SYSTÈMES MICROPROCESSEURS ET INTERFACES
GEL-16383 – SECTION A
SESSION HIVER 2000

Projet :
Rapport final

Remis à: Dr U.S. Ganguly

Par: Sébastien Demers
Stéphane Drouin
Simon Dufour
Martin Gagnon
Yannick Laroche
Pascal Marchand
François Ménard
Marc Villemure

TABLE DES MATIÈRES

DESCRIPTION

DESCRIPTION GÉNÉRALE

Ce projet est réalisé dans le cadre du cours "Système Microprocesseurs et Interfaces", GIF-16383, donné par le Dr. Udaya S. Ganguly lors de la session d'Hiver 2000 à l'Université Laval, département de Génie Informatique.

Le but d'un tel projet est de s'initier à l'interfaçage de matériel et de logiciel, en particulier l'utilisation de périphériques. Plus spécifiquement, notre projet, désigné sous le nom de CALMAT 64, consiste en la réalisation d'une « calculatrice matricielle » basée sur un micro-contrôleur. La calculatrice permettra de réaliser différentes opérations sur des matrices de nombres réels.

MODÈLE GÉNÉRAL

La calculatrice permet de faire différentes opérations mathématiques classiques sur des matrices de nombres réels. L'implantation s'est réalisée à l'aide de composants standards.

Entrées

Un clavier permet l'entrée des nombres ainsi que leur manipulation.

Ce clavier est de type matriciel, avec 16 touches (4 x 4). Il s'agit du même type de clavier que ceux employé en téléphonie.

Sorties

Les résultats sont affichés sur un écran à cristaux liquides.

Nous avons employé un écran de 2 lignes à 16 caractères, type affichage de caractères (par exemple basé sur un contrôleur Hitachi 44780).

Traitements

Les calculs et l'ensemble des opérations sont réalisés et contrôlés par un microprocesseur Motorola 68HC16.

Ce processeur a l'avantage de permettre les calculs en virgule flottante de manière native et de bonne précision (16 bits), ainsi que de permettre le développement à l'aide du langage C. De plus le développement a été facilité par sa compatibilité avec le MC68HC11, pour lequel il existe de nombreux outils de développement (entre autres des simulateurs logiciels facilitant le partage du travail).

APPROCHE DE DESIGN

Cette calculatrice est opérée de manière « post-fixée ». Elle emploie donc une pile des valeurs employées, de même que quelques registres pour faciliter les calculs.

Le clavier permet l'entrée des nombres, l'appel de fonctions ainsi que les autres opérations nécessaires.

L'écran permet l'affichage des valeurs complète. En ce sens il est possible de faire défiler une « grande valeur » (une matrice) à l'écran (« scroll »)

Le développement a été réalisé en décomposant le problème en ses constituants essentiels, lesquels ont été assemblés suivant les besoins.

CARACTÉRISTIQUES

Nous avons identifié 2 jeux de caractéristiques : *essentielles*, *visées*.

Les *caractéristiques essentielles* forment l'implantation minimale de notre projet et ont été l'objet de la première intégration (voir l'échéancier dans la section « Plan de travail »).

- Opérations de base, scalaire (addition, soustraction, multiplication, division);
- Génération de matrices élémentaires (Nulle, Un, Identité);
- Lecture du clavier pour touche simple;
- Affichage avec défilement de valeur.

Les *caractéristiques visées* forment le produit que nous voulions livrer à la fin de ce projet.

- Opérations de base, **matricielles** (addition, soustraction, multiplication, division);
- Lecture du clavier pour touches composées.

DÉMARCHE

Nous avons divisé les composantes de ce projet dans les groupes *Théorie*, *Logiciel*, *Matériel*, lesquels sont assemblés sous le groupe « *Système et intégration* ». Nous avons formé des équipes pour réaliser les différentes activités nécessaires au succès du projet.

Afin de nous aider à travailler en groupes indépendants tout en partageant les résultats et devis communs nous avons monté deux sites informatiques. Le premier est un site FTP pour stocker les fichiers de travail. Le second est un site Web pour publier aisément nos documents de travail.

Nous avons visé à ce que chacun des groupes puisse poursuivre le projet de manière relativement indépendante (bien qu'il y ait un ordre dans l'ordre des résultats à obtenir). Pour chacun des groupes nous avons nommé un responsable parmi les membres. Cette personne devait voir à faire respecter l'échéancier établi ainsi qu'à coordonner les activités des membres à l'intérieur du groupe ainsi qu'entre les groupes.

GROUPES

Système/Intégration

Le Système concerne l'aspect « haut niveau » du projet, et en particulier il s'agit de l'agencement des différentes composantes. Quant à l'Intégration il s'agit d'assembler effectivement ces composantes et de maintenir les résultats de test.

Coordonnateur: Simon Dufour

Théorie

Le groupe Théorie a établi les méthodes de calculs de chacune des fonctions à implanter, les calculs de base requis pour l'implantation de ces fonctions, de même que les analyses de complexité des calculs (traitement, taille mémoire) et les impacts de précision dans les calculs réalisés.

Coordonnateur: Marc Villemure

Groupe : Sébastien Demers, Martin Gagnon, Pascal Marchand, aidés de François Ménard.

Logiciel

Ce groupe s'est occupé du design et de l'implantation des logiciels de la calculatrice dans le microcontrôleur.

Coordonnateur : Yannick Laroche

Groupe : François Ménard, Stéphane Drouin, aidés de Martin Gagnon, Marc Villemure, Pascal Marchand.

Matériel

Ce groupe a vu aux aspects matériels du projet, leur programmation d'interactivité et le design et l'implantation de l'électronique de support.

Coordonnateur : Simon Dufour

Aidé par Stéphane Drouin, Sébastien Demers.

DÉPENDANCES DES ÉLÉMENTS DU PROJET

Ce tableau liste les dépendances entre les groupes, en particulier au niveau des résultats.

	PROVIENT DE				ATTENDU PAR			
ITEM	**SYS**	**THE**	**LOG**	**MAT**	**SYS**	**THE**	**LOG**	**MAT**
Spécifications du projet	S				S	T	L	M
Métriques des opérations mathématiques		T					L	M
Test de compatibilité HC16 vs HC11				M			L	
Capacité de calcul flottant-IEEE sur HC16				M		T	L	
Usage des interruptions avec HC16 (clavier)				M			L	
Opération autonome de Dunfield sans le Freedom16				M			L	
Développement en C pour le HC16	S						L	M
Opération correcte d'écran 128x128			M		S			M

PLAN DE TRAVAIL

Note: Les lettres entre crochets ("[" et "]") désignent les parties suivantes:
S = Système : le système dans son entier, l'intégration des parties.
T = Théorie : les bases théoriques pour le traitement mathématique ou numérique.
L = Logiciel : les aspects programmés.
M = Matériel : les aspects physiquement câblés.

DATE	ANALYSE	DESIGN	IMPLANTATION	INTÉGRATION	RÉSULTATS
2000/03/07-Ma	Énoncé des fonctionnalités [S]				
2000/03/09 -Je		Sélection et validation des traitements [T] Définition des interfaces [S]	Sélection du matériel [M] Sélection de l'environnement de développement [S]		
2000/03/14 -	Adoption des fonctionnalités	Adoption des	Adoption du		SPÉCIFICATIONS

Ma	[S]	traitements [T] Adoption des interfaces [S]	matériel [M] Adoption de l'environnement de développement [S]		[S] INTERFACES [S]
2000/03/16 -Je		Design et prototypage [L] Design [M]	Implantation des procédures de traitement [T]		
2000/03/21 - Ma			Présentation des prototypes logiciels [L] Présentation des prototypes matériels [M]		TRANSFORMATIONS [S]
2000/03/23 -Je			Développement et codage [L] Développement [M]		
2000/03/28 - Ma					
2000/03/30 -Je					
2000/04/04- Ma					
2000/04/06 -Je					
2000/04/11 - Ma					
2000/04/13 -Je				Intégration logicielle [L] Intégration matérielle [M]	
2000/04/18 - Ma				Intégration logicielle/matérielle [S]	BILAN DE QUALITÉ [S]
2000/04/20 -Je					
2000/04/25 - Ma	* * * REMISE FINALE * * * REMISE FINALE * * * REMISE FINALE * * *				RAPPORT FINAL [S]

THÉORIE

Les éléments principaux de la théorie ont été l'analyse des opérations que notre calculatrice devait effectuer, l'élaboration d'algorithme, ainsi qu'une estimation du temps de calcul nécessaire.

OPÉRATIONS SCALAIRES (MULTIPLICATION ET DIVISION)

Définition : Les opérations scalaires appliquées sur les matrices consistent en une série d'opérations singulières effectuées sur chacun des éléments de la matrice en cause vis-à-vis d'un élément semblable.

Forme : **A** [op] x = **B** , où [op] est l'opération * ou /, et x est un scalaire.

Matrices :

Opérande :

$$\mathbf{A} = \begin{bmatrix} A_{0,0} & \cdots & A_{0,n} \\ \vdots & \ddots & \vdots \\ A_{m,0} & \cdots & A_{m,n} \end{bmatrix}$$

Réponse :

$$\mathbf{B} = \begin{bmatrix} B_{0,0} & \cdots & B_{0,n} \\ \vdots & \ddots & \vdots \\ B_{m,0} & \cdots & B_{m,n} \end{bmatrix}$$

Scalaire : x

Algorithme :

```
for (j = 0; j < n; j++) {
        for (i = 0; i < m; i++) {
            B_{i,j} = A_{i,j} [op] x
        }
    }
```

Complexité :

La complexité due aux nombres d'opérations est exactement égale au nombre d'éléments n*m dans la matrice opérande, donc linéaire.

$T = \Theta (m*n)$

Restrictions :

Les restrictions sur les paramètres n et m (lignes et colonnes) sont à déterminer dépendamment de l'espace mémoire disponible. Il en sera de même pour toutes les opérations.

L'ordre de grandeur du scalaire doit être tel que la multiplication de celui-ci avec l'élément de la matrice donne un résultat sur 32 bits.

Note : Nous pourrions aussi ajouter les [op] + et -, si elles s'avéraient utiles.

OPÉRATIONS MATRICIELLES (ADDITION ET SOUSTRACTION)

Forme : **A** [op] **B** = **C** , où [op] est l'opération + ou -.

Matrices (considérant qu'elles sont de dimensions égales) :

1ere opérande :

$$\mathbf{A} = \begin{bmatrix} A_{0,0} & \cdots & A_{0,n} \\ \vdots & \ddots & \vdots \\ A_{m,0} & \cdots & A_{m,n} \end{bmatrix}$$

2e opérande :

$$\mathbf{B} = \begin{bmatrix} B_{0,0} & \cdots & B_{0,n} \\ \vdots & \ddots & \vdots \\ B_{m,0} & \cdots & B_{m,n} \end{bmatrix}$$

Réponse :

$$\mathbf{C} = \begin{bmatrix} C_{0,0} & \cdots & C_{0,n} \\ \vdots & \ddots & \vdots \\ C_{m,0} & \cdots & C_{m,n} \end{bmatrix}$$

Algorithme :

```
if ((nbre_lignes(A) != nbre_lignes(B)) || (nbre_colonnes(A) != nbre_colonnes(B))) {
        message d'erreur
}
else {
        for (j = 0; j < n; j++) {
                for (i = 0; i < m; i++) {
                        C_{i,j} = A_{i,j} [op] B_{i,j}
                }
        }
}
```

Complexité :

Ici aussi, la complexité due aux nombres d'opérations est exactement égale au nombre d'éléments n * m dans les matrices opérandes, donc linéaire.

$T = \Theta (m*n)$

OPÉRATION MATRICIELLE (MULTIPLICATION)

Forme : **A** * **B** = **C**

Matrices (considérant qu'elles sont de dimensions compatibles) :

1ere opérande :

$$\mathbf{A} = \begin{bmatrix} A_{0,0} & \cdots & A_{0,n} \\ \vdots & \ddots & \vdots \\ A_{m,0} & \cdots & A_{m,n} \end{bmatrix}$$

2e opérande :

$$\mathbf{B} = \begin{bmatrix} B_{0,0} & \cdots & B_{0,p} \\ \vdots & \ddots & \vdots \\ B_{n,0} & \cdots & B_{n,p} \end{bmatrix}$$

Réponse :

$$\mathbf{C} = \begin{bmatrix} C_{0,0} & \cdots & C_{0,p} \\ \vdots & \ddots & \vdots \\ C_{m,0} & \cdots & C_{m,p} \end{bmatrix}$$

Algorithme :

```
if (nbre_colonnes(A) != nbre_lignes(B)) {
        message d'erreur
}
else {
        for (j = 0; j < p; j++) {
                for (i = 0; i < m; i++) {
                        C_{i,j} = 0
                        for (k = 0; k < n; k++) {
                                C_{i,j} += A_{i,k} * B_{k,j}
                        }
                }
        }
}
```

Complexité :

La complexité de cet algorithme se calcul de la façon suivante :

$$\text{Nombre d' opérations} = Q = \frac{(m * n) * (n * p)}{n}$$

$T = \Theta (Q)$

TRANSPOSITION DE MATRICE

Forme : $\mathbf{A}^T = \mathbf{B}$

Matrices :

Opérande :

$$\mathbf{A} = \begin{bmatrix} A_{0,0} & \cdots & A_{0,n} \\ \vdots & \ddots & \vdots \\ A_{m,0} & \cdots & A_{m,n} \end{bmatrix}$$

Réponse :

$$\mathbf{B} = \begin{bmatrix} B_{0,0} & \cdots & B_{0,m} \\ \vdots & \ddots & \vdots \\ B_{n,0} & \cdots & B_{n,m} \end{bmatrix}$$

Algorithme :

```
for (j = 0; j < n; j++) {
        for (i = 0; i < m; i++) {
                B_{i,j} = A_{j,i}
        }
}
```

Complexité :

Comme il n'y a aucune opération mathématique à effectuer, on ne peut pas vraiment parler de complexité tel que vu précédemment dans ce cas-ci. On peut néanmoins mentionner que l'algorithme effectuera m*n affectations avant de se terminer.

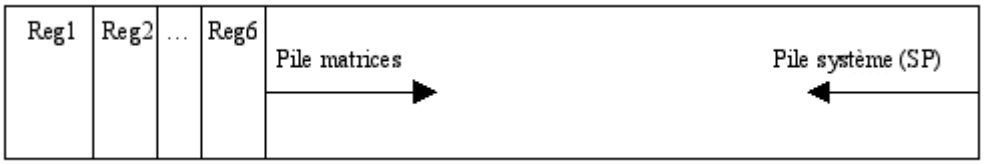
LOGICIEL

Cette section présente les spécifications et fonctions utilisées pour la partie programmée du projet.

SPÉCIFICATIONS

Une première liste de points à respecter est la suivante :

- Étant donné qu'il existe du code déjà fait pour des nombres en point flottant sur 32 bits, tous nos nombres seront représentés sur 32 bits.
- Nous limitons la dimension des matrices à N x N selon la mémoire dont nous disposerons. De façon pratique, 16 x 16 semble suffisant.
- Les registres sont placés à une extrémité de la mémoire, et les piles (matrices et système) vont en sens inverses l'une de l'autre. Voici de quoi pourrait avoir l'air la mémoire :



- Gestion de pile des matrices (opérandes) :
  - On place d'abords le contenu d'une matrice sur la pile.
  - On place ensuite les dimensions de cette matrice sur la pile. On pourra ainsi récupérer les dimensions de la matrice en premier.
- Les modifications des nombres sur la pile des opérandes directement est permise à condition de ne pas modifier les dimensions de la matrice.
- Affichage :
  - Ce qui est entré au clavier est copié sur la pile et dans un buffer pour l'affichage.
  - Pour afficher les résultats, on copie la zone mémoire des résultats dans la zone mémoire de l'affichage.
  - On doit pouvoir afficher des messages d'erreur en plus des nombres des matrices.
- Opérations :
  - Les opérations sont de type " post-fixe " i.e. : chaque opération utilise les deux opérandes sur le dessus de la pile opérandes.
  - Les registres peuvent être utilisés par l'usager selon son bon vouloir.
  - À la suite d'une opération, les opérandes utilisées disparaissent de la pile et le résultat calculé est placé sur cette pile.
- Les 32 premiers bits d'une matrice servent à sauvegarder la dimension (nombre de rangées et de colonnes) et la position (rangée et colonne actuelles), la disposition des bits étant la suivante :
  - 00 à 07 : nombre de rangées

- 08 à 15 : nombre de colonnes
- 16 à 23 : position en x (# rangée)
- 24 à 32 : position en y (# colonne)

FONCTIONS

Les fonctions suivantes ont été implantées dans le projet.

STORE_MATRIX_FROM_STACK_TO_REGISTER
IN : # du registre, la dernière matrice sur la pile
OUT : matrice dans un registre
DESC : prend la dernière matrice de la pile et la copie dans un registre

STORE_MATRIX_FROM_REGISTER_TO_STACK
IN : # du registre
OUT : matrice sur la pile
DESC : prend la matrice d'un registre et la copie sur la pile

STORE_MATRIX_FROM_KEYBOARD_TO_STACK
IN : clavier
OUT : matrice sur la pile
DESC : lie une matrice au clavier et la copie sur la pile

STORE_IDENTITY_MATRIX_ON_STACK
IN : nombre de lignes / colonnes
OUT : matrice sur la pile
DESC : ajoute une matrice identité sur la pile

STORE_0S_MATRIX_ON_STACK
IN : nombre de lignes, nombres de colonnes
OUT : matrice sur la pile
DESC : ajoute une matrice de zéros sur la pile

STORE_1S_MATRIX_ON_STACK
IN : nombre de lignes, nombres de colonnes
OUT : matrice sur la pile
DESC : ajoute une matrice de uns sur la pile

MODIFY_MATRIX_ON_STACK
IN : # ligne, # colonne, nouvelle valeur
OUT : -
DESC : modifie la matrice sur la pile

DELETE_ONE_MATRIX_ON_STACK
IN : -
OUT : -
DESC : enlève la dernière matrice de la pile

DELETE_ALL_MATRICES_ON_STACK
IN : -
OUT : -
DESC : reset le pointeur de la pile

CLEAR_ONE_REGISTER
IN : # du registre
OUT : -
DESC : efface la dimension de la matrice du registre

CLEAR_ALL_REGISTERS
IN : -
OUT : -
DESC : efface les dimensions des matrices des registres

ADD_MATRICES
IN : les deux dernières matrices sur la pile
OUT : une matrice sur la pile
DESC : additionne les deux dernières matrices sur la pile et les remplacent par le résultat (addition).

SUB_MATRICES
IN : les deux dernières matrices sur la pile
OUT : une matrice sur la pile
DESC : soustrait les deux dernières matrices sur la pile et les remplacent par le résultat (soustraction).

MULTIPLY_MATRICES
IN : les deux dernières matrices sur la pile
OUT : une matrice sur la pile
DESC : multiplie les deux dernières matrices sur la pile et les remplacent par le résultat (multiplication).

TRANSPOSE_MATRIX
IN : la dernière matrice sur la pile
OUT : une matrice sur la pile
DESC : transpose la dernière matrice sur la pile et la remplace par le résultat (transposé).

MULTIPLY_MATRIX_BY_SCALAR
IN : la dernière matrice sur la pile
OUT : une matrice sur la pile
DESC : multiplie la dernière matrice par un scalaire et la remplace par le résultat.

DIVIDE_MATRIX_BY_SCALAR
IN : la dernière matrice sur la pile
OUT : une matrice sur la pile
DESC : divise la dernière matrice par un scalaire et la remplace par le résultat.

READ_KEYBOARD
IN : clavier
OUT : touche appuyée
DESC : lie une touche au clavier et retourne sa valeur

WRITE_SCREEN
IN : valeur, position, message
OUT : affichage
DESC : écrit à l'écran la valeur et la position et le message

MOVE_LEFT
IN : -
OUT : valeur, nouvelle position, message
DESC : retourne la valeur a gauche de l'élément actuel, sauve la nouvelle position. Si la position est non-valide, rester à la position actuelle.

MOVE_RIGHT
IN : -
OUT : valeur, nouvelle position, message
DESC : retourne la valeur a droite de l'élément actuel, sauve la nouvelle position. Si la position est non-valide, rester à la position actuelle.

MOVE_DOWN
IN : -
OUT : valeur, nouvelle position, message
DESC : retourne la valeur au-dessous de l'élément actuel, sauve la nouvelle position. Si la position est non-valide, rester à la position actuelle.

MOVE_UP
IN : -
OUT : valeur, nouvelle position, message
DESC : retourne la valeur au-dessus de l'élément actuel, sauve la nouvelle position. Si la position est non-valide, rester à la position actuelle.

IS_STACK_EMPTY
IN : -
OUT : oui / non
DESC : dit s'il reste des éléments dans la pile

READ_DIM
IN : Un pointeur sur un espace mémoire de 8 bits
OUT : Retourne 1 si la lecture est un succès (le nombre est valide), 0 sinon.
DESC : Lire une dimension (char: 0..255)

READ_NUMBER
IN : Un pointeur sur un espace mémoire de 32 bits. Une touche initiale (CL_0 à CL_9, CL_POINT, CL_EXP, CL_SIGN ou CL_NULL)
OUT : Retourne 1 si la lecture est un succès (le nombre est valide), 0 sinon.
DESC : Compléter la lecture d'un nb 32 bits

PUT_KEYBOARD

IN : Le numéro de touche logique à placer dans le buffer

OUT : -

DESC :Placer une touche logique ds le buffer clavier. Ne fait rien si buffer plein.

READ_KEYBOARD

IN : -

OUT : Le code logique d'une touche ou CL_NULL s i le buffer est vide.

DESC : Lire une touche logique ds le buffer clavier

M_PUSH

IN : Nombre de lignes (rangées) de la matrice, nombre de colonnes de la matrice, pointeur au début de la matrice à piler (premier nombre, position (1,1))

OUT : retourne 1 si push avec succès et 0 s'il n'y a pas assez d'espace sur la pile.

DESC : Copie la matrice et ses dimensions sur la pile et place le pointeur de pile après la nouvelle matrice.

M_POP

IN : -

OUT : Tous les pointeurs dans la zone mémoire de la pile : pointeur sur le nombre de ligne de la matrice, pointeur sur le nombre de colonnes de la matrice, pointeur sur la position ligne courante de la matrice, pointeur sur la position colonne courante de la matrice, pointeur au début de la matrice dans la pile (premier nombre). Retourne 1 si pop avec succès et 0 si la pile est vide (rien à dépiler).

DESC : Retourne un pointeur sur la matrice *dans* la pile. Place le pointeur de pile avant la matrice dépilée.

M_TOP

IN : -

OUT : Tous les pointeurs dans la zone mémoire de la pile : pointeur sur le nombre de ligne de la matrice, pointeur sur le nombre de colonnes de la matrice, pointeur sur la position ligne courante de la matrice, pointeur sur la position colonne courante de la matrice, pointeur au début de la matrice dans la pile (premier nombre). Retourne 1 si pop avec succès et 0 si la pile est vide (rien à dépiler).

DESC : Retourne un pointeur sur la matrice *dans* la pile. Ne modifie pas le pointeur de pile.

MAIN

IN : -

OUT : -

DESC : boucle de lecture du clavier et dispatch des fonctions reliées

ÉTAT DU CALCULATEUR

Pour un bon fonctionnement, le calculateur devra se rappeler dans quel état il est. Ci-dessous est présenté un ensemble de ‘flags’ servant à reconnaître l’état du calculateur.

Bit #	Nom	Description
0	S1	Le Shift1 est actif. Flag activé en appuyant sur la touche Shift1. Flag désactivé en appuyant sur n’importe quelle touche.
1	S2	Le Shift2 est actif. Flag activé en appuyant sur la touche Shift2. Flag désactivé en appuyant sur n’importe quelle touche.
2	M1	Le Menu1 est actif. Flag activé en appuyant sur la touche Shift1, puis sur la touche 7. Flag désactivé en appuyant sur n’importe quelle touche.
3	M2	Le Menu2 est actif. Flag activé en appuyant sur la touche Shift2, puis sur la touche 7. Flag désactivé en appuyant sur n’importe quelle touche.
4	AF	On est en mode affichage (affichage de la matrice sur le dessus de la pile des opérandes). Flag activé par la fin correcte de toutes les fonctions de calcul, par les fonctions/touches Top, Identité, Zéro, Matrice et Recall.
5	SA	On est en mode saisie (d’un nombre). Flag activé par une touche " nombre " pendant que le flag AF est aussi actif. Flag désactivé par la touche " Enter ".
6	CA	On est en mode calcul. Flag activé par le début d’une routine de calcul. Flag désactivé par la fin de ladite routine. Pourrait être utile pour annuler un long calcul...
7	ERR	Une erreur a été produite et n’a pas été affichée. Flag activé par un la rencontre d’une erreur, au sens large. Flag désactivé par la routine qui traite l’erreur (affichage d’une note, ...).

La combinaison AF/SA permet de saisir facilement une matrice : on commence par créer une matrice de zéros d’une dimension quelconque, on place l’élément courant comme étant l’élément (1,1) et on permet de modifier chaque nombre avec l’activation de SA au besoin.

Le flag CA est optionnel, si on veut pouvoir annuler un calcul. Si ce n'est pas le cas, il n'a pas d'utilité évidente.

Le flag ERR permet de séparer la détection d'une erreur et son traitement. Cela augmente la modularité et facilite le développement. Les flags S1/S2/M1/M2 ne devraient être modifiés que par la fonction de traitement de l'interruption clavier.

Un octet de mémoire est donc suffisant pour conserver cette structure. Cet espace mémoire a été appelé CALFLAG.

UTILISATION D'UN BUFFER POUR LE CLAVIER

(Le lecteur est prié de se référer à la section « Matériel » qui suit pour un diagramme des touches du clavier)

Comme l'utilisateur peut utiliser le clavier en tout temps et ce même pendant les traitements, on introduit la notion d'interruption. On présume que le clavier peut générer une interruption matérielle et que le HC16 saura y répondre. La routine de traitement d'interruption a quelques tâches à accomplir :

1) déterminer quelle touche physique est enfoncée;
2) connaissant l'état du calculateur, faire correspondre une touche logique;
3) placer cette valeur logique dans le buffer clavier qui sera lu par la fonction READ_KEYBOARD, fonction qui serait préférablement non bloquante.

La dimension du buffer est définie arbitrairement, et nous avons présumé qu'une taille équivalente au nombre de chiffres significatifs plus *x* devrait suffire. Un buffer de taille 10 devrait être suffisant pour nos besoins. Quand le buffer est plein, on n'accepte plus de caractères.

TOUCHE PHYSIQUE, TOUCHE LOGIQUE

Dans le logiciel, on s'intéresse aux touches logiques, i.e. " Clear ", " + " ou " 7 ". Ce sont des touches logiques que doit retourner la fonction READ_KEYBOARD. En référence à la section précédente, les touches shift et menu ne produisent pas de touche logique puisqu'elles ne font que changer l'état du calculateur. La liste des touches logiques est présentée ci-dessous, ainsi que les codes correspondants. La colonne " État du calculateur " donne le nombre binaire CALFLAG dans l'ordre ERR/CA/SA/AF/M2/M1/S2/S1. Seuls les 4 derniers sont d'intérêt pour produire une touche logique. Tous les codes logiques commencent par CL_ afin d'en faire des noms de constantes claires. Lors de l'activation de toutes les touches logiques (sauf les 4 premières), les flags M2/M1/S2/S1 sont remis à 0.

Touche physique	État du calculateur	Code logique	Commentaires
Shift1	xxxxxxxx	(Aucune touche)	Toggle le drapeau S1
Shift2	xxxxxxxx	(Aucune touche)	Toggle le drapeau S2
7	xxxx0001	(Aucune touche)	Toggle le drapeau M1
7	xxxx0010	(Aucune touche)	Toggle le drapeau M2
Enter	xxxx0000	CL_ENTER	
7	xxxx0000	CL_7	
8	xxxx0000	CL_8	
9	xxxx0000	CL_9	
Store	xxxx0000	CL_STORE	Copie de pile au registre *x*
4	xxxx0000	CL_4	
5	xxxx0000	CL_5	
6	xxxx0000	CL_6	
CLR/On	xxxx0000	CL_CLR	Reset AF
1	xxxx0000	CL_1	
2	xxxx0000	CL_2	
3	xxxx0000	CL_3	
0	xxxx0000	CL_0	
.	xxxx0000	CL_POINT	
Enter	xxxx0001	CL_MATRICE	
7	xxxx0001	Voir haut du tableau	
8	xxxx0001	CL_UP	
9	xxxx0001	CL_LEFT	
Store	xxxx0001	CL_RECALL	Copie registre *x* à pile
4	xxxx0001	CL_A	
5	xxxx0001	CL_C	
6	xxxx0001	CL_E	
CLR/On	xxxx0001	CL_RESET	Flush la pile, efface les registres
1	xxxx0001	CL_ADD	
2	xxxx0001	CL_MULT	
3	xxxx0001	CL_SIGN	
0	xxxx0001	CL_ZERO	
.	xxxx0001	CL_DELETE	Efface une matrice de la pile
Enter	xxxx0010	(Réservé)	(Spot libre sur le clavier)
7	xxxx0010	Voir haut du tableau	
8	xxxx0010	CL_DOWN	
9	xxxx0010	CL_RIGHT	
Store	xxxx0010	CL_CLEAR	Efface le contenu du registre *x*
4	xxxx0010	CL_B	
5	xxxx0010	CL_D	
6	xxxx0010	CL_F	
CLR/On	xxxx0010	CL_OFF	
1	xxxx0010	CL_SUB	
2	xxxx0010	CL_DIV	
3	xxxx0010	CL_EXP	
0	xxxx0010	CL_IDENTITY	
.	xxxx0010	CL_TOP	Passe en mode affichage
Enter	xxxx0100	(Réservé)	(Fonction menu 1 à définir)

7	xxxx0100	(Réservé)	(Fonction menu 1 à définir)
8	xxxx0100	(Réservé)	(Fonction menu 1 à définir)
9	xxxx0100	(Réservé)	(Fonction menu 1 à définir)
Store	xxxx0100	(Réservé)	(Fonction menu 1 à définir)
4	xxxx0100	(Réservé)	(Fonction menu 1 à définir)
5	xxxx0100	(Réservé)	(Fonction menu 1 à définir)
6	xxxx0100	(Réservé)	(Fonction menu 1 à définir)
CLR/On	xxxx0100	(Aucune touche)	CLR : sortir du menu... ?
1	xxxx0100	(Réservé)	(Fonction menu 1 à définir)
2	xxxx0100	(Réservé)	(Fonction menu 1 à définir)
3	xxxx0100	(Réservé)	(Fonction menu 1 à définir)
0	xxxx0100	(Réservé)	(Fonction menu 1 à définir)
.	xxxx0100	(Réservé)	(Fonction menu 1 à définir)
Enter	xxxx1000	(Réservé)	(Fonction menu 2 à définir)

7	xxxx1000	(Réservé)	(Fonction menu 2 à définir)
8	xxxx1000	(Réservé)	(Fonction menu 2 à définir)
9	xxxx1000	(Réservé)	(Fonction menu 2 à définir)
Store	xxxx1000	(Réservé)	(Fonction menu 2 à définir)
4	xxxx1000	(Réservé)	(Fonction menu 2 à définir)
5	xxxx1000	(Réservé)	(Fonction menu 2 à définir)
6	xxxx1000	(Réservé)	(Fonction menu 2 à définir)
CLR/On	xxxx1000	(Aucune touche)	CLR : sortir du menu... ?
1	xxxx1000	(Réservé)	(Fonction menu 2 à définir)
2	xxxx1000	(Réservé)	(Fonction menu 2 à définir)
3	xxxx1000	(Réservé)	(Fonction menu 2 à définir)
0	xxxx1000	(Réservé)	(Fonction menu 2 à définir)
.	xxxx1000	(Réservé)	(Fonction menu 2 à définir)

CODES D'ERREUR

Dès qu'on rencontre une erreur, on active le flag ERR. Il faut aussi écrire le code de l'erreur rencontrée quelque part en mémoire. Cette zone mémoire a été appelée CALERR. Un byte devrait être suffisant, soit 256 codes d'erreur. 255 si on affecte au code 0 la valeur « tout est correct ». Il n'y a qu'un espace en mémoire, ce qui implique que toute erreur est traitée avant qu'une nouvelle erreur n'apparaisse. À chaque code d'erreur, on associe un traitement d'erreur à faire et/ou un message à afficher. On considère des lignes de 20 caractères, donc des messages dont la longueur ne dépasse pas 20. Voici la liste des codes d'erreur présentement utilisés :

Numéro	Nom	Traitement/Message
0	-	Aucune erreur
	ERR_CALCUL	Pour une raison plus ou moins connue (plutôt moins), une erreur de calcul survient... Ce code est pour se donner une porte de sortie dans un cas qu'on n'aurait pas prévu... On fait un reset de la calculatrice : flush la pile des opérandes et les registres, reset le CALFLAG.
	ERR_DIV0	Division par zéro... Laisser la pile des opérandes dans son état original et afficher " Division par 0 "
	ERR_DIM_INC	Dimensions incompatibles pour le calcul matriciel demandé. Laisser la pile des opérandes dans son état original et afficher " Dim. incompatibles "
	ERR_OPER	Nombre d'opérandes insuffisant. Par exemple, une seule matrice sur la pile des opérandes. Laisser la pile des opérandes dans son état original et afficher : " Opérande manquante "
	ERR_TOUCHE	Une touche incorrecte a été utilisée. Par exemple, une touche non assignée ou une fonction de menu non définie ou lors de la saisie d'un nom de registre. Ne rien faire (ignorer la touche, attendre la suivante).
	ERR_OVERFLOW	Un calcul dépasse la capacité. Laisser la pile des opérandes dans son état original et afficher " Overflow ".
	ERR_DIM	Les dimensions d'une matrice saisie sont incorrectes (trop grande, 0). Refaire la saisie des dimensions.

Dans plusieurs situations, on désire le comportement " Laisser la pile des opérandes dans son état original ". Ceci implique qu'on ne modifie pas la pile des opérandes avant que le calcul ne soit complété et que les erreurs comprenant cette condition ne soient plus possibles.

MATÉRIEL

Ce projet n'a pas demandé de développement de matériel proprement dit car nous avons exploité un kit de développement pour 68HC16 et des composantes discrètes courantes (clavier matriciel et affichage à LCD).
Cette section « Matériel » couvre donc le travail fait pour exploiter ce matériel, i.e. presque essentiellement de la programmation bas niveau.

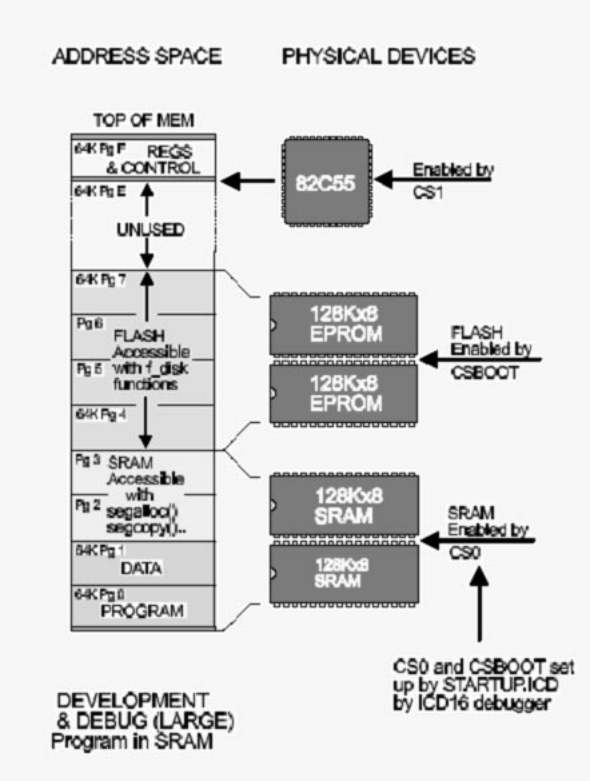
ENVIRONNEMENT DE DÉVELOPPEMENT

La base matérielle employée a été un kit de développement pour 68HC16 de Dunfield Systems, le modèle Freedom16. Les périphériques employés ont été un clavier matriciel et un afficheur à LCD.
La programmation a été faite pour l'essentiel en C, avec quelques parties en assembleur. Nous avons utilisé les bibliothèques de routines disponibles dans l'environnement choisi. Cet environnement était le Dunfield C.
Nous avons tenté d'implanter une bibliothèque permettant le calcul en nombres flottants suivant la norme IEEE. Cette bibliothèque n'est pas dans la version livrée du calculateur. Les détails se trouvent un peu plus loin dans ce rapport.

Kit Freedom16

Il s'agit d'une carte de développement contenant l'électronique de support pour l'interfaçage du micro-contrôleur.

La carte contient la mémoire vive (2 banques de 128ko SRAM) et morte (2 banques de 128ko Flash) ainsi que les « chip select » (basé sur un 8255) de même que d'autres composantes pour la configuration de la carte et les « latch » sur les ports.
La carte supporte aussi la connexion direct au port « EVB » du 68HC16, lequel est branché au PC hôte par un connecteur contenant l'électronique de support.



Dunfield C et bibliothèques

Nous avons utilisé l 'environnement de développement « Dunfield C » en version 2.5. Cet environnement est créé pour le DOS et permet le codage en assembleur ainsi qu'en une version simplifiée du langage C.

La démarche de développement de cet environnement est standard au sens où il s'agit d'écrire une source, de la compiler puis de faire l'édition des liens. Malgré des résultats initiaux prometteurs, nous n'avons pas réussi à faire l'édition de liens de plusieurs sources.
Par ailleurs, cet environnement est plutôt simple : types « int » et «char » seulement, détection d'erreurs minimales au niveau de l'espace de noms, pas de validation des typages au moment de la compilation.

Les librairies de cet environnement semblent destinées au développement rapide d'applications de contrôle de taille petite.
Elles permettent des entrées-sorties sur des périphériques ainsi que sur la console de l'ordinateur hôte. Les fonctions présentes sont des versions minimales des fonctions standard.

Cet environnement demeure malgré tout excellent, quoiqu'un peu vieux.

Bibliothèque de nombres flottants

Nous avons pu obtenir sur l'Internet une bibliothèque permettant le calcul en nombres flottants suivant la norme IEEE.
Cette bibliothèque en assembleur a été originellement conçue pour le 68HC11, mais nous n'avons pas réussi à l'adapter complètement pour le 68HC16 avec un modèle de développement « langage C » à temps pour la livraison du projet.
Au départ du projet nous croyions pouvoir intégrer cette bibliothèque parmi les bibliothèques standard de l'environnement de développement. Malgré des essais initiaux prometteurs, nous n'avons pas réussi à configurer convenablement l'éditeur de lien de l'environnement Dunfield. La solution suivante a été de tenter d'implanter la bibliothèque en tant que partie intégrée à la source de notre projet. Pour une question de temps, ceci n'a pas été un succès.

Il s'agit de la bibliothèque « math11 » créée par Scott Wagner, de Rochester Instrument Systems (1992). Cette bibliothèque est une révision du code original « fp11 » écrit par Gordon Doughman de Motorola (1986).
La source est en assembleur pour 68HC11 et utilise largement des globales pour le stockage des nombres manipulés, ainsi que de nombreuses fonctions de support en plus des fonctionnalités arithmétiques.

Adaptation au 68HC16

Le défi pour l'adaptation au 68HC16 était sur trois plans : conversion des opcodes assembleur, conversion de la logique des globales, conversion de la logique algorithmique.

La conversion des opcodes a été la partie la plus facile puisqu'il y a moins de 10 instructions à remplacer par d'autres instructions fonctionnellement compatibles. Dans notre cas il a suffit de convertir les opcodes suivants : PSHX, INX, INS, PULX, SEC et CLC.

La conversion de la logique des globales a aussi été simple car nous avons choisi de ne pas la convertir mais de plutôt les gérer entre chaque appel en les sauvant sur la pile préalablement à un appel d'une fonction en nombre flottant.
Cette approche n'est pas optimale car une solution meilleure consisterait à simplement passer ces globales par la pile comme il se doit pour une programmation dans la logique du langage C. Nous n'avons pas procédé ainsi car cela demandait une modification de chacune des fonctions employées dans la bibliothèque.
Ce choix de ne pas toucher toutes les fonctions a été gouverné par la contrainte de la logique algorithmique de la bibliothèque.

La conversion algorithmique s'est avéré le plus grand défi, et éventuellement la raison justifiant que la bibliothèque n'est pas intégrée dans la version livrée du calculateur.
La bibliothèque obtenue fait un usage très libre des sauts à travers le code de la bibliothèque, où les segments de code sont appelés sans arrêt d'une fonction à l'autre. Ceci nous a causé un problème lors de la division des routines en « fonctions C ». Le problème n'est pas une difficulté technique mais une question d'ordre de grandeur. La bibliothèque contient 57 points de branchements différents, de même que 6 globales et 9 constantes, nous n'avons pas eu le temps de tout mettre en ordre de manière convenable.

CLAVIER

Voici le clavier tel qu'il a été accepté :

Matrice **Enter**	Menu1 Menu2 **7**	↓ **8**	← → **9**
Recall Clear **Store**	A B **4**	C D **5**	E F **6**
Reset Off **CLR/On**	+ - **1**	× ÷ **2**	± EXP **3**
 Shift1	Zero Identité **0**	Delete Top **.**	 **Shift2**

Dans le projet le clavier s'intègre comme un périphérique virtuel, tel que décrit dans la section Logiciel de ce rapport. Ceci signifie que le code de bas niveau doit réaliser la translation des touches physiques employées en code de touches virtuelles.

Translation des codes

Cette translation est réalisée à l'aide de tableaux de 16 entrées, contenant le code de la touche virtuelle associée à une touche physique. Le bon tableau de translation est choisi suivant la

valeur d'une variable d'état du clavier, laquelle est fixée par la pression de « clefs mortes » sur le clavier.
Ces clefs mortes sont implantées en tant que bascules, où une pression active l'état, et toute pression consécutive d'une touche désactive alors l'état.

Nous avons donc 3 tableaux chacun correspondant à un état du clavier, tel que présenté dans la figure précédente où il y a une couleur de texte différente pour chacun des 3 états.

Lecture

La lecture du clavier est réalisée de manière traditionnelle par parcours des signaux des lignes et colonnes. L'implantation n'a pas utilisé d'interruption et le clavier doit être examiné de manière régulière et cyclique. Le délai est évalué par un simple compteur car notre micro-contrôleur ne nous offrait pas de valeur de temps réel de précision suffisante pour évaluer ce délai (l'horloge en temps réel du kit de développement ne compte que des secondes).
Le clavier a un « debouncing » simple implanté par un délai de relecture de l'état. Les frappes sont détectées suivant le fait que l'interrupteur d'une clef est fermé ou non. Une méthode nettement meilleure serait de détecter plutôt les transitions sur les clefs, i.e. entre les états « ouvert vers fermé » et « fermé vers ouvert ».

Dans une implantation ultérieure il serait préférable d'une part d'examiner les transitions d'état des touches plutôt que l'état lui-même, et d'autre part de gérer le « debouncing » par un délai en temps réel. Il serait aussi très intéressant d'ajouter une capacité d'interruption de manière à permettre l'usage du clavier de manière indépendante du reste de la charge du micro-contrôleur.

AFFICHAGE

Nous examinons ici la question de l'opération de l'affichage tel que vu par l'usager du calculateur, puis l'implantation réalisée pour permettre cette opération.

Opération visée

On dispose d'un écran d'une dimension de 2 lignes de 20 caractères. Cet écran est tout à fait suffisant pour nos besoins. Une façon d'utiliser les deux lignes est de réserver la première ligne pour afficher l'état du calculateur (shift, menu, etc.) et d'utiliser la deuxième ligne pour afficher les valeurs/messages d'erreur et faire l'écho du clavier.

La première ligne
Partant du fait que le nombre de lignes et de colonnes est limité à 255 par les choix faits dans le document " Fonctions de la calculatrice matricielle ", on utilise au maximum 9 caractères pour afficher la position du nombre courant dans une matrice : (iii,jjj). D'autre part, on se réserve deux caractères pour afficher l'état des shifts, un autre caractère pour afficher si la matrice du dessus a été directement entrée sur la pile ou si elle vient d'un registre, auquel cas on affiche le nom de ce registre. Il reste assez de caractères à réserver pour l'affichage du nom du menu courant. Le plan de la première ligne est donc :

I	I	I	I	I	I	I	I	I		R		M	M	M	M	M		S	S

Avec
I : caractères réservés à l'information de matrice
R : caractère réservé à l'affichage d'un nom de registre ou de pile (A-F ou P)
M : caractères réservés à l'affichage du nom du menu courant (MENU1, MENU2)
S : caractères réservés à l'affichage de l'état des shifts

Shift (S)
Les deux derniers caractères de la première ligne sont utilisés pour afficher l'état des shifts. On peut utiliser les petits symboles et les afficher ou non, selon l'état du Shift1 et Shift2, respectivement. Sinon, on peut afficher S1 ou S2 ou rien, selon l'état du calculateur. Pour des raisons de simplicité, cette dernière alternative est utilisée dans le reste du document, mais une ou l'autre peut être utilisée dans la calculatrice.

Information (I)
Il y a trois cas à considérer :
1) On fait la saisie d'une matrice, il faut demander les dimensions. On affiche " lin = " puis " col = ", pour indiquer qu'on fait la saisie du nombre de lignes et du nombre de colonnes.
2) On fait la saisie/modification d'un nombre. Il faut afficher l'indice du nombre dans la matrice ou indiquer qu'il s'agit d'un scalaire. On affiche " (i,j) " ou " Scalaire ", respectivement.
3) On veut utiliser un registre, il faut en saisir le nom. On affiche " Reg A-F "

La deuxième ligne
La deuxième ligne est complètement libre. On l'utilise pour afficher les nombre et les messages d'erreur. Dans le cas de l'utilisation d'un menu, on peut afficher " Choix de l'option ". Cependant, si on a peu d'option par menu (4 ou moins), on utilisera probablement les touches du haut pour choisir ces options. On pourrait alors afficher des flèches au-dessus des touches auxquelles on a affecté des options.

Implantation réalisée

Nous avons employé un afficheur de caractères à LCD de 2 lignes de 20 colonnes, contrôlé par un 44780 de Hitachi. Ce contrôleur peut être opéré en mode « 4 bits » ou « 8 bits », ceci décrivant la largeur des données envoyées au contrôleur. Nous avons employé le mode « 8 bits ».

Pour utiliser ce contrôleur il s'agit d'abord d'initialiser le chip et par la suite de lui donner des commandes. Nous avons employé les commandes de positionnement, d'affichage d'un caractère et d'effacement de l'affichage. Il y a 2 approches possibles pour l'opération de chaque commande envoyée au contrôleur. D'abord on peut attendre un certain délai afin de laisser le temps au contrôleur de réaliser son travail, ou alors on peut examiner de manière active son « acknowledge ». Considérant que notre projet ne demande pas d'animation rapide à l'écran nous avons choisi l'approche où nous attendons un délai raisonnable pour émettre une nouvelle commande. Cette approche est moins contraignante au niveau de la logique de contrôle, en particulier cela ne requiert pas d'interruption générée à l'ordre de la micro-seconde près ni d'optimisation des séquences d'opérations du micro-contrôleur.

Le reste de cette sous-section présente les approches employées pour l'initialisation et l'opération commande par commande du 44780, de même que les fonctions implantées pour l'opération de l'affichage dans le contexte plus général du projet.

Opération générale

Dans notre kit l'écran est branché sur le port E du microcontrôleur. Il s'agit donc d'écrire les bonnes valeurs sur ce port pour commander le 44780. Comme nous ne considérons pas les résultats donnés par le 44780, nous n'avons à utiliser le port qu'en lecture seulement.

Mode contrôle :
- 0-5 : code de commande
- 6 : signal RS
- 7 : signal EN

Mode caractère :
- 0 – 7 : code du caractère

Il est à noter que ces 2 modes sont exclusifs et qu'à tout moment le port E ne sert qu'à l'un des deux (i.e. que les adresses données et contrôles sont séparées, c'est le 8255 qui fait la distinction au niveau câblage).

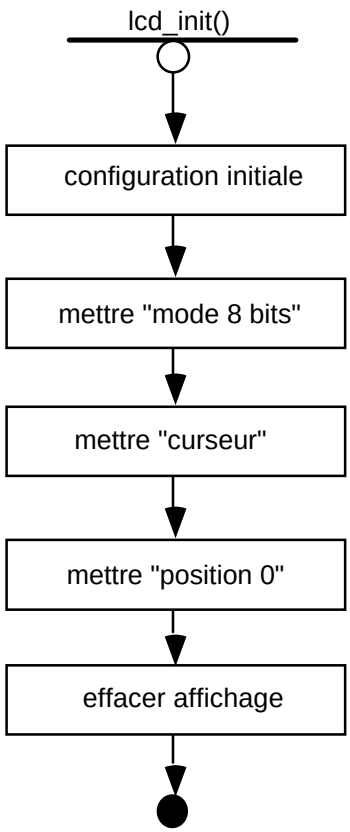
Initialisation

Pour l'initialisation du contrôleur il s'agit essentiellement de le mettre en mode d'entrée de caractères, 8 bits puis d'ajuster les caractéristiques du curseur.

Chacune de ces étapes est réalisée à l'aide d'une routine gérant l'opération de l'écran, tel que décrite au point suivant. Le diagramme de flot de l'initialisation est présenté à la figure ci-contre.

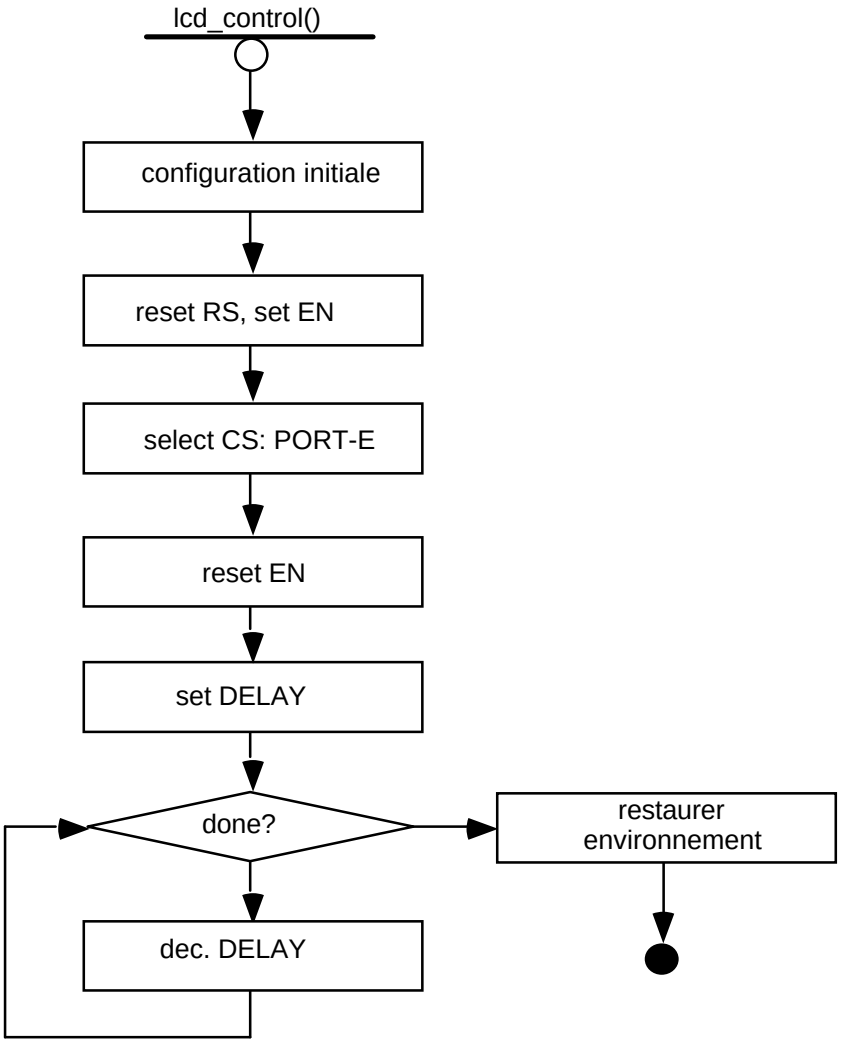
Pour la configuration initiale et finale il s'agit simple de s'assurer que l'environnement est pilé puis dépilé.

Cette routine est appelée dans le code C par la fonction « lcd_init() ».



Opération

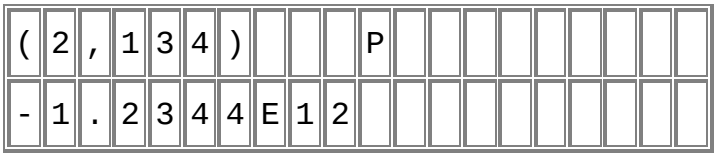
Pour chaque opération sur le contrôleur il s'agit d'activer les signaux RS et EN convenablement puis d'écrire la donnée correspondant à la commande au contrôleur. Il faut ensuite soit attendre un acknowledge ou attendre un délai raisonnable; c'est cette dernière approche que nous employons. Le diagramme de flot est présenté à la figure ci-contre.



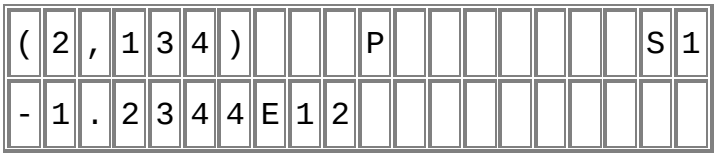
Dans le code C cette routine est appelée par la fonction « lcd_control(int code) », où le code correspond à la commande à donner.
Le DELAY est une boucle active dont le compte est ajusté pour obtenir un délai de 40 micro-secondes. Nous attendons 100 fois ce compte.

QUELQUES EXEMPLES DE FONCTIONNEMENT

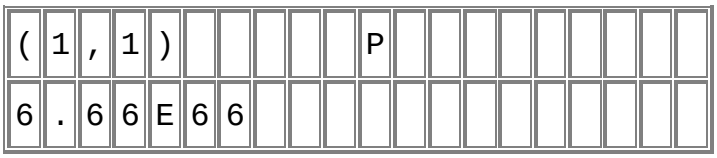
Exemple 1 : on affiche le nombre contenu à la position (2,134) de la matrice, qui a été directement entrée sur la pile. Ce nombre est -1.2344 x 10^{12}.



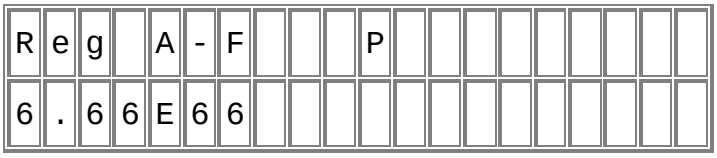
Exemple 2 : Après l'exemple 1, on décide de multiplier les deux matrices qui sont sur le dessus de la pile. On a la séquence suivante.
Shift1



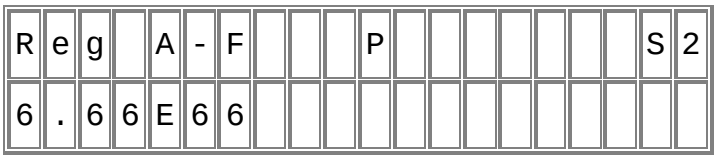
Touche 2. Après la multiplication, on a une nouvelle matrice. On affiche son premier élément. Par un pur hasard, ce nombre est 6.66 x 10^{66}.



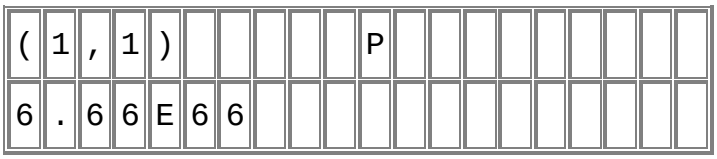
Exemple 3 : On désire sauvegarder le précieux résultat obtenu à l'exemple 2 dans le registre F. On a la séquence suivante.
On appuie sur Store. Il faut préciser le registre.



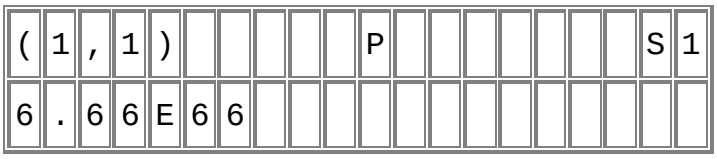
On fait Shift2



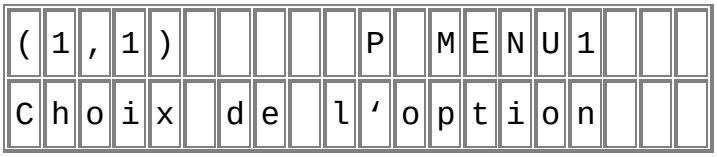
On appuie sur 6 et on revient à l'état initial. On affiche encore " P " dans le champ registre, puisque la matrice n'a pas été récupérée d'un registre.



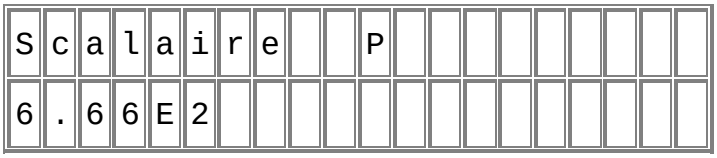
Exemple 4 : On veut calculer le déterminant de la matrice. On suppose que la fonction pour calculer le déterminant est dans le menu1, touche Enter.
On fait Shift1.



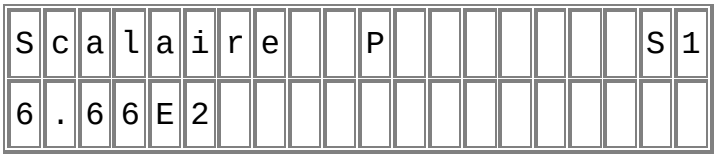
On fait 7.



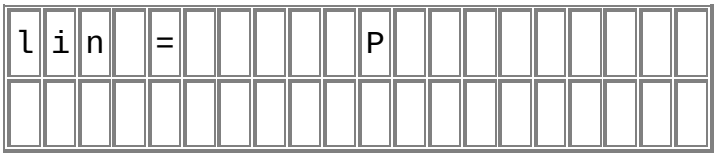
On fait Enter. Le déterminant est un scalaire. Par un autre pur hasard, le déterminant est 666!



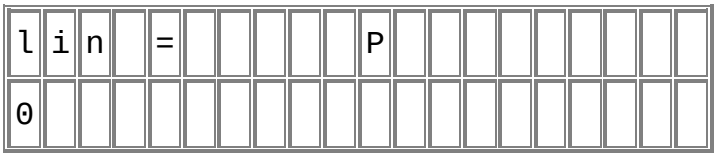
Exemple 5 : On veut saisir une matrice 2x3, mais on a des problèmes à utiliser le clavier! Shift 1...



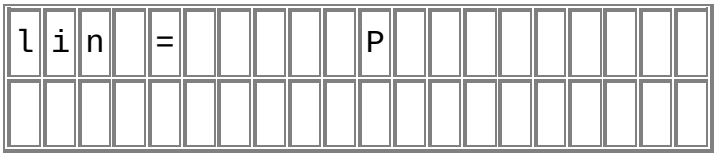
Enter...



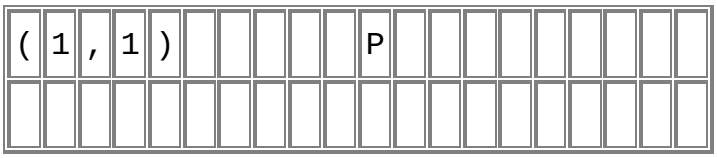
On veut écrire 2, mais un spasme fait qu'on écrit 0.



On fait Enter... Erreur, dimension incorrecte... On redemande le nombre de lignes...



(etc.) Finalement, on commence à entrer les valeurs à partir de (1,1)



CONCLUSION

Le développement du projet de calculatrice matricielle CALMAT 64 fût un défi de taille pour l'ensemble de notre équipe. La répartition des tâches en groupes (théorie, logiciel, matériel) a permis de répartir le travail entre les membres de façon efficace, ce qui a permis une progression constante.

L'utilisation de ressources matérielle incluant le Motorola 68HC16, un clavier, un écran, et bien sûr le kit de développement Freedom 16 nous a donné l'occasion de réaliser concrètement notre projet. À la base, cependant, son bon fonctionnement vient de notre effort à produire un code source exécutant le plus possible les tâches qu'on attendrait d'un calculateur matriciel. Cette partie logicielle est le reflet de nombreuses heures de discussion afin d'élaborer, en théorie pour commencer, les caractéristiques de notre système. L'ensemble de cette démarche donne un résultat que nous avons perfectionné jusqu'à la dernière minute, et dont des exemples de fonctionnement ont été donné à la section précédente.

Le développement d'un projet complexe tel que celui-ci nous a appris à bien gérer nos ressources, autant humaines que matérielles et temporelles. En avoir l'occasion, nous aurions même des idées pour produire des versions améliorées de notre produit. Or, cela serait une toute autre histoire…

BIBLIOGRAPHIE

CPU16 Reference Manual, *Motorola*, doc. réf. CPU16RM/AD révision 2

Freedom16 Advanced Microcontroller User's Manual, *Intec Corporation*

HC11 Technical Data, *Motorola*

HD44780U (LCD-II) Reference, *Hitachi,* doc. réf. ADE-207-272(Z), Révision 0.0, '99.9

HD61830 (LCDC) Reference, *Hitachi*, doc. réf. ADE-207-275(Z), révision 0.0 '99.9

How to use Intelligent LCDs, *Julian Ilet*, Everyday Practical Electronics, Février 1997, accessible sur le Web à http://www.maxmon.com