5/6

Micro-informatique

Table des matières

5/6.1

Micro-ordinateur MPS 65

Pourquoi le MPS 65? Presque tous les micro-ordinateurs sont bâtis autour des micro-processeurs 6502 et Z80 (ou 8080). Nous avons donc été amenés à choisir entre ces deux "unités centrales". Parmi les modèles simples d'initiation (monocartes), on trouve plus facilement des 6502, ce qui nous a conduit à retenir le MPS 65. Mais si vous possédez ou recevez un monocarte 6502 différent, l'extrapolation de ce qui suit est relativement aisée. Il suffira de trouver ou d'établir la correspondance des noms des fonctions (E, B, G, etc. dans ce qui suit), ce qui peut même se faire par tâtonnements. Mais revenons au MPS 65.

Le micro-ordinateur MPS 65 est un ordinateur monocarte sur carte de format européen. Il offre un grand nombre de possibilités d'emploi, par exemple pour la commande d'un réseau de chemin de fer

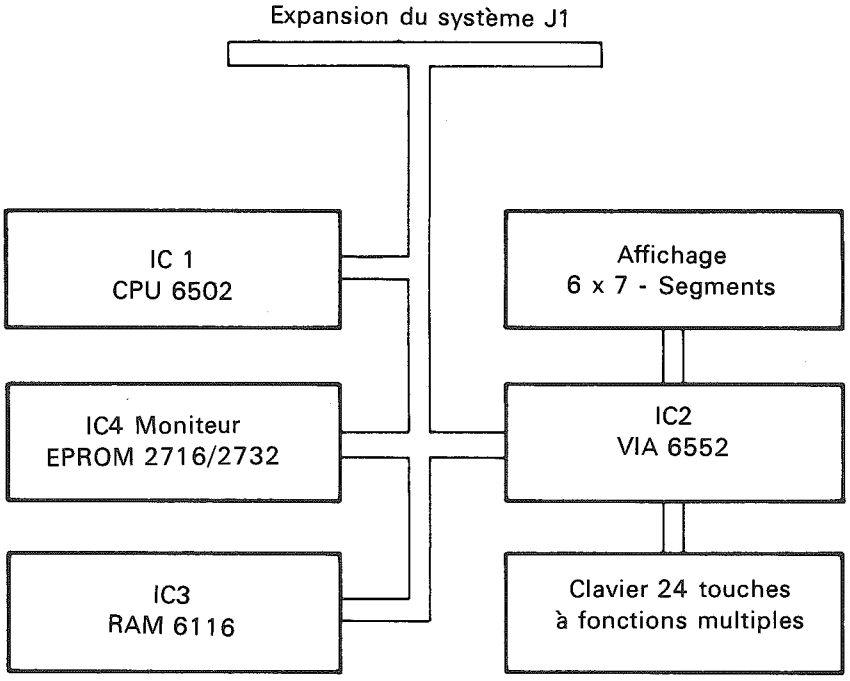


Fig. 1 : Schéma synoptique

6.1 Micro-ordinateur MPS 65

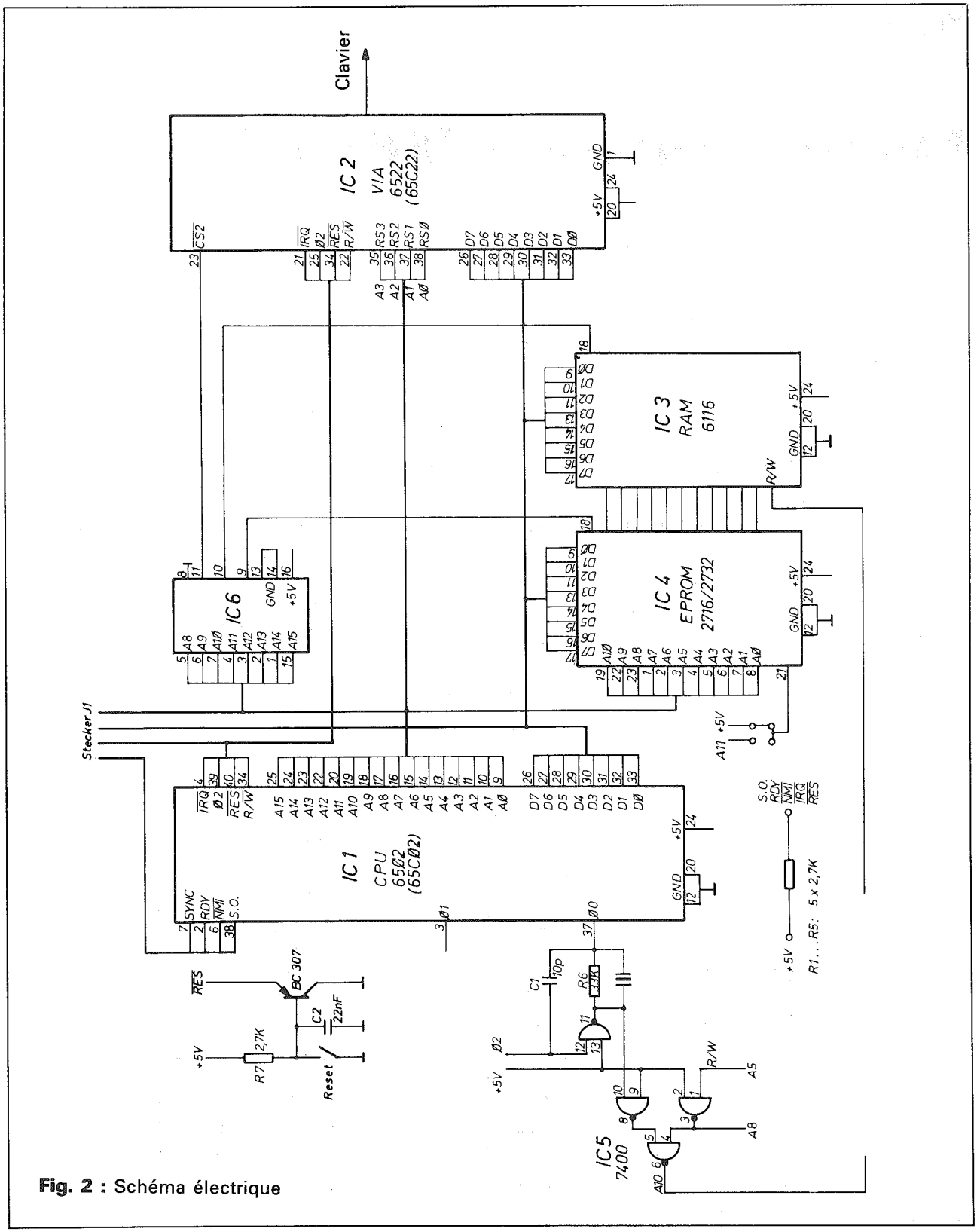


Fig. 2 : Schéma électrique

6.1 Micro-ordinateur MPS 65

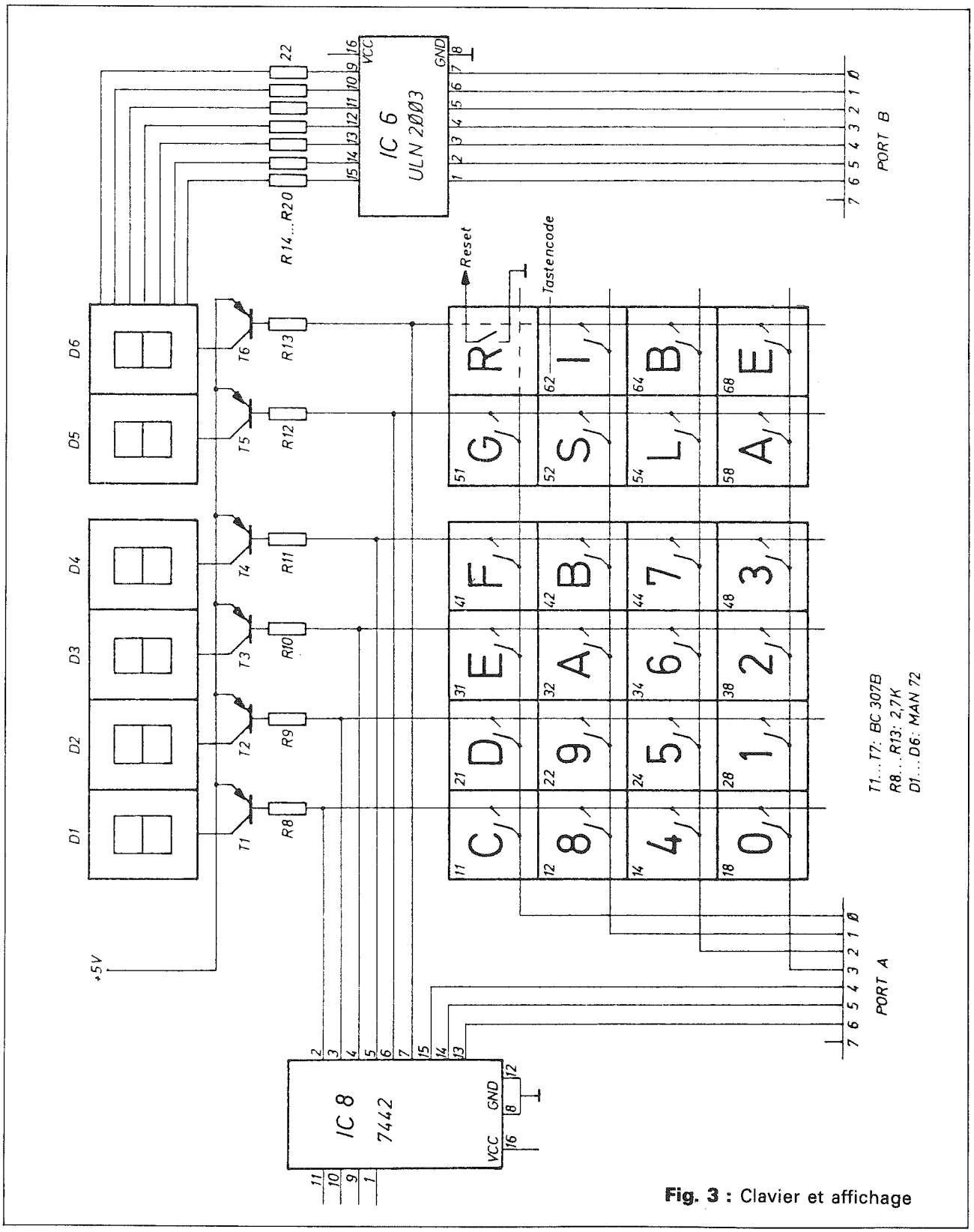


Fig. 3 : Clavier et affichage

6.1 Micro-ordinateur MPS 65

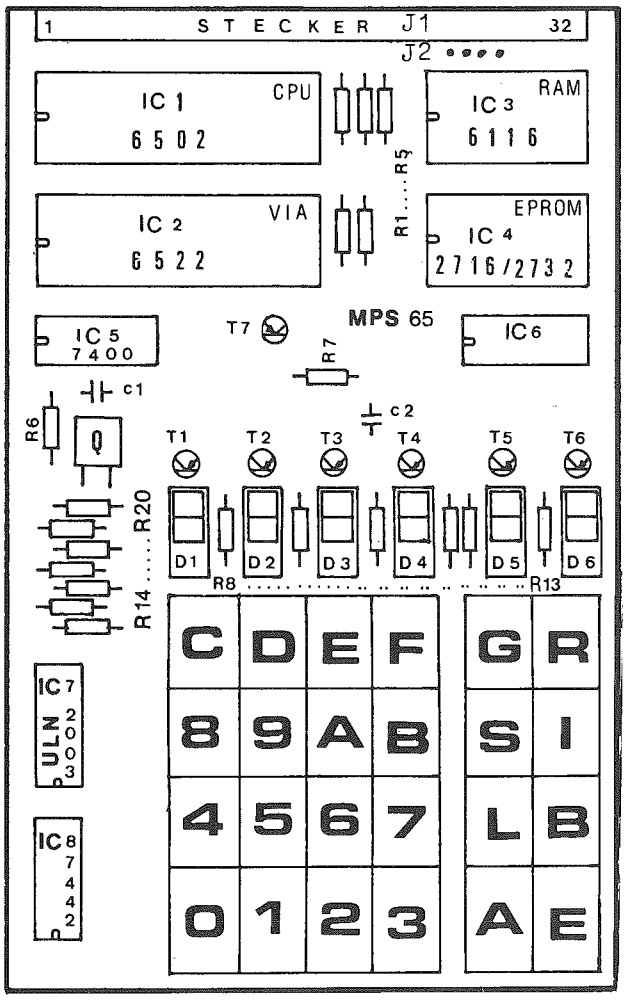


Fig. 4 : Plan d'implantation

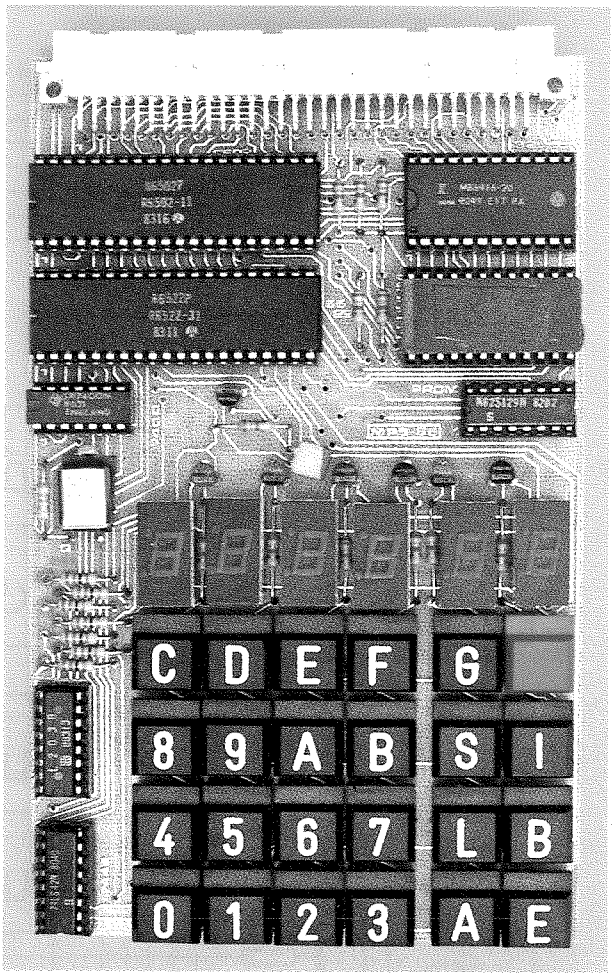


Fig. 5 : La carte format européen avec ses composants

modèle réduit ou pour l'apprentissage d'un langage de programmation. Le système constitue un ordinateur autonome, pouvant être complété par différentes platines reliées à travers le bus SMP. Avec le MPS 65, il est possible de créer à peu de frais la base matérielle d'initiation à l'informatique.

Les principales caractéristiques électriques :

Unité centrale :	CPU 6502 (Rockwell)
Commande d'interface :	VIA 6522
Affichage :	6 afficheurs à 7 segments
Mémoire vive :	RAM 6116 (2 Koctets)
Mémoire morte :	EPROM 2716/2732 (programme moniteur)
Système de bus :	Système SMP étendu
Alimentation :	5 V/0,4 A
Dimensions :	Circuit imprimé 10 x 16 cm (carte format européen)

Mise en marche du MPS 65

Alimentation

Le système microprocesseur MPS 65 peut fonctionner comme unité autonome.

Il suffit de le relier à une alimentation secteur fournissant une tension continue de 5 V. L'alimentation s'effectue par J2.

La tension d'alimentation doit être de 5 V = (± 0,25 V). La consommation du MPS 65 est d'environ 0,4 A.

Lors du branchement d'un système de BUS, l'alimentation s'effectue à travers J1 (+ 5 V : a32 – GND : c2, c31).

Processus de mise en marche

Après mise sous tension, le MPS 65 affiche

µPS-65

Après action sur la touche E-Enter, il affiche 0200xx avec le contenu existant dans 0200H qui, à la mise sous tension, est de 00H (la zone de 0200H à 07FFH est remplie, à la mise sous tension, de 00H).

0200 00

Nota : si l'affichage reste obscur, démarrez le MPS 65 par action sur la touche R-RESET.

Fonctions du clavier

Ø-F Touches hexadécimales

Clavier hexadécimal.
Le clavier hexadécimal sert à l'introduction de données et d'adresses sous forme hexadécimale (Ø, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F).
Une action sur la touche INDEX donne une seconde fonction à certaines des 16 touches - voir touche INDEX.

E Touche ENTER

La touche ENTER est la plus importante de toutes les touches, car elle remplit plusieurs fonctions.

1. Un pas en avant, single step. La pression sur la touche provoque l'incrémentation de l'affichage d'adresses (augmenté de 1).
   L'affichage montre le contenu de la nouvelle cellule de mémoire.
2. Transfert d'une donnée introduite dans la cellule de mémoire affichée, et incrémentation de l'adresse.
3. Fin des programmes moniteur
   Mise à l'heure de l'horloge
   Affichage de l'heure
   Fonction BREAK

B Touche BACKSTEP

La touche BACKSTEP a simplement pour fonction de décrémenter l'adresse (la diminuer de 1). Indication sur l'afficheur de données : contenu de cette cellule de mémoire. Une donnée modifiée n'est pas mise en mémoire, ce qui est important pour qu'en cas d'introduction erronée, l'ancienne valeur soit conservée.

G Touche GO

La touche GO fait démarrer les programmes utilisateur. L'adresse de démarrage est l'adresse indiquée par l'afficheur d'adresses.

Modification de l'adresse de démarrage : voir touche A.

Après démarrage d'un programme utilisateur, les afficheurs indiquent

= = = = = =

(A condition que le programme ne fasse pas appel à l'affichage.)

Nota :
Le programme moniteur et l'horloge système fonctionnent comme programme d'interruption pendant que le programme utilisateur se déroule.
On peut donc scruter les touches (KEYIN, KEYIN1) et modifier les afficheurs (ADR16, ADR16+1, DAT8 ou DISP1 à DISP6).

S Touche SAVE

L Touche LOAD

Ces touches sont utilisées avec quelques cartes d'extension du système.

I Touche INDEX

La touche INDEX modifie la fonction de quelques-unes des 16 touches hexadécimales (0 à F).

Affichage — — — — In

INDEX B Conversion en adresses relatives
INDEX C Déplacement de blocs de Ø2ØØ à Ø7FF
INDEX D Affichage de l'horloge système
INDEX E Démarrage de programme à EØØØ
INDEX F Mise à l'heure de l'horloge système

Fonction BREAK

L'instruction BREAK est un ordre intéressant du 6502. Le BREAK est un interrupt de software et, dans le système CT65, un auxiliaire de test de programmes. Si le système détecte un BREAK (ØØ) dans un programme en cours d'exécution, le programme est interrompu et les contenus des registres X, Y et A du CPU sont affichés.

Affichage : (Reg. X) (Reg. Y) (ACCU)

Mais la réaction du système en présence d'un BREAK peut aussi être modifiée par l'utilisateur.

Fonctions indexées

INDEX F

Mise à l'heure de l'horloge système.
Après enfoncement de la touche INDEX F, on peut mettre l'horloge système à l'heure.
On introduit les heures et minutes par les touches Ø à 9. Lorsqu'on appuie sur la dernière touche, l'horloge démarre.
La touche E-ENTER permet de supprimer l'affichage de l'heure. L'horloge continue alors de fonctionner. Elle n'est arrêtée que par RESET et SAVE/LOAD, mais continue ensuite de fonctionner.
L'heure peut être demandée dans un programme (H, MN, S : adresses ØØA9, ØØAA, ØØAB).

INDEX D

Affichage de l'horloge système interne.
A terminer par E-ENTER.

INDEX B

Conversion d'une adresse absolue en distance de saut (adresse relative).
Les instructions de branchement (p. ex. BEQ, BPL, etc.) sont des instructions à 2 octets dont le premier constitue l'instruction proprement dite et le second la distance de saut vers l'adresse de destination.
Par exemple :

FØ	BEQ
Ø9	9 pas en avant

Afin de pouvoir calculer cette distance de saut avec certitude (surtout pour les sauts en arrière, à cause de la distance négative du saut), on dispose de la fonction INDEX B.
L'instruction, p. ex. BEQ = FØ est introduite normalement et transférée par E-ENTER.
A l'endroit où la distance de saut doit être introduite, vous appuyez sur INDEX B et donnez l'adresse absolue de saut à quatre digits. Vous obtenez la distance de saut correctement calculée et continuez en agissant sur E-ENTER.

Fonctions indexées Auto-start

Exemple :

```
Ø22Ø     FØ  ØE  BEQ  Ø23Ø
.
.
Ø23Ø     Introduction : INDEX/B/Ø23Ø/E-ENTER
```

INDEX C

Déplacement de blocs dans la zone Ø2ØØ à Ø7FF.
Si vous désirez introduire ultérieurement une instruction dans un programme, la fonction INDEX C sert à décaler le bloc de programme commençant à l'adresse affichée d'un pas vers le haut. Le trou ainsi produit est rempli par NOP = EA et peut donc à nouveau recevoir une instruction. Le déplacement n'est possible que dans la zone Ø2ØØ à Ø7FF.

INDEX E

Le système effectue un saut vers EØØØ.

Un programme commençant à partir de l'adresse EØØ5 sera démarré par le système après initialisation du système, à condition qu'il y ait le code Ø1 et FF aux adresses EØØ3 et EØØ4.

EØØ3	01
EØØ4	FF
EØØ5	Début du programme

Les adresses EØØØ à EØØ2 sont libres pour permettre de sauter au début du programme par un JMP. Utilisation : INDEX E.

Code d'erreurs

Le système MPS 65 émet divers signaux d'erreurs. Selon le type d'erreur, les chiffres 1, 2, 3, etc. codifient l'erreur.

Signification des codes d'erreurs :

Error 1 Indication d'adresse illogique (début > fin)

Error 2 Saut relatif trop loin en arrière (> 128)

Error 3 Saut relatif trop loin en avant (> 127) ou sur une distance de saut correspondant à sa propre adresse

Error 4 Absence d'ordre Branch

Error 6 Erreur dans la manipulation des vecteurs d'interruption

Après action sur la touche E-ENTER, vous pouvez recommencer l'introduction.

Tableau 1

Nom	Adresse	Reg.	Description
KEYIN	F800	A	Scrutation du clavier avec attente. Le code de la touche est transféré dans l'accumulateur.
KEYIN1	F803	A	Scrutation du clavier sans attente. Le code de la touche est dans l'accumulateur.
DISPEX	F806	—	Mise en fonction "Display Extern" (commande individuelle des segments).
DISPSY	F809	—	Mise en fonction "Display System" (représentation hexadécimale de 0 à F).
PHAXY	F80C	—	Les contenus de l'accu, reg. x et reg. y sont sauvegardés dans la pile.
PLAXY	F80C	A,X,Y	Les contenus de l'accu, reg. x et reg. y sont extraits de la pile.
ASK2	F812	A,X	Introduction au clavier d'un nombre hexadécimal à 2 digits. Le code des touches est en 00A0.
ASK4	F815	A,X	Introduction au clavier d'un nombre hexadécimal à 4 digits. Le code des touches est en 00A0 et 00A1.
MON65	F818	—	Retour au programme moniteur du MPS 65. Utilisation : p. ex. FIN.
MON65E	F81b	—	Retour au programme moniteur du MPS 65 après action sur la touche E-Enter.
T01S	F81E	—	Boucle de temporisation de durée 0,1 s x (accu).
INC16X	F821	—	Un nombre de 16 bits (deux adresses de mémoire) est incrémenté de 1.
DEC16X	F824	—	Un nombre de 16 bits (deux adresses de mémoire) est décrémenté de 1.
ERR	F827	A,X,Y	Indication : "Error (A)". (A) est le contenu de l'accu (de 0 à F).
STRICH	F82D	—	Affichage : – – – – – –
SUCHEX	F830	A	Scrutation du clavier avec attente (touche enfoncée dans l'accu 0 à F).

Appels système (description)

Les appels système sont des sous-programmes du programme moniteur qui sont à la disposition de l'utilisateur.

Les parties de programmes apparaissant souvent sont réunies ici et peuvent être appelées comme sous-programmes (par JSR). Les appels sont des programmes ou des parties de programmes au fonctionnement impeccable qui, là où c'est important, tiennent compte du comportement du système dans le temps.

L'accumulateur sert dans la plupart des cas au transfert de paramètres (p. ex. code de la touche vers l'accumulateur lors d'un KEYIN). Les registres X et Y ne sont pas modifiés lorsqu'ils ne sont pas nécessaires au transfert (exception : ERR-F827).

Les adresses du système se trouvent à partir de F8ØØ.

L'utilisateur recherche un tableau de sauts (à partir de F8ØØ) qui le guide ensuite vers le sous-programme désiré.

Ce procédé offre plusieurs avantages :

— Les adresses d'appel sont successives et faciles à mémoriser.

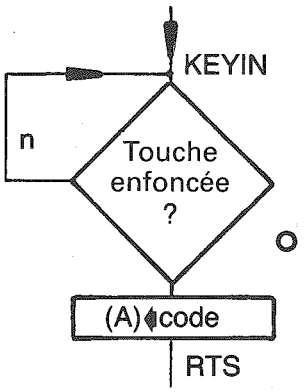
— Les adresses d'appel restent conservées, même lorsque le programme moniteur doit être amélioré ou complété.

De sorte que tous les programmes existants restent valables avec des versions plus étendues du moniteur.

==KEYIN==

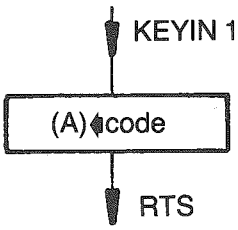
L'appel KEYIN cherche le code de la touche enfoncée. Si aucune touche n'est actionnée, KEYIN attend jusqu'à ce qu'une touche soit enfoncée. Le code de la touche est alors transféré dans l'accumulateur.

Registres modifiés :



==KEYIN 1==

Cet appel a une fonction analogue à KEYIN, mais n'attend pas qu'on enfonce une touche.



==DISPEX==

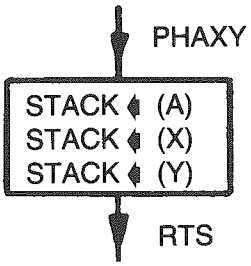
Conversion du mode moniteur vers la commande externe des six afficheurs 7 segments (DISP1 à DISP6), c'est-à-dire commande de chacun des segments (voir page 14).
Cet état est automatiquement supprimé lors d'un démarrage de programme par le système.
DISPEX n'est donc nécessaire que pour le retour à la normale après une utilisation en mode DISPEX.

En outre, les afficheurs à 7 segments, s'ils étaient coupés (OUTPB arrête l'affichage), sont mis en service.
Registres modifiés : aucun.

==**PHAXY**==

Push accu, registre X, registre Y on stack.

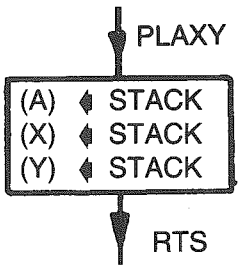
Les registres A, X, Y sont sauvegardés dans la pile.
Registres modifiés : aucun.



==PLAXY==

Pull accu, registre X, registre Y from stack.

Les contenus de registres sauvegardés par PHAXY sont extraits de la pile (PLAXY doit être appelé par JSR).
Registres modifiés : A, X, Y.



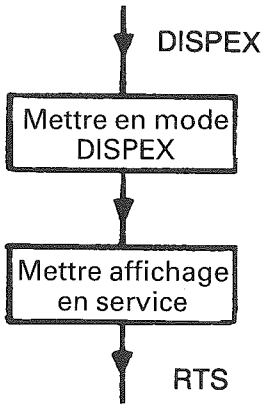
==ASK2==
==ASK4==

Possibilité d'introduction d'un nombre à 4 (2) digits. Le nombre introduit par le clavier se trouve, après déroulement du programme système, dans les cellules mémoires.

ØØAØ	ADR16	⬅ seulement ASK2
ØØA1	ADR16+1	

Registres modifiés : A, X.

Nota : les valeurs en ADR16 et ADR16+1 sont effacées par le système après RESET et JMP MON 65, puisque l'affichage est modifié.



==**DISPSY**==

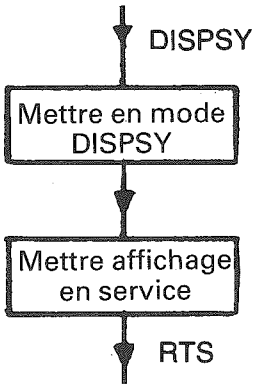
Branchement du mode moniteur sur la représentation système des six afficheurs à 7 segments (possibilité d'affichage : Ø, 1, 2, 3... E, F; voir page14).

Les données à afficher doivent être logées dans les trois adresses.

ØØAØ	ADR16
ØØA1	ADR16+1
ØØA1	DAT8

(p. ex. STA ØØAØ).
En outre, les afficheurs à 7 segments sont mis en service s'ils étaient arrêtés.
Registres modifiés : aucun.

6.1 Micro-ordinateur MPS 65

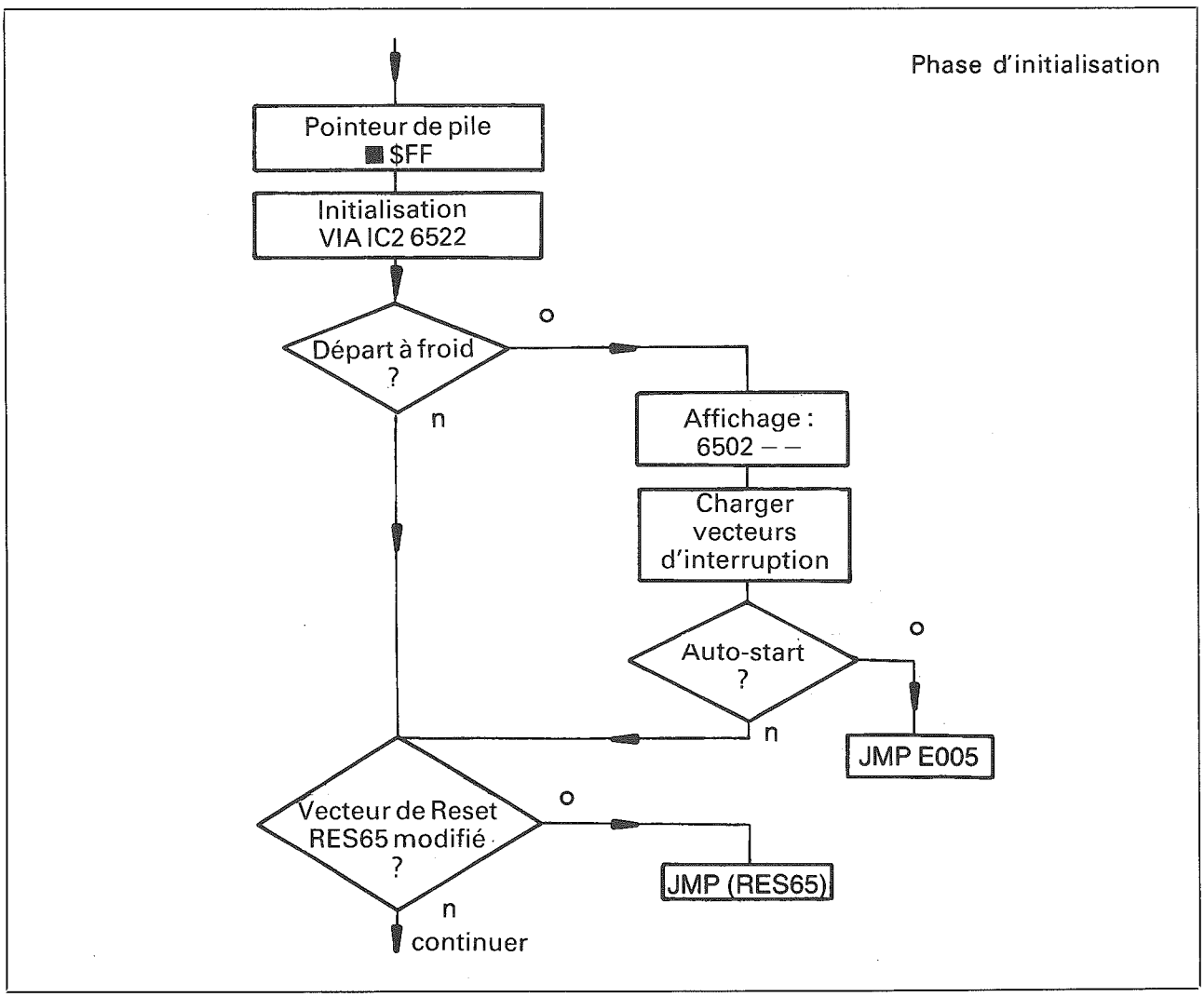


==MON65==

Retour au programme moniteur du MPS65.
Le VIA 6522 IC2 est initialisé.
Le pointeur de pile est mis à FF.

Utilisation :
Fin d'un programme utilisateur par JMP MON65.

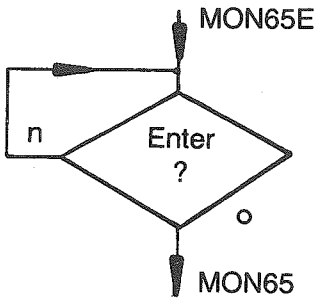
.
Programme utilisateur
.
.
4C 18 F8 JMP MON65



==MON65E==

Saut vers MON65 lorsqu'on agit sur la touche E-Enter.

Utilisation :
Fin d'un programme utilisateur lorsqu'on désire conserver le dernier affichage.



..
Programme utilisateur
..
..
4C 1B F8 JMP MON65E

==TØ1S==

Boucle de temporisation de durée Ø,1 seconde multipliée par le contenu de l'accumulateur. Cet appel produit une temporisation de 0,1 à 25,5 secondes.

La durée est déterminée par le contenu de l'accumulateur.

C'est ainsi que la durée est de 0,9 s lorsque l'accu contient Ø 9.

Exemple :

.
.
Programme utilisateur
..
..
A9 Ø9 LDA #$ temporisation
2Ø 1E F8 JSR TØ1SØ 9s
Programme utilisateur.

==INC16X==

.Incrémente 16 bits, adresse dans le registre X.

INC16X incrémente une donnée à 16 bits qui se trouve à deux adresses successives de la page zéro. L'adresse est transcrite dans le registre X.

P. ex. une donnée 16 bits est aux adresses mémoire ØØ35 et ØØ36 et doit être incrémentée. La donnée est 342E.

Adr.	Donnée
ØØ35	2E
ØØ36	34

La disposition des octets est habituelle, d'abord l'octet de faible poids, puis celui de poids fort.

Dans le programme utilisateur, la succession des instructions serait :

.
.
A2 35 LDX #$35
2Ø 21 F8 JSR INC16X
.
.

Registres modifiés : aucun.

==DEC16X==

Décrémente 16 bits, adresse dans le registre X.
Le reste comme INC16X.

==ERR==

Indication d'erreur avec le code d'erreurs (p. ex. Error3).

Le code d'erreurs est transféré dans l'accu. Codes d'erreurs possibles : 16 possibilités Error1, Error2, ..., ErrorE, ErrorF.

Exemple : Représentation de Error3.

```
A9 Ø3      LDA #$Ø3
2Ø 27 F8   JSR ERR
```

Le sous-programme ERR met le mode DISPSY en service.

Registres modifiés : A, X.

Organisation de la page zéro

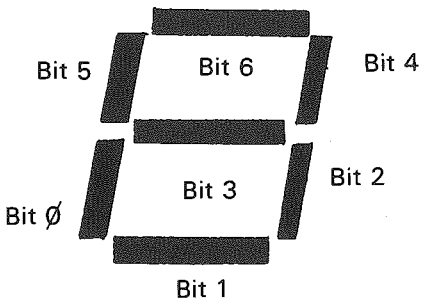
ADR16	= ØØAØ
ADR16+1	= ØØA1
DAT8	= ØØA2
DISP1	= ØØA3
DISP2	= ØØA4
DISP3	= ØØA5
DISP4	= ØØA6
DISP5	= ØØA7
DISP6	= ØØA8
STD	= ØØA9 heures (H)
MIN	= ØØAA minutes (MN)
SEC	= ØØAB secondes (S)
GA	= ØØAD
GX	= ØØAE
GY	= ØØAF
REFS	= ØØCØ
RES65	= ØØC1, ØØC2
IRQF	= ØØC3
IRQ65	= ØØC4, ØØC5
NMIF	= ØØC6
NMI65	= ØØC7, ØØC8
IRQMF	= ØØC9
IRQM	= ØØCA, ØØCB
BRKUF	= ØØCC
BRKU	= ØØCD, ØØCE

Nota : toutes les adresses sont en hexadécimal.

Représentation sur les afficheurs à 7 segments

						Modus:
DISP1	DISP2	DISP3	DISP4	DISP5	DISP6	DISPEX
ADR16+1		ADR16		DAT8		DISPSY

Commande de segments en mode DISPEX :



Exemple :

	Bit	Binaire	Hexadécimal
I	0, 5	0010	21
C	0, 1, 5, 6	0110	63
H	0, 2, 3, 4, 5	0011	3 D

Traitement des interruptions

Le traitement des vecteurs d'interruption du système CT-65

Signification des adresses :

RES65	Vecteur de RESET
IRQ65	Vecteur IRQ
NMI65	Vecteur NMI
BRKU	Vecteur d'interruption pour l'utilisateur (avec programme moniteur)
IRQM	Vecteur IRQ pour l'utilisateur (avec programme moniteur) (ce vecteur est appelé cycliquement toutes les 2,5 ms)

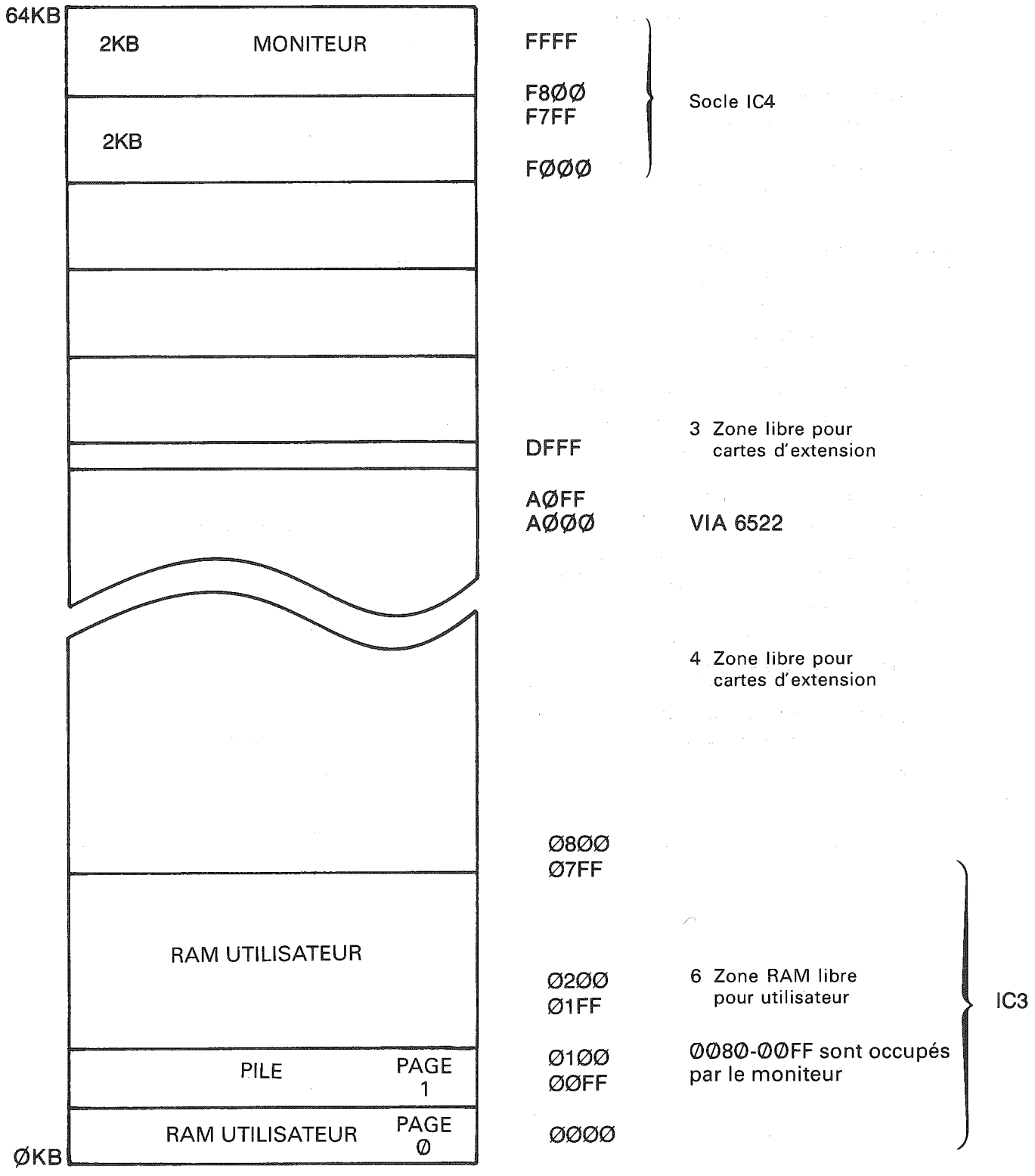
Travail correct avec les vecteurs :

1. Inscrire l'adresse de début du programme d'interruption dans les adresses des vecteurs (p. ex. NMI65 = ØØC7 et ØØC8).
2. Placer l'adresse correspondante de flag à ØØ (ici l'adresse ØØC6).

Les programmes d'interruption doivent présenter la structure suivante :

```
RES65   ..
        ..
        (Programme)
        ..
        JSR PLAXY
        RTI
IRQ65   ..
        ..
        (Programme)
        ..
        JSR PLAXY
        RTI
NMI65   ..
        ..
        (Programme)
        ..
        JSR PLAXY
        RTI
BRKU    ..
        ..
        (Programme)
        ..
        ..
        JSR PLAXY
        RTI
IRQM    ..
        ..
        (Programme)
        ..
        ..
        RTS!
```

6.1 Micro-ordinateur MPS 65



Architecture

L'ordinateur MPS 65 est un circuit imprimé de 10 cm sur 16 cm (carte au format européen) avec clavier, affichage et tous les circuits intégrés nécessaires à son fonctionnement.
La CPU (central processing unit = unité centrale) est un 6502 de Rockwell.
Le clavier et les afficheurs à 7 segments sont commandés par le circuit port 6522, également nommé VIA (versatile interface adapter).
Le connecteur J1 constitue le bus du système (lignes d'adresses, de données et de commande).
Le connecteur à 24 broches d'IC4 est équipé d'une EPROM 2716 de 2048 x 8 bits contenant le programme moniteur (système d'exploitation).

IC3 est une RAM 2114 (6116) nécessaire au fonctionnement du système (pile, mémoire intermédiaire). La zone de RAM de ØØØØH à ØØ7FH et de Ø2ØØH à Ø7FFH est libre pour l'utilisateur.

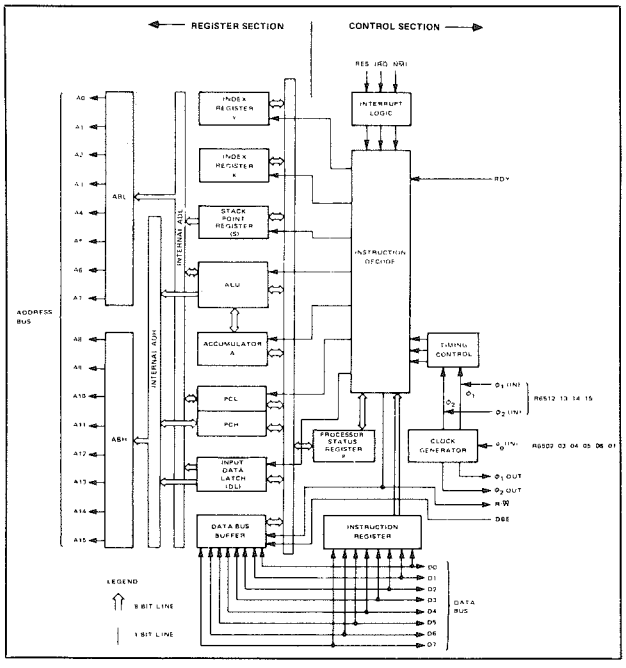


Fig. 6 : Constitution interne de la CPU 6505

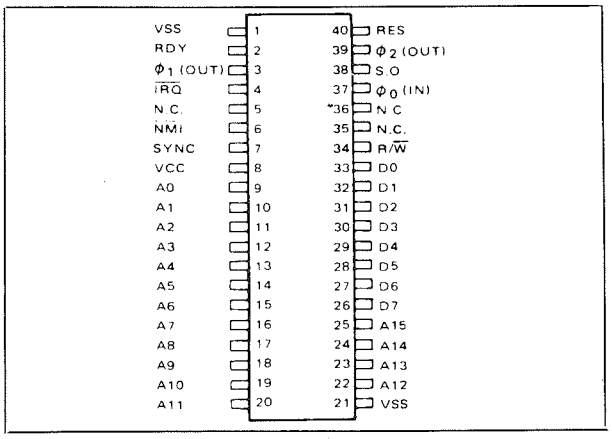


Fig. 7 : Brochage du boîtier DIL 40 broches du 6502

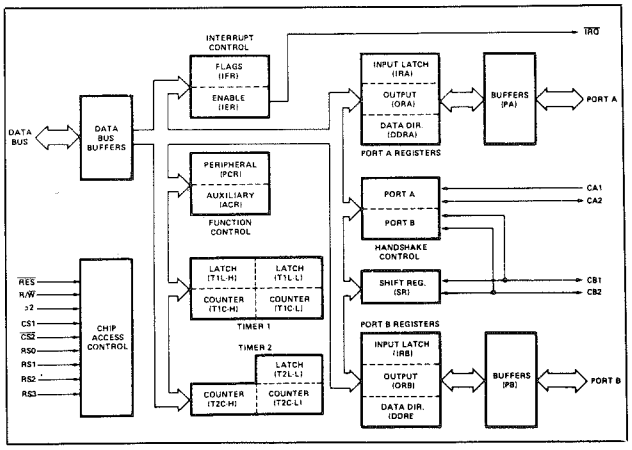


Fig. 8 : Schéma synoptique de la VIA R 6522

Principales caractéristiques techniques :

1. Deux entrées/sorties bidirectionnelles 8 bits.
2. Deux timer/compteurs programmables 16 bits.
3. Entrée série de données.
4. Alimentation par 5 V.
5. Compatible TTL.
6. Lignes de commandes de périphérie compatibles CMOS.
7. Fonctionnement sous 1 et 2 MHz.

6.1 Micro-ordinateur MPS 65

Les principales instructions du 6502

LDA LDX LDY	A9 A2 AØ	A5 A6 A4	AD AE AC	B5 B4	BD BC	A1	
STA STX STY		85 86 84	8D 8E 8C	95 94	9D	81	
CMP CPX CPY	C9 EØ CØ	C5 E4 C4	CD EC CC	D5	DD		
AND ORA EOR	29 Ø9 49	25 Ø5 45	2D ØD 4D	35 15 55	3D 1D 5D	21 Ø1 41	
ADC SBC	69 E9	65 E5	6D ED	75 F5	7D FD	61 E1	
INC DEC		E6 C6	EE CE	F6 D6	FE DE		
ASL LSR ROL ROR	A ØA A 4A A 2A A 6A	Ø6 46 26 66	ØE 4E 2E 6E	16 56 36 76	1E 5E 3E 7E		
JMP JSR			4C 20				6C
BCC BCS	9Ø BØ	BEQ BNE	FØ DØ	BPL BMI	1Ø 3Ø	BVS BVC	7Ø 5Ø
CLC SEC CLD SED CLI SEI	18 38 D8 F8 58 78	TAX TXA TAY TYA TSX TXS	AA 8A A8 98 BA 9A	INX DEX INY DEY NOP BRK	E8 CA C8 88 EA ØØ	PHA PLA PHP PLP RTS RTI	48 68 Ø8 28 6Ø 4Ø

Exemples de programmation

Exemple 1 :
Le signe "L" doit être représenté sur les afficheurs 1 et 5.

"L" n'est pas représentable avec les caractères définis Ø à F, il faut commander les segments individuellement, ce que l'on obtient en mode DISPEX, qui est normalement mis en fonction par le système (page 14).

Pour le signe "L", il faut poser les bits Ø, 1, 5 (page 14) - ce qui signifie 23 en hexadécimal.
Les afficheurs 1 et 5 sont commandés par les adresses DISP1 et DISP5 (ce qui correspond à ØØA3 et ØØA7).

Il faut donc inscrire la constante 23 aux adresses mémoires ØØA3 et ØØA7.
Le programme se termine par le retour au programme moniteur : JMP MON65E.

Le programme :		**Le code hexadécimal:**	
LDA	#$23	A9	23
STA	DISP1	85	A3
STA	DISP5	85	A7
JMP	MON65E	4C	1b F8

Introduisez le programme (le code hexadécimal) dans la mémoire vive.
Chaque introduction sera terminée par E-Enter.
Mettre l'adresse à Ø2ØØ à l'aide de A-Adresse.
Démarrez le programme par G-Go.
L'affichage doit prendre l'aspect suivant :

L – – – L –

La touche E-Enter vous permet de revenir au programme moniteur.

Exemple 2 :
Représentation de 6 caractères quelconques sur les afficheurs 1, 2, 3, 4, 5, 6. Les caractères se trouvent dans le tableau "TAB" qui se situe en page zéro à partir de l'adresse ØØØØ.

La longueur de saut après l'instruction BPL doit être introduite à l'aide de I-Index B.

Le programme :

```
        LDX  #$Ø5
SCHL:   LDA  TAB, x
        STA  DISP1, x
        DEX
        DPL  SCHL
        JMP  MON65E
```

Le code hexadécimal :

Ø2ØØ	A2	Ø5	
Ø2Ø2	b5	ØØ	
Ø2Ø4	95	A3	
Ø2Ø6	CA		
Ø2Ø7	1Ø	F9	⧫ avec Index B
Ø2Ø9	4C	1b	F8

Les 6 caractères à représenter sont inscrits dans les adresses de mémoire ØØØØ à ØØØ5.

Exemple : ICH

TAB:	ØØØØ ØØ		DISP1
	ØØØ1 21		DISP2
	ØØØ2 63	⧫	DISP3
	ØØØ3 3D		DISP4
	ØØØ4 ØØ		DISP5
	ØØØ5 ØØ		DISP6

Par A-Adresse vous revenez au début du programme (Ø2ØØ) et démarrez le programme par G-Go.

Exemple 3 :
Représentation du code clavier sur les afficheurs 5 et 6 :

Le programme se met d'abord en mode Display System (DISPSY - affichage direct des caractères hexadécimaux).

On scrute le clavier par KEYIN puis on transfère la valeur contenue dans l'accumulateur à l'adresse ØØA2. Le système indique, sur les afficheurs 5 et 6, le contenu de l'adresse ØØA2.

```
Ø2ØØ  2Ø  Ø9 F8          JSR DISPSY
Ø2Ø3  2Ø  ØØ F8   SCHL : JSR KEYIN
      85  A2             STA DAT8
      4C  Ø3 Ø2          JMP SCHL
```

Lorsque le programme est démarré à partir de Ø2ØØ, les afficheurs 5 et 6 affichent le code de la touche enfoncée (sauf RESET).

Exemple 4 :
Incrémentation hexadécimale sur les afficheurs 1, 2 :

```
Ø2ØØ  2Ø  Ø9 F8
Ø2Ø3  A9  Ø3
      2Ø  1E F8
      E6  A1
      4C  Ø3 Ø2

        JSR  DISPSY
SCHL:   LDA  #$Ø3
        JSR  TØ1S   :Boucle de
                     temporisation
        INC  A1     :Incrémentation (A1)
        JMP  SCHL
```

Exemple 5 :
Programme dépendant de l'heure :

Certaines utilisations exigent qu'un programme défini commence à une heure déterminée, comme par exemple une commande de chauffage. Nous nous proposons de montrer ici comment résoudre un tel problème à l'aide de l'horloge à quartz incorporée. On suppose au départ que l'heure de démarrage se trouve aux adresses ØØ, Ø1, Ø2 de la page zéro.

L'heure réelle se trouve dans les adresses STD (heures), MIN (minutes) et SEK (secondes) c'est-à-dire en A9, AA et AB.

```
Ø2ØØ  A2  Ø2
Ø2Ø2  B5  ØØ
Ø2Ø4  D5  A9
Ø2Ø6  DØ  F8
Ø2Ø8  CA
Ø2Ø9  1Ø  F7
Ø2ØB  ...
LDX   #Ø2
LDA   ØØ,X    :Chercher l'heure de
                consigne
CMP   STD,X   :Comparaison avec l'heure
                réelle
BNE   $2ØØ    :Différente : recommencer
DEX   ...
BPL   $2Ø2    :Toujours différente
Ø2ØB  ...
```

A l'adresse Ø2ØB commence le programme utilisateur proprement dit, qui démarre lorsqu'on atteint l'heure figurant aux adresses ØØØØ à ØØØ2.

Utilisation des adresses de mémoire GA, GX et GY

L'utilisateur peut inscrire des données dans les adresses GA, GX et GY. Lorsqu'on démarre un programme par la touche G, ces données sont transférées dans l'accumulateur et les registres X et Y. De sorte qu'on démarre un programme avec des valeurs définies. On peut ainsi tester certaines parties du programme sans que d'autres calculs les influencent. A titre d'exemple, on peut placer certaines valeurs dans GA et sauter au sous-programme TØ1S.

Brève initiation au système micro-ordinateur MPS 65

Introduction au système de numération hexadécimale

Un système informatique ne fonctionne qu'avec des impulsions logiques, comme MARCHE ou ARRÊT, représentées par 1 ou Ø.

Une telle information se nomme un "bit" Pour mieux manier les ensembles de bits, ceux-ci sont toujours regroupés par quatre dans le système hexadécimal, et ils sont représentés par des chiffres ou des lettres. Ces quatre bits s'appellent des "nibbles". Huit bits ou deux nibbles donnent un "octet", ou "byte" en américain et en allemand.

Le tableau suivant présente les systèmes décimal, binaire et hexadécimal.

décimal	binaire	hexadécimal
Ø	ØØØØ	Ø
1	ØØØ1	1
2	ØØ1Ø	2
3	ØØ11	3
4	Ø1ØØ	4
5	Ø1Ø1	5
6	Ø11Ø	6
7	Ø111	7
8	1ØØØ	8
9	1ØØ1	9
10	1Ø1Ø	A
11	1Ø11	B
12	11ØØ	C
13	11Ø1	D
14	111Ø	E
15	1111	F

Comme le montre le tableau, la représentation hexadécimale correspond exactement, pour les combinaisons de bits ØØØØ à 1ØØ1, aux nombres décimaux.

Pour les combinaisons 1Ø1Ø à 1111, il faut ajouter un second chiffre dans le système décimal. Ce qui est matériellement correct, mais pose des problèmes pour les tableaux et les liaisons logiques. Pour pouvoir conserver une représentation à un seul chiffre, les valeurs 1Ø à 15 du système décimal sont figurées par les lettres A, B, C, D, E, F dans le système hexadécimal.

Nous donnons ci-après quelques exemples de calcul avec le système hexadécimal.

Pour éviter les confusions, les nombres hexadécimaux sont précédés du $ (dollar) ou suivis d'un H (pour hexadécimal).

Dans les exemples qui suivent, nous n'utiliserons que le $, qui correspond par ailleurs à la représentation normale avec les systèmes équipés d'un 6502 et 6800.

Exemple 1
Fonction ADN (ET, AND) sur 4 bits

binaire	hexadécimal
Ø1Ø1	$5
^ Ø11Ø	$6
Ø1ØØ	$4

Exemple 2
Fonction ADN (ET, AND) sur 8 bits

binaire	hexadécimal
11Ø1.11ØØ	$DC
^ 11ØØ.Ø111	$C7
11ØØ.Ø1ØØ	$C4

Exemple 3
Fonction ORA (OU, OR)

binaire	hexadécimal
Ø111.ØØØØ	$7Ø
ˇ ØØ11.Ø1ØØ	$C7
Ø111.Ø1ØØ	$74

Exemple 4
Fonction EOR (OU exclusif, EXOR)

binaire	hexadécimal
Ø1Ø1.ØØØØ	$7Ø
ˇ ØØ11.Ø1ØØ	$34
Ø11Ø.Ø1ØØ	$64

Exemple 5
Fonction ADD (ADDITION)

binaire	hexadécimal
Ø1Ø1.ØØ1Ø	$52
+ Ø1Ø1.11ØØ	$5C
1Ø1Ø.111Ø	$AE

Exemple 6
Fonction ADD avec CARRY
(ADDITION avec REPORT)

binaire	hexadécimal
Ø1Ø1.ØØ1Ø	$52
+ 11Ø1.Ø11Ø	$D6
1.ØØ1Ø.1ØØØ	$1.28

↑ Bit de carry (report)

Registres CPU

Le CPU (central processing unit, unité centrale) 6502 comporte plusieurs registres dans lesquels on peut réaliser des opérations logiques. L'un de ces registres se nomme "accumulateur", accu en abrégé.
Cet accumulateur a une longueur de 8 bits.

De sorte que tous les exemples (1 à 6) peuvent facilement être exécutés. On dispose même d'un bit de report (exemple 6).

Le microprocesseur doit d'abord recevoir une instruction disant quel nombre doit être inscrit dans l'accu. Cette instruction se dit "Charge l'accu avec une constante".

Le constructeur de la CPU l'a abrégée en "LDA" (Load Accu). Toutes les instructions de la CPU s'abrègent ainsi, avec trois lettres.

Exemple 3 sous forme de programme

Traduisons l'exemple 3 pour en faire un programme à introduire dans l'ordinateur. Le programme se "prononcerait" ainsi :

1. Charge l'accu avec la constante hexadécimale 7Ø (chargement direct).
2. Exécute une fonction OU avec le nombre hexadécimal 34.
3. Arrête-toi et affiche le résultat.

Cette façon d'écrire un programme serait rapidement cause de confusions inextricables. C'est la raison pour laquelle le fabricant de la CPU a codifié des abréviations (nommées "mnémoniques") qui donnent l'aspect suivant au programme :

```
LDA  #$7Ø
ORA  #$34
BRK
```

LDA, ORA et BRK sont les instructions, suivies des constantes.
\# indique qu'il s'agit d'un chargement immédiat.
$ indique qu'il s'agit d'un nombre hexadécimal.

Dans le tableau 1 nous trouvons les mnémoniques des instructions et dans le tableau 2 le code hexadécimal.

A l'aide des codes du tableau, nous pouvons maintenant introduire le programme dans le MPS 65. La constante se trouve immédiatement après chaque code d'instruction :

Adresses	Code hexa	Instruction	Constante
Ø2ØØ	A9	LDA	#$7Ø
Ø2Ø1	7Ø		
Ø2Ø2	Ø9	ORA	#$34
Ø2Ø3	34		
Ø2Ø4	ØØ	BRK	

— Introduisez ce programme.
— Retournez à l'adresse de démarrage Ø2ØØ.
— Démarrez le programme par G-Go.

Le programme est alors exécuté des adresses Ø2ØØ à Ø2Ø4 et s'arrête à Ø2Ø4.

L'instruction BRK (Break) provoque l'affichage des registres du CPU (les afficheurs 5 et 6 concernent l'accu).

Affichage :

(Registre X)	Registre (Y)	(Accu)	
0 0	0 0	7 4	Résultat (exemple 3)

Exemple 5
Programme d'addition.

Afin de transformer correctement l'exemple 5 en programme, il faut s'assurer que le bit de report est effacé avant l'addition. L'instruction CLC (clear carry) efface le bit de report (le met à Ø).

A l'aide du tableau 1, on peut maintenant écrire le programme pour l'exemple 5 :

```
LDA   #$52
CLC
ADC   #$5C
BRK
```

et l'introduire en code hexadécimal à partir de l'adresse $Ø2ØØ :

Ø2ØØ	A9	LDA	#$52
Ø2Ø1	52		
Ø2Ø2	18	CLC	
Ø2Ø3	69	ADC	#$5C
Ø2Ø4	5C		
Ø2Ø5	ØØ	BRK	

A la fin du programme le résultat de l'addition apparaît sur les afficheurs 5 et 6 (contenu de l'accu).

Faites l'expérience avec diverses valeurs.

Maintenant que nous avons donné des exemples d'opérations logiques simples, où nous avons toujours travaillé sur des "constantes", il faut montrer comment traiter des données tirées d'autres parties de l'ordinateur, et pas seulement de celles qui se trouvent déjà dans le programme.

Une telle possibilité de sélection s'appelle "type d'adressage".

Nous connaissons déjà ce type d'adressage, l'adressage direct ou immédiat, avec les exemples 1 à 5.

Nous connaissons également l'adressage "implicite", l'instruction CLC (clear carry - efface le bit de report) de l'exemple 5 utilisant ce type d'adressage.

L'adressage implicite ne nécessite pas d'indication d'adresses ou de constante et se compose d'un seul octet en code hexadécimal.

Les instructions comme CLC, CLD, TAX, etc. sont des instructions à adressage implicite.

L'adressage absolu est un autre type d'adressage, qui permet de combiner des valeurs à partir d'adresses diverses de l'ordinateur.

Les instructions utilisant ce type d'adressage se composent toujours de trois octets. Le code d'instruction est toujours suivi d'une adresse à 16 bits (2 x 8 bits).

Exemple 6 (mot à mot) :
Charge l'accu avec le contenu de l'adresse Ø3ØØ.

La mnémonique s'écrit :
LDA = Ø3ØØ

Le code hexa s'écrit :
AD (code d'instruction)
ØØ (partie de faible poids de l'adresse à 16 bits)
Ø3 (partie de poids élevé de l'adresse à 16 bits)

Exemple 7
Les contenus des adresses 0300 et 0301 doivent être utilisés pour une opération logique ET. Le résultat doit être inscrit à l'adresse 0302. Notre programme d'initiation commencera comme d'habitude à l'adresse #0200.

Ø2ØØ AD	(Code d'instruct.)	LDA $0300
Ø2Ø1 ØØ Ø2Ø2 Ø3	(Adresse à 16 bits)	
Ø2Ø3 2D	(Code d'instruct.)	ADN $0301
Ø2Ø4 Ø1 Ø2Ø5 Ø3	(Adresse à 16 bits)	
Ø2Ø6 8D	(Code d'instruct.)	STA $0302
Ø2Ø7 Ø2 Ø2Ø8 Ø3	(Adresse à 16 bits)	
Ø2Ø9 ØØ	(Code d'instruct.)	BRK

Notre exemple sera :

Ø11Ø.1ØØ1	$69
▲ ØØ11.11ØØ	$3C
ØØ1Ø.1ØØØØ	$28

Mettez les valeurs $69 et $3C dans les adresses $Ø3ØØ et $Ø3Ø1. N'oubliez pas, après introduction, de terminer par E-Enter.

Démarrez le programme à partir de l'adresse $Ø2ØØ.

Si, à la fin du programme, vous trouvez la valeur correcte de l'opération ($28) à l'adresse $Ø3Ø2, l'exemple a bien été suivi. Sinon, vérifiez le programme et les adresses utilisées.

Exercez-vous sur des exemples similaires avec ORA (OU), EOR (OU exclusif) et d'autres adresses.

Le type d'adressage "page zéro" (page Ø)

On nomme page une zone d'adresses qui peut être sélectionnée à partir des 8 bits de faible poids du comptage d'adresses.
L'adressage "page zéro" correspond en principe exactement à l'adressage absolu, à la petite différence près qu'il ne concerne que les adresses de $ØØØØ à $ØØFF.

L'instruction LDA $ØØ1Ø
(en code hexa AD 1Ø ØØ)
peut aussi s'écrire, comme elle concerne la page zéro :

LDA $1Ø (en code hexa A5 1Ø).

Les instructions d'adressage en page zéro se composent donc, contrairement à l'adressage absolu, de deux octets seulement au lieu de trois.
L'avantage de ce mode d'adressage spécifique réside dans le faible encombrement de mémoire et dans le temps d'exécution plus court.
L'indication spécifiant qu'il s'agit d'un adressage en page zéro se trouve dans le code spécial d'instruction (ici A5 au lieu de AD).

Exemple 8

Ø2ØØ A5	(Code d'instruct.)	LDA $1Ø
Ø2Ø1 1Ø	(Adresse à 8 bits)	
Ø2Ø2 25	(Code d'instruct.)	ADN $11
Ø2Ø3 11	(Adresse à 8 bits)	
Ø2Ø4 85	(Code d'instruct.)	STA $12
Ø2Ø5 12	(Adresse à 8 bits)	
Ø2Ø6 ØØ	(Code d'instruct.)	BRK

Le programme combine (par une fonction OU) les contenus des adresses $1Ø et $11 et enregistre le résultat à partir de l'adresse $12.
Introduisez le programme dans l'ordinateur à partir de l'adresse $Ø2ØØ et testez-en le fonctionnement à l'aide d'un déroulement. Modifiez les valeurs dans les adresses $1Ø et $11.

Brochage du connecteur de bus J1

Le brochage du connecteur J1 du MPS 65 est une extension du BUS SMP.
Les standards (adresses, données) et signaux particuliers (lignes de commande, alimentation) sont reliés aux fiches du bus.
Le tableau montre comment les diverses lignes sont reliées aux diverses broches. Les tirets indiquent que les broches sont disponibles pour d'autres signaux. La rangée b n'est pas utilisée.

Bus SMP étendu du MPS 65

1	−15V	1	−12V
2	−5V	2	GND
3	s.o.	3	+5V
4	Ø2	4	—
5	R/W	5	A12
6	RES	6	AØ
7	SYNC	7	A13
8	R/W	8	A1
9		9	A14
10	RAM R/W	10	A2
11	—	11	A15
12	READY	12	A3
13	BUSEN	13	—
14	DØ	14	A4
15	—	15	—
16	D1	16	A5
17	—	17	—
18	D2	18	A5
19	—	19	—
20	D3	20	A7
21	—	21	—
22	D4	22	A8
23	IRQ	23	—
24	D5	24	A9
25	—	25	—
26	D6	26	A1Ø
27	—	27	—
28	D7	28	A11
29	—	29	—
30	—	30	—
31	+15V	31	GND
32	+5V	32	+12 V
Rangée a		Rangée c	

Liste des instructions du microprocesseur MCS 650 X

ADC	Add Memory to Accumulator with Carry	JSR	Jump to New Location Saving Return Address
AND	"AND" Memory with Accumulator		
ASL	Shift Left One Bit (Memory or Accumulator)	LDA	Load Accumulator with Memory
		LDX	Load Index X with Memory
BCC	Branch on Carry Clear	LDY	Load Index Y with Memory
BCS	Branch on Carry Set	LSR	Shift Right One Bit (Memory or Accumulator)
BEQ	Branch on Result Zero		
BIT	Test Bits in Memory with Accumulator	NOP	No Operation
BMI	Branch on Result Minus	ORA	"OR" Memory with Accumulator
BNE	Branch on Result not Zero		
BPL	Branch on Result Plus	PHA	Push Accumulator on Stack
BRK	Force Break	PHP	Push Processor Status on Stack
BVC	Branch on Overflow Clear	PLA	Pull Accumulator from Stack
BVS	Branch on Overflow Set	PLP	Pull Processor Status from Stack
CLC	Clear Carry Flag	ROL	Rotate One Bit Left (Memory or Accumulator)
CLD	Clear Decimal Mode	ROR	Rotate One Bit Right (Memory or Accumulator)
CLI	Clear Interrupt Disable Bit	RTI	Return from Interrupt
CLV	Clear Overflow Flag	RTS	Return from Subroutine
CMP	Compare Memory and Accumulator		
CPX	Compare Memory and Index X	SBC	Subtract Memory from Accumulator with Borrow
CPY	Compare Memory and Index Y	SEC	Set Carry Flag
		SED	Set Decimal Mode
DEC	Decrement Memory by One	SEI	Set Interrupt Disable Status
DEX	Decrement Index X by One	STA	Store Accumulator in Memory
DEY	Decrement Index Y by One	STX	Store Index X in Memory
		STY	Store Index Y in Memory
EOR	"Exclusive-Or" Memory with Accumulator		
		TAX	Transfer Accumulator to Index X
INC	Increment Memory by One	TAY	Transfer Accumulator to Index Y
INX	Increment Index X by One	TSX	Transfer Stack Pointer to Index X
INY	Increment Index Y by One	TXA	Transfer Index X to Accumulator
		TXS	Transfer Index X to Stack Pointer
JMP	Jump to New Location	TYA	Transfer Index Y to Accumulator