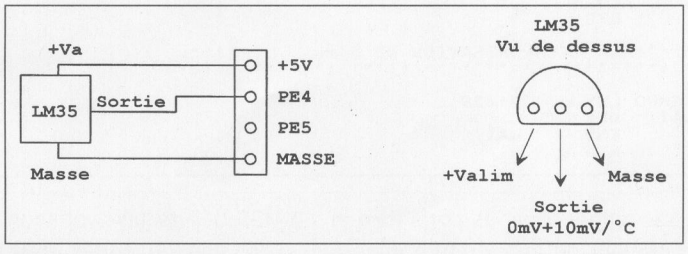
Figure 14

```
*************************************************************
******   Programme de gestion du convertisseur 8 bits  ******
******     lecture du résultat sur AD1 et affichage    ******
*************************************************************

PORTB          EQU $1004       (le PORTB est a l'adresse $1004)
OPTION         EQU $1039
ADCTL          EQU $1030
AN0            EQU $1031
DIGIT          EQU $10         (variable DIGIT a l'adresse $10)
PILE           EQU $C0

               ORG $00      (La table commence a l'adresse $00)

      FCB      $3F,$06,$5B,$4F,$66,$6D,$7D,$07
      FCB      $7F,$6F,$77,$7C,$39,$5E,$79,$71

      ORG $20                (Début du programme: adresse $20)

      LDS      #PILE
      LDAA     #%10000000
      STAA     OPTION
      LDAA     #%00100000
      STAA     ADCTL
SUITE LDAA     AN0
      STAA     DIGIT
      LDAB     DIGIT
      ANDB     #%00001111
      BSR      TRANS
      STAA     PORTB
      BSR      TEMPO
      LDAB     DIGIT
      LSRB
      LSRB
      LSRB
      LSRB
      BSR      TRANS
      ORA      #%10000000
      STAA     PORTB
      BSR      TEMPO
      BRA      SUITE

****** TRANSCODAGE HEXA/7 SEGMENTS ******
*****************************************

TRANS LDX      #$0000
      ABX
      LDAA     ,X
      RTS

******     TEMPORISATION DE 5 ms     ******
*****************************************

TEMPO LDY      #1428
WAIT  DEY
      BNE      WAIT
      RTS
```

Figure 15 : câblage du capteur de température sur le kit



Ecrire dans les registres AD1 à AD4 n'a aucun effet sur leur contenu. Il est possible d'accéder à un résultat avant la fin d'un cycle complet à condition de respecter le timing de la figure 11 (par exemple, le résultat de AD1 est accessible 33 cycles après une écriture dans le registre ADCTL).

Adresses des résultats :

AD1 : [$1031] AD2 : [$1032]
AD3 : [$1033] AD4 : [$1034]

UTILISATION DU CONVERTISSEUR SUR UNE APPLICATION SIMPLE

Pour débuter avec ce périphérique, nous allons utiliser le potentiomètre P1 dont le curseur est câblé sur l'entrée PE0 (AN0) du port E.

Le câblage externe permet d'envoyer sur AN0 une tension comprise entre 0 et 2,5 V. Après conversion, le résultat sera lu sur AD1, qui pourra donc prendre une valeur comprise entre $00 et $FF suivant la position du curseur de P1. On va initialiser le convertisseur dans le mode «conversions continues» sur l'entrée AN0, afin de visualiser le résultat sur les afficheurs. On en déduit l'organigramme très simple de la figure 13. Dans l'exemple d'application proposée, on commence par une lecture du résultat avant la fin de la première conversion. Ce premier résultat sera donc faux au démarrage du programme, mais il sera remplacé dès la deuxième boucle (au bout de 10 ms) par la bonne valeur. Le rebouclage perpétuel permet de prendre en compte toutes les 10 ms la position du curseur de P1.

Le listing correspondant est indiqué en figure 14, qu'il vous suffit de tester sur le kit selon la procédure habituelle. On remarque, au début du listing, la zone de déclaration des variables.

Cette démarche permet de n'utiliser ensuite que des noms de variable, bien plus explicites que des adresses. On arrive ensuite à la zone d'initialisation du système: après avoir initialisé la pile à l'adresse $C0, on alimente le convertisseur en mettant le bit D7 du registre OPTION à 1, puis on définit dans le registre ADCTL le mode de fonctionnement du convertisseur. Le bit D5 (SCAN) de ce registre étant positionné à 1, le convertisseur commence une série de conversions ininterrompues. Il ne reste plus qu'à lire le résultat dans AD1 et lancer la procédure d'affichage multiplexée.

Figure 16 :
Thermomètre Numérique :
conversion sur AN4 et affichage décimal

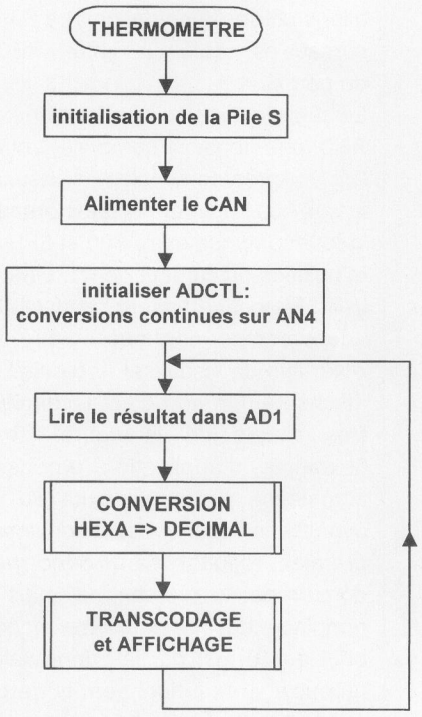


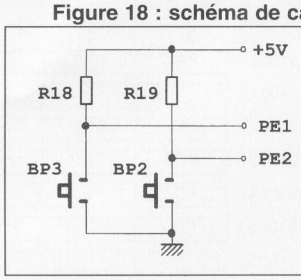
Figure 17

```
*************************************************************
******      Thermomètre 0°C à 99°C avec sonde LM35      ******
*************************************************************

PORTB          EQU $1004       (le PORTB est a l'adresse $1004)
OPTION         EQU $1039
ADCTL          EQU $1030
AN0            EQU $1031
DIGIT          EQU $10         (variable DIGIT a l'adresse $10)
PILE           EQU $C0

       ORG $00              (La table commence a l'adresse $00)

       FCB     $3F,$06,$5B,$4F,$66,$6D,$7D,$07
       FCB     $7F,$6F,$77,$7C,$39,$5E,$79,$71

       ORG $20        (Début du programme: adresse $20)

       LDS     #PILE
       LDAA    #%10000000
       STAA    OPTION
       LDAA    #%00100100
       STAA    ADCTL
SUITE  LDAA    AN0
       STAA    DIGIT
       BSR     HEDEC
       LDAB    DIGIT
       ANDB    #%00001111
       BSR     TRANS
       STAA    PORTB
       BSR     TEMPO
       LDAB    DIGIT
       LSRB
       LSRB
       LSRB
       LSRB
       BSR     TRANS
       ORA     #%10000000
       STAA    PORTB
       BSR     TEMPO
       BRA     SUITE

******   CONVERSION HEXA/DECIMAL   ******
*****************************************

HEDEC  CLRA
       LDAB    DIGIT
       LDX     #10
       IDIV
       STAB    DIGIT
       XGDX
       LSLB
       LSLB
       LSLB
       LSLB
       ORB     DIGIT
       STAB    DIGIT
       RTS

****** TRANSCODAGE HEXA/7 SEGMENTS ******
*****************************************

TRANS  LDX     #$0000
       ABX
       LDAA    ,X
       RTS

******    TEMPORISATION DE 5 ms    ******
*****************************************

TEMPO  LDY     #1428
WAIT   DEY
       BNE     WAIT
       RTS
```

MESURE DE TEMPÉRATURE

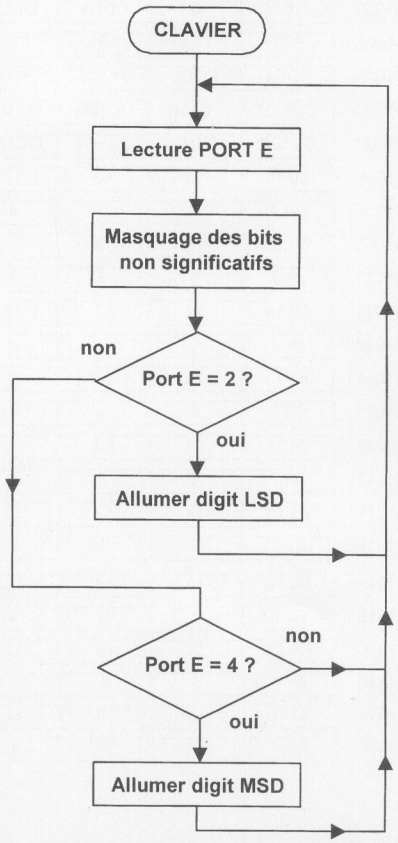
Pour en finir avec ce convertisseur, nous allons exploiter le connecteur de 4 broches installé sur le kit, près du 68HC11 : il supporte la masse, l'alimentation 5 V du kit et deux entrées analogiques dirigées sur les ports PE4 et PE5. Un capteur de température intégré comme le LM35 peut donc être directement câblé sur ce connecteur, sans nécessiter le moindre composant supplémentaire. Le schéma particulièrement simple de la figure 15 va nous permettre de réaliser un petit thermomètre ambiant limité toutefois à l'intervalle [0°..+99°]. Le LM35 délivre une variation de 10mV/°C, la tension de sortie étant nulle à 0°C. Pour notre plage d'utilisation de 0°C à 99°C, la tension délivrée par le capteur variera entre 0 V et 990 mV. Puisque la plage de conversion est définie à 2,5 V par VRH, un pas de conversion sera exactement de 2,5 V / 255 = 9,8 mV. On peut donc considérer que un pas de

Figure 18 : schéma de câblage des contacts de touche



BP3	BP2	PE1	PE2
repos	repos	1	1
repos	actif	1	0
actif	repos	0	1
actif	actif	0	0

Figure 19 :
lecture du clavier à deux touches



conversion correspond (approximativement) à 1°C. L'erreur totale est de 2 %. Ce qui signifie que pour l'affichage de 99°, la température réelle est en fait de 101°C. Pour obtenir une meilleure précision, il suffirait de régler VRH à 2,55 V. Dans ce cas, on obtiendrait exactement un pas de conversion de 10mV, soit 1 LSB par degré. Il reste encore un détail à régler : le résultat de mesure fourni par le convertisseur est exprimé en hexadécimal, alors que l'utilisateur doit lire une température en base décimale. Notre programme devra donc prévoir une procédure de conversion hexa/décimal avant d'envoyer le résultat de conversion aux afficheurs. L'organigramme répondant à notre application est indiqué en figure 16. C'est pratiquement le même programme que pour la mesure de tension aux bornes du potentiomètre, avec les deux modifications suivantes: l'acquisition est effectuée sur AN4 (PE4), et la procédure de conversion Hexa/décimal est intercalée entre la lecture du résultat sur AD1 et l'appel de la routine de transcodage Hexa/7 segments (qui reste valable en décimal).

PRINCIPE DE LA CONVERSION HEXA => DÉCIMAL

Une valeur hexadécimale représente la même quantité d'unités que son équivalent en base décimale : seul le mode de représentation change. Pour passer de la base hexadécimale à la base décimale, il suffit de diviser la valeur d'origine par 10 : le résultat donnera la part des dizaines, le reste la part des unités en base 10. Le listing de la figure 17 met en application ce principe, comme on peut le constater dans le sous-programme de conversion. La fonction **IDIV** du 68HC11 assure la division du contenu de l'accumulateur D par le contenu du registre d'index X. Le résultat de la division est placé dans X, et son reste dans D. D est un registre de 16 bits, et on travaille uniquement avec des nombres sur huit bits : en conséquence, le chargement de D sera effectué à partir de B, qui est au format 8 bits et représente le poids faible de D. On procédera de la même façon pour la lecture du résultat, qui sera récupéré dans B. L'algorithme de division est le suivant :

1. Chargement de B avec le contenu de AD1, mise à zéro de A (poids fort de D)

2. Chargement de X avec la valeur 10

3. Exécution de la division (IDIV)

4. Stockage du contenu de B dans DIGIT (c'est l'unité)

5. **XGDX** = Echange entre X et D (pour récupérer le résultat de division dans B)

6. Exécuter 4 décalages à gauche de B (pour placer la dizaine sur le quartet de poids fort)

7. Assembler les parties «unité» et «dizaine» par une opération logique «OU»

8. Stocker le résultat final dans la variable DIGIT.

Voilà ! Ce n'est pas très simple à comprendre du premier coup, mais ça marche très bien, du moins jusqu'à 99. En effet, à partir de 100, il faudrait effectuer une division supplémentaire par 10 pour séparer le chiffre des dizaines avec celui des centaines. Cependant, dans tous les cas, le principe de conversion restera identique. Le test de ce programme peut être également effectué à partir de l'entrée AN0, sur laquelle est câblé le potentiomètre. Dans ce cas, au-dessus de 99, l'affichage sera erroné. C'est peut-être aussi l'occasion de compléter le sous-programme de conversion !. Encore une remarque : si la conversion est effectuée à partir de l'entrée **AN4** pour la mesure de température, il faut positionner le bit **CC** du registre **ADCTL** à 1 (à vérifier dans le tableau de la figure 12). Si vous préférez utiliser l'entrée **AN5,** il faut positionner **CC** et **CA** à 1.

Figure 20

```
*****************************
***  gestion de clavier   ***
*****************************

PORTB  EQU  $1004
PORTE  EQU  $100A

       ORG  $00

DEBUT  LDAA     PORTE
       ANDA     #%00000110
       CMPA     #%00000010
       BNE      TEST2
       LDAB     #$7F
       STAB     PORTB
       BRA      DEBUT
TEST2  CMPA     #%00000100
       BNE      DEBUT
       LDAB     #$FF
       STAB     PORTB
       BRA      DEBUT
```

GESTION DE CLAVIER

Nous terminerons cette session par la prise en compte des deux touches câblées sur le port E du 68HC11, qui fonctionne également avec des signaux logiques. On ne peut pas vraiment considérer ces deux touches comme un clavier à part entière : nous aurons l'occasion de gérer un vrai clavier matriciel ultérieurement. Les touches qui nous intéressent aujourd'hui sont reliées directement aux entrées PE1 et PE2, selon le câblage indiqué en figure 18. Un registre implanté à l'adresse $100A permet d'y accéder par une instruction de lecture . Pour cette application, contentons nous d'un programme de test du clavier assez simple : si la touche de droite est enfoncée, on allume l'afficheur de droite, et si on actionne la touche gauche, c'est l'afficheur de gauche qui doit s'allumer. Au repos, remarquons sur le schéma que chaque touche envoie un niveau haut sur le port E. Après la lecture du port E, on force à zéro les bits qui ne nous intéressent pas.
La configuration obtenue en fonction de l'état des touches est indiquée ci-dessous :

au repos : % 00000110 ($06)
droite : % 00000010 ($02)
gauche : % 00000100 ($04)

On compare ensuite ce résultat avec les valeurs qu'on doit obtenir lorsque l'une ou l'autre des touches est enfoncée :

- Si le résultat est 2 (bouton poussoir de droite activé), on allume l'afficheur des unités en envoyant la combinaison $7F sur le port B.

- Si le résultat est 4 (bouton poussoir de gauche activé), on allume l'afficheur des dizaines en envoyant la combinaison $FF sur le port B.

On en déduit l'organigramme de la figure 19. Le listing de cette routine, qui se passe de commentaires, est fourni en figure 20.

NOTRE PROCHAIN RENDEZ-VOUS...

Voici un bref aperçu des festivités qui vont suivre.
Nous présenterons les interruptions du 68HC11, qui permettront d'augmenter la souplesse et la puissance de nos programmes. Nous en profiterons pour réaliser la gestion de l'affichage multiplexé sous interruption, qui sera vraiment plus pratique à utiliser. Même remarque pour la gestion de clavier. Ensuite, il sera temps de présenter le TIMER du 68HC11, qui nous permettra d'obtenir des temporisations très précises. Il pourra être associé à des mesures de périodes, de fréquences, ou simplement à la réalisation de signaux logiques calibrés. Enfin, nous implanterons nos programmes en EEPROM, afin de réaliser des applications entièrement autonomes.

à suivre...

Bernard Dalstein

	Bit 7	6	5	4
$1000	PA7	PA6	PA5	PA4
$1001				
$1002	STAF	STAI	CWOM	HNDS
$1003	PC7	PC6	PC5	PC4
$1004	PB7	PB6	PB5	PB4
$1005	PCL7	PCL6	PCL5	PCL4
$1006				
$1007	DDC7	DDC6	DDC5	DDC4
$1008	0	0	PD5	PD4
$1009	0	0	DDD5	DDD4
$100A	PE7	PE6	PE5	PE4
$100B	FOC1	FOC2	FOC3	FOC4
$100C	OC1M7	OC1M6	OC1M5	OC1M4
$100D	OC1D7	OC1D6	OC1D5	OC1D4
$100E	Bit 15	14	13	12
$100F	Bit 7	6	5	4
$1010	Bit 15	14	13	12
$1011	Bit 7	6	5	4
$1012	Bit 15	14	13	12
$1013	Bit 7	6	5	4
$1014	Bit 15	14	13	12
$1015	Bit 7	6	5	4
$1016	Bit 15	14	13	12
$1017	Bit 7	6	5	4
$1018	Bit 15	14	13	12
$1019	Bit 7	6	5	4
$101A	Bit 15	14	13	12
$101B	Bit 7	6	5	4
$101C	Bit 15	14	13	12
$101D	Bit 7	6	5	4
$101E	Bit 15	14	13	12
$101F	Bit 7	6	5	4

Annexe 1 (1)

Dans notre précédent numéro, en page 25, la colonne de gauche a été «décapitée» lors de la mise en page. Nous republions donc une partie de l'annexe 1 (1) que vous pourrez, si vous le souhaitez, coller sur la page 25 du N° 145 afin d'avoir toutes les données.