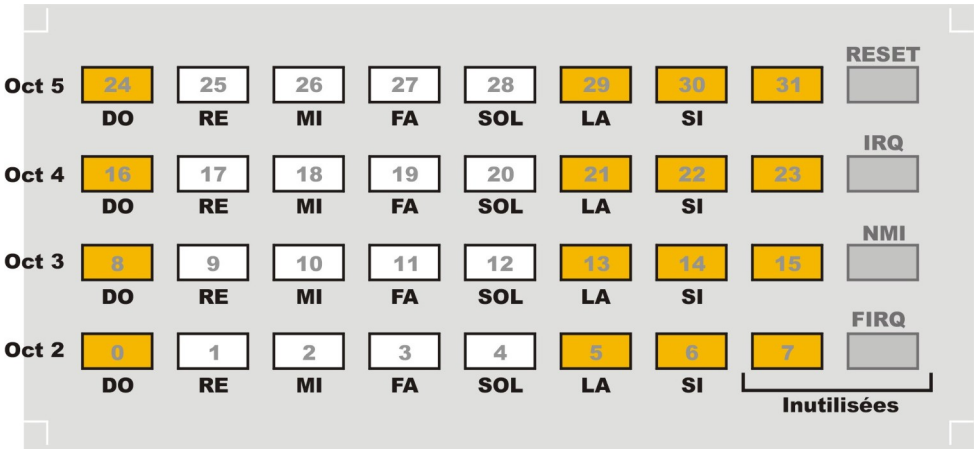


Figure 10: Affectation des notes au clavier du kit

La fonction de lecture du clavier renvoie un nombre N indiquant sur quelle touche l'utilisateur a appuyé. Si aucune touche n'est appuyée, la fonction retourne 255 ($FF).

Pour traduire la touche appuyée en note, il peut être pratique d'utiliser une table associée à un adressage indexé afin de simplifier le code.
Chaque note est définie par une valeur TP codée sur 12bits (2 octets) donc si X contient l'adresse du début de la table, il suffit de lire les 2 octets placés aux adresses X+2*N et X+2*N+1.
Attention, il reste 1 colonne inutilisée à droite correspondant aux codes N=7, 15, 23 et 31. Cela n'est pas très grave car il suffit d'ajouter un 0 dans la table pour associer cette touche à un silence.
Au final, on obtient le programme suivant :

```
Addr    Hex code        Label       Instruction
0200                                org $200
0200    BD F3 00        main        jsr $F300           ; INITSND
0203    86 0F                       lda #$0F
0205    BD F3 29                    jsr $F329           ; VOLUME
0208    BD F0 5A        loop        jsr $F05A           ; DELAY2
020B    BD F1 BE                    jsr $F1BE           ; SCANKB
020E    B7 80 00                    sta $8000
0211    2B 0C                       bmi note_off
0213    10 8E 02 30                 ldy #table
0217    48                          asla
0218    AE A6                       ldx a,y
021A    BD F3 3D                    jsr $F33D           ; PLAY_A
021D    20 E9                       bra loop
021F    8E 00 00        note_off    ldx #$0000
0222    BD F3 3D                    jsr $F33D           ; PLAY_A
0225    20 E1                       bra loop

0230    01 DE 01 A9     table       fcb $01DE, $01A9, $017B, $0166,
        01 7B 01 66
                                    ….
                                    end
```

Entrez ce programme puis complétez la table en n'oubliant pas d'ajouter un zéro après le SI de chaque octave. Testez votre programme.

5.12 Exercice 11 – Divertissement musical

Le kit 6809 peut aussi jouer de la musique en exploitant les 4 canaux du synthétiseur. Pour cela, il faut produire un flux de données alimentant le synthétiseur en continu. Cet exercice nécessite l'extension 1.

Les ordinateurs familiaux des années 1980-90 utilisaient massivement le synthétiseur AY3-8910 et on trouve aujourd'hui un grand nombre de fichiers musicaux au format YM. Ce format regroupe les 14 registres du circuits échantillonnés à une fréquence de 50 Hz. Ces fichiers sont compressés afin de prendre moins de place.

Ce format ne peut pas être pris en compte directement par le kit 6809, il faut d'abord le convertir au format MYM qui permet une décompression des données à la volée lors de la lecture.

Le programme permettant de lire les données se trouve dans la ROM du moniteur tandis que les données musicales sont stockées dans la ROM de l'extension. La lecture d'un morceau consiste à indiquer au programme le N° du morceau puis il suffit d'appeler la routine. Pour que le programme fonctionne, il faut placer l'interrupteur d'interruption sur INT.

```
Addr    Hex code    Label    Instruction
0200                         org $200
0200    C6 29                ldb #$29          ; BOULDER DASH
0202    BD F4 05             jsr $F405         ; PLAYMYMR
0205    3F                   swi
```